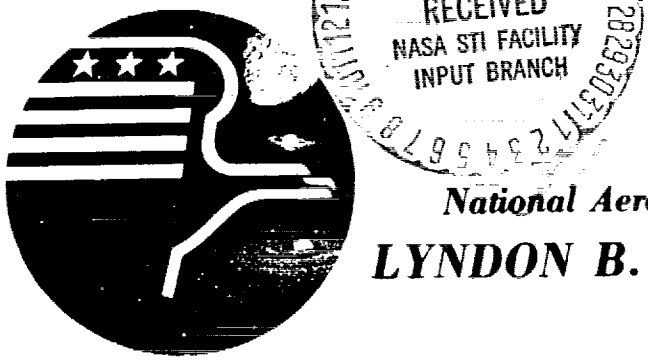
CR 151244

Space Shuttle Orbiter Avionics Software

ALT SYSTEM SOFTWARE DESIGN SPECIFICATION

Prepared By IBM
Under
NAS9-14444



National Aeronautics and Space Administration

LYNDON B. JOHNSON SPACE CENTER

Houston, Texas

February 28, 1977

SPACE SHUTTLE

ORBITER AVIONICS SOFTWARE

NAS 9-14444

APPROACH AND LANDING TEST (ALT)
SYSTEM SOFTWARE DESIGN SPECIFICATION

IBM Approval:

L. K. Lovinggood, Manager
Flight Software

D. F. McCabe, Chairman
Software Review Board

J. O. Lilly, Manager
Shuttle Project Office

NASA Approval:

R. P. Parten, Chief
Spacecraft Software Division

IBM FEDERAL SYSTEMS DIVISION, HOUSTON, TEXAS

77-SS-3576
IRD No. 2d4

TYPE I
2/28/77

FOREWORD

In response to the Space Shuttle Approach and Landing Test (ALT) Orbiter Avionics Flight Software requirements defined by NASA and documented in the Level A Computer Program Development Specification (CPDS) the IBM Federal Systems Division has developed this Detailed Design Specification (DDS).

DOCUMENT STRUCTURE

The System Software Design Specification (SSDS) replaces Part I, the Functional Design Specification, and Part II the Detailed Design Specification of the previous SSDS. In addition, the SSDS is Volume II of the total Flight Software Design Documentation.

The SSDS is divided into the following parts:

Part 1 - Flight Computer Operating System (FCOS)
Part 2 - User Interface (UI)
Part 3 - System Control (SC).

A list of acronyms and abbreviations has been placed at the end of Volume II.

TABLE OF CONTENTS

1. INTRODUCTION

This document presents the FCOS detail design descriptions of modules to be used for the Approach and Landing Test (ALT). The design presented reflects requirements as specified in the Computer Program Development Specification Level A (CPDS), Book 1 (Software), dated July 23, 1976 and all approved Change Requests.

The remainder of the INTRODUCTION, Section 1, discusses the documents purpose and Section 2 explains the format used to describe each module.

The MODULE DESCRIPTION Section, Section 3, contains the ALT detailed design for each module in FCOS.

Appendices contain additional information concerning the FCOS design:

Appendix A contains a FCOS Requirements - Detailed Design Cross Reference.

Appendix B contains the FCOS resource allocation summary.

Appendix C contains the FCOS module list.

Appendix D contains a table of error conditions and the default system action to be performed in FCOS.

Appendix E contains the data descriptors table.

Appendix F contains a table of modules which are OPS dependent.

Appendix G contains the FCOS I/O tables.

Appendix H contains a list of the FCOS Supervisor Calls.

Appendix I contains a description of the I/O profile.

A detailed knowledge of the HAL/S Realtime Statements and a general knowledge of the AP-101 flight computer are assumed. The following list of documents contain information regarding these topics:

- HAL/S Language Specification
- Interface Control Document: HAL/FCOS
- Space Shuttle Model AP-101 C/M Principles of Operation
  IBM File #6246156A
- Advanced System/4 Pi Model AP-101 Computer Software Systems Manual,
  IBM File #62280045

- FCOS User's Guide

1.1 PURPOSE

The Detail Design Specification provides explicit module descriptions of the implementation of the FCOS requirements. The detailed breakout of the modules, the detailed logic flows and the detailed data descriptions provide the final level of design description preceding the actual program listing.

1.2 SCOPE

The Flight Computer Operating System (FCOS) is a permanently resident set of software which manages the DPS computer hardware complex, allocates computer resources to all Flight Software functions and provides control and service functions to computing processes in the Flight Software. The FCOS can be divided into three major areas:

Process Management is the control and allocation of all internal computer resources. It includes a CPU scheduling algorithm, support for HAL/S realtime statements, Time and Event Management services, support for software system status monitoring.

I/O Management controls the initiation and the response to the I/O system hardware, provides servicing of all the application program requests for I/O, and provides processing of IOP hardware error interrupts and the detection and processing of I/O transmission errors.

DPS Configuration Management is the FCOS software support for the control of GPC and IOP Configuration changes. This includes control of internal configuration of a single GPC as well as control over the configuration changes resulting from internally detected hardware failures, software directive, or crew input, support for memory overlays, main/mass memory modification, and IOP bus configuration changes, and redundant computer management.

2. FORMAT EXPLANATION

The individual detail design description of each module consists of a Module Writeup which contains interfaces with other modules, a process description and any limitations the processing may have. The input and output to the module is presented in the Module Data Table and Module I/O Table. Finally, the detailed design is presented in the form of a structured control flow in the Module Control Flow.

2.1 MODULE IDENTIFIERS

Several methods for referring to a program have been defined. The program can be identified by its HAL/Assembler name, English_Equivalent_Name or its three digit ID.

2.1.1 English_Equivalent_Names

An English_Equivalent_Name is provided for each HAL/S or assembly language program. Each name conforms to the following standards.

a. It shall be uniquely defined.

b. It shall not contain other than accepted English abbreviations, acronyms defined in the Level A CPDS, or the acronyms Appendix to the SDDS.

c. It shall be concise.

d. It shall be indicative of its use.

e. Underscore connectors shall be used between words within the English name. Each new word will begin with a capital letter.

Example:

1. Error_Logging_Routine

2. LDB_Message_Router

2.1.2 Module Identification (ID) Codes

All programs, procedures, and segments (where segments are convenient modularizations of complex control flows) are identified by a numerical ID of the format XXX.XX. These codes are utilized in the Source/Destination section of the Data Table (see Section 2.3) and in the appropriate parts of the Control Flow (see Section 2.4).

The module ID's shall conform to the following standards:

a. A unique three digit code shall be invented for each Program and Procedure. The numbers are assigned in the following way:

FCOS	100-199	Process Management
	200-299	I/O Management
	300-399	Configuration Management
UI	400-699	
SC	700-999	

b. Segment ID codes shall consist of the same three digit prefix as the module it is a part of, followed by a decimal and a two digit number (e.g., 112.01). Segments are numbered consecutively even if they are referenced only by another segment.

2.2 MODULE WRITEUPS

The title at the beginning of each section shall consist of the section number, followed by the module English_Equivalent_Name, the module HAL/S or assembly language name and the module ID.

Example:

X.X.X.X. English_Equivalent_Name (MODULE1) (XXX)

Each module writeup is divided into the parts described below.

Function - This paragraph provides a very general description of the functions performed by the module.

a. Control Interface - This paragraph identifies all known users of the module and if SC or UI, its form of invocation. Priority and rate parameters are specified as symbolic parameters defined in Software Awareness Memo (SAM) 10.

Example:

Control Interface
1. Called by (ID) English_Equivalent_Name (Assembler Name)
2. Hardware PSW swap because of _ _ interrupt

b. Input - In most cases this is a reference to the module data table. However, if special interfaces need to be described they are described here.

c. Process Description - This is a narrative which functionally describes the processing performed by the module as depicted in its Control Flow. If data parameters are referenced, they are referenced by their English_Equivalent_Names.

d. Outputs - In most cases this is a reference to the module Data Table. However, if special interfaces need to be described, they are described here.

e. Module References - This section identifies all other modules which this module references. Each module is identified by a line consisting of the three digit ID, English_Equivalent_Name and HAL/S or Assembly language name. Modules are CALLed or SCHEDULEd. It also includes the FCOS service routine name of FCOS Supervisor Calls that are invoked.

Example:

1. (001) Reset_Event_Processor (FPMRESET) is CALLed.
2. (002) Lights_and_Alarms_Processing (DLA_LIGHT_ALARM_PROC) is CALLed.

f. Module Attributes

This section specifies whether a module is a program or an external procedure. An external procedure is a common subroutine which can be invoked by a CALL from a HAL/S module. All other modules are programs.

g. Template References - This section is not applicable to FCOS.

h. Error Handling

This section describes the error handling techniques employed by this module. For each error detected by this module, an explanation of the error and the action taken by the module is provided. If the module outputs an error message (via the User Interface Annunciation function), because an error was detected in another FCOS module, the error is briefly discussed and the error message number (see Appendix D) is enclosed in parenthesis.

i. Constraints and Assumptions - This section describes the programming limitations to which this module is subject and delineates the assumptions which are reflected in the design of this module.

j. Detailed Implementation - This section is used to provide a text extension of the logic statements which are usually enclosed within a processing block of a Control Flow. These logic statements may be in HAL/S form, but assembler language code is not allowed. The process block contains a reference number which relates to a number in this section of the module writeup.

2.3 MODULE DATA TABLE

All data input and output to a module or the hardware and internal parameters necessary for the understanding of a processing event (i.e., all data referenced in the control flow, narrative, or detailed implementation section) are itemized in a tabular format (see Figure 2.3-1). Each item in the table is explained below.

1 Name - The module English_Equivalent_Name along with the HAL/S or assembler name in parenthesis.

Example:

MCDS_Input_Processor (DMI_MCDS_IN)
Configuration_Management_SVC_Service_Routine (FCMSVC)

2 Item Number - Items are sequentially numbered in the general order of their appearance in the associated control flow.

3 Item - The English_Equivalent_Name assigned to the data in the Data Descriptors Table (see Appendix E).

4 Description ID - The ID assigned to the data in the Data Descriptors Table (see Appendix E).

5 Activity Type - This column denotes the usage of the parameter. Codes are:

I - An external memory location referenced by the module.
O - An external memory location changed by the module.
L - Local variable
Z - COMPOOL constant
R - Input from hardware (FCOS only)
W - Output to hardware (FCOS only)
T - Temporary Variable (Variables used only within a HAL DO group)

Any number of activity types that apply are specified in this column.

6 Source - A listing of the three digit ID's of all modules updating the parameter or hardware source of the parameter. If more than four modules update the parameter, a reference to the Data Descriptors Appendix is inserted instead of the list.

7 Destination - A listing of the three digit ID's of all modules referencing the parameter or the hardware source to which the parameter is being written. If more than four modules reference the parameter, a reference to the Data Descriptors Appendix is inserted instead of the list.

8 HAL/S Assembler Name - The HAL or Assembler name is used in coding. (Include the HAL equate name in parenthesis where applicable).

UI/SC - Column is HALNAME.
FCOS - Column is assembler Name.

If no name is assigned the column is blank.

9 MML - The Master Measurements List number in the requirements source. If not specified the column is left blank.

10 D - An 'X' is placed in the column if the parameter is available to be downlisted.

11 C - An 'X' is placed in the column if the parameter is available to be displayed on a CRT.

2.4 MODULE CONTROL FLOW

2.4.1 Flow Charts

System Software modules are documented by Structured Control Flows. Where possible/practical, the structure of the flow follows the structure of the program listing. Each page of flowcharting presents a complete picture of a primary processing objective with subsequent expansions of detail appropriately referenced. The expansion is accomplished either through a segment that is presented in detail on later pages of the control flow or through a reference to the Detailed Implementation section of the module writeup. The first page of each flow contains one elliptical block specifying "Enter" and a block specifying "Return". Logic flow is generally from top to bottom and left to right thus no arrowheads are required on connecting lines. See Figure 2.4-1 for the Control Flow legend.

The initial page of the control flow is titled with the same title as the module writup; that is English_Equivalent_Name and HAL/S or assembler name.

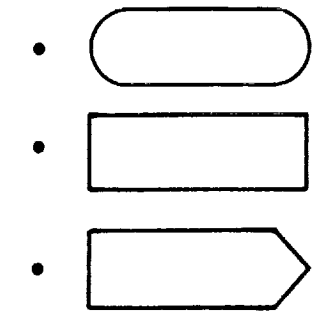
Example:

Figure X.X.X.X - English_Equivalent_Name (HAL_NAME)

Subsequent pages of a flow have the same title as the initial page without the HAL/S or assembler name. Additionally, they have a subtitle consisting of an English_Equivalent_Name for the segment and the ID defined in the process box which references the segment.

Example:

Figure X.X.X.X-2 English_Equivalent_Name
English_Equivalent_Segment_Name (XXX.1)

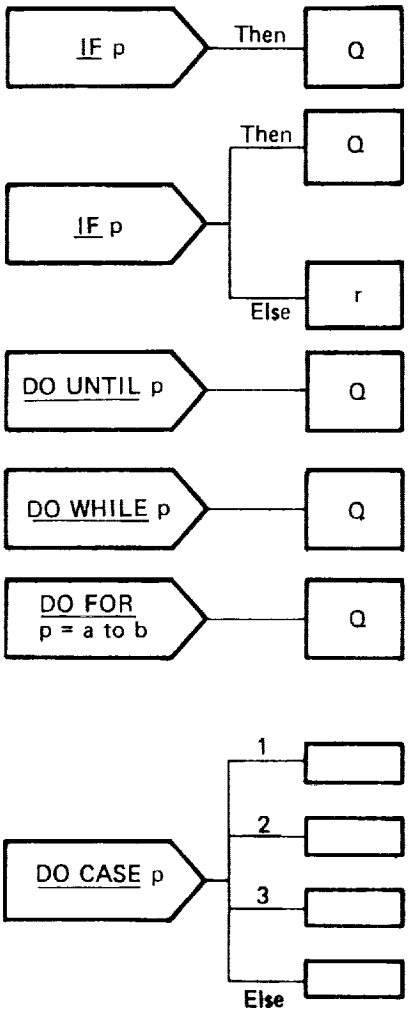


Terminal - identifies ENTER/RETURN point (such as HAL/S executable units of compilation and internal procedures and functions)

Mathematical statement/process control block which sometimes references another flow for a lower level of detail

Decision statement block

Documentation Examples



If p true then do Q, and return in-line*. (The word THEN is required.)

If p true then do Q and return in-line*, else do r, and return in-line*.

Do Q until p true, then return in-line*.
Note: The decision on p is made after doing Q.

Do Q while p true, then return in-line*.
Note: The decision on p is made before doing Q.

Do Q while p increments from a to b, and return in-line*.

Do one of specified processes based on value of p, and return in-line.* NOTE: Some identification of each case choice is given on the line leading into box, such as an index number, or the word ELSE (for the branch taken if the case index is outside its valid range).

* - Return in-line means return to the statement immediately below the decision. If there is no such statement, retrace path to the last previous decision and repeat this logic.

Figure 2.4-1. Control Flow Legend

3. FCOS MODULE DESCRIPTIONS

Flight Computer Operating System

FCOS is that portion of the System Services software that has direct control over the GPC and IOP hardware. It initiates and provides basic hardware control services for System Control. It also provides general processing control, input and output, and configuration management services to the User Interface, System Control, and application software.

GPC/IOP Initialization

When power is applied to a GPC, one of the FCOS initialization sequences is performed. First, if the entire GPC is to be reloaded, an IPL is performed after power on in the HALT mode and software is automatically loaded from memory by the hardware. The second is a SYSTEM RESET interrupt which corresponds to the normal restart sequence. In this case, FCOS: (1) sychronizes the internal timers with the MTU; and (2) initiates the System Control function.

At this time FCOS begins accepting interrupts associated with System Control and/or software descendants. After each interrupt the highest priority process is activated until a power-down interrupt occurs. At that time the hardware stores the GPC status in main memory to be available at the next power-on interrupt. Again FCOS is started and one of the two initialization sequences executed.

Processing

FCOS processes all interrupts and returns GPC control to the highest priority process. The only exception to the activation of the highest priority process is when a lower priority process is holding a shared resource required by the highest priority process. The shared resource could be either the execution of an exclusive procedure, use of mass memory, or a data update block being executed by a process that becomes a lower priority at the time of interrupt. If this occurs the lower priority process is activated until the shared resource is released.

The processing of an interrupt within FCOS cannot be interrupted except by Power Off, Machine Check, Instruction Monitor, and Program Interrupts. The seven blocks shown in the flow (Figure 3-1) represent the prime GPC and the basic types of interrupts. In the failure and processing exception cases, the FCOS performs one of the following actions:

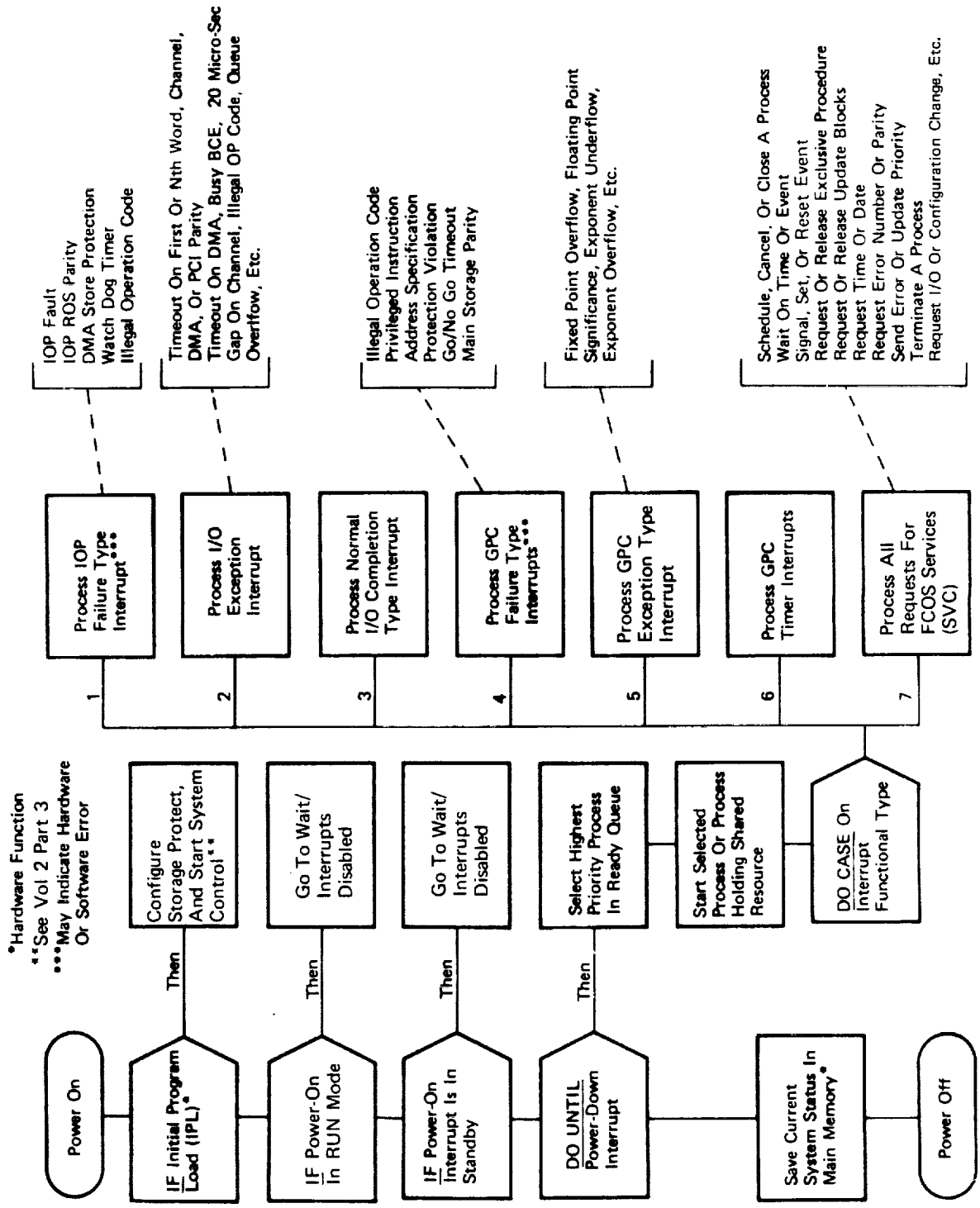


Figure 3-1. Flight Computer Operating System

- Ignores the interrupt and continues.
- Sends the indication to the application.
- Forces close of the process associated with interrupt.

After each interrupt is processed FCOS again activates the highest priority process on the ready queue. A process may be added to or removed from the ready queue within a service request interrupt routine. The services are defined by the real time statements of HAL/S and other functions of System Services.

GPC/IOP Configuration

The initial configuration allows System Control to access the DEU's and mass memories. System Control activates User Interface functions which can recognize user input that implies system reconfiguration. If it involves memory overlay and bus reconfiguration within a single GPC or single set of redundant GPC's, User Interface issues the request for FCOS to make the change. Otherwise the request goes to the Reconfiguration function in System Control.

FCOS acquires the overlay from the specified mass memory or another GPC's memory. Therefore, FCOS contains functions to both request and send requested overlays via the MM/LDB buses.

The remainder of the Functional Description is divided into the three major areas of FCOS (Figure 3-2):

a. Process Management - The allocation and control of internal computer resources.

b. I/O Management - The servicing of application process requests for input/output and the control of the initiation of and response to the interfaces of the I/O system hardware.

c. Configuration Management - The FCOS software support required for the control of GPC and IOP configuration changes.

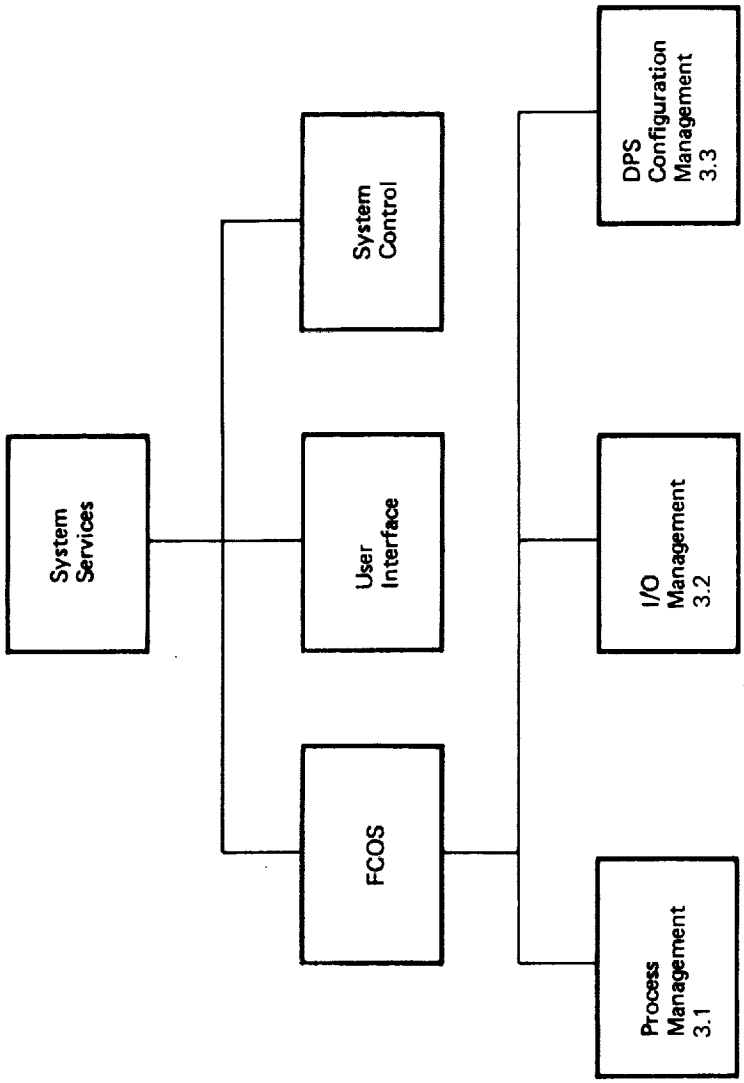


Figure 3-2. Functional Overview Hierarchy Diagram

3.1 PROCESS MANAGEMENT

Process Management is the control and allocation of computer resources to application software. Process Management maintains a process run queue that is ordered by predefined priorities of processes supplied to FCOS at process SCHEDULE time. This queue contains control and status information for all currently scheduled processes and is the basis of the FCOS process switching and dispatching functions. A process switch occurs whenever the active process goes into a wait state or completes (normally or abnormally), or a higher priority process is readied. When an interrupt occurs, after servicing the interrupt, the process dispatching function allocates the CPU to the highest priority ready process.

Process Management includes FCOS support for HAL/S realtime statements. Included are Time and Event Management services such as the queuing of time/event dependent requests and the activation of these requests when the time/event occurs. Additional Time Management functions such as maintenance of software clocks, program counters, GO/NO-GO timer, MTU and handling of application requests for date/time are also provided.

Also supported is the capability to allow a cyclic process to be initialized such that a predefined I/O operation will be automatically initiated by the FCOS upon receipt of the timer interrupt that readies the process.

FCOS Process Management also includes support for error exception handling. Application error recovery overrides are supported as well as system default response actions when no application overrides exist. Figure 3.1 illustrates the elements of Process Management and shows in which sections detailed descriptions of these elements can be found.

The description of Process Management is divided into the following area:

a. Process Control - The allocation of the CPU to application processes according to predefined rules.

b. Time Management - Software support for the program counters, the GO/NO-GO timer, the Master Timing Unit (MTU support included in this area of FCOS does not include input/output support), the software clocks, and the maintenance and activation of time-dependent requests.

c. Event Management - FCOS support for the changing of the status of application event variables and the maintenance and activation of event dependent requests.

d. Process Error Management - FCOS support for the handling of error conditions occurring during FCOS/application execution.

e. Idle_Time_Processor - A routine which approximates the amount of idle time calculated by the CPU duty cycle.

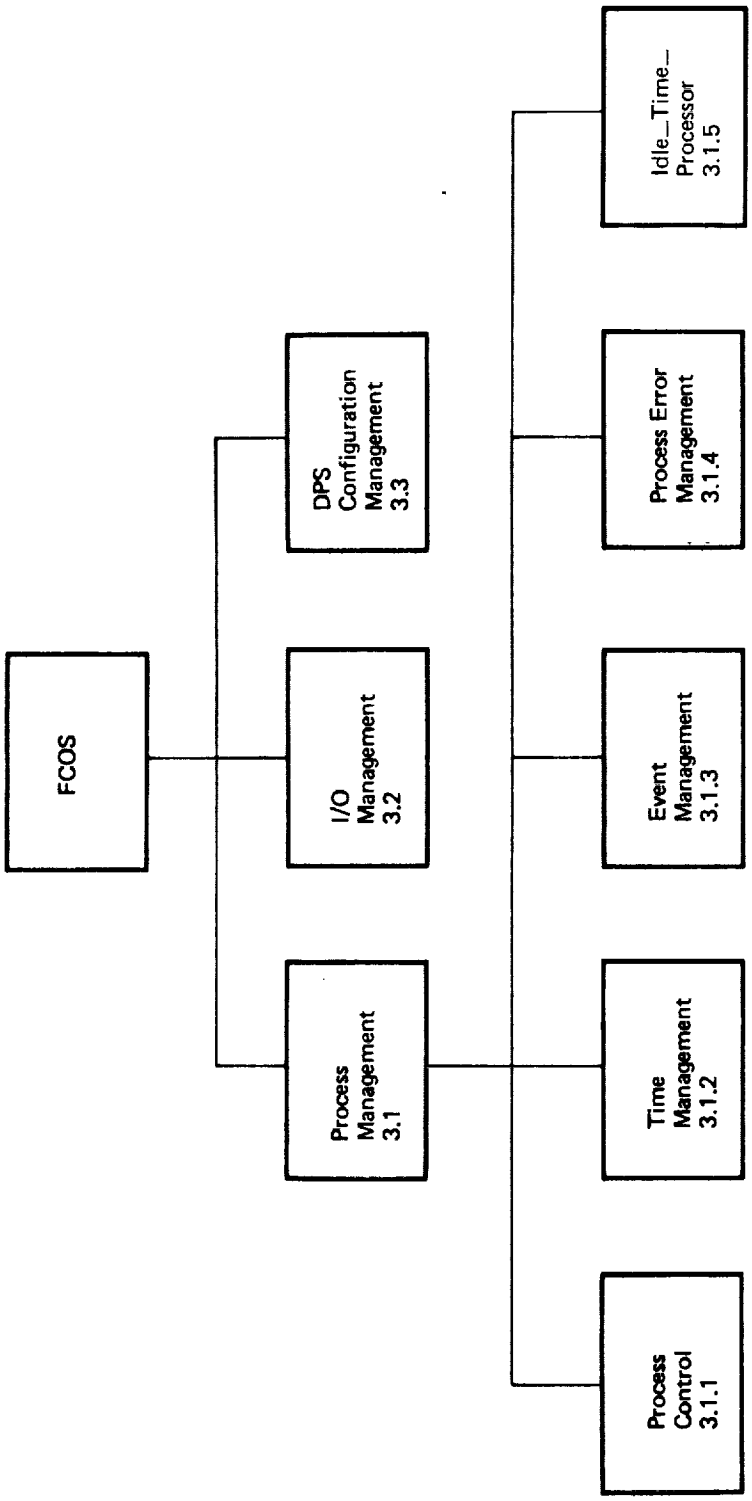


Figure 3.1. Process Management Hierarchy Diagram

3.1.1 Process Control

Process Control consists of the interface between the HAL/S application requests and FCOS, the servicing of process control related SVC requests, and the allocation of control of the CPU to application processes. Figure 3.1.1-1 represents the control flow which occurs when a supervisor call (SVC) is issued.

The application program interfaces with FCOS services via the SVC instruction which causes an interrupt with control being passed to the SVC_Handler. The SVC_Handler routes the SVC parameter list associated with the request to the appropriate FCOS routine to service the request. After the SVC request has been serviced, the Process_Dispatcher passes control of the CPU to the highest priority ready process. To indicate to the Process Dispatching function when a process switch should occur, FCOS routines utilize the Process_Switcher which indicates in an FCOS control block (Communications_Vector_Table) when a process switch is to occur. In cases when a process is readied, Process Switching is invoked to determine if the readied process is now highest in priority.

The main source of status and control information needed to perform Process Control functions is the process run queue. The process run queue is a priority-ordered queue of Process_Control_Tables (PCT). The ordering is based on pre-assigned priorities communicated to FCOS at process SCHEDULE time. When a process is scheduled, a PCT is initialized and maintained on the process run queue until the process completes (normally or abnormally). A scheduled process may exist in one of three states:

a. An active state in which the process has control of the CPU.

b. A ready state in which the process is capable of utilizing the CPU but does not have control.

c. A wait state in which the process must have some time pass, an event occur, or its I/O to complete before it becomes ready.

A process is said to be within an execution cycle after it has been readied to begin a cycle and before it issues its CLOSE statement to complete the cycle.

The information saved in the PCT comes from two major sources: the SCHEDULE_Parameter_List and the Process_Directory_Entry (PDE) associated with the process being scheduled. Each process which is to be scheduled must be known to FCOS via a PDE (generated as a CSECT by the HAL/S compiler for each process).

In addition to information in the Process_Control_Table and the Process_Directory_Entry, the FCOS Process Control functions maintain status information in the Communications_Vector_Table (CVT). Contents of this table which are of primary importance to Process Control are:

- A pointer to the beginning of the process run queue.
- A pointer to the process control table of the active process.
- A pointer to the process control table of the next to execute process.

FCOS routines manipulate the next-to-execute field to indicate when another process (other than the active process) should be given control. Whenever this next-to-execute field contains a PCT address, this PCT is the highest priority ready process. After an interrupt has been serviced, control of the CPU is given to a process other than the interrupted process when:

- The interrupted process has entered the wait state.
- The interrupted process has completed (normally or abnormally).
- The interrupted process is no longer the highest priority ready process.

The contents of the tables mentioned can be found in Appendix E. The remainder of the detailed description of Process Control is divided into the following major areas (Figure 3.1.1-2):

a. SVC_Handler - Routing of Supervisor Call requests to appropriate FCOS service routines.

b. Process_Scheduler - Servicing of an application SCHEDULE request.

c. Process_Switcher - Determining whether a readied process is the highest priority ready process.

d. Process_Dispatcher - Passing control of the CPU to the appropriate process.

e. WAIT_Processor - Servicing of an application request to delay processing.

f. UPDATE/EXCLUSIVE_Reserve_Processor and UPDATE/EXCLUSIVE Release Processor - FCOS control of resources associated with UPDATE blocks and EXCLUSIVE procedures.

g. CLOSE_Processor - Clean-up and/or reinitialization functions necessary at the end of the current execution cycle of a process.

h. TERMINATE_Processor - Servicing of an application request to terminate itself or other process(es).

i. CANCEL_Processor - Servicing of an application request to cancel itself or other process(es).

j. OPS_CANCEL_Processor - Servicing of an application request to cancel all the processes within a major function except for the specified process(es).

k. Select_I/O_Processor - Servicing of an application request to associate timer initiated I/O with a cyclic process.

l. Hybrid_Dispatching_Routine - Provides the applications with the capability to call application functions (HAL/S External Procedures) at a set frequency and phase relationship and to provide active/inactive flags for each function.

m. Chain_PCT_Routine, Release_PCT_Routine, and Free_PCT_Routine - PCT handling routines.

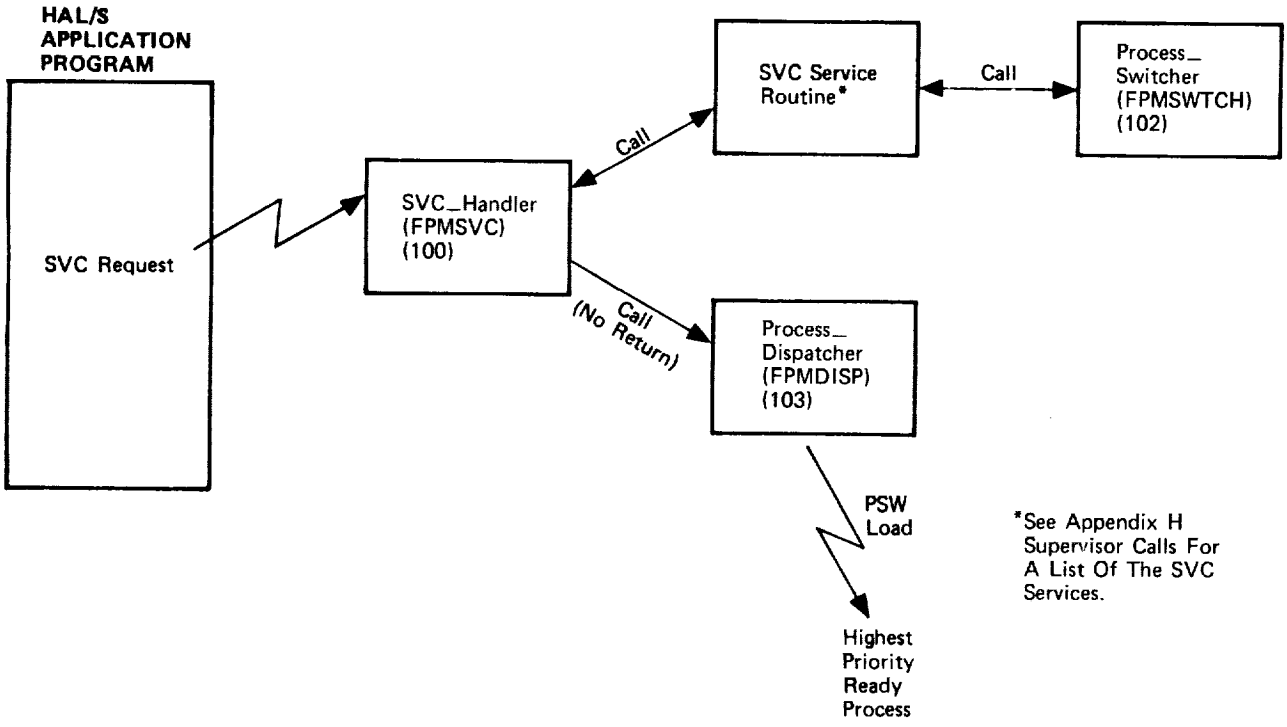


Figure 3.1.1-1 SVC Control Flow

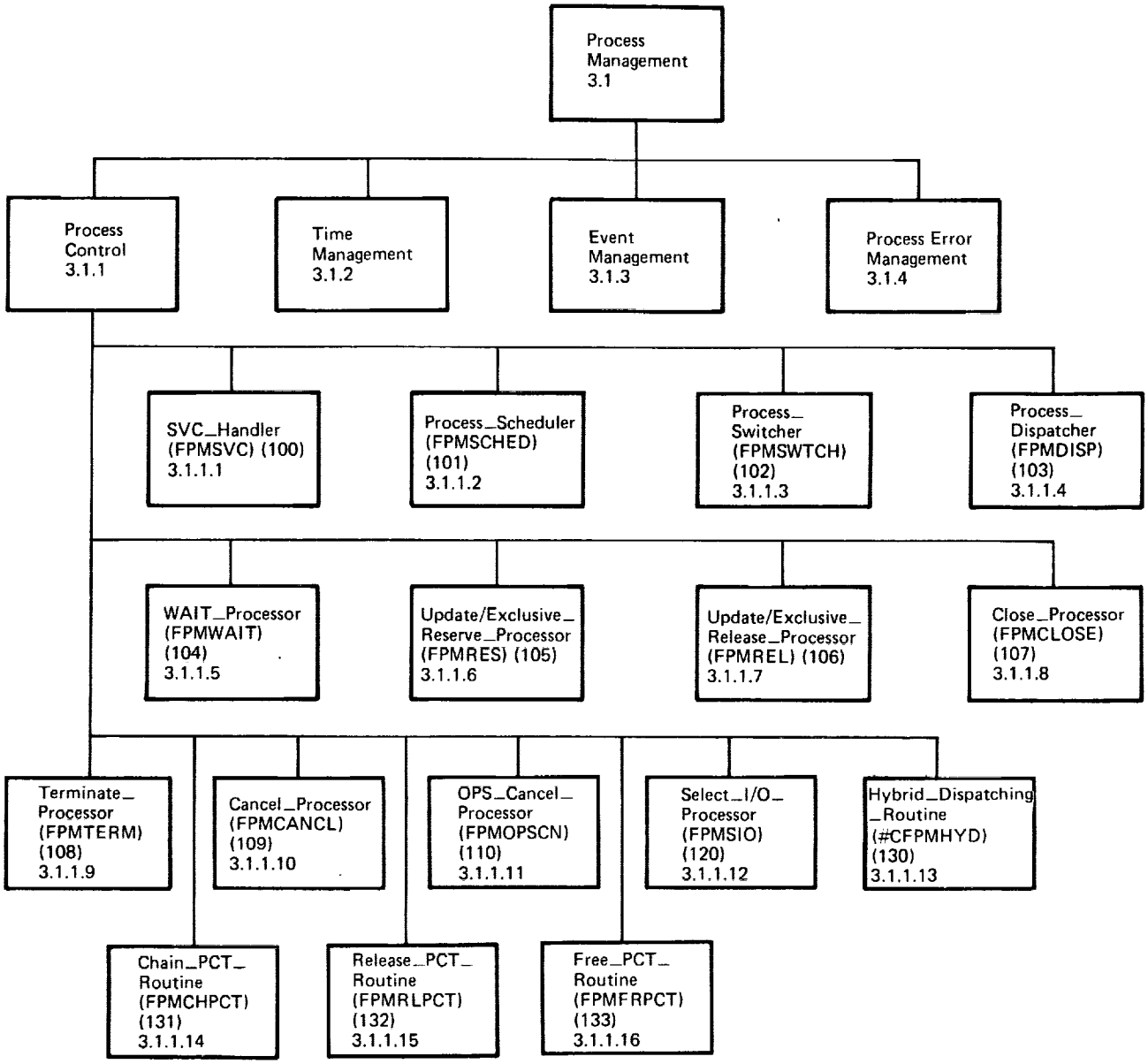


Figure 3.1.1-2. Process Control Hierarchy Diagram

3.1.1.1 SVC_Handler (FPMSVC) (100)

FPMSVC receives control whenever an SVC is issued. It passes the request to the proper SVC routine for servicing.

a. Control Interface - Hardware PSW swap because of SVC interrupt.

b. Input - See Table 3.1.1.1-1.

c. Process Description - The SVC_Handler intercepts all SVC's. The address of the SVC parameter list which contains the SVC number identifying the service being requested and other parameters needed to service the request is located in the PSW_Interrupt_Code of the SVC_Old _PSW. The SVC number is obtained from the parameter list and according to this number, the appropriate FCOS module is invoked to service the request. After the SVC request has been serviced, the Active_PCT_Address and the Next_To_Execute_PCT_Address are compared to determine whether the process issuing the SVC should be returned control of the CPU (the fields are equal) or a process switch should occur (the fields are not equal). The control flow for this module is presented in Figure 3.1.1.1-1.

d. Output - Register 0 contains the address of the SVC parameter list. See Table 3.1.1.1-1.

e. Module References -

1. Appropriate SVC Service Routine is CALLed (See Appendix H).
2. (103) Process_Dispatcher (FPMDISP) is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - The SVC number of the service is assumed by the SVC_Handler to be valid (No validity checking is performed).

j. Detailed Implementation -

1. Control is passed via a PSW load of the Interrupt_PSW_Of_Process.

2. The Process_Dispatcher passes control to the appropriate process; therefore there is no return to the SVC_Handler after the CALL.

DATA TABLE 3.1.1.1-1

NAME SVC Handler (FPMSVC)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	PSW_Interrupt_Code	&002.26	I		180, 242, 320				
2	SVC_Old_PSW	&001.67	I		See App. E	TPSASOP			
3	Interrupt_PSW_Of_Process	Q003.5	O	See App. E	See App. E	TPCTPSW			
4	Active_PCT_Address	Q001.2	I	103	See App. E	TCVTOLD			
5	SVC_Entry_Point_Table	@005	I		100	FPMSVCEP			
6	Next_To_Execute_PCT_Address	Q001.3	I	See App. E	See App. E	TCVTNEW			

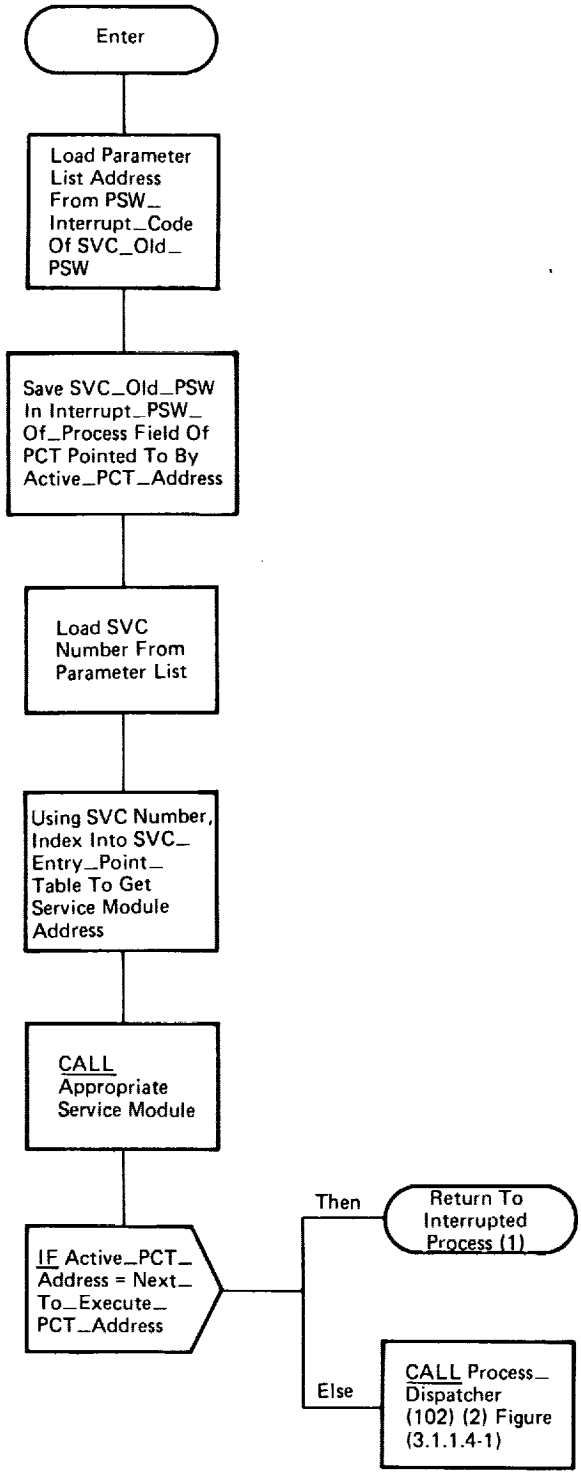


Figure 3.1.1.1-1. SVC_Handler (FPMSVC)

3.1.1.2 Process_Scheduler (FPMSCHED(101))

FPMSCHED processes an application request to add a process to the process run queue.

a. Control Interface - CALLed by (100) SVC_Handler (FPMSVC).

b. Input - See table 3.1.1.2-1.

Floating point registers 0, 1 contain the AT or IN time value if specified. The AT value will be specified in GMT; the IN value will be expressed as a delta time value from current GMT.

Floating point registers 2, 3 contain the REPEAT EVERY or REPEAT AFTER time value if specified. The time value will be expressed as a delta time.

Floating point registers 4, 5 contain the UNTIL time value if specified. The UNTIL time value will be expressed as GMT. (All time will be double precision in seconds and fractions of seconds to the nearest millisecond).

c. Process Description - The Process_Scheduler invokes the SVC_Synchronization_Processor to synchronize the redundant set GPC's. Upon return, the Process_Scheduler obtains the address of the Schedule_Parameter_List from the SVC_Old_PSW. If the Assigned PCT_Address for this process indicates that the process is already scheduled, no further processing is done (i.e., control is returned to the SVC_Handler). Otherwise the following processing is done.

An unused PCT, located by PCT_Free_Pool_Address, is initialized for the process being scheduled using information from the Schedule_Parameter_List and from the PDE for this process (indicated by Process_PDE_Address). If the SSIP_Schedule_Indicator shows that this is the SSIP process which is being scheduled, the SSIP_PCT_Indicator is set.

If Cancellation_Criteria and/or Initial_Conditions have been specified, necessary TQE's/EQE's are initialized and the Timer_Queue_Generator and/or the Event_Queue_Generator are invoked to determine if the specified conditions are met and to queue the appropriate queue elements if necessary. If Cancellation_Criteria are already met and the "until event" has not been specified, no further processing is done (since no queue elements have been queued and no indications have been made denoting the process as scheduled, nothing is necessary but to return to the SVC_Handler. Phased scheduling is the adjustment of the start time of a cyclic process so that the timer interrupts for its subsequent cycles have a specified phase relationship to interrupts of other cyclic processes. FCOS performs phased scheduling when the specified AT time has already

passed for a REPEAT EVERY process. This processing is done in Timer_Queue_Generator when invoked to process the AT time. If initial conditions are not met, the Schedule_Wait_Indicator is set.

If Repeat_Options are specified, the delta time specified in floating point registers 2 and 3 is converted to FCOS fixed point time format by invoking the Convert_To_Fixed_Point_Routine. The returned result is placed in the Repeat_Delta_Time in the PCT. For a process with the REPEAT EVERY option, the Timer_Initiated_I/O_Table is examined to determine if the process is to have timer initiated I/O associated with it. If so, the Time_Initiated_I/O_Parameter_List_Address field of the PCT is set to the appropriate address. The Timer_Queue_Generator is invoked to queue the REPEAT EVERY TQE if the process PCT_Wait_Indicators are zero.

Pre-allocated areas of memory are used as stacks (temporary storage areas) by application processes. These areas are used as register save areas when calling subroutines, as data storage areas, and as error environment definition areas. When the Stack_Size/Address_Indicator indicates that the process does not have a pre-assigned stack, the Process_Scheduler assigns one of sufficient size to the process from a pre-allocated pool of stack areas starting at Stack_Pool_Address. Whether or not a process has a pre-assigned stack and the size of the area needed is determined at compile and linkedit time.

When the PCT is initialized, it is placed into the PCT run queue at a level reflecting the PCT_Priority. If the process is not in a wait state (PCT_Wait_Indicators=0), the Process_Switcher is invoked to determine if this process is now the highest priority ready process. If the process is in a wait state, no further processing is done at this time. The process will be readied by Time/Event Management, depending on the Initial_Conditions specified.

d. Outputs - See Table 3.1.1.2-1

e. Module References

1. (363) SVC_Synchronization_Processor (FCMSSYNC) is CALLed.
2. (131) Chain_PCT_Routine (FPMCHPCT) is CALLed.
3. (140) Timer_Queue_Generator (FPMTMENQ) is CALLed.
4. (160) Convert_To_Fixed_Point_Routine (FPMCVTFX) is CALLed.
5. (173) Event_Queue_Generator (FPMEVENQ) is CALLed.
6. (174) Event_Evaluator (FPMEVAL) is CALLed.
7. (102) Process_Switcher (FPMSWTCH) is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - The Process_Scheduler assumes that there is a PDE existing for the process being scheduled. No check is made to verify that the Process_PDE_Address is not zero.

j. Detailed Implementation

1. The number of entries in the Timer_Initiated_I/O_Table is currently 2. It is assumed that no more than 2 requests will be made to associate I/O with a process before issuing the SCHEDULE statement for one or both of the associated processes. Once the SCHEDULE is done for one of the processes, the entry in the Timer_Initiated_I/O_Table is available for associating another process with I/O.

DATA TABLE 3.1.1.2-1

NAME Process_Scheduler (FPMSCHED)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Schedule_Parameter List	S001			101				
2	PSW_Interrupt_Code	&002.26			180, 242,320				
3	SVC_Old_PSW	&001.67	I		See App. E	TPSASOP			
4	Process_PDE_Address	S001.11	I		101	TSCDPDE			
5	Assigned_PCT_Address	Q002.2	I	101, 133	101, 108	TPDEPCT			
			O		109, 110				
6	Process_Entry_Point	Q002.3	I	101	101	TPDEP			
			O		107				
7	PSW_Data_Sector_Register	&002.11							
8	SVC_New_PSW	&001.68	I	300	101	TPSASNP			
					140				
9	PCT_Free_Pool_Address	Q001.9	I	131, 132	101, 280	TCVTPCTP			
				305	320				
10	PCT_Flags	Q003.29	O	101, 142	107, 133	TPCTFLGS			
				320	140, 225				
11	Process_Schedule_Options	S001.3	I		101	TSCDFLGS			
12	PCT_Priority	Q003.2	O	101, 105	See App. E	TPCTPRI			
				106					
13	Process_Priority	S001.1	I		101	TSCDPRIO			
14	PDE_Address	Q003.4	O	101	See App. E	TPCTPDE			

DATA TABLE 3.1.1.2-1 (Cont'd)

NAME Process_Scheduler (FPMSCHED)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
15	Outstanding_I/O_Count	Q003.25	O	101,200, 201	133	TPCTIORQ			
16	Original_Priority	Q003.26	O	101,105, 106,320	106,131	TPCTOPRI			
17	Last_Error_Group_Code	Q003.27	O	101, 107	185	TPCTERR			
				183					
18	Program_Interrupt_Count	Q003.28	O	101, 107	182	TPCTECNT			
19	PCT_Wait_Indicators	Q003.42	O	See App. E.	See App. E	TPCTWAIT			
			I						
20	Interrupt_PSW_of_Process	Q003.5	O	See App. E.	See App. E	TPCTPSW			
21	PSW_Instruction_Address	&002.1			183,320				
22	PSW_Branch_Sector_Register	&002.10			183				
23	PSW_Program_Mask	&002.2			182				
24	PSW_System_Mask	&002.12			320				
25	PSW_Register_Set	&002.22		182	320				
26	PSW_Machine_Check_Mask	&002.23			320				
27	PSW_Wait_State_Indicator	&002.24		300	320				
28	PSW_Run_State_Control	&002.25			180,181, 320				
29	SSIP_Schedule_Indicator	#002.10	I	101	101	TFCMFLAG			
			O						
30	SSIP_PCT_Indicator	Q003.38	O	101	101,142	TPCTFLGS			

DATA TABLE 3.1.1.2-1 (cont'd)

NAME Process_Scheduler (FPMSCHED)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
31	Cancellation_Criteria	Q003.35	O	101	101	TPCTFLGS			
			I						
32	Canceled_PCT_Indicator	Q003.30	O	101, 109	See App. E	TPCTFLGS			
			I	142	225				
33	Initial_Conditions	Q003.39	O	101,	101	TPCTFLGS			
			I	174	174				
34	Repeat_Options	Q003.36	O	101	101, 107	TPCTFLGS			
			I	131,174	131,174				
35	Stack_Size/Address_Indicator	Q002.6	I		101	TPDEFLGS			
36	Stack_Pool_Address	Q001.13	I		101, 300	TCVTSTOR			
					305				
37	Stack_Information	Q002.4	I		101, 107	TPDESTAK			
38	FCOS_Assigned_Stack_Address	Q003.3	O	101	107, 133	TPCTSTOR			
39	Process_Interrupt_Register_0	Q003.6	O	101, 103	103, 108	TPCTGPRO			
				107, 182	182				
40	Stack_Frame	@002	O	101	182,	TFTSA			
					290				
41	PSW_Program_Mask_Field	@002.2	O	101	182	TTSAPGMM			
42	Previous_Stack_Frame_Address	@002.3	O	101	108,182	TTSAPREV			
					184				

DATA TABLE 3.1.1.2-1 (cont'd)

NAME Process_Scheduler (FPMSCHED)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
43	Local_Data_Area_Address	@002.9	O	101	108, 182	TTSALCDA			
					184, 290				
44	Event_State	@007.2		See App. E	See App. E				
45	Process_Event	Q1002.1	O	See App. E	See App. E	TPDEVENT			
46	Event_Used_Indicator	@007.1			See App. E				
47	TQE_Free_Pool_Address	Q001.11	I	142, 143	See App. E	TCVTTQEP			
				166, 305					
48	TQE_Parent_PCT_Address	Q005.2	O	See App. E	See App. E	TTQEPCT			
49	TQE_Type_Indicator	Q005.10	O	101, 104	140, 142,	TTQEFLGS			
				107	148,174				
50	TQE_ Status	Q005.7	O	101, 143	142	TTQEFLGS			
51	TQE_Request_Type	Q005.9	O	101, 104	140	TTQEFLGS			
				107, 140					
52	Null_TQE_Indicator	Q005.8	O	101, 142	140	TTQEFLGS			
53	EQE_Free_Pool_Address	Q001.10	I	173, 175	101, 104	TCVTEQEP			
					174				

DATA TABLE 3.1.1.2-1 (Cont'd)

NAME Process_Scheduler (FPMSCHED)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
54	EQE_Parent_PCT_Address	Q006.2	O	101, 104	173, 174	TEQEPCT			
				175	175				
55	EQE_Type	Q006.11	O	101, 104	173, 174	TEQETYPE			
				173,175	175				
56	Option_Event_Type	Q006.12	O	101, 104	173,174	TEQETYPE			
57	Cancelation_Event_Expression_Address	S001.13	I		101	TSCDEXP2			
58	Operator_Field	Q006.3	O	101, 104	173, 174	TEQEOPS			
59	Count_of_Operators	Q006.4	O	101, 104	173	TEQEOPS			
60	Event_Expression_Operators	Q006.5	O	101, 104		TEQEOPS			
61	Event_Variable_1	Q006.6	O	101, 104	173, 174	TEQEVAR1			
				173, 175	175				
62	Event_Variable_2	Q006.7	O	101, 104	174	TEQEVAR2			
				173					
63	Event_Variable_3	Q006.8	O	101, 104	173,174	TEQEVAR3			
64	Event_Variable_4	Q006.9	O	101, 104	173,174	TEQEVAR4			
65	Event_Variable_5	Q006.10	O	101, 104	173,174	TEQEVAR5			
66	ON_EVENT_Expression_Address	S001.12	I		101	TSCDEXP1			
67	Scheduled_Wait_Indicator	Q003.48	O	101, 107	See App. E	TPCTWAIT			
				142					
68	Delta_Time_30_Minute_Portion	Q003.23	O	101	140	TPCTRPTH			

DATA TABLE 3.1.1.2-1 (Cont'd)

NAME Process_Scheduler (FPMSCHED)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
69	Delta_Time_Half_Hour_Counter	Q003.24	O	101	140	TPCTRPTM			
70	Time_Initiated_I/O_Parameter_List_Address	Q003.49	O	101	140	TPCTIOPP			
71	Time_Initiated_I/O_Table	Y025	I	101	101	FPMTIOT			
			O		120				
72	Time_Initiated_I/O_Parent_PDE_Address	Y025.1	I	101	101	TTIOTPDE			
			O	120	120				
73	Process_I/O_Parameter_List_Address	Y025.2	I	120	101	TTIOTIO			

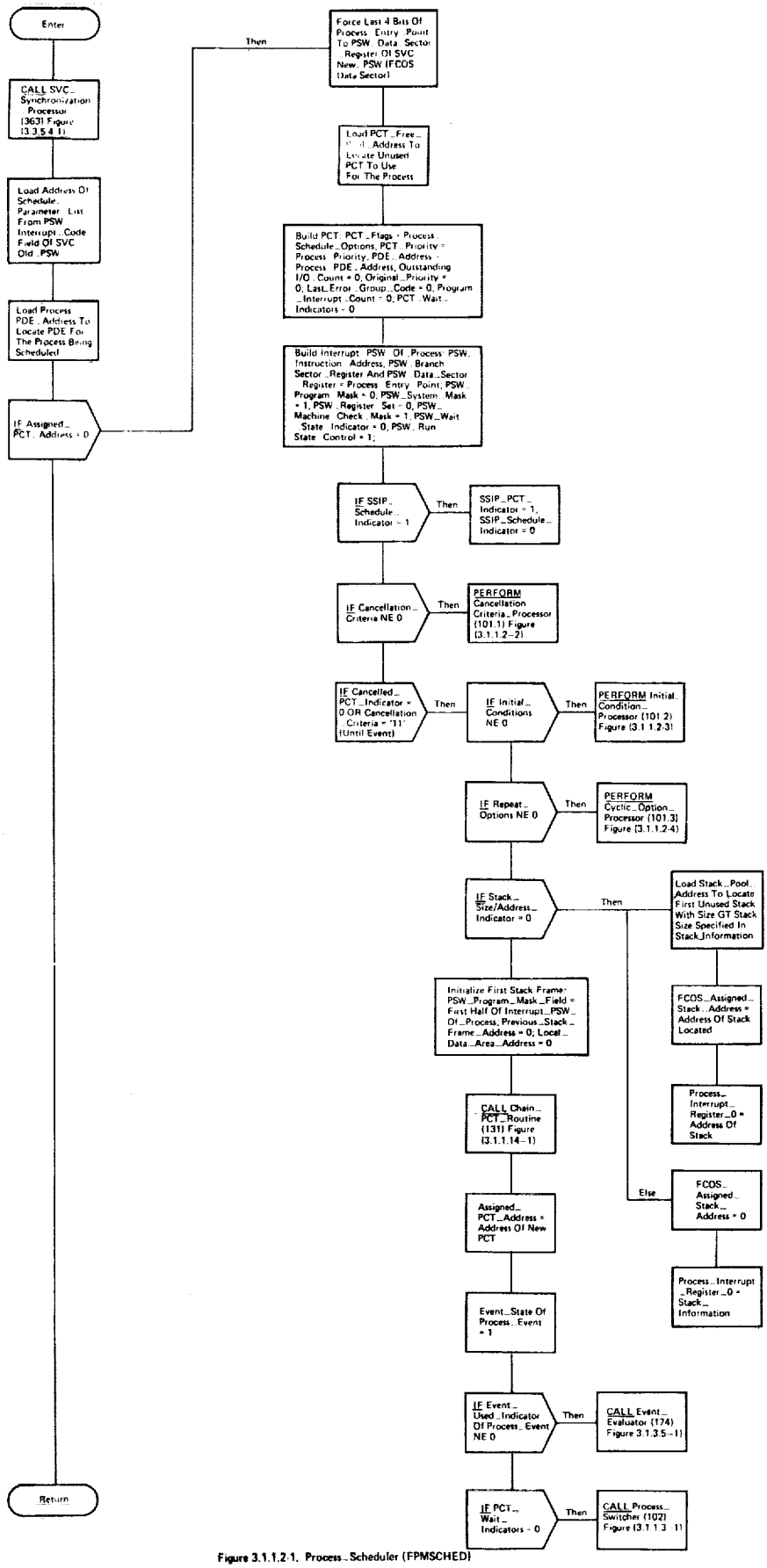


Figure 3.1.1.2-1. Process_Scheduler (FPMSCHED)

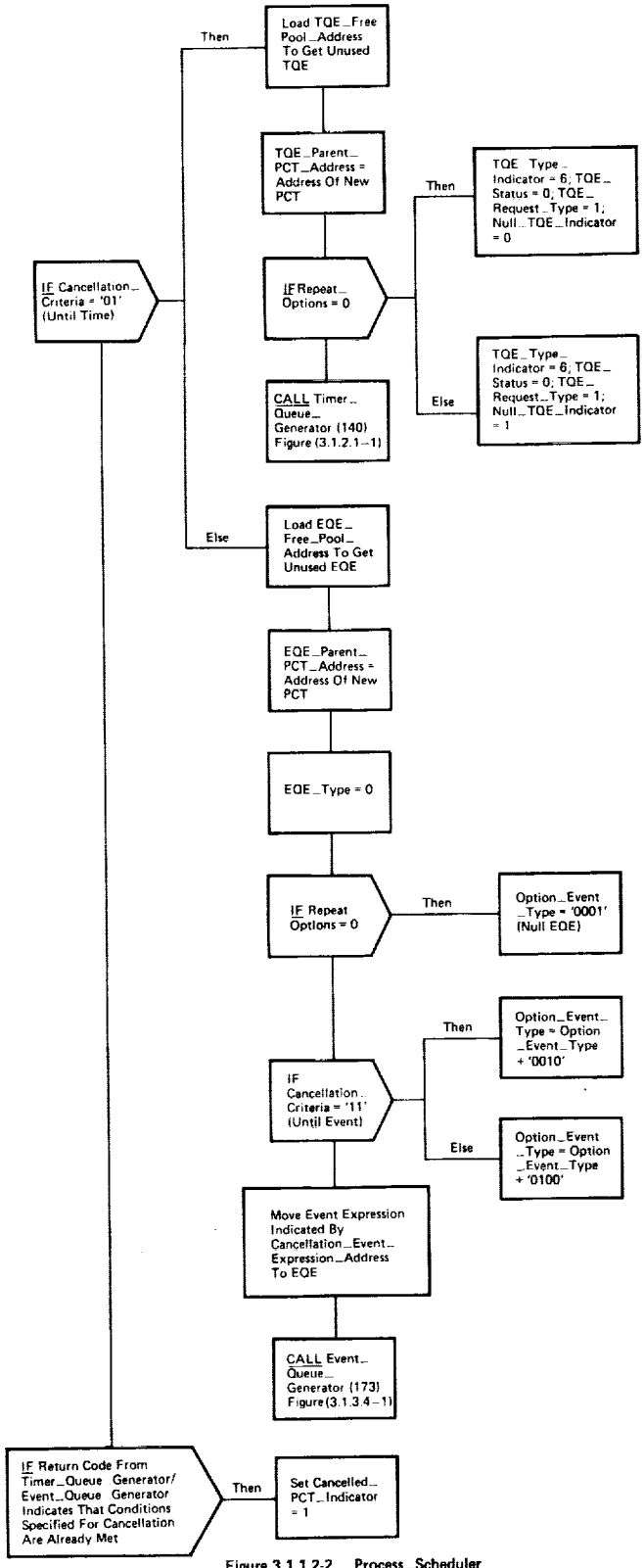


Figure 3.1.1.2-2. Process_Scheduler Cancellation_Criteria_Processor (101.1)

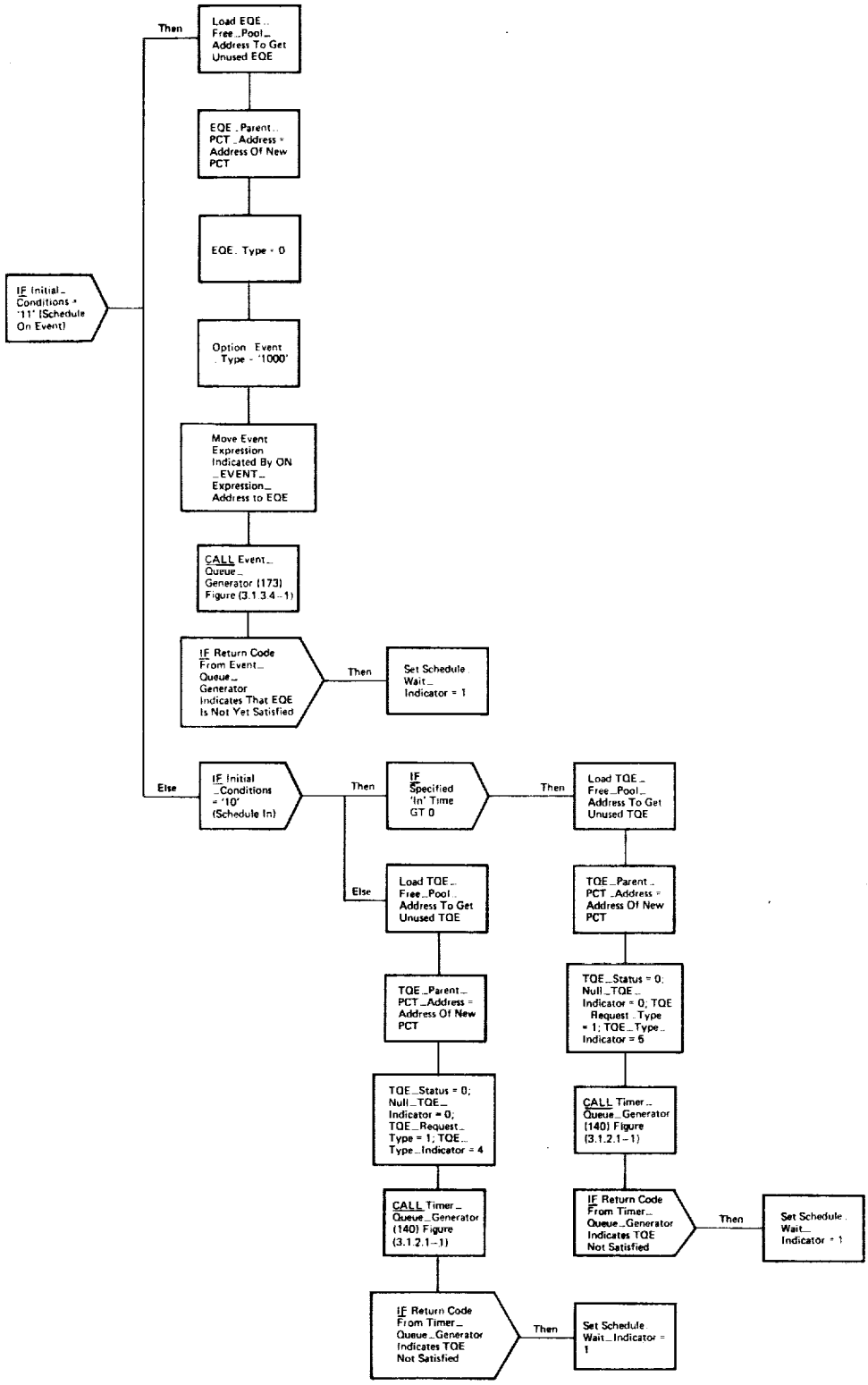


Figure 3.1.1.2-3. Process_Scheduler Initial_Condition_Processor (101.2)

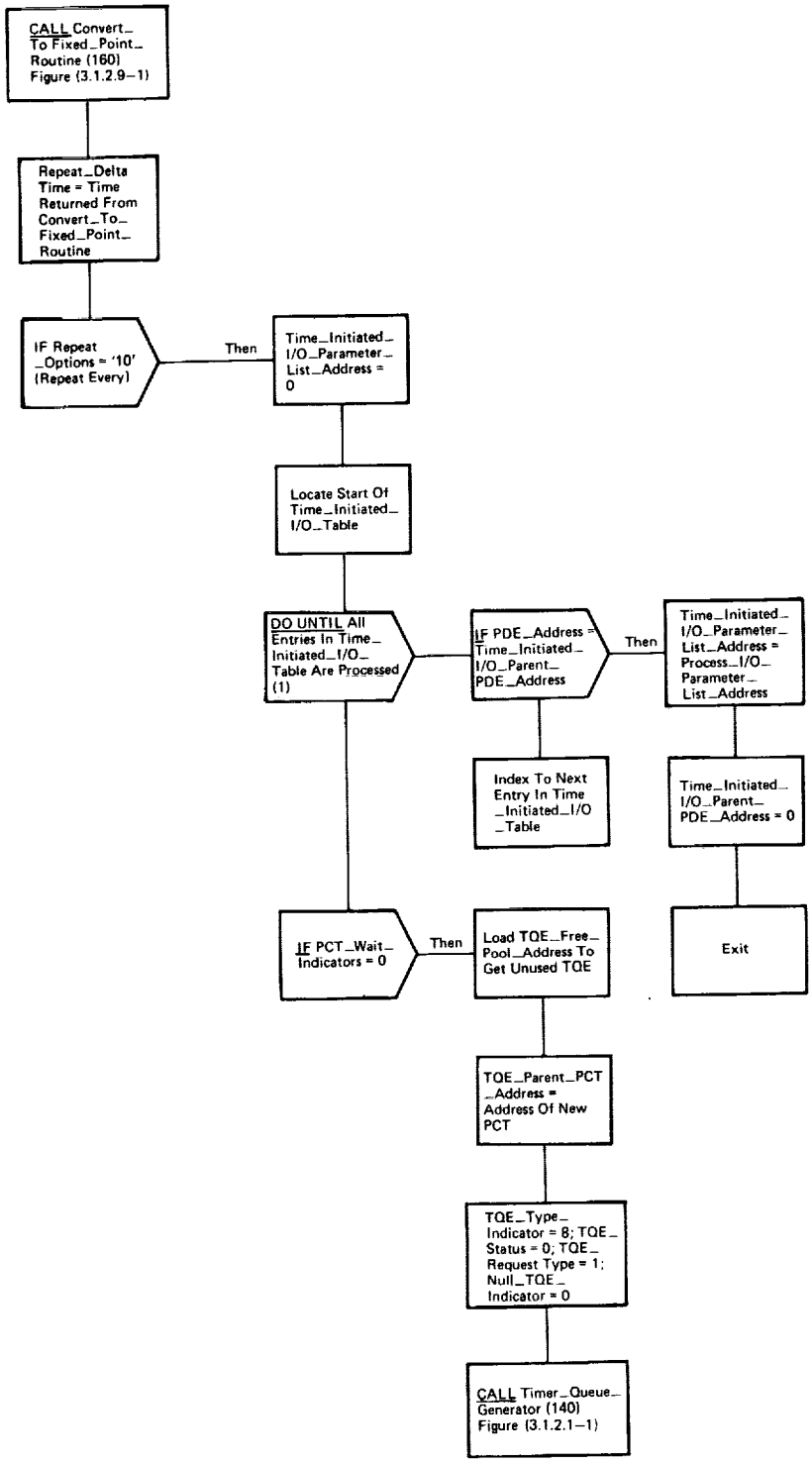


Figure 3.1.1.2-4. Process_Scheduler Cyclic_Option_Processor (101.3)

3.1.1.3 Process_Switcher (FPMSWTCH) (102)

FPMSWTCH determines if a readied process is the highest priority ready process.

a. Control Interface -

1. CALLed by (101) Process_Scheduler (FPMSCHED)
2. CALLed by (174) Event_Evaluator (FPMEVAL)
3. CALLed by (142) TQE_Expiration_Processor (FPMIHPC2)
4. CALLed by (220) I/O_Completion_Processor (FIOCMPLT)
5. CALLed by (280) Mass_Memory_Manager (FIOMMMGR)

b. Input - Register 0 contains the address of the input PCT. See Table 3.1.1.3-1 for other input.

c. Process Description - The Process_Switcher compares the PCT_Priority of the readied process (input PCT) to the PCT_Priority of the process identified in the Next_To_Execute_PCT_Address and/or to the PCT_Priority of the process identified in the Active_PCT_Address. Since the priority of the next to execute process is always greater than that of the current process, there is not a need to check any further if the readied process does not have a priority greater than the current process. If the readied process is the highest in priority, the Next_To_Execute_PCT_Address is set equal to the input PCT address. The control flow for this module is shown in Figure 3.1.1.3-1.

d. Outputs - See Table 3.1.1.3-1.

e. Module References - NONE

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - NONE

i. Constraints and Assumptions - NONE

j. Detailed Implementation - NONE

DATA TABLE 3.1.1.3-1

NAME Process_Switcher (FPMSWTCH)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Active_PCT_Address	Q001.2	I	103	See App. E	TCVTOLD			
2	PCT_Priority	Q003.2	I	101, 105 106	See App. E	TPCTPRI			
3	Next_To_Execute_PCT_Address	Q001.3	I,O	See App. E	See App. E	TCVTNEW			
4	Next_PCT_Address	Q003.1	I	101, 131 132	See App. E	TPCTNXT			

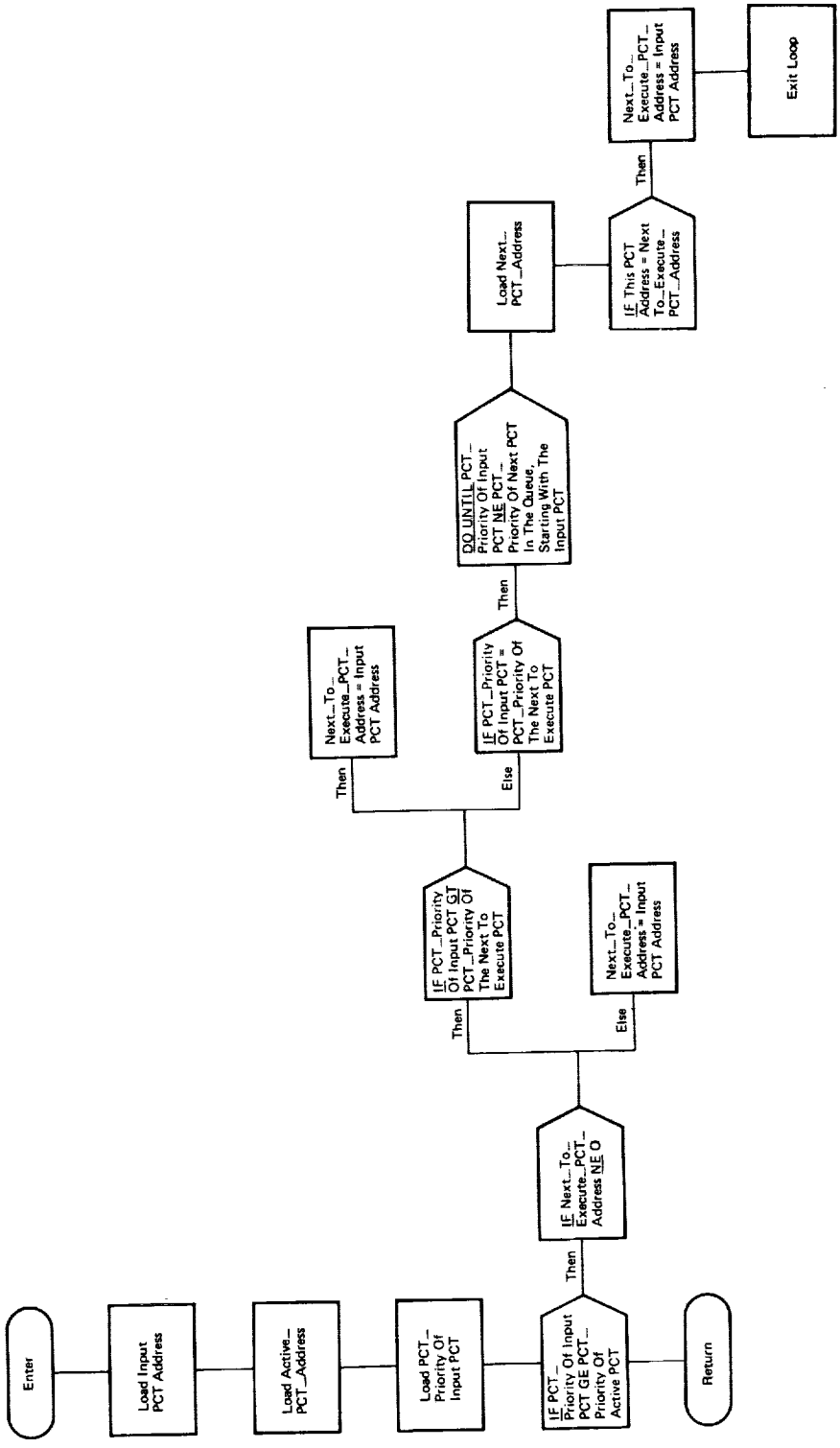


Figure 3.1.1.3-1. Process_Switcher (FPMSWTCH)

3.1.1.4 Process_Dispatcher (FPMDISP) (103)

FPMDISP determines the highest priority ready process and passes control of the CPU to it.

a. Control Interface -

1. CALLed by (100) SVC_Handler (FPMSVC).
2. CALLed by (107) Close_Processor (FPMCLOSE) at entry point FPMDISP2.
3. CALLed by (142) TQE_Expiration_Processor (FPMIHPC2).
4. CALLed by (180) Program_Interrupt_Handler (FPMIHPGM).
5. CALLed by (181) Instruction_Monitor_Interrupt_Handler (FPMIHIM).
6. CALLed by (220) I/O_Completion_Processor (FIOCMPLT).
7. CALLed by (280) Mass_Memory_Manager (FIOMMMGR).
8. CALLed by (300) Software_System_Loader (FCMINSSL).
9. CALLed by (305) Normal_Initialization_Processor (FCMNINIT).
10. CALLed by (320) Overlay_Processor (FCMPOVLY).

b. Input - See Table 3.1.1.4-1

c. Process Description - The Process_Dispatcher saves the register contents of the interrupted process in the Process_Control_Table (PCT) indicated by the Active_PCT_Address. If entry is from the Close_Processor or the Normal_Initialization_Processor, this saving of registers is bypassed. The highest priority ready process is then found beginning the search with either the Next_To_Execute_PCT_Address or the Active_PCT_Address. Before control is passed to the process found in the search, the application registers are reloaded with the settings that were saved in its PCT.

It should be noted that there is always a special priority zero process (Idle_Time_Processor) on the process run queue that is ready to be dispatched. This process consists of a loop in which Idle_Time is incremented by an amount which provides an approximation of the amount of time spent in this process. Idle_Time will be used by the TQE_Expiration_Processor to compute duty cycle. Thus when no other process is ready to be dispatched, the Idle_Time_Processor will be dispatched.

The control flow for this module is presented in Figure 3.1.1.4-1.

d. Output - See Table 3.1.1.4-1.

e. Module References - None.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - The appropriate interrupt old PSW must be saved in the interrupted PCT indicated by the Active_PCT_Address by the associated interrupt handler prior to executing FPMDISP.

j. Detailed Implementation -

1. Control is passed via a PSW load of the Interrupt_PSW_Of_Process.

DATA TABLE 3.1.1.4-1

NAME Process_Dispatcher (FPMDISP)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Active_PCT_Address	Q001.2	I O	103	See App. E	TCVTOLD			
2	Process_Interrupt_Register_0	Q003.6	O I	101, 103 107, 182	103 108, 182	TPCTGPR0			
3	Process_Interrupt_Register_1	Q003.7	O I	103 182,320	103 320	TPCTGPR1			
4	Process_Interrupt_Register_2	Q003.8	O I	103 182,320	103 320	TPCTGPR2			
5	Process_Interrupt_Register_3	Q003.9	O I	103, 182, 320	103 320	TPCTGPR3			
6	Process_Interrupt_Register_4	Q003.10	O I	103, 320	103, 320	TPCTGPR4			
7	Process_Interrupt_Register_5	Q003.11	O I	103 320	103 320	TPCTGPR5			
8	Process_Interrupt_Register_6	Q003.12	O I	103	103	TPCTGPR6			
9	Process_Interrupt_Register_7	Q003.13	O I	103	103	TPCTGPR7			
10	Interrupt_Floating_Point_Register_0	Q003.14	O I	103	103	TPCTFPR0			
11	Interrupt_Floating_Point_Register_1	Q003.15	O I	103	103	TPCTFPR1			
12	Interrupt_Floating_Point_Register_2	Q003.16	O I	103		TPCTFPR2			
13	Interrupt_Floating_Point_Register_3	Q003.17	O I	103	103	TPCTFPR3			
14	Interrupt_Floating_Point_Register_4	Q003.18	O I	103	103	TPCTFPR4			
15	Interrupt_Floating_Point_Register_5	Q003.19	O I	103	103	TPCTFPR5			
16	Interrupt_Floating_Point_Register_6	Q003.20	O I	103	103	TPCTFPR6			
17	Interrupt_Floating_Point_Register_7	Q003.21	O I	103	103	TPCTFPR7			
18	Next_To_Execute_PCT_Address	Q001.3	O I	See App. E		TCVTNEW			
19	PCT_Wait_Indicators	Q003.42	I	See App. E		TPCTWAIT			
20	Next_PCT_Address	Q003.1	I	101,131, 132	See App. E	TPCTNXT			

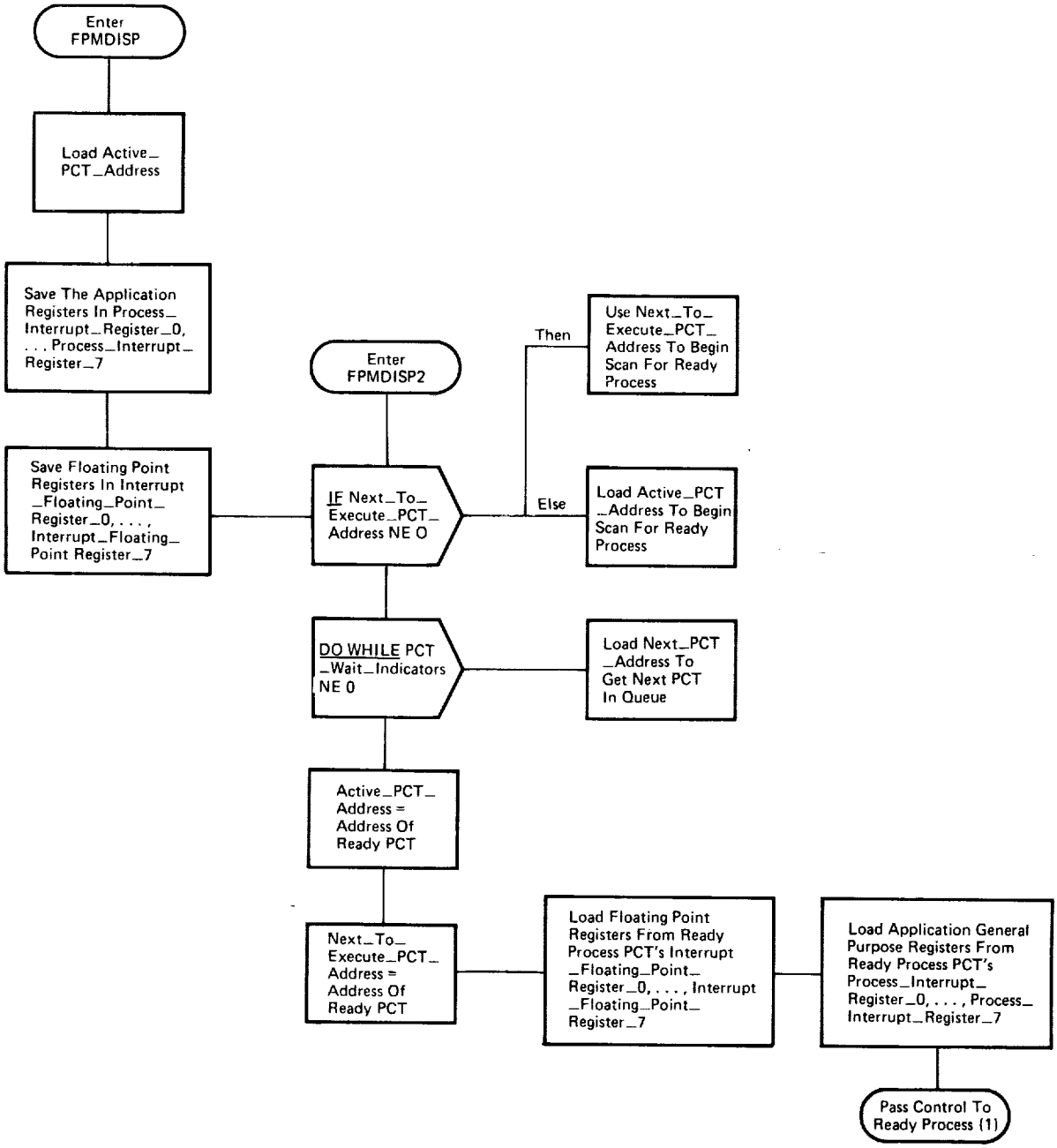


Figure 3.1.1.4-1. Process_Dispatcher (FPMDISP)

3.1.1.5 Wait_Processor (FPMWAIT) (104)

FPMWAIT processes an application request to delay its processing.

a. Control Interface - CALLed by (100) SVC_Handler (FPMSVC)

b. Input - Floating point register 0,1 will contain a delta time or UNTIL time. Also, see Table 3.1.1.5-1.

c. Process Description - The Wait_Processor, upon receiving control from the SVC_Handler, calls the SVC_Synchronization_Processor for synchronization. Then, the Wait_Processor determines the type of wait from the WAIT-SVC_Number from the WAIT_SVC_Parameter_List whose address is in the SVC_Old_PSW. If a time wait is being requested, the Timer_Queue_Generator is called to set up a TQE to expire when the wait time is satisfied, and the PCT_Wait_Indicators are set. If an event wait is requested, the Event_Queue_Generator is called to set up an EQE to be monitored until the event condition is satisfied and the PCT_Wait_Indicators are set. If the PCT_Wait_Indicators indicate the wait condition is satisfied when the Wait_Processor receives control, the WAIT SVC is ignored. The control flow for this module is shown in Figure 3.1.1.5-1.

d. Output - See Table 3.1.1.5-1.

e. Module References -

1. (363) SVC_Synchronization_Processor (FCMSSYNC) is CALLed.
2. (140) Timer_Queue_Generator (FPMTMENQ) is CALLed.
3. (173) Event_Queue_Generator (FPMEVENQ) is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.1.1.5-1

NAME Wait_Processor (FPMWAIT)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	SVC_Old_PSW	&001.67	I		See App. E	TPSASOP			
2	Wait_SVC_Number	S006.1	I		100, 104				
3	Delta_Time_Wait_Indicator	Q003.47	O	104	103, 105	TPCTWAIT			
4	Absolute_Time_Wait_Indicator	Q003.44	O	104	103, 105	TPCTWAIT			
5	PCT_Wait_Indicators	Q003.42	I,O	See App. E	See App. E	TPCTWAIT			
6	Next_To_Execute_PCT_Address	Q001.3	O	See App. E	See App. E	TCVTNEW			
7	TQE_Free_Pool_Address	Q001.11	I	142,143, 166,305	See App. E	TCVTTQEP			
8	Active_PCT_Address	Q001.2	I	103	See App. E	TCUTOLD			
9	TQE_Parent_PCT_Address	Q005.2	O	See App. E	See App. E	TTQEPCT			
10	TQE_Request_Type	Q005.9	O	101,104 107,140	140	TTQEFLGS			
11	TQE_Type_Indicator	Q005.10	O	101, 104 107	140,142, 148,174	TTQEFLGS			
12	Wait_Event_Expression_Address	S006.2	I		104				
13	EQE_Free_Pool_Address	Q001.10	I	173 175	101 104, 174	TCVTEQEP			
14	Operator_Field	Q006.3	O	101 104	173 174	TEQEOPS			
15	Count_Of_Operators	Q006.4	O	101 104	173	TEQEOPS			
16	Event_Expression_Operators	Q006.5	O	101 104		TEQEOPS			
17	Event_Variable_1	Q006.6	O	101,104 173,175	173, 174, 175	TEQEVAR1			
18	Event_Variable_2	Q006.7	O	101,104 173	174	TEQEVAR2			
19	Event_Variable_3	Q006.8	O	101 104	173, 174	TEQEVAR3			
20	Event_Variable_4	Q006.9	O	101 104	173, 174	TEQEVAR4			

DATA TABLE 3.1.1.5-1 (Cont'd)

NAME Wait_Processor (FPMWAIT)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
21	Event_Variable_5	Q006.10	O	101 104	173, 174	TEQEVAR5			
22	Option_Event_Type	Q006.12	O	101 104	173,174	TEQETYPE			
23	Expression_Type	Q006.13	O	104,173	173,174	TEQETYPE			
24	EQE_Satisfied_Flag	Q006.15	O	104		TEQETYPE			
25	Active_EQE_Flag	Q006.16	O	104		TEQETYPE			
26	EQE_Parent_PCT_Address	Q006.2	O	101, 104 175	173, 174 175	TEQEPCT			
27	Event_Wait_Indicator	Q003.46	O	104	103, 105	TPCTWAIT			

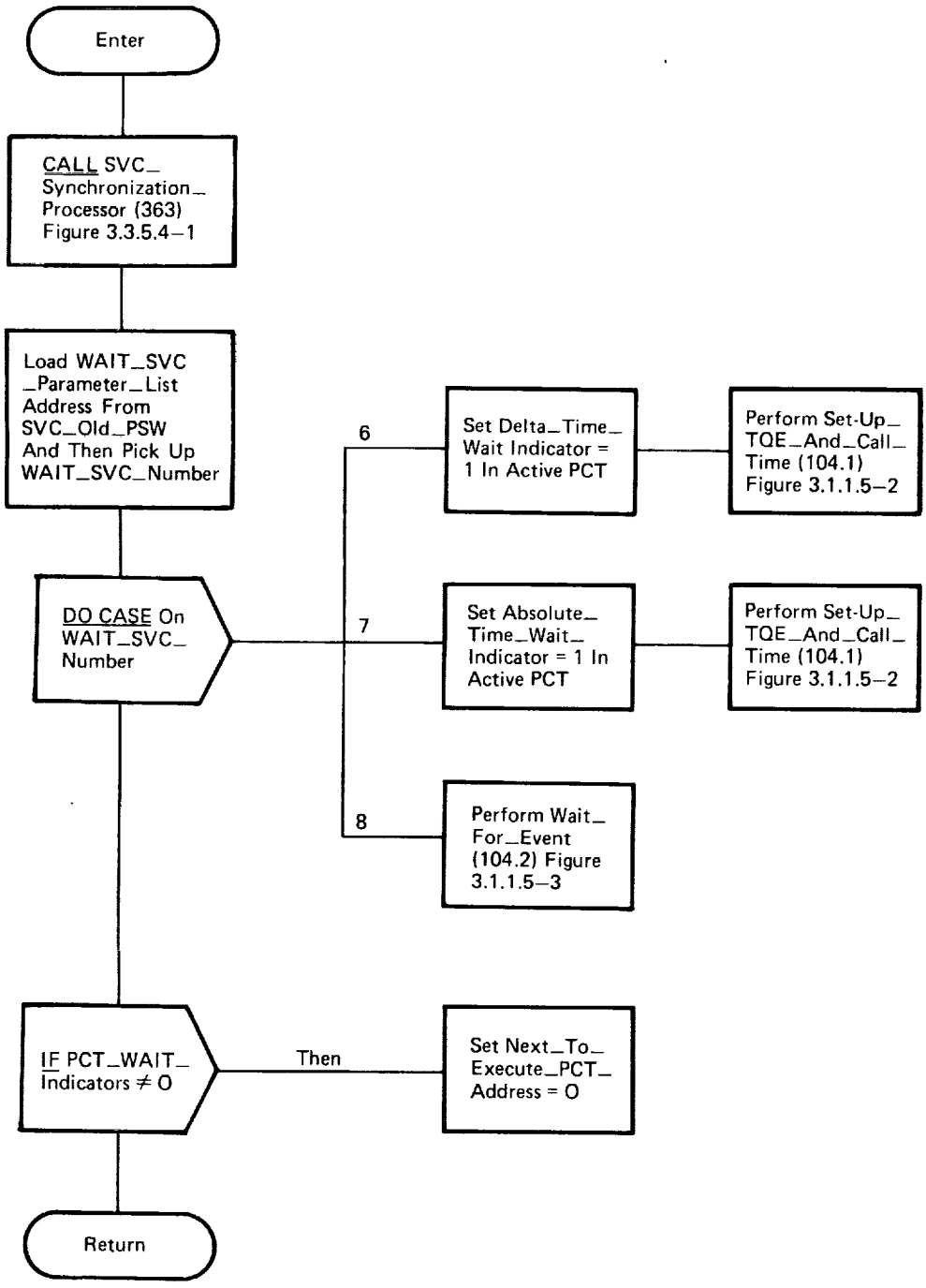


Figure 3.1.1.5-1. Wait_Processor (FPMWAIT)

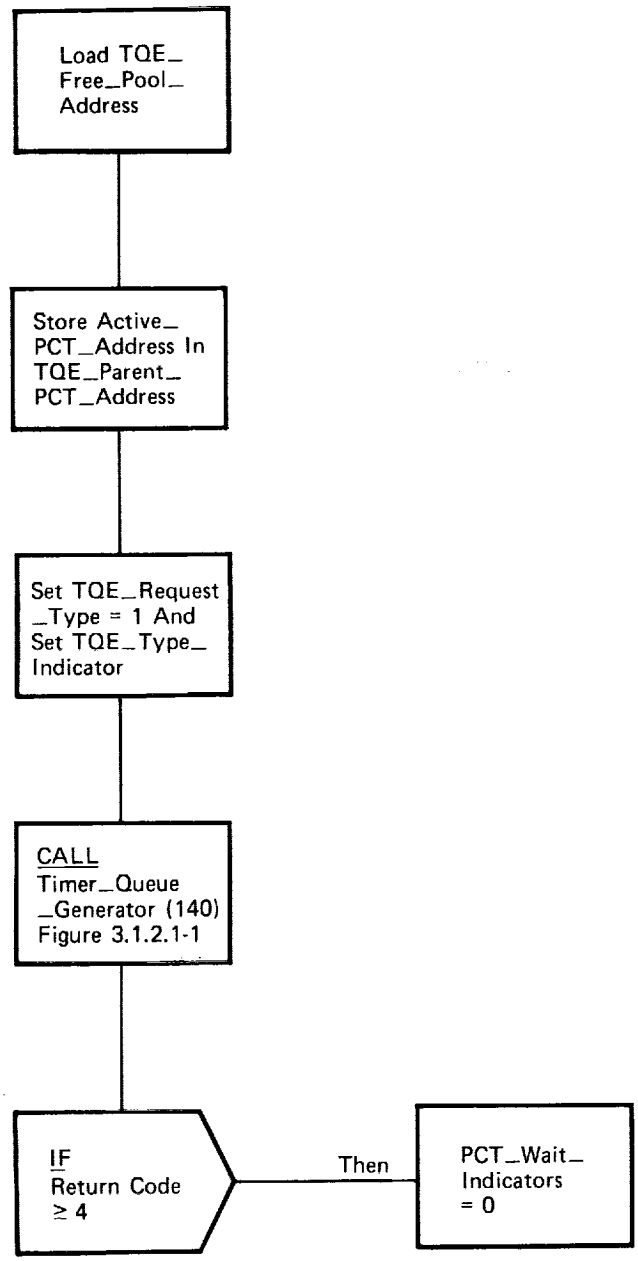


Figure 3.1.1.5-2. Wait_Processor_Set_Up_TQE_And_CALL_Time (104.1)

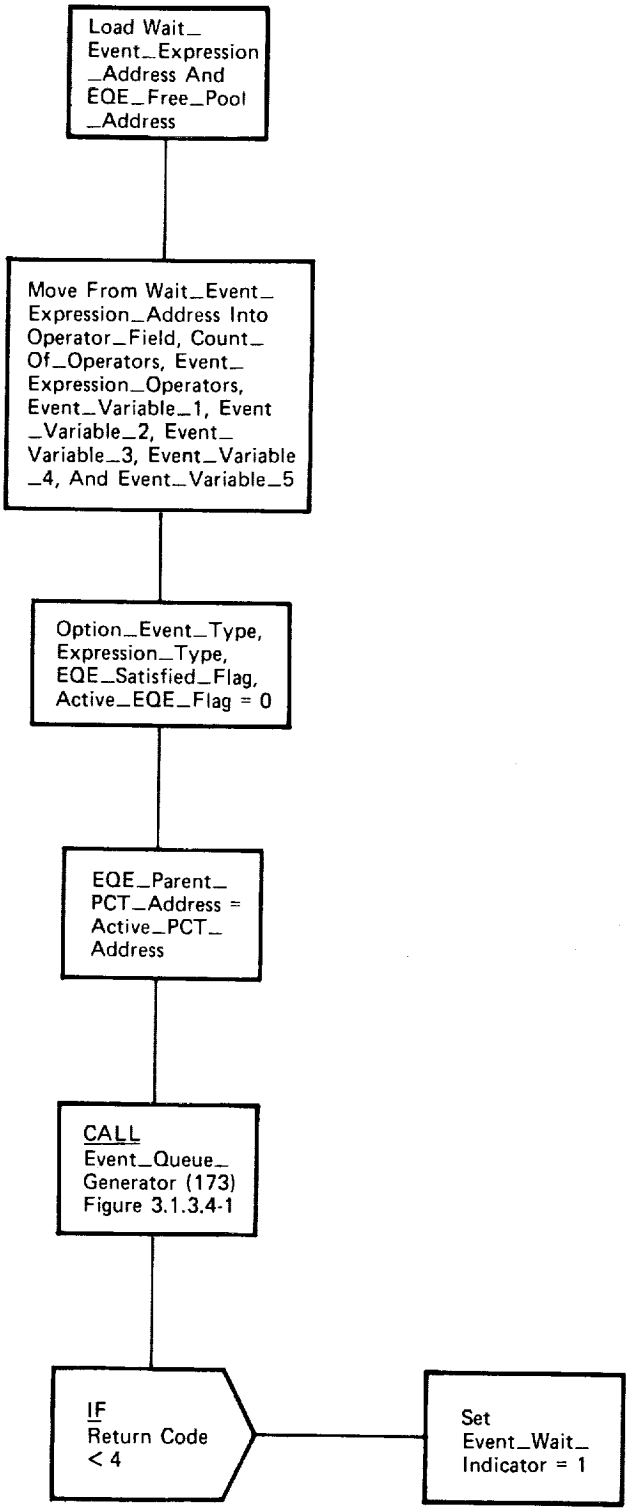


Figure 3.1.1.5-3. Wait_Processor_Wait_For_Event (104.2)

3.1.1.6 Update/Exclusive_Reserve_Processor (FPMRES) (105)

FPMRES processes application requests to use lock groups or EXCLUSIVE procedures/functions.

a. Control Interface -

1. CALLed by (100) SVC_Handler (FPMSVC) at entry FSVC0016 for lock group reservation.

2. CALLed by (100) SVC_Handler (FPMSVC) at entry point FSVC0015 for EXCLUSIVE procedure/function reservation.

b. Input - Register 0 contains the address of the parameter list. Also, see Table 3.1.1.6-1.

c. Process Description - The Update/Exclusive_Reserve_Processor, upon receiving control from the SVC_Handler, attempts to give the requesting (active) process exclusive control of the locks identified in Resource_Identification by placing the Active_PCT_Address in the appropriate Lock_Group_Control_Words entry. If another process has control of any of the locks, that process is forced to complete its use of the lock by temporarily raising the PCT_Priority of the holder of the resource to the PCT_Priority of the requesting process. For lock group processing, the process promoted is the process using the requested lock which is the first one examined and in use. The Interrupt_PSW_Of_Process which needs the lock is reset to cause it to reissue the reserve SVC. If the requested resource identified in Resource_Identification is available, the SVC_Synchronization_Processor is called. The control flow for this module is shown in Figure 3.1.1.6-1.

d. Output - See Table 3.1.1.6-1.

e. Module References - (363) SVC_Synchronization_Processor (FCMSSYNC) is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.1.1.6-1

NAME Update/Exclusive_Reserve_Processor (FPMRES)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Lock-Group-Control-Table-Address	Q001.19	I		105, 106	TCVTLOCK			
2	Resource_Identification	S015.6	I		105, 106				
3	Mask_Of_Used_Lock_Groups	Q004.1	I, O	105, 106	105, 106	FPMLOKCW			
4	Active_PCT_Address	Q001.2	I	103	See App.E	TCVTOLD			
5	Lock_Group_Control_Words	Q004.2	I,O	105	105, 106	FPMLOKCW			
6	SVC_Old_PSW	&001.67	I		See App.E	TPSASOP			
7	Interrupt_PSW_Of_Process	Q003.5	O	See App. E	See App.E	TPCTPSW			
8	PCT_Priority	Q003.2	I,O	101,105 106	See App E	TPCTPRI			
9	Original_Priority	Q003.26	O	101,105 106,320	106 131	TPCTOPRI			
10	PCT_Wait_Indicators	Q003.42	I	See App. E	See App.E	TPCTWAIT			
11	Next_To_Execute_PCT_Address	Q001.3	O	See App E	See App E	TCVTNEW			
12	Next_PCT_Address	Q003.1	I	101, 131,132	See App E	TPCTNXT			

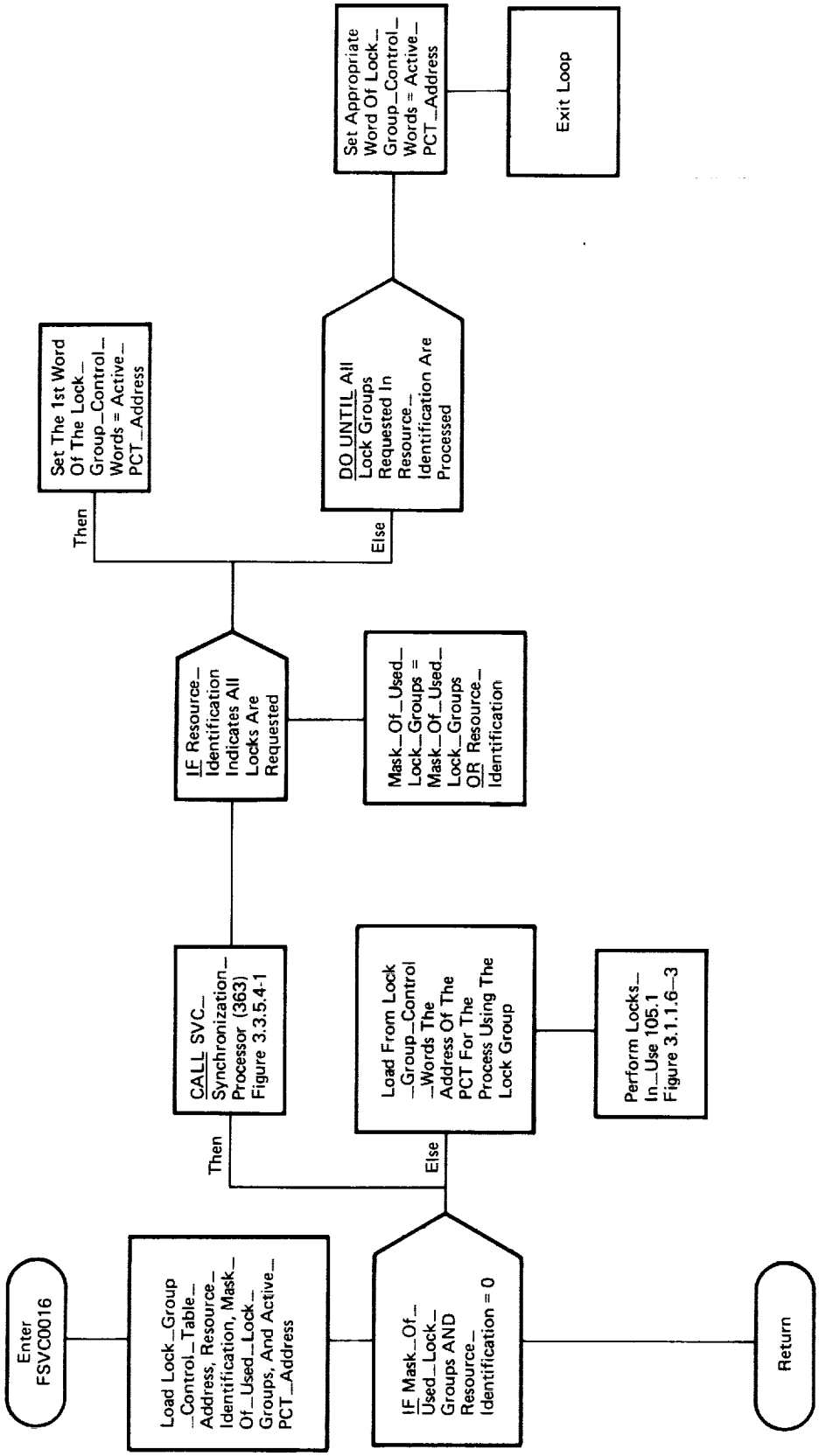


Figure 3.1.1.6-1. Update/Exclusive_Reserve_Processor (FPMRES)

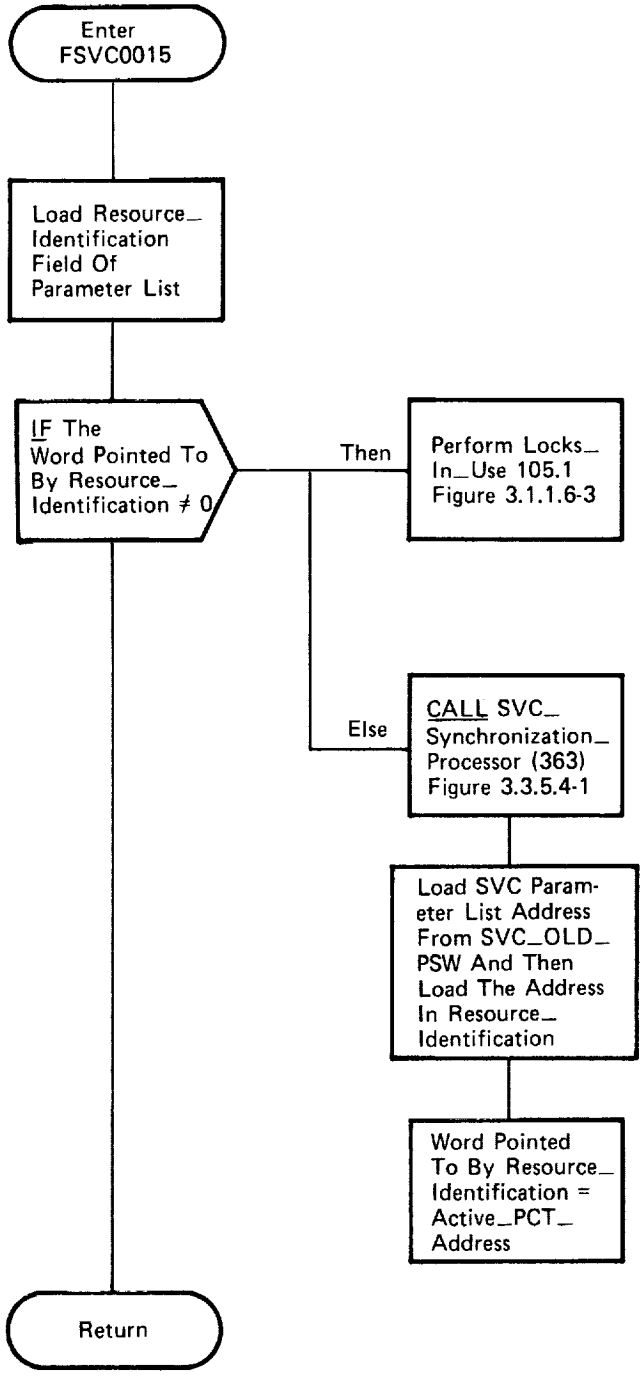


Figure 3.1.1.6-2. Update/Exclusive_Reserve_Processor_Entry_FSVC0015

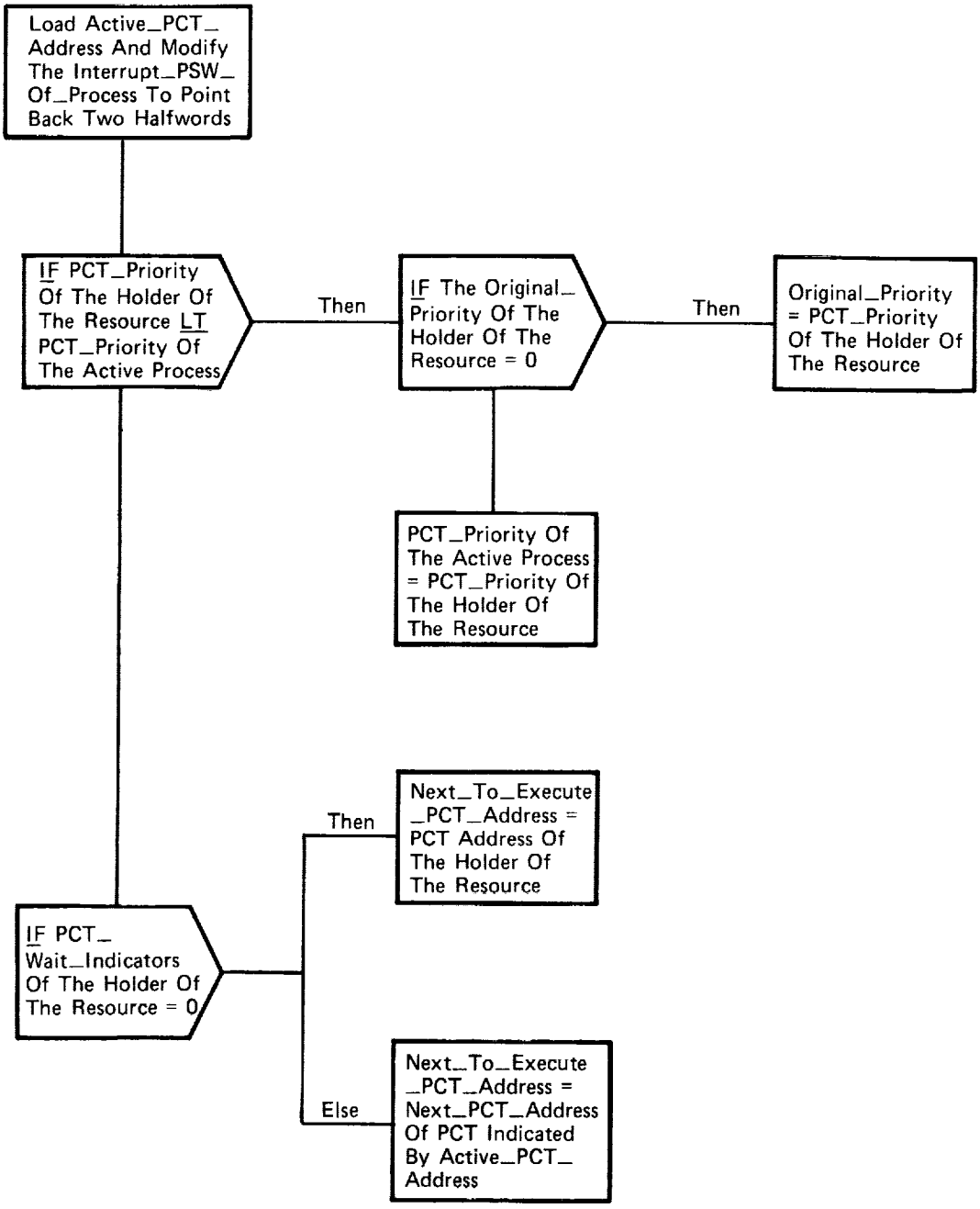


Figure 3.1.1.6-3. Update/Exclusive_Reserve_Processor_Locks_In_Use (105.1)

3.1.1.7 Update/Exclusive_Release_Processor (FPMREL) (106)

FPMREL releases lock groups or EXCLUSIVE procedures/functions for use by other processes.

a. Control Interface -

1. CALLed by (100) SVC_Handler (FPMSVC) at entry point FSVC0017 for normal release of EXCLUSIVE procedure/function.

2. CALLed by (100) SVC_Handler (FPMSVC) at entry point FSCV0018 for normal release of lock groups

3. CALLed by (108) Terminate_Processor (FPMTERM)

4. CALLed by (182) Process_Error_Recovery_Processor (FPMSDERR)

5. CALLed by (184) Forced_Cease_Processor (FPMFCLOS)

b. Input - Register 0 contains the address of the appropriate SVC parameter list. Also, see Table 3.1.1.7-1.

c. Process Description - When called by FCOS programs other than the SVC_Handler to release resources (for example, when the process is being terminated), the Release_SVC_Number is examined to determine whether code or data is to be released and the proper block of code is performed. The code to release data and the code to release code each have entry points which receive control directly from the SVC_Handler when normal release SVC's are issued.

The Exclusive_Release_Processor releases EXCLUSIVE procedures/functions by zeroing the EXCLUSIVE control word field which was pointed to by the address in Resource_Identification. The PCT of the process releasing the resource is restored to its Original_Priority if necessary. If the process had been priority promoted, the Next_To_Execute_PCT_Address is set to the PCT_Run_Queue_Address to force the dispatcher to the top of the queue to find the process wanting the resource.

The UPDATE_BLOCK_Release_Processor releases lock groups by removing the lock group identifiers in the Resource_Identification from the Mask_Of_Used_Lock_Groups. The PCT of the process releasing the resource is restored to its Original_Priority if necessary (if the process had been priority promoted) and the Next_To_Execute_PCT_Address is set to the PCT_Run_Queue_Address (if priority promotion had been performed).

e. Module References - None

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.1.1.7-1

NAME Update/Exclusive_Release_Processor (FPMREL)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Release_SVC_Number	S015.5	I		106				
2	Resource_Identification	S015.6	I		105, 106				
3	Lock_Group_Control_Table_Address	Q001.19	I		105, 106	TCVTLOCK			
4	Mask_Of_Used_Lock_Groups	Q004.1	I,O	105, 106	105, 106	FPMLOKCW			
5	Lock_Group_Control_Words	Q004.2	I	105	105, 106	FPMLOKCW			
6	Original_Priority	Q003.26	I,O	101,105, 106,320	106, 131	TPCTOPRI			
7	PCT_Priority	Q003.2	O	101,105 106	See App E	TPCTPRI			
8	Next_To_Execute_PCT_Address	Q001.3	O	See App E	See App E	TCVTNEW			
9	PCT_Run_Queue_Address	Q001.1	I	131,132 305	See App E	TCVTPCT			

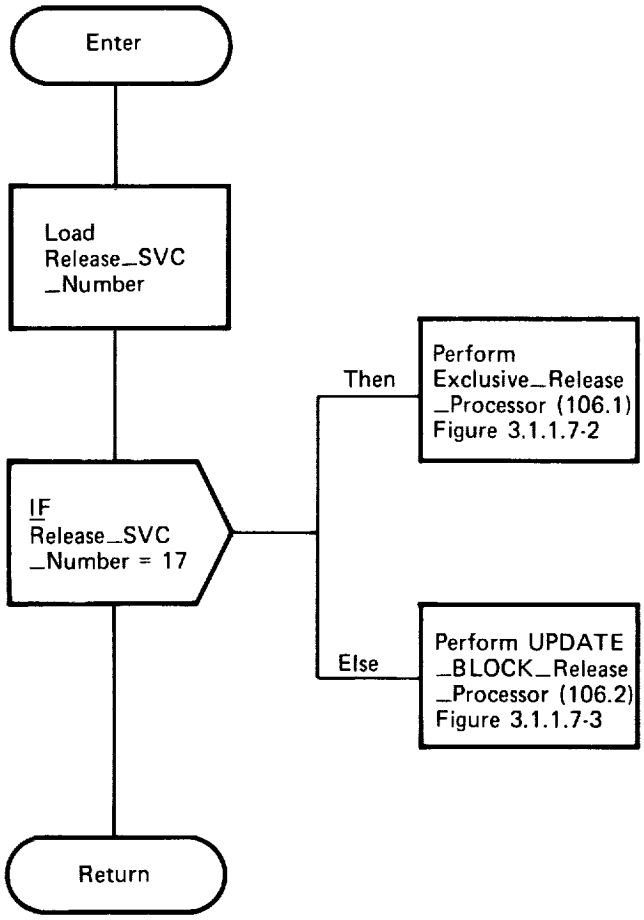


Figure 3.1.1.7-1. Update/Exclusive_Release_Processor (FPMREL)

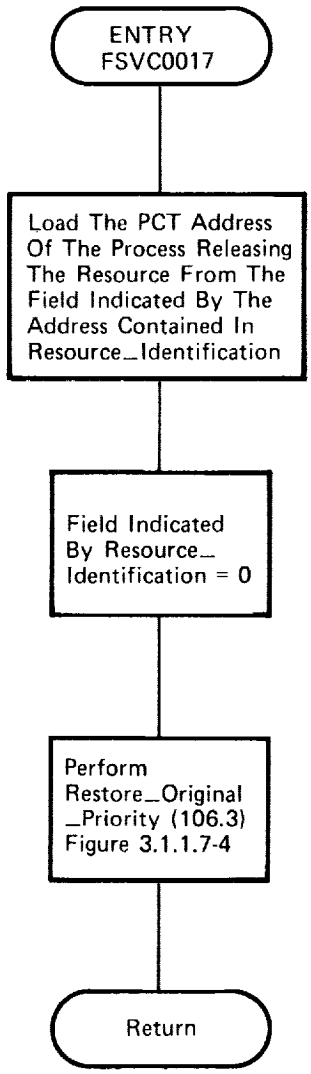


Figure 3.1.1.7-2. Update/Exclusive_Release_Processor Exclusive_Release_Processor (106.1)

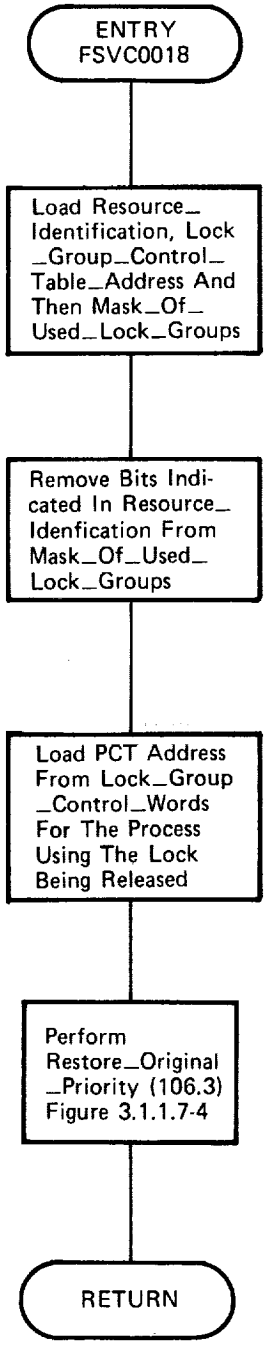


Figure 3.1.1.7-3. Update/Exclusive_Release_Processor Update_Block_Release_Processor (106.2)

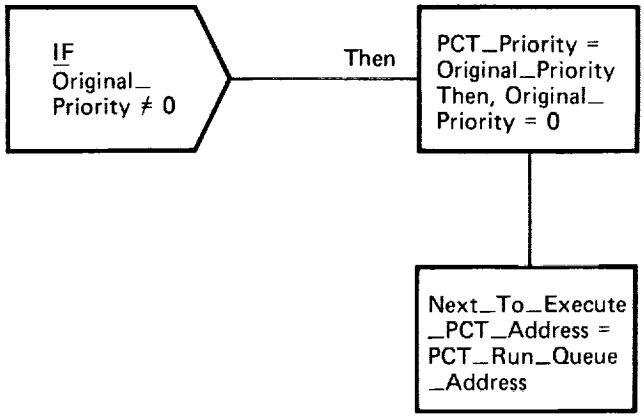


Figure 3.1.1.7-4. Update/Exclusive_Release_Processor Restore_Original_Priority (106.3)

3.1.1.8 Close_Processor (FPMCLOSE) (107)

FPMCLOSE is executed at the end of each program's schedule cycle. It closes the current execution cycle and initiates another cycle if necessary.

a. Control Interface -

1. CALLed by (100) SVC_Handler (FPMSVC)
2. CALLed by (184) Forced_Close_Processor (FPMFCLOS)

b. Input - See Table 3.1.1.8-1.

c. Process Description - The Close_Processor, upon receiving control, CALLs the SVC_Synchronization_Processor for synchronization. If Repeat_Options are zero (non-cyclic process) or if Cancelled_PCT_Indicator is on then Clean_Up processing is done. Clean_Up consists of CALLing the Free_PCT_Routine and the Event_Evaluator if Process_Event is non-zero. If Repeat_Options are non-zero (cyclic process) and if Cancelled_PCT_Indicator is zero then PCT_Reinitialization processing is done for reentry to the process. According to Repeat_Options specified at schedule time, the process is placed in the wait state and if repeat after is specified the Timer_Queue_Element for its next execution is set up. If the process must wait for its next cycle, a process switch is indicated. The Close_Processor exits by CALLing the Process_Dispatcher directly, bypassing register save logic. The control flow for this module is shown in Figure 3.1.1.8-1.

d. Output - See Table 3.1.1.8-1.

e. Module References -

1. (363) SVC_Synchronization_Processor (FCMSSYNC) is CALLed.
2. (103) Process_Dispatcher (FPMDISP) is CALLed.
3. (133) Free_PCT_Routine (FPMFRPCT) is CALLed.
4. (174) Event_Evaluator (FPMEVAL) is CALLed.
5. (140) Timer_Queue_Generator (FPMTMENQ) is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - The Process_Dispatcher is CALLed to dispatch an application process; therefore, there is no return from this call.

DATA TABLE 3.1.1.8-1

NAME Close_Processor (FPMCLOSE)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Active_PCT_Address	Q001.2	I	103	See App E	TCVTOLD			
2	PDE_Address	Q003.4	I	101	See App E	TPCTPDE			
3	Repeat_Options	Q003.36	I	101	101,107 133,174	TPCTFLGS			
4	Cancelled_PCT_Indicator	Q003.30	I	101,109 142	See App. E	TPCTFLGS			
5	Process_Event	Q002.1	I,O	See App.E	See App E	TPDEVENT			
6	Event_State	@007.2	O	See App E	See App. E				
7	Event_Used_Indicator	@007.1	I		See App. E				
8	Interrupt_PSW_Of_Process	Q003.5	O	See App E	See App. E	TPCTPSW			
9	Process_Entry_Point	Q002.3	I	101	101 107	TPDEP			
10	FCOS_Assigned_Stack_Address	Q003.3	I	101	107,133	TPCTSTOR			
11	Process_Interrupt_Register_0	Q003.6	O	101,103 107,182	103,108 182	TPCTGPRO			
12	Stack_Information	Q002.4	I		101,107	TPDESTAK			
13	Last_Error_Group_Code	Q003.27	O	101,107 183	185	TPCTERR			
14	Program_Interrupt_Count	Q003.28	O	101,107	182	TPCTECNT			
15	Scheduled_Wait_Indicator	Q003.48	O	101,107 142	See App. E	TPCTWAIT			
16	TQE_Free_Pool_Address	Q001.11	I	142,143, 166,305	See App E	TCVTTQEP			
17	TQE_Parent_PCT_Address	Q005.2	O	See App E	See App E	TTQEPCT			
18	TQE_Flags	Q005.6	O	104,107 140,142	104,143 148,174	TTQEFLGS			
19	TQE_Request_Type	Q005.9	O	101,104 107,140	140	TTQEFLGS			
20	TQE_Type_Indicator	Q005.10	O	101,104 107	140,142, 148,174	TTQEFLGS			

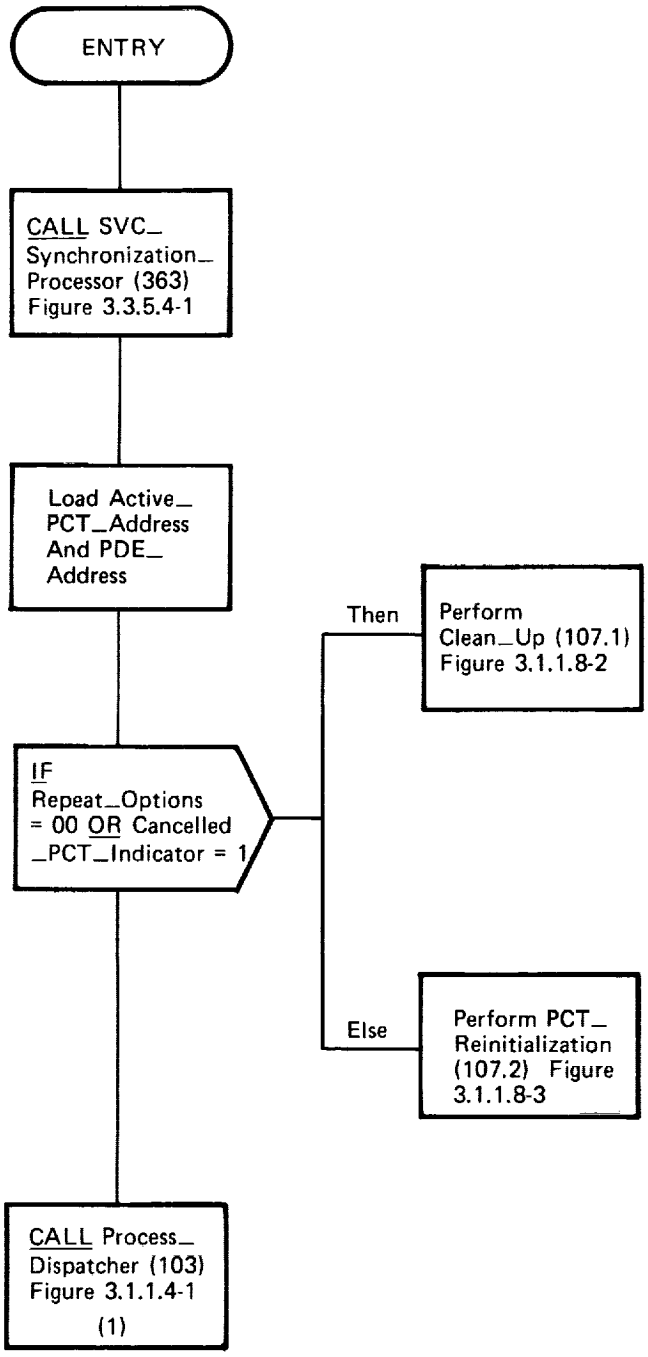


Figure 3.1.1.8-1. Close_Processor (FPMCLOSE)

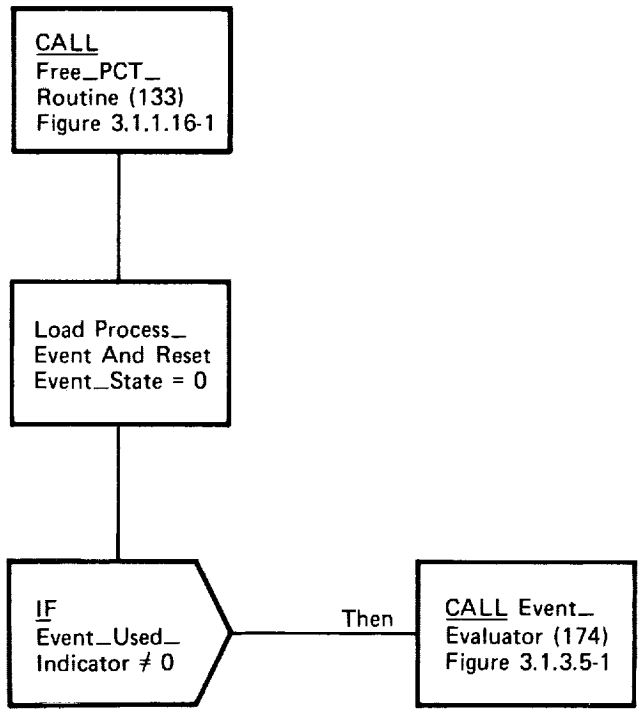


Figure 3.1.1.8-2. Close_Processor Clean_Up (107.1)

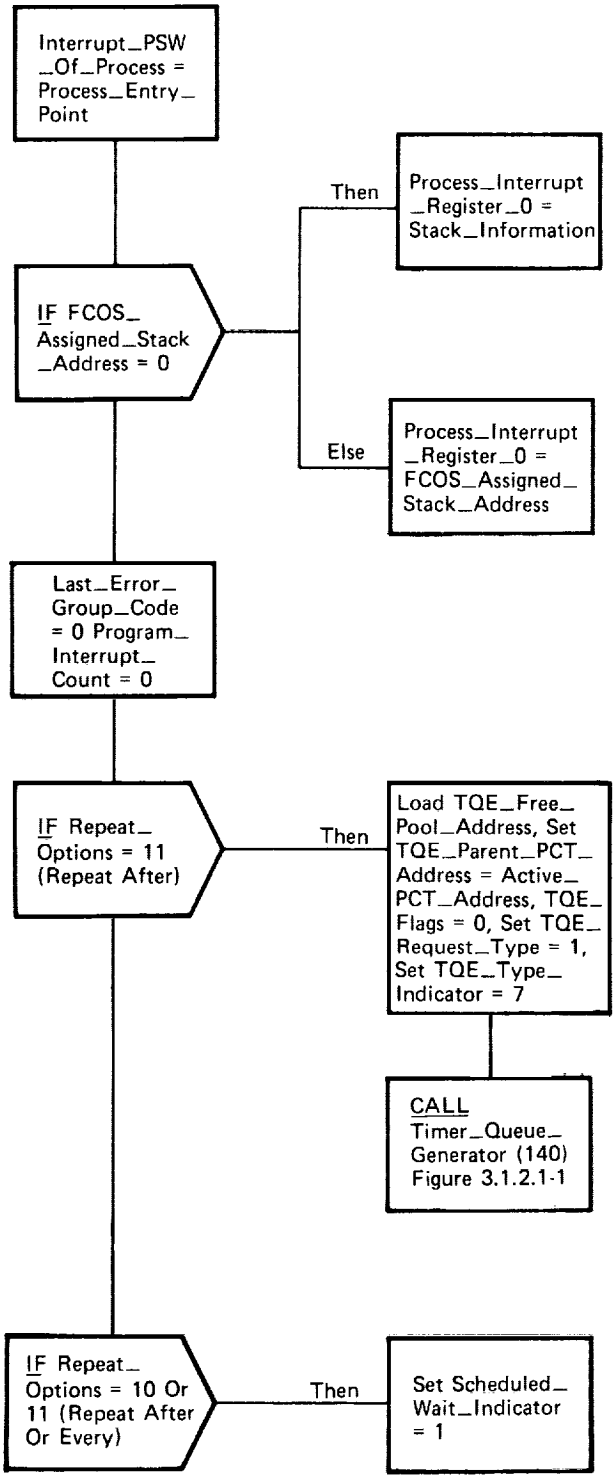


Figure 3.1.1.8-3. Close_Processor PCT_Reinitialization (107.2)

3.1.1.9 Terminate_Processor (FPMTERM) (108)

FPMTERM immediately halts execution of the specified process(es).

a. Control Interface -

1. CALLed by (305) Normal_Initialization_Processor (FCMNINIT) at the initial entry point.

2. CALLed by (100) SVC_Handler (FPMSVC) at entry point FSVC0003 for terminate with a list of processes.

3. CALLed by (100) SVC_Handler (FPMSVC) at entry point FSVC0002 for terminate without a list of processes.

b. Input - Bits 0-15 of register 0 contain either the address of the SVC parameter list (for a CALL by the SVC_Handler) or the address of the PCT to be terminated (for a CALL by Normal_Initialization).

c. Process Description - The Terminate_ Processor, upon receiving control from the SVC_Handler (with one or more processes to terminate), CALL's the SVC_Synchronization_Processor for synchronization. After that (and upon receiving control from the Normal_Initialization_Processor), the process(es) is/are terminated by setting the Terminated_PCT_Indicator and CALLing the Free_PCT_Routine. If an UPDATE block or EXCLUSIVE procedure is involved (Update/Exclusive_Indicator is on) then Update/Exclusive_Release_Processor is CALLed to release the resources. If a process event is indicated (Event_Used_Indicator is on), then the Event_Evaluator is CALLed. The control flow for this module is shown in Figure 3.1.1.9-4.

d. Output - See Table 3.1.1.9-1.

e. Module References -

1. (363) SVC_Synchronization_Processor (FCMSSYNC) is CALLed.
2. (106) Update/Exclusive_Release_Processor (FPMREL) is CALLed.
3. (133) Free_PCT_Routine (FPMFRPCT) is CALLed.
4. (174) Event_Evaluator (FPMEVAL) is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.1.1.9-1

NAME Terminate_Processor (FPMTERM)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	SVC_Old_PSW	&001.67	I		See App E	TPSASOP			
2	Terminate_Process_Count	S002.1	I		108				
3	Terminate_Process_List	S002.3	I		108				
4	Assigned_PCT_Address	Q002.2	I	101 133	101,108 109,110	TPDEPCT			
5	Active_PCT_Address	Q001.2	I	103	See App E	TCVTOLD			
6	Process_Interrupt_Register_0	Q003.6	I	101,103 107,182	103,105 182	TPCTGPRO			
7	Previous_Stack_Frame_Address	@002.3	I	101	108,182, 184	TTSAPREV			
8	Local_Data_Area_Address	@002.9	I	101	108,182, 184,290	TTSALCDA			
9	Update/Exclusive_Indicator	@001.3	I		108,182, 184	TLDAEINF			
10	Release_SVC_Parameter_List	@001.7	I		108,182, 184	TLDARELP			
11	Terminated_PCT_Indicator	Q003.33	O	108 142	225	TPCTFLGS			
12	PDE_Address	Q003.4	I	101	See App E	TPCTPDE			
13	Process_Event	Q002.1	I,O	See App. E	See App E	TPDEVENT			
14	Event_State	@007.2	O	See App E	See App E				
15	Event_Used_Indicator	@007.1	I		See App.E				

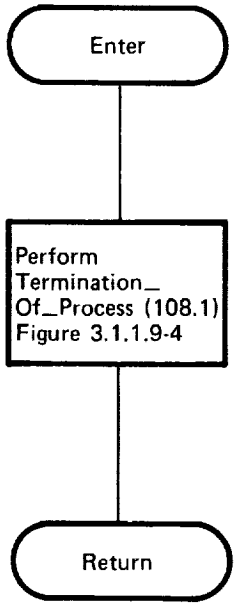


Figure 3.1.1.9-1. Terminate_Processor (FPMTERM)

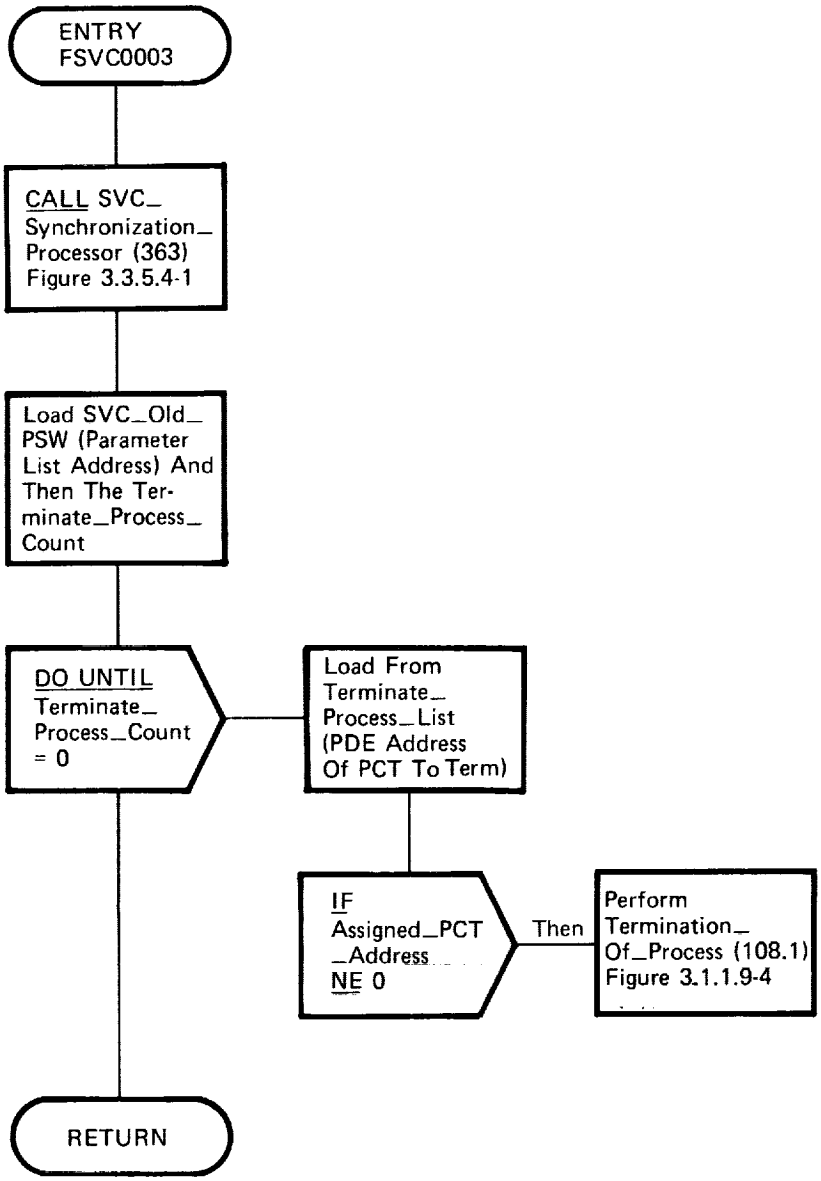


Figure 3.1.1.9-2. Terminate_Processor Terminate_Multiple_Processes

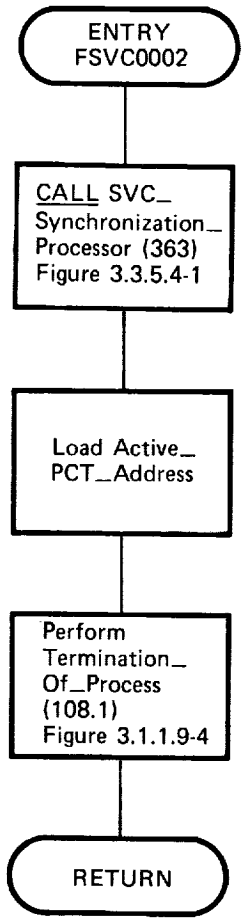


Figure 3.1.1.9-3. Terminate_Processor
Terminate_Requested_Process

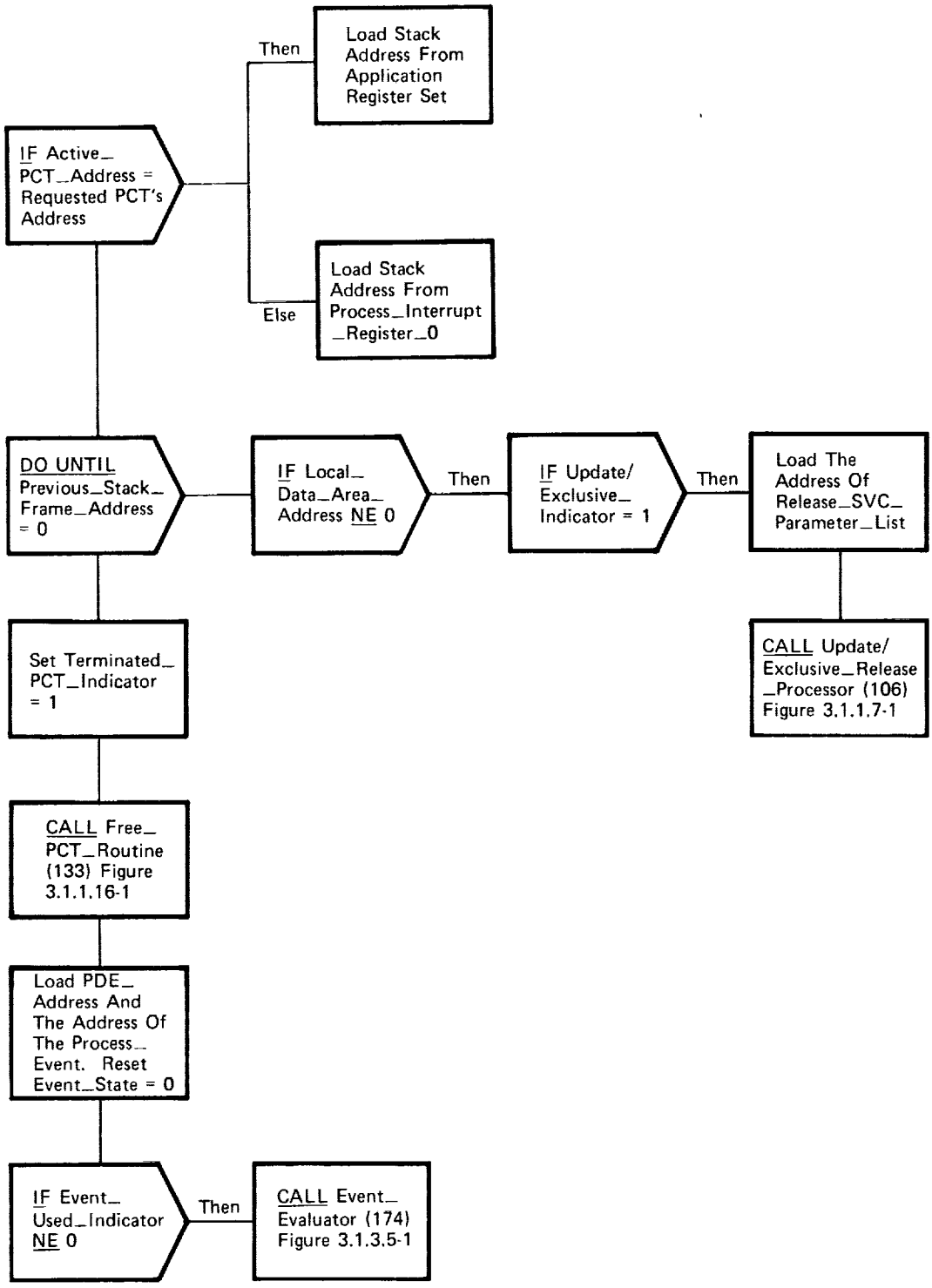


Figure 3.1.1.9-4. Terminate_Processor Termination_Of_Process (108.1)

3.1.1.10 Cancel_Processor (FPMCANCL) (109)

FPMCANCL services an application request to remove specified processes from the FCOS process run queue.

a. Control Interface - Called by the (100) SVC Handler when a supervisor call 4 or 5 is executed.

b. Input - See Table 3.1.1.10-1

c. Process Description - The Cancel_Processor services all supervisor call CANCEL requests. Upon receiving control it calls the SVC_Synchronization_Routine. If the Cancel_SVC_Number was a 4, it is a request to cancel the current process which is pointed to by the Active_PCT_Address. Therefore, the Cancel_Processor sets the Cancelled_PCT_Indicator in its PCT.

If the Cancel_SVC_Number was a 5, a list of PDE addresses (Cancel_Process_List) was provided, and it indicates which processes to cancel. The Cancel_Processor steps through the list of PDE's and

1. Sets the Cancelled_PCT_Indicator in each PCT which has the Scheduled_Wait_Indicator not equal to 0.

2. If the Scheduled_Wait-Indicator is 0, the Cancel_Processor calls the Free_PCT_Routine to release the PCT. It then resets the Process_Event and calls the Event_Evaluator if a process is waiting for the event to change state.

The Cancel_Processor then returns to the caller. The control flow for this module is presented in Figure 3.1.1.10-1.

d. Output - See Table 3.1.1.10-1

e. Module References -

1. (363) SVC_Synchronization_Processor (FCMSSYNC) is called.
2. (133) Free_PCT_Routine (FPMFRPCT) is called.
3. (174) Event_Evaluator (FPMEVAL) is called.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.1.1.10-1

NAME Cancel Processor (FPMCANCL)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Cancel_SVC_Number	S004.2	I		100 109				
2	Cancelled_PCT_Indicator	Q003.30	O	101, 109 142	See App.E				
3	Active_PCT_Address	Q001.2	I	103	See App. E	TCVTOLD			
4	Cancel_Process_Count	S004.1	I		109				
5	Cancel_Process_List	S004.3	I		109				
6	Assigned_PCT_Address	Q002.2	I	101 133	101, 108 109, 110	TPDEPCT			
7	Scheduled_Wait_Indicator	Q003.48	I	101, 107 142	See App.E				
8	Process_Event	Q002.1	I O	See App.E	See App. E	TPDEVENT			
9	Event_State	@007.2	O	See App. E	See App. E				
10	Event_Used_Indicator	@007.1	I		See App. E				

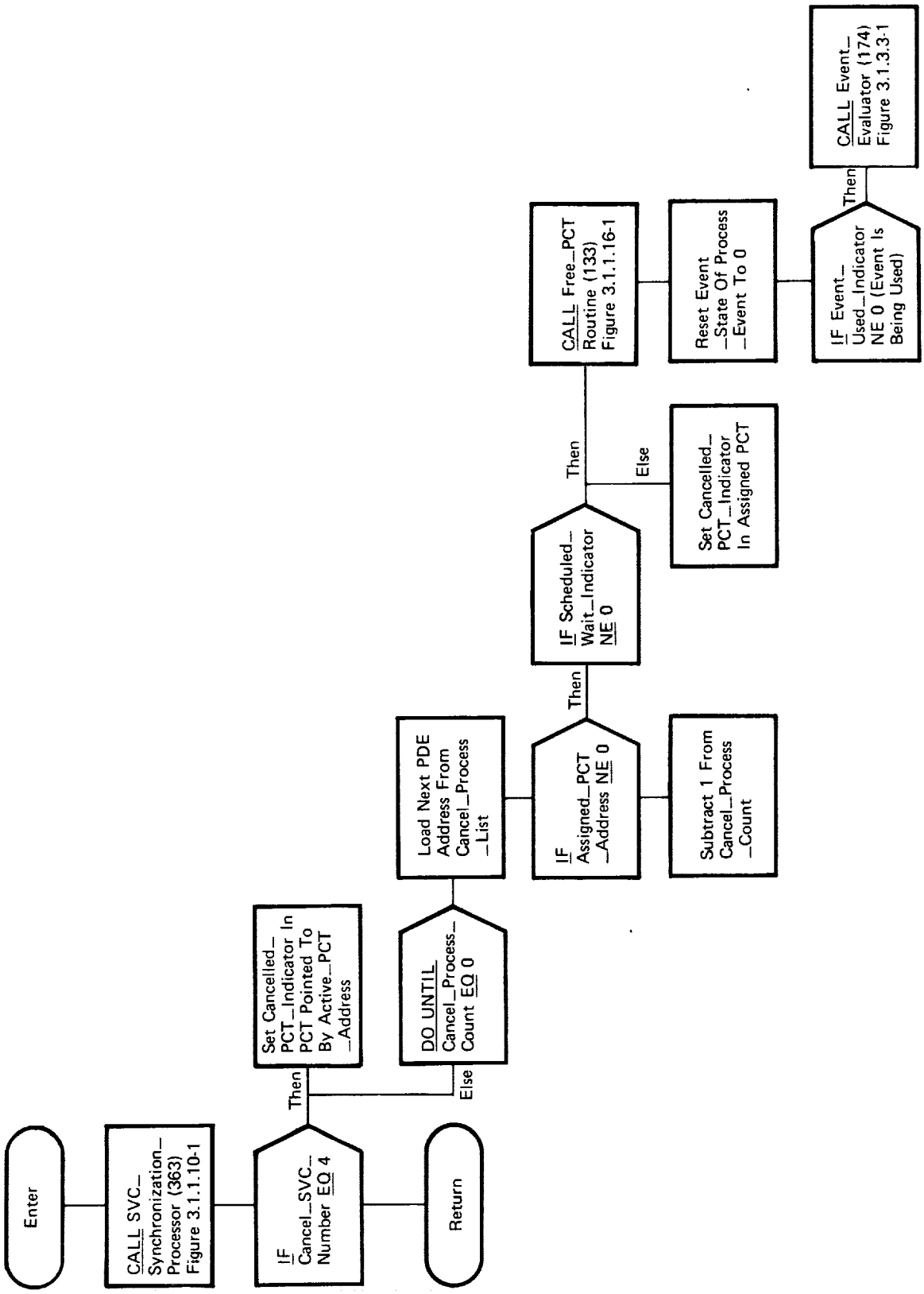


Figure 3.1.1.10-1. Cancel_Processor (FPMCANCL)

3.1.1.11 OPS_Cancel_Processor (FPMOPSCN) (110)

FPMOPSCN provides the applications with the capability to specify that all but a list of processes for a certain major function should be cancelled.

a. Control Interface -

CALLed by (100) SVC_Handler (FPMSVC)

b. Input -

Register 0 contains the address of the OPS_Cancel_Parameter_List.

See table 3.1.1.11-1

c. Process Description -

The active Process_Control_Table (PCT) queue is scanned and the Process_Directory_Entry (PDE) for each PCT is located. As each PDE is located, it is examined to determine if it is associated with the OPS_Major_Function_Id for which processes are to be cancelled. If it is the PDE_Address is compared with each PDE address in the list of processes not to be cancelled. (Pointed to by OPS_Process_List_Address) If the PDE is not in the list, the process associated with the PDE is cancelled, unless the process is currently within an execution cycle. If the process to be cancelled is within an execution cycle, the PCT is updated to indicate that the process is to be cancelled at close time. If the process is not in an execution cycle, the Free_PCT_Routine is called, and the Process_Event is set to false and the EVENT_ Evaluator is called if necessary. The control flow for this program is shown in Figure 3.1.1.11-1.

d. Outputs -

See Table 3.1.1.11-1.

e. Module References -

1. (133) Free_PCT_Routine (FPMFRPCT) is CALLed.

2. (174) Event_Evaluator (FPMEVAL) is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.1.1.11-1

NAME OPS_Cancel_Processor (FPMOPSCN)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	PCT_Run_Queue_Address	Q001.1	I	131, 132	See App. E	TCVTPCT			
				305					
2	Next_PCT_Address	Q003.1	I	101, 131	See App. E	TPCTNXT			
				132					
3	PDE_Address	Q003.4	I	101	See App. E	TPCTPDE			
4	Process_Major_Function_ID	Q002.8	I		110	TPDEFLGS			
5	OPS_Major_Function_ID	S033.3	I		110	TOPSMFID			
6	OPS_Cancel_Count	S033.1	I		110	TOPSCSVC			
7	OPS_Process_List_Address	S033.4	I		110	TOPSPGMA			
8	Scheduled_Wait_Indicator	Q003.48	I	101, 107	See App. E	TPCTWAIT			
				142					
9	Canceled_PCT_Indicator	Q003.30	I O	101, 109	See App. E	TPCTFLGS			
				142					
10	Process_Event	Q002.1	I O	See App E	See App. E	TPDEVENT			
11	Event_State	@007.2	I O	See App. E	See App. E				

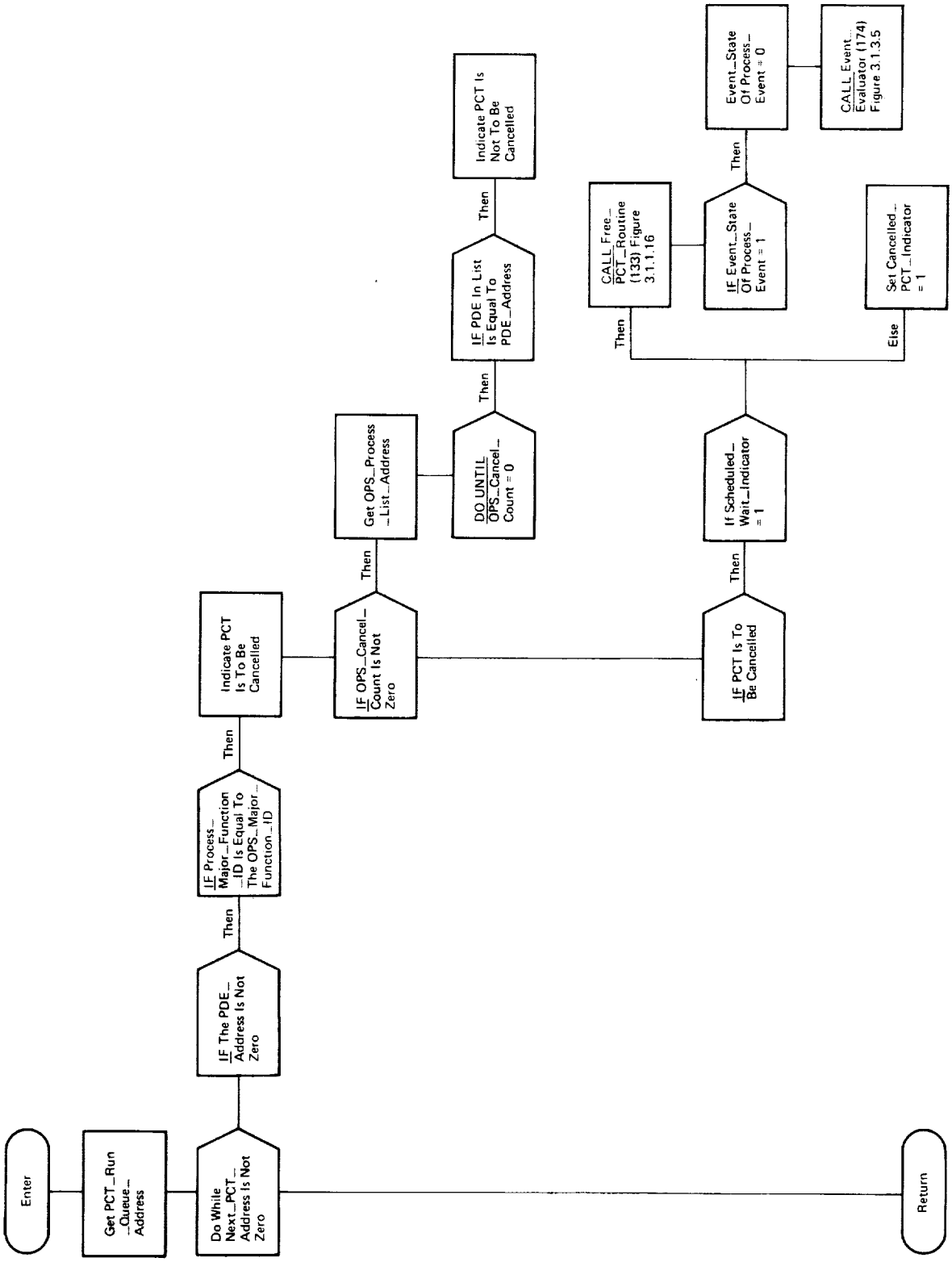


Figure 3.1.1.11-1. OPS_Cancel_Processor (FPMOPSCN)

3.1.1.12 Select_I/O_Processor (FPMSIO) (120)

The Select_I/O_Processor is called to associate I/O with a cyclic process. The association takes place when the process is scheduled.

a. Control Interface - CALLed by (100) SVC_Handler (FPMSVC)

b. Input - See Table 3.1.1.12-1.

c. Process Description - On entry, the SVC_Synchronization_Processor is called to perform redundant set GPC synchronization.

The Time_Initiated_I/O_Table is then examined to determine if an entry already exists for the target PDE. If so, the existing entry is overlayed by the new request and control is returned to the SVC_Handler.

If no entry exists, the new request is stored in the first unused slot in the Time_Initiated_I/O_Table or, if the table is full, in the first slot of the table before returning to the SVC_Handler.

Once a Time_Initiated_I/O_Table entry has been created for a process (PDE), a SCHEDULE of the process will cause the I/O operation to be initiated each time the process begins an execution cycle. See the Process_Scheduler description for further information. The control flow for this module is presented in Figure 3.1.1.12-1.

d. Outputs - See Table 3.1.1.12-1.

e. Module References - (363) SVC_Synchronization_Processor (FCMSSYNC) is CALLed.

f. Module Attributes - Program

g. Template References - None

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.1.1.12-1

NAME Select_I/O_Processor (FPMSIO)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Time_Initiated_I/O_Table	Y025	I	101	101 120	FPMTIOT			
2	Time_Initiated_I/O_Parent_PDE_Address	Y025.1	I,0	101, 120	101, 120	TTIOTPDE			
3	Select_I/O_PDE_Address	S041.2	I		120	SVCPDE			
4	I/O_Parameter_List_Address	S041.3	I		120	SVCIO			
5	Process_I/O_Parameter_List_Address	Y025.2	0	120	101	TTIOTIO			

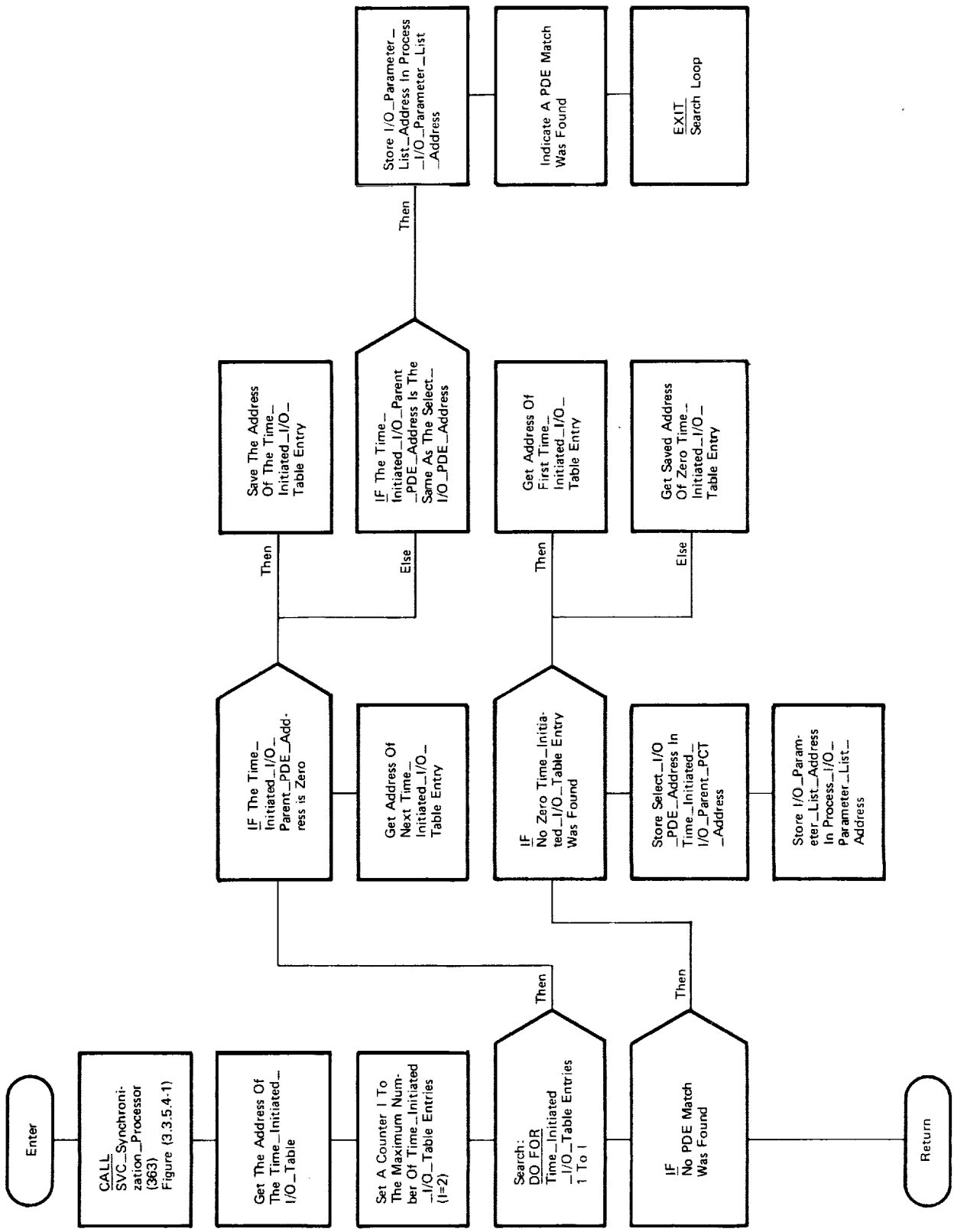


Figure 3.1.1.12-1. Select_I/O_Processor (FPMSIO)

3.1.1.13 Hybrid_Dispatching_Routine (#CFPMHYD)(130)

To Be Provided

3.1.1.14 Chain_PCT_Routine (FPMCHPCT) (131)

FPMCHPCT determines where in the PCT run queue to place a newly created PCT and chains it in the proper place.

a. Control Interface -

1. CALLed by (101) Process_Scheduler (FPMSCHED)
2. CALLed by (280) Mass_Memory_Manager (FIOMMMGR)
3. CALLed by (320) Overlay_Processor (FCMPOVLY)

b. Input - Register 0 contains the address of the new PCT. See Table 3.1.1.14-1.

c. Process Description - The Chain_PCT_Routine loads the PCT_Run_Queue_Address to begin its scan of PCT's to determine where to chain the newly created PCT. The PCT_Priority of the new PCT is compared to the PCT_Priority of the PCT from the run queue. If the new PCT's PCT_Priority is greater, the new PCT is chained into the run queue immediately preceding the PCT currently being processed. If the new PCT's PCT_Priority is not greater, the Next_PCT_Address is loaded to continue the scan. This process is repeated until a PCT is found which has a priority less than the new PCT. When the new PCT has been chained into the PCT run queue, it is removed from the free pool by updating the PCT_Free_Pool_Address. The control flow for this module is presented in Figure 3.1.1.14-1.

d. Output - Registers 0, 1, 3, 4, and 5 are not preserved across a CALL to this program. See Table 3.1.1.14-1.

e. Module Reference - None

f. Module Attributes - Program

g. Template Reference - N/A

h. Error Handling - None

i. Constraints and Assumptions - Attempts to chain a PCT with a priority of 0 will have unpredictable results since the scan of the PCT run queue will go past the end of the queue.

j. Detailed Implementation - N/A

DATA TABLE 3.1.1.14-1

NAME Chain_PCT_Routine (FPMCHPCT)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	PCT_Priority	Q003.2	I	101, 105	See App. E	TPCTPRI			
				106					
2	PCT_Run_Queue_Address	Q001.1	I	131, 132	See App. E	TCVTPCT			
			0	305					
3	Next_PCT_Address	Q003.1	I	101, 131	See App. E	TPCTNXT			
			0	132					
4	Original_Priority	Q003.26	I	101, 105	106, 131	TPCTOPRI			
				106, 320					
5	PCT_Free_Pool_Address	Q001.9	0	131, 132	101, 280	TCVTPCTP			
				305	320				

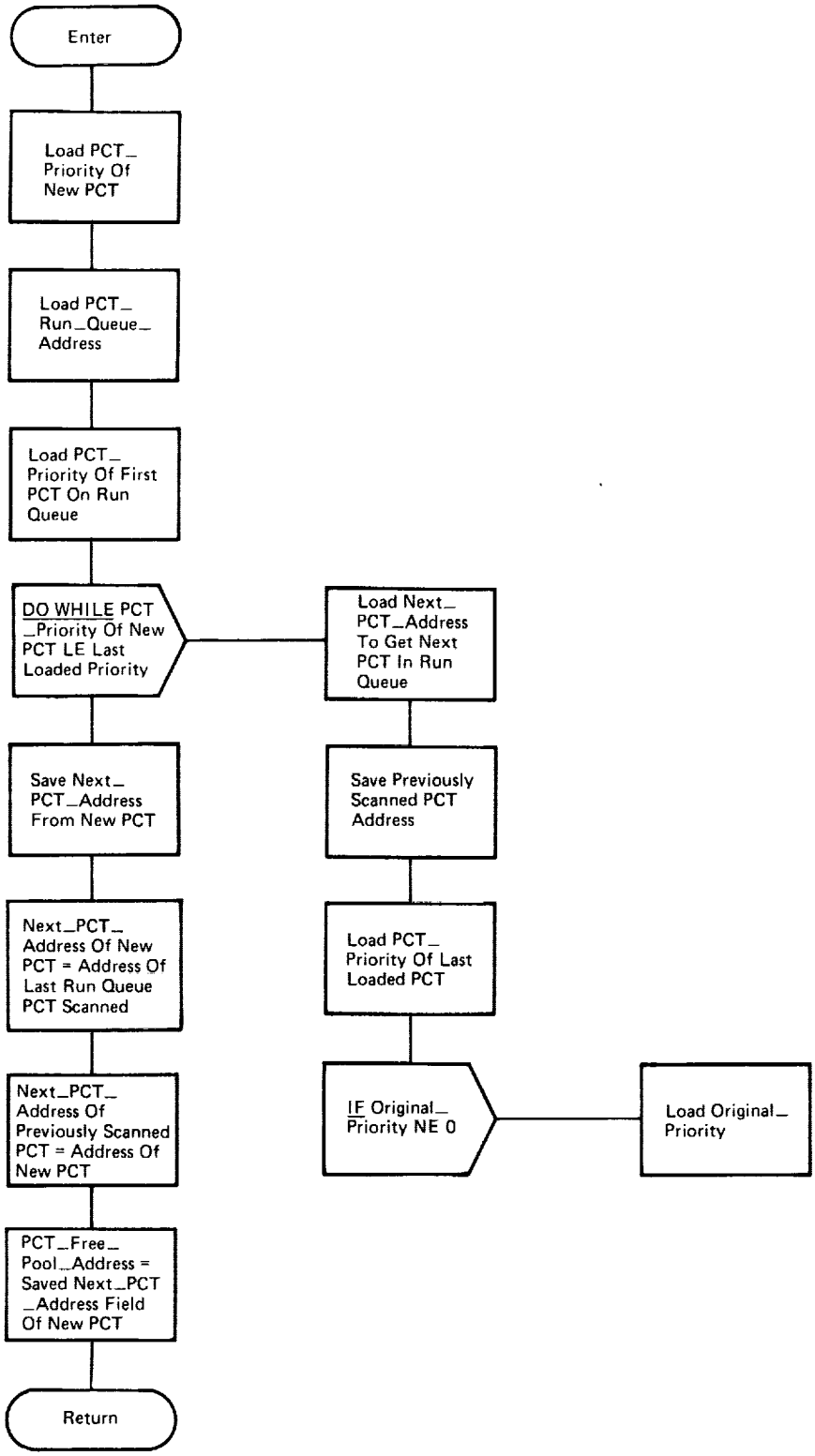


Figure 3.1.1.14-1. Chain_PCT_Routine (FPMCHPCT)

3.1.1.15 Release_PCT_Routine (FPMRLPCT) (132)

FPMRLPCT removes a PCT from the PCT run queue and adds it to the PCT free pool.

a. Control Interface -

1. CALLed by (133) Free_PCT_Routine (FPMFRPCT)
2. CALLed by (280) Mass_Memory_Manager (FIOMMMGR)
3. CALLed by (320) Overlay_Processor (FCMPOVLY)

b. Input - Register 5 contains the address of the PCT to be removed from the queue. See Table 3.1.1.15-1.

c. Process Description - The Release_PCT_Routine scans the PCT run queue beginning with the PCT_Run-Queue_Address to find the location of the input PCT in the queue and unchains it from the run queue. The input PCT address is then stored as the PCT_Free_Pool_Address and chained to the other unused PCT's. The control flow for this module is presented in Figure 3.1.1.15-1.

d. Output - Registers 0, 1, 4, and 5 are not preserved across CALLs to this program. See Table 3.1.1.15-1.

e. Module References - None

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - N/A

DATA TABLE 3.1.1.15-1

NAME Release_PCT_Routine (FPMRLPCT)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	PCT_Run_Queue_Address	Q001.1	I	131, 132	See App. E	TCVTPCT			
			O	305					
2	Next_PCT_Address	Q003.1	I	131, 132	See App. E	TPCTNXT			
			O	101					
3	PCT_Free_Pool_Address	Q001.9	O	131, 132	101, 280	TCVTPCTP			
				305	320				

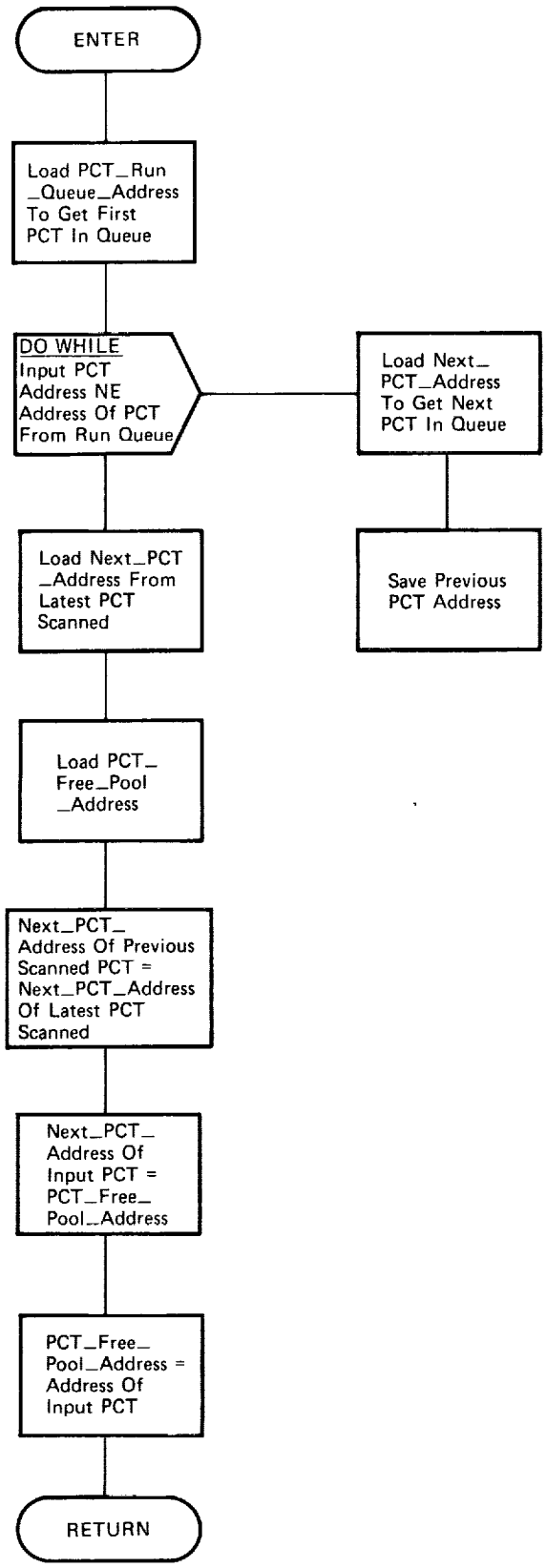


Figure 3.1.1.15-1. Release_PCT_Routine (FPMRLPCT)

3.1.1.16 Free_PCT_Routine (FPMFRPCT) (133)

FPMFRPCT terminates use of a PCT by coordinating the removal of associated queue elements from their respective queues.

a. Control Interface -

1. CALLed by (107) Close_Processor (FPMCLOSE)
2. CALLed by (108) Terminate_Processor (FPMTERM)
3. CALLed by (109) Cancel_Processor (FPMCANCL)
4. CALLed by (110) OPS_Cancel_Processor (FPMOPSCN)
5. CALLed by (142) TQE_Expiration_Processor (FPMIHPC2)
6. CALLed by (174) Event_Evaluator (FPMEVAL)

b. Input - Register 0 contains the address of the PCT to be released. See table 3.1.1.16-1.

c. Process Description - If the PCT_Flags field of the input PCT indicate that entry to the Free_PCT_Routine is not resulting from a close of a non-cyclic process, the TQE_Dequeue_Processor and the EQE_Dequeue_Processor are invoked to remove any outstanding TQE's/EQE's associated with this process.

The I/O_Termination_Processor is then invoked to either terminate outstanding IOQE's for this process (if process is being terminated) or flag outstanding IOQE's for this process to indicate invalid parent PCT (if process is not being terminated).

The Release_PCT_Routine is then invoked to remove the PCT from the PCT run queue and return it to the free pool.

The PDE_Address of the input PCT is used to locate the process PDE in order to zero the Assigned_PCT_Address. If FCOS has assigned a temporary storage area to this process, the area is released for use by other processes. If the PCT being released is indicated by the Next_to_Execute_PCT_Address, the Next_To_Execute_PCT_Address is set to the PCT_Run_Queue_Address.

The control flow for this module is presented in figure 3.1.1.16-1.

d. Output - See table 3.1.1.16-1.

e. Module References -

1. (132) Release_PCT_Routine (FPMRLPCT) is CALLed.
2. (143) TQE_Dequeue_Processor (FPMTMDEQ) is CALLed.
3. (175) EQE_Dequeue_Processor (FPMEVDEQ) is CALLed.
4. (225) I/O_Termination_Processor (FIOPURGE) is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - N/A

DATA TABLE 3.1.1.16-1

NAME Free_PCT_Routine (FPMFRPCT)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Repeat_Options	Q003.36	I	101	101, 107	TPCTFLGS			
					133,174				
2	Outstanding_EQE_Indicators	Q003.32	I	133,173, 174,175	See App. E.	TPCTFLGS			
3	Outstanding_I/O_Count	Q003.25	I	101,200, 201	133	TPCTIORQ			
4	PDE_Address	Q003.4	I	101	See App. E	TPCTPDE			
5	Assigned _PCT_Address	Q002.2	O	101, 133	101, 108	TPDEPCT			
					109, 110				
6	FCOS_Assigned_Stack_Address	Q003.3	I	101	107, 133	TPCTSTOR			
7	Next_to_Execute_PCT_Address	Q001.3	I	See Appendix E		TCVTNEW			
			O						
8	PCT_Run_Queue_Address	Q001.1	I	131, 132	See App. E	TCVTPCT			
				305					

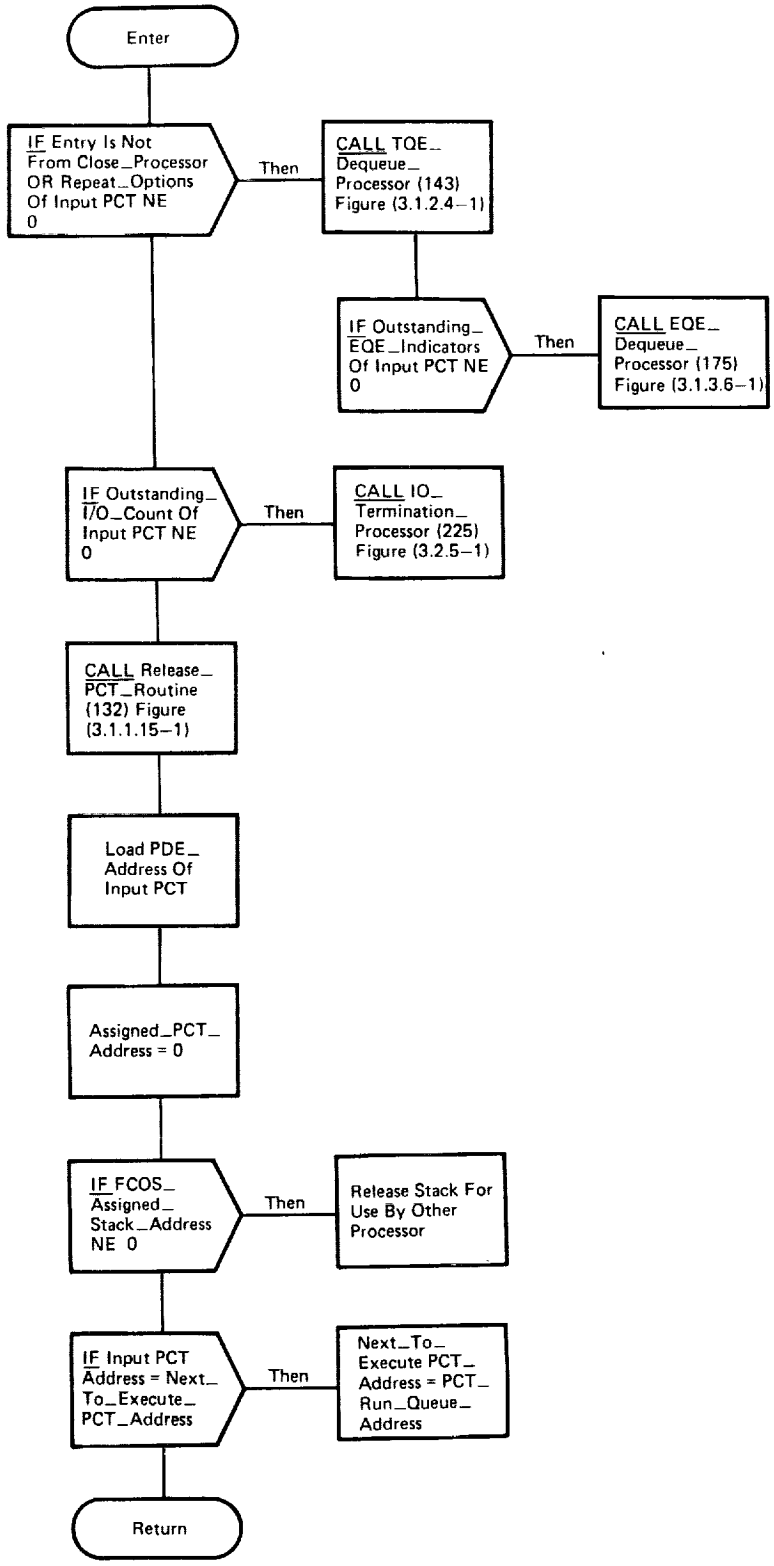


Figure 3.1.1.16-1. Free_PCT_Routine (FPMFRPCT)

3.1.2 Time Management

Time Management includes the FCOS support for all HAL/S realtime statements which have time as a parameter, the FCOS support of the AP-101 GO/NO-GO (Watchdog) timer. the FCOS support of Master Timing Unit (MTU) updates, MTU/software clock synchronization, the FCOS support of a software backup to the MTU, and the FCOS support of timer initiated I/O.

In support of these functions, the FCOS builds table entries called Timer_Queue_Elements (TQE's). A TQE contains information which identifies its purpose, the time at which it is to expire or a time increment, and the process to which it applies. When a TQE has been initialized with this information, it is said to be "active". Active TQE's are chained together and arranged by time such that the top TQE in the chain is next to expire. Active TQE's contain the time at which they are to expire.

AP-101 Program Counter 2 (PC2) is used as an interval timer to cause interrupts at the times specified in the active TQE's. PC2 is always set using the top TQE in the active TQE chain. When PC2 expires, the information in the top TQE is used to identify what action is to be taken.

The FCOS maintains a Greenwich Mean Time (GMT) clock as a software backup to the MTU. The software GMT clock is also used to provide finer granularity and faster response in the processing of HAL/S SCHEDULE and WAIT statements which have time options. The software clock has microsecond granularity and is maintained in the CVT. The clock is updated using AP-101 Program Counter 1 and the SSIP TOE interrupt (every 25th cycle).

The FCOS maintains a 16 bit GO/NO-GO(Watchdog) timer located in the IOP. The FCOS loads the GO/NO-GO timer with 180 milliseconds every 160 milliseconds (every 4th SSIP cycle). This timer is used to detect possible software errors and hardware failure. If an error condition occurs which prevents the FCOS from reloading the timer, the timer will expire and the IOP will interrupt the GPC to notify FCOS of the error. The user will be notified of the error, and the FOCS will attempt to continue processing.

The SSIP TQE interrupt is also used to initiate updates to the software clock and to initiate the FCOS duty cycle computation. Every twenty-fifth time the SSIP TQE expires, bit 11 of AP-101 Program Counter 1 (PC1) is interrogated (only bits 11-31 of PC1 are used, bits 0-10 are always zero). If the bit is zero, it is reset to 1 and 1.048576 seconds is added to the software clock. Since the software clock is only updated periodically, current GMT or MET is determined by summing the software clock value and the difference between the maximum PC1 value and the current PC1 value. The FCOS duty cycle computation is also performed every twenty-fifth SSIP TQE interrupt.

Figure 3.1.2-2 presents the control flow of Time Management. Whenever a request to perform an action at a time is made, a TQE is built and passed to the Timer_Queue_Generator process to add the TQE to the active TQE chain.

The Timer_Dequeue-processor is entered whenever a process is removed from the process run queue.

Time Management is also entered whenever AP-101 program counter 2 expires.

The functional description of Time Management presented in the following paragraphs is divided into the following areas: (See Figure 3.1.2-1)

a. Timer_Queue_Generator - The process of creating the active TQE chain.

b. GPC_Clock_Expiration_Processor - Results of a PC1 interrupt.

c. TQE_Expiration_Processor - Results of a PC2 interrupt.

d. TQE_Dequeue_Processor - Removing expired TQE's from the active TQE chain.

e. Time/Date_Application_Requests_Processor - Supplies time and date to requesting applications.

f. MTU_Update_Processor - FCOS involvement in updates to the MTU.

g. MTU_Redundancy_Manager - Synchronization of MTU and software clocks.

h. Time_Conversion_SVC_Services - Handles conversion SVC's.

i. Convert_To_Fixed_Point_Routine and Convert_To_Floating_Point_Routine - Conversion routines.

j. Current_GMT_Routine - Supplies current GMT.

k. Program_Counter_2_Update_Routine - Updates PC2.

l. Fixed_To_MTU_Format_Conversion_Routine and MTU_To_Fixed_Format_Conversion_Routine - Conversion routines.

m. Chain_TQE_Routine - Handles TQE's.

n. Expiration_Time_Update_Routine - Updates expiration time.

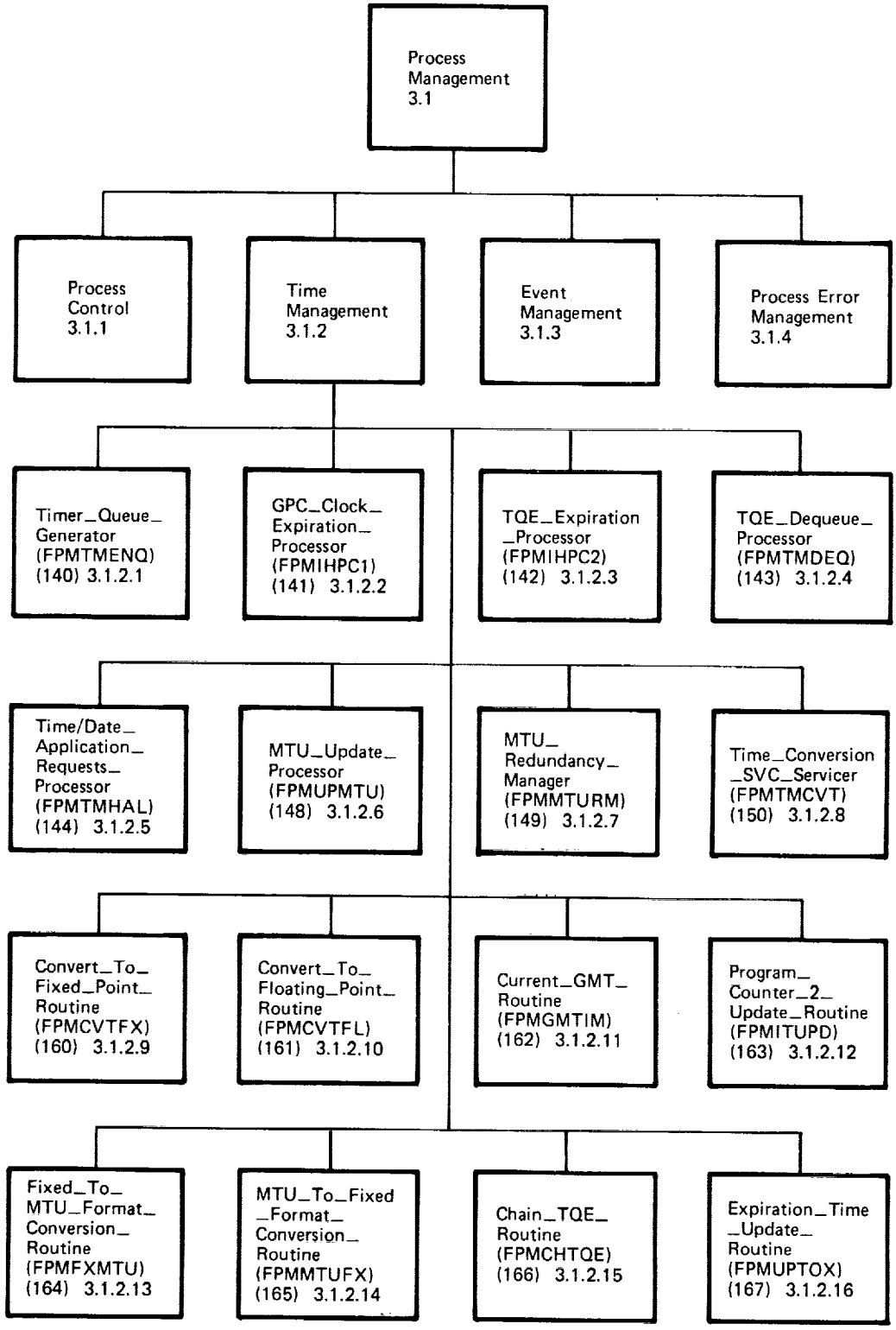


Figure 3.1.2–1. Time Management Hierarchy Diagram

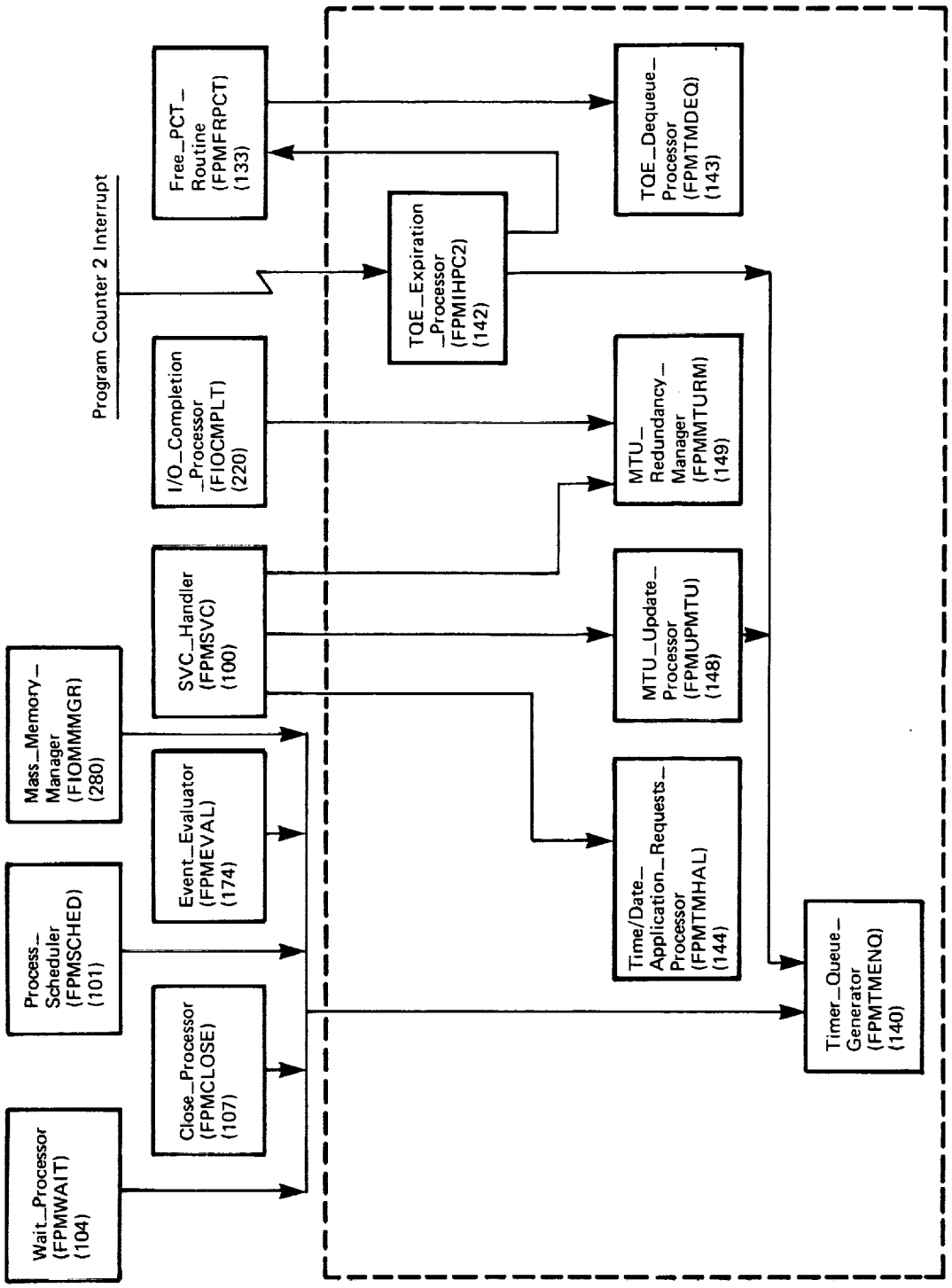


Figure 3.1.2-2. Time Management Control Flow

3.1.2.1 Timer_Queue_Generator (FPMTMENQ) (140)

Timer_Queue_Generator places interval timer requests on the time queue.

a. Control Interface -

1. CALLed by (101) Process_Scheduler (FPMSCHED)
2. CALLed by (104) Wait_Processor (FPMWAIT)
3. CALLed by (107) CLOSE_Processor (FPMCLOSE)
4. CALLed by (142) TQE_Expiration_Processor (FPMIHPC2)
5. CALLed by (148) MTU_Update_Processor (FPMUPMTU)
6. CALLed by (174) Event_Evaluator (FPMEVAL)
7. CALLed by (280) Mass_Memory_Manager (FIOMMMGR)

b. Input -

The top free pool Timer_Queue_Element contains the queue request information. For initial timer requests, the floating point registers will contain the following:

FPR0,1 - The Schedule At or Wait Until time of the Schedule In or Wait Delta time.

FPR2,3 - The Repeat Every delta time.

FPR4,5 - The Schedule Until time.

See table 3.1.2.1-1.

c. Process Description

Establish addressability of the top free pool TQE from the TQE_Free_Pool_Address. This TQE has been initialized by the caller of this program. TQE_Flags is loaded and the case macro is executed based on the TQE_Type_Indicator.

For a TQE_Type_Indicator of 1 and 8 CTOX_for_Repeat_Every is performed to compute TQE_Time_of_Expiration for a Repeat Every TQE. If the Cycle_Overrun_Indicator is not on, then the Time_Initiated_I/O_Parameter_List is checked for timer initiated I/O and I/O_SVC_Service_Processor is CALLed if there is timer initiated I/O. The Repeat_Delta_Time is obtained from the PCT and Expiration_Time_Update_Routine is CALLed to update the TQE_Time_of_Expiration with the Repeat_Delta_Time.

For a TQE_Type_Indicator of 2 and 5 CTOX_for_Wait_Schedule_In is performed to compute TQE_Time_of_Expiration for a Wait or Schedule In TQE. Current time is obtained by Get_Top_TQE_Time and updated by CALLing Expiration_Time_Update_Routine.

For a TQE_Type_Indicator of 3 and 4 CTOX_for_Wait_Until_Schedule_At is performed to compute TQE_Time_of_Expiration for a Wait Until or a Schedule At TQE. The time is obtained and converted to fixed point format by Convert_To_Fixed_Point_Routine and saved in TQE_Time_of_ Expiration.

For a TQE_Type_Indicator of 6 CTOX_for_Schedule_Until is performed to compute TQE_Time_of_expiration for a Schedule Until TQE. The time is converted by Convert_To_Fixed_Point_Routine and saved in TQE_ Time_of_Expiration.

For a TQE_Type_Indicator of 7 CTOX_for_Repeat_After is performed to compute TQE_Time_of_Expiration for a Repeat After TQE. Current time is obtained by Get_Top_TQE_Time and saved in TQE_Time_of_Expiration. The Repeat_Delta_Time is obtained from the PCT and added to TQE_Time_ of_Expiration by Expiration_Time_Update_Routine.

For a TQE_Type_Indicator of 9 Queue_Mass_Memory_TQE is performed. Current time is obtained by Get_Top_TQE_Time, updated by Expiration_ Time_Update_Routine and saved in TQE_Time_of_Expiration.

For a TQE_Type_Indicator of 10 and 12 CTOX_for_Runtime_GMT_Requests is performed to compute TQE_Time_of_Expiration for a Runtime or GMT update request. The coincident time is passed in the TQE_Parent_ PCT_Address. The coincident time is adjusted to the proper format and saved in TQE_Time_of_Expiration. Update_Time is CALLed to add in the Software_Clock_Offset to the TQE_Time_of_Expiration.

For a TQE_Type_Indicator of 11 CTOX_for_MET_Update is performed to compute TQE_Time_of_Expiration for a MET update request. MTU_Update_ Time is obtained and saved in TQE_Time_of_Expiration.

After TQE_Time_of_Expiration has been computed for the appropiate TQE_Type_Indicator the return code is set to 2 and TQE_Request_Type is checked to see if this is an initial TQE. If this is not an initial TQE, Chain_TQE_Routine is CALLed to requeue the TQE to the chain. Otherwise current time is obtained by Get_Top_TQE_Time. Next, TQE_ Time_of_Expiration is compared to current time. If TQE_Time_of_ Expiration is less than or equal to current time, then the TQE is considered past due. If the TQE is not past due ADD_TQE is CALLed to add the TQE to the active TQE chain. If the TQE is past due, set the return code to 4. If the TQE is a Schedule At and Repeat Every, then it is Phase Scheduled.

Phase Scheduling is the computation of a new Schedule At time which would be the next repeat interval in the future for the process. Compute_New_Time is called to handle this.

Compute_New_At_Time computes the new At time by figuring the number of intervals between the old At time and the current time and adds one repeat interval. If this time is in the future and the process is not SSIP, one additional repeat interval is added. The control flow for this module is shown in figure 3.1.2.1-1.

d. Output -

Bits 16-31 of register 7 save area word contain the return code. Explanation of return codes;

0 - Timer request successful, interval timer was updated.
2 - Timer request successful, interval timer was not updated.
4 - Timer request unsuccessful.

Also see table 3.1.2.1-1.

e. Module References -

1. (160) Convert_To_Fixed_Point_Routine is CALLed.
2. (161) Convert_To_Floating_Point_Routine is CALLed.
3. (162) Current_GMT_Routine is CALLed.
4. (163) Program_Counter_2_Update_Routine is CALLed.
5. (166) Chain_TQE_Routine is CALLed.
6. (167) Expiration_Time_Update_Routine is CALLed.
7. (200) I/O_SVC_Service_Processor is CALLed.
8. (201) Pre-Initialized_I/O_SVC_Processor (FIOSVCP) is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation -

1. CT is current time
   INT is intervals
   AT is the schedule AT time
   RE is the repeat interval

2. CTOX means Compute Time of expiration

DATA TABLE 3.1.2.1-1

NAME Timer_Queue_Generator (FPMTMENQ)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Next_To_Execute_PCT_Address	Q001.3	O	See App. E		TCVTNEW			
2	Top_TQE_Address	Q001.4	I	142,143, 166, 305	140, 142	TCVTTTQE			
3	TQE_Free_Pool_Address	Q001.11	I	142, 143, 166, 305	See App. E	TCVTTQEP			
4	TQE_Parent_PCT_Address	Q005.2	O	See App. E		TTQEPCT			
5	TQE_Time_Of_Expiration	Q005.3	O	140, 142, 148, 167	See App. E	TTQETOXH TTQETOXM			
6	TQE_Flags	Q005.6	I	104, 107, 140, 142	140, 143, 148, 174	TTQEFLGS			
7	Null_TQE_Indicator	Q005.8	I	101, 142	140				
8	TQE_Request_Type	Q005.9	I	101,104 107,140	140				
9	TQE_Type_Indicator	Q005.10	I	101, 104 107	140, 142, 148, 174				
10	Repeat_Delta_Time	Q003.22	I		140				
11	Delta_Time_30_Minute_Portion	Q003.23	I	101	140	TPCTRPTH			
12	Delta_Time_Half_Hour_Counter	Q003.24	I	101	140	TPCTRPTM			
13	PCT_Flags	Q003.29	I	101,142 320	107,133, 140,225	TPCTFLGS			
14	Cycle_Overrun_Indicator	Q003.34	I	142					
15	Time_Initiated_I/O_Parameter_List_Address	Q003.49	I	101	140	TPCTIOPP			
16	MTU_Update_Time	#022	I	148	140, 142				
17	MTU_Update_Half_Hour_Portion	#022.1	I	148	140, 142	TFCMMTUH			
18	MTU_Update_Half_Hour_Count	#022.2	I	148	140, 142	TFCMMTUM			
19	Software_Clock_Offset	#023	I	148	140, 148	TFCMOFST			
20	Last_TQE_Half_Hour_Count	#025.2	I	142, 149	140, 144	TFCMLTQM	V91W1999C	X	

DATA TABLE 3.1.2.1-1 (Cont'd)

NAME Timer_Queue_Generator (FPMTMENQ)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
21	SVC_New_PSW	&001.68	I	300	101, 140	TPSASNP			

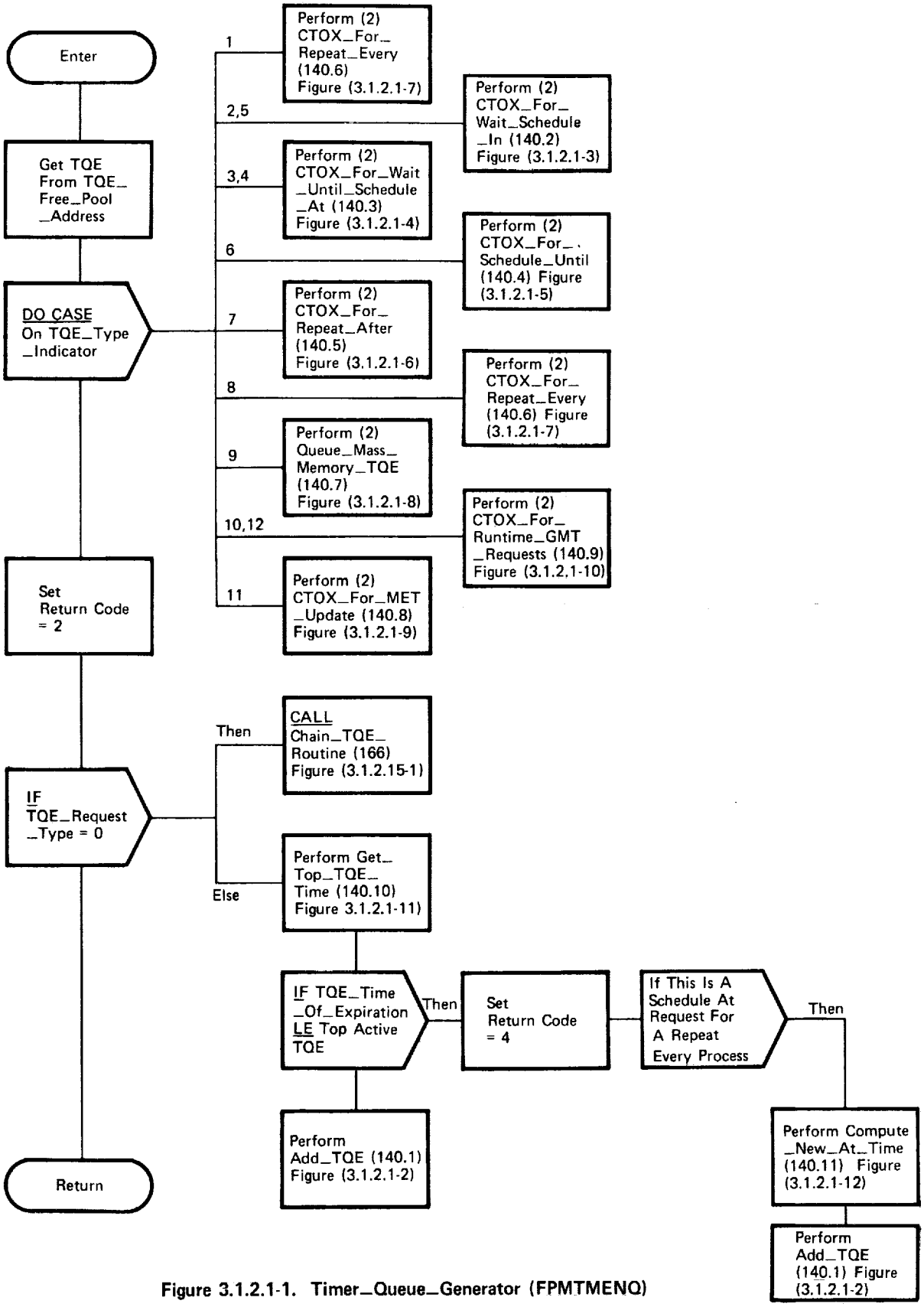


Figure 3.1.2.1-1. Timer_Queue_Generator (FPMTMENQ)

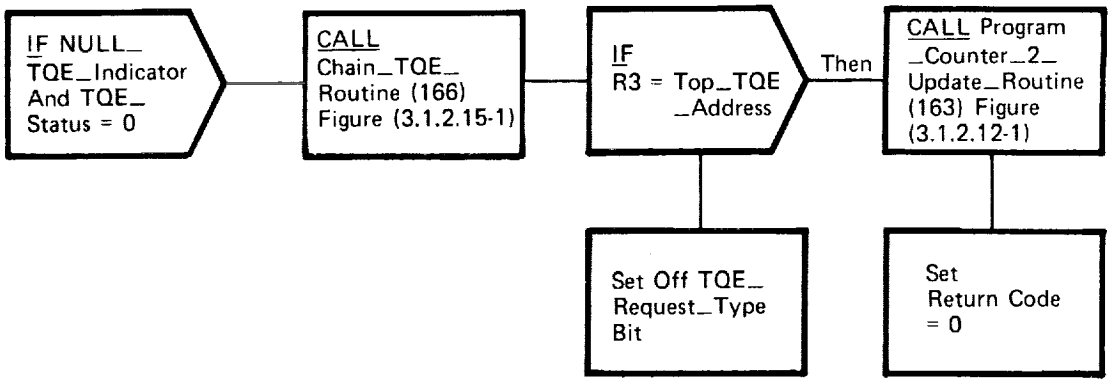


Figure 3.1.2.1-2. Timer_Queue_Generator Add_TQE (140.1)

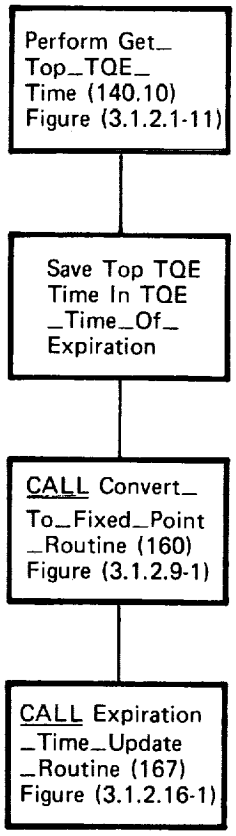


Figure 3.1.2.1-3. Timer_Queue_Generator CTOX_For_Wait_Schedule_In (140.2)

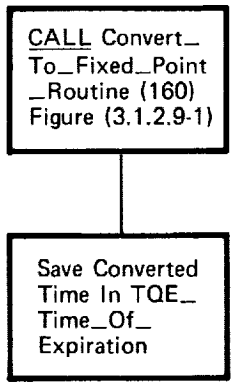


Figure 3.1.2.1-4. Timer_Queue_Generator CTOX_For_Wait_Until_Schedule_At (140.3)

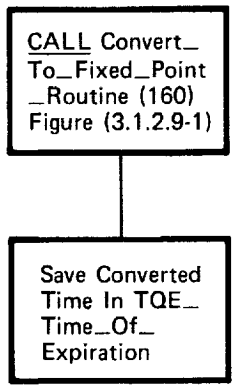


Figure 3.1.2.1-5. Timer_Queue_Generator CTOX_For_Schedule_Until (140.4)

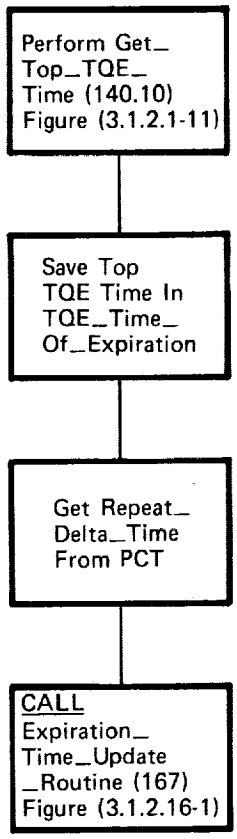


Figure 3.1.2.1-6. Timer_Queue_Generator CTOX_For_Repeat_After (140.5)

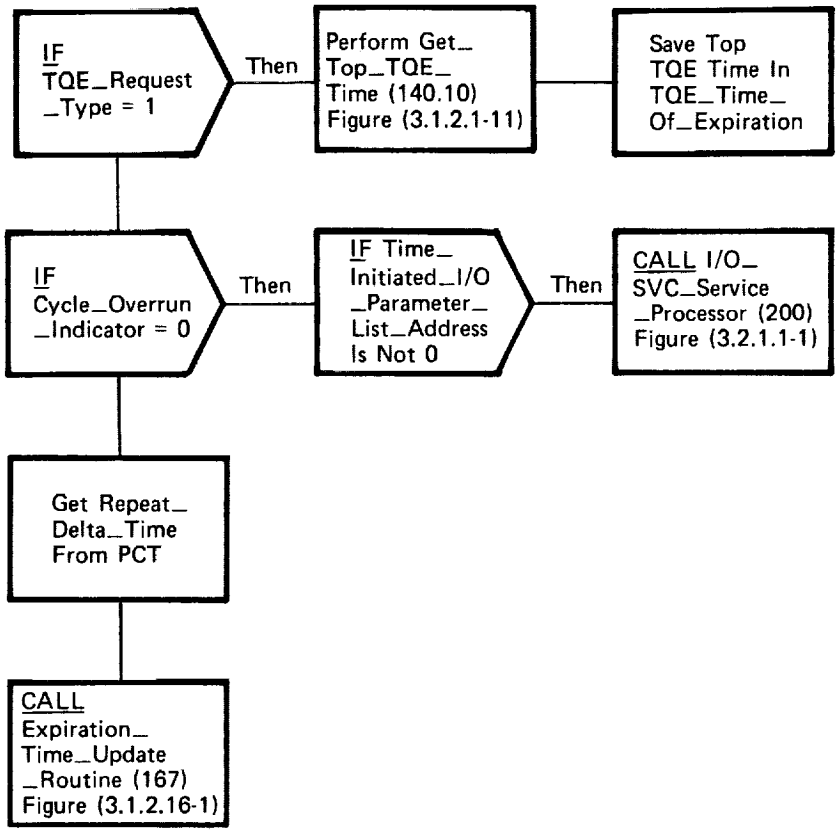


Figure 3.1.2.1-7. Timer_Queue_Generator CTOX_For_Repeat_Every (140.6)

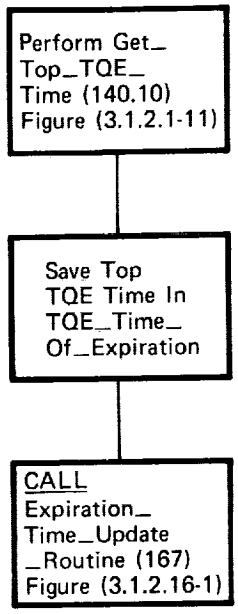


Figure 3.1.2.1-8. Timer_Queue_Generator Queue_Mass_Memory_TQE (140.7)

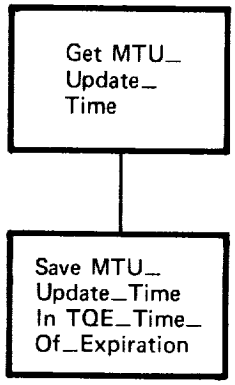


Figure 3.1.2.1-9. Timer_Queue_Generator CTOX_For_MET_Update (140.8)

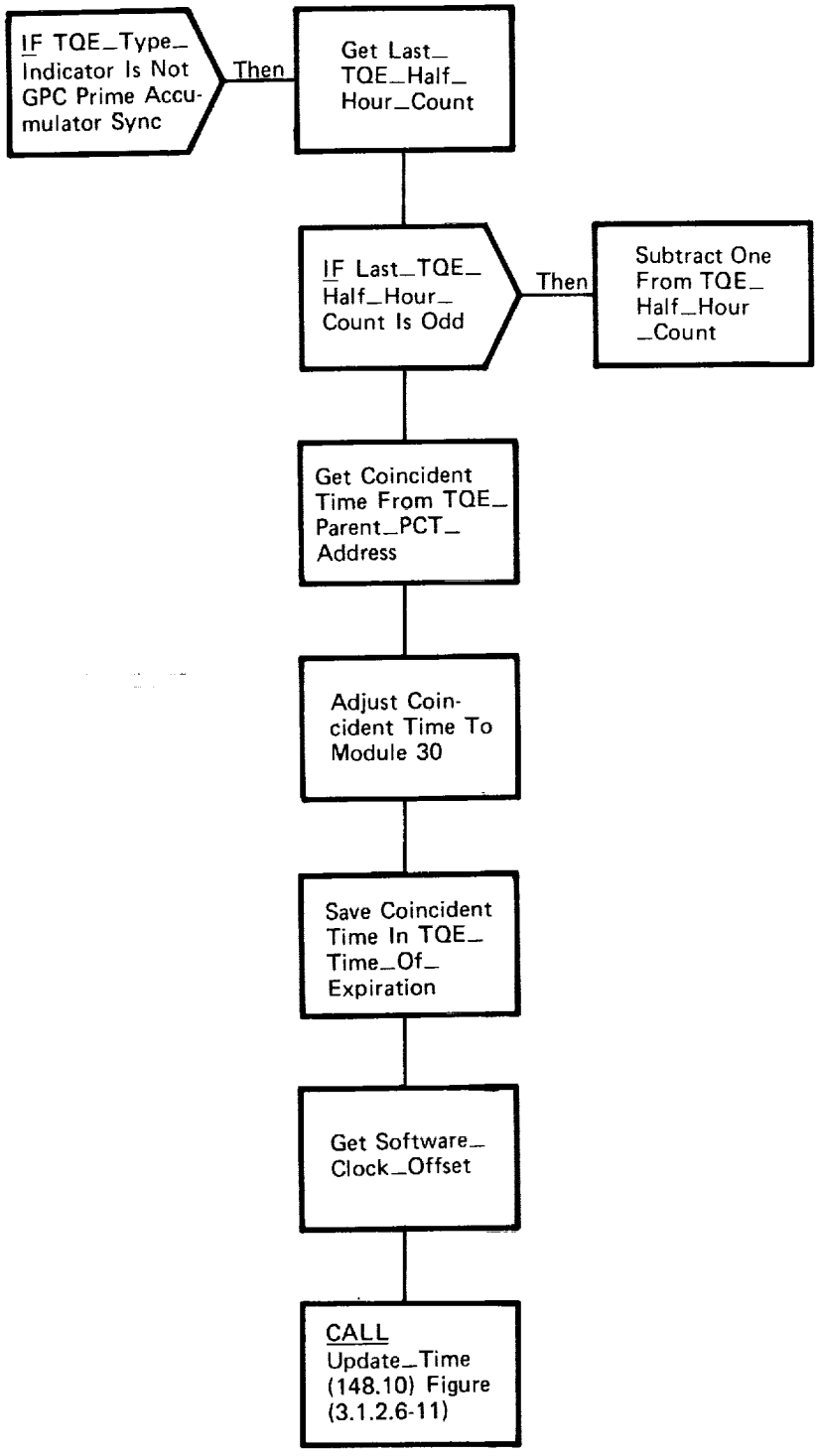


Figure 3.1.2.1-10. Timer_Queue_Generator CTOX_For_Routine_GMT_Requests (140.9)

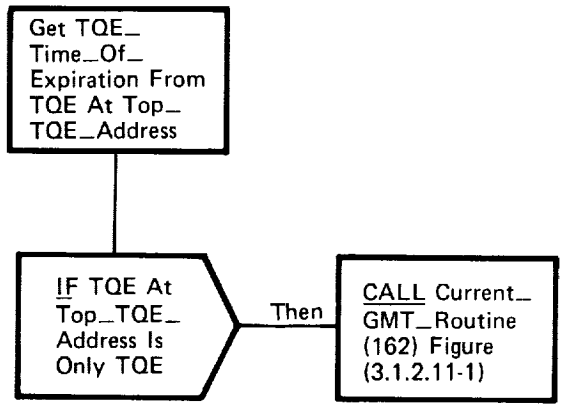


Figure 3.1.2.1-11. Timer_Queue_Generator Get_Top_TQE_Time (140.10)

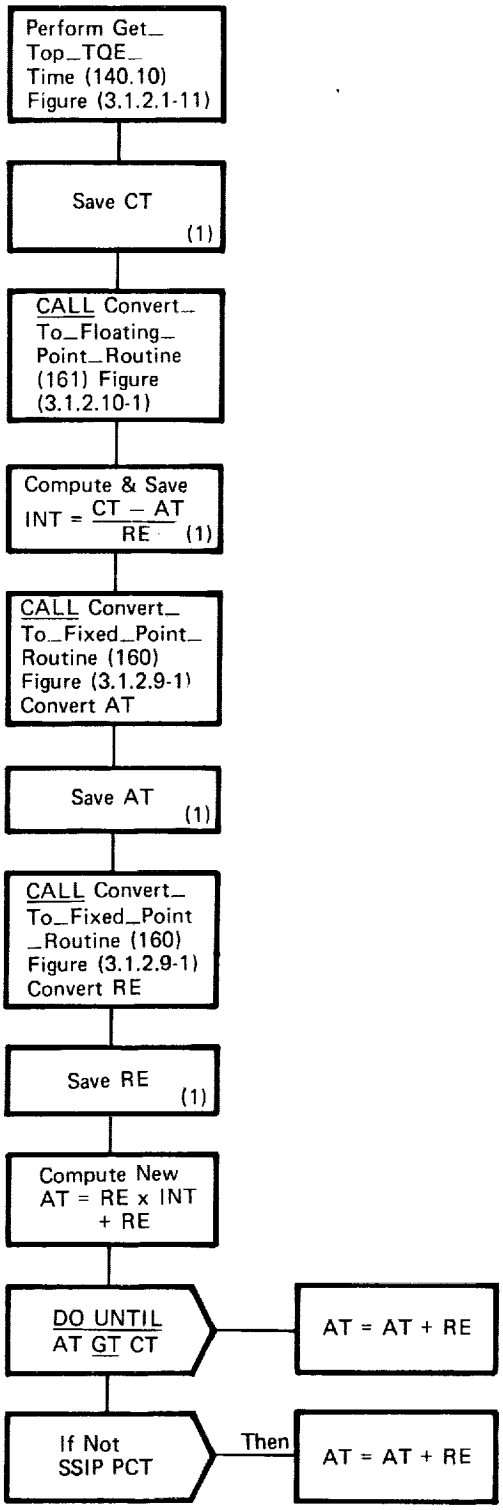


Figure 3.1.2.1-12. Timer_Queue_Generator Compute_New_At_Time (140.11)

3.1.2.2 GPC_Clock_Expiration_Processor (FPMIHPC1) (141)

GPC_Clock_Expiration_Processor handles PC1 interrupts and reloads the PC1 clock.

a. Control Interface - This processor gets control when a PC1 interrupt causes a PSW swap.

b. Input - See table 3.1.2.2-1.

c. Process Description - When a PC1 interrupt occurs PC1_Software_Portion is loaded with X'001F' and the GPC_Software_Clock is updated by 2.097152 seconds. The control flow for this module is shown in figure 3.1.2.2-1.

d. Output - See table 3.1.2.2-1.

e. Module References - None

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.1.2.2-1

NAME GPC_Clock_Expiration_Processor (FPMIHPC1)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	PC1_Software_Portion	&001.86	0	141,142, 300	142	TPSAPC1			
2	GPC_Software_Clock	Q001.86	0	141, 142 149, 305	149, 162 163	TCVTSWCH TCVTSWCM			
3	Software_Clock_30_Minute_ Portion	Q001.87	0	141, 142 149, 305	162, 163	TCVTSWCH			
4	Software_Clock_Half_Hour_Counter	Q001.88	0	141, 142 149, 305	149, 162 242	TCVTSWCM			

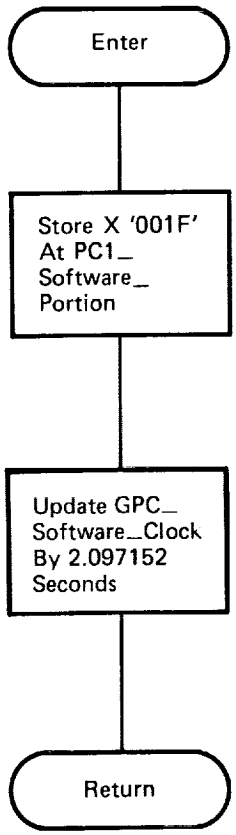


Figure 3.1.2.2-1. GPC_Clock_Expiration _Processor (FPMIHPC1)

3.1.2.3 TQE_Expiration_Processor (FPMIHPC2) (142)

TQE_Expiration_Processor is responsible for fielding GPC Program Counter 2 (PC2) interrupts and initiating the appropriate action based on the purpose of the TQE to which the interrupt applies.

a. Control Interface - PSW swap from hardware via Program Counter 2 interrupt.

b. Input - The top TQE in the active TQE chain is the TQE to which the interrupt applies. See Table 3.1.2.3-1.

c. Process Description - Issue a Null Sync code and if a SVC Sync is in progress, then backup Interrupt_PSW_of_Process two halfwords and reset SVC_Sync_In_Progress_Indicator. If SVC Sync is not in progress, then check the I/O_Sync_In_Progress_Indicator. If I/O Sync is in progress save Program_Counter_2_Old_PSW at the Interrupt_PSW_of_Process in the PCT.

If the PC2_Software_Portion is negative then this is a valid interrupt. If the TQE_Type_Indicator is not SSIP and the Redundant_Set_Sync_Mask is not zero then issue a Timer Sync code. If the Redundant_Set_Sync Mask is zero, CALL the Timer_Synchronization_Processor. Next loop through the active TQE chain handling the expired TQEs by performing TQE Case.

After handling the expired TQEs check the I/O_Sync_In_Progress_Indicator. If not zero, zero I/O_Sync_In_Progress_Indicator and CALL the I/O_Completion_Processor.

Finally control is relinguished by returning to the interrupted process or CALLing the Process_Dispatcher to give control to the highest priority PCT.

Expired TQEs are handled by TQE_Case who performs the following functions, removes the expired TQE from the active TQE chain and adds it to the free pool. If the TQE is active then the TQE_Time_of_Expiration is saved at Last_Expired_TQE_Time and control is passed to the appropriate subroutine based on the TQE_Type_Indicator.

For a TQE_Type_Indicator of 1, the Process_SSIP_TQE is performed. Normal_SSIP_Synchronization_Processor is CALLed and the Go/No_Go_Phase_Count is decremented and checked. If the Go/No_Go_Phase_Count is zero, then the Go/No_Go_Clock_Value is reloaded and saved. The Duty_Cycle_Computation_Phase_Count and perform Compute_Duty_Cycle. Compute_Duty_Cycle computes the percent of CPU usage and performs Software_Clock_Maintenance. Software_Clock_Maintenance resets PC1 clock. If the PCT is not in a scheduled wait, set the Cycle_Overrun_Indicator and CALL Process_Error_Recovery_

Processor. If the PCT is in a scheduled wait, the PCT_Wait_Indicators and Cycle_Overrun_Indicator are zeroed and the PCT_Priority is checked to see if a higher priority PCT is waiting to run. Then CALL Timer_Queue_Generator to rechain SSIP TQE.

For a TQE_Type_Indicator of 2, 3 and 7, Process_Wait_Repeat is performed. Process_Wait_Repeat zeros the PCT_Wait_Indicators and CALLs the Process_Switcher.

For a TQE_Type_Indicator of 4 and 5 Process_Schedule_TQE is performed. If this is the SSIP PCT and TQE_Type_Indicator is 4 then set SSIP flag in TQE_Flags and perform Process_SSIP_TQE, else zero PCT_Wait_Indicators. If this is the SSIP PCT CALL Reset_IOP_Processor and CALL Normal_SSIP_Synchronization_Processor. If it is not the SSIP PCT and PCT_Flags indicate Repeat Every set TQE_Flags and CALL Timer_Queue_Generator. Then CALL Process_Switcher.

For a TQE_Type_Indicator of 6, Schedule_Until_TQE is performed. Cancelled_PCT_Indicator is set to one and if the Scheduled_Wait_Indicator is one, Free_PCT_Routine is CALLed. Then the Event_Flag of Process_Event is set to zero and the Event_Evaluator is CALLed. Then the return code is set to four.

For a TQE_Type_Indicator of 8, Repeat_Every_TQE is performed. First a check is made of the Cycle_Overrun_Indicator. If on, CALL Process_Error_Recovery_Processor. Otherwise zero PCT_Wait_Indicators and check the priority of the PCT at Next_To_Execute_PCT_Address. Then Timer_Queue_Generator is CALLED to requeue the TQE.

For a TQE_Type_Indicator of 9, Mass_Memory_TQE is performed to CALL the Mass_Memory_Manager.

For a TQE_Type_Indicator of 10 RUNTIME_Update_TQE is performed. The time source of the update is checked and the appropriate update time is obtained. The TQE_Time_of_Expiration is adjusted and saved in the GPC_Software_Clock.

For a TQE_Type_Indicator of 11, MET_Update_TQE is performed. The MET_Reference_Time is obtained, modified, and saved. Perform Signal_Event and set return code to four.

For a TQE_Type_Indicator of 12, GMT_Update_TQE is performed. MTU_Update_Time is obtained, modified, and saved at GPC_Software_Clock. PC1 is reset to its maximum value and the System_Reset_Emulation_Flag_Word is set. Then the PSW is loaded with System_Reset_PSW.

d. Output - See Table 3.1.2.3-1.

e. Module References -

1. (102) Process_Switcher (FPMSWTCH) is CALLed.
2. (103) Process_Dispatcher (FPMDISP) is CALLed.
3. (133) Free_PCT_Routine (FPMFRPCT) is CALLed.
4. (140) Timer_Queue_Generator (FPMTMENQ) is CALLed.
5. (163) Program_Counter_2_Update_Routine (FPMITUPD) is CALLed.
6. (174) Event_Evaluator (FPMEVAL) is CALLed.
7. (182) Process_Error_Recovery_Processor (FPMSDERR) is CALLed.
8. (220) I/O_Completion_Processor (FIOCMPLT) is CALLed.
9. (244.5) Reset_IOP_Processor (FIORESET) is CALLed.
10. (280) Mass_Memory_Manager (FIOMMMGR) is CALLed.
11. (361) Normal_SSIP_Synchronization_Processor (FCMCSYNC) is CALLed.
12. (364) Timer_Synchronization_Processor (FCMTSYNC) is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - none

i. Constraints and Assumptions - none

j. Detailed Implementation - none

DATA TABLE 3.1.2.3-1

NAME TQE_Expiration_Processor (FPMIHPC2)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Null_Sync_Code_Sending_Pattern	Q001.73	I		See App. E	TCVTNULS			
2	SYC_Sync_In_Progress_Indicator	Q001.49	O	220,305, 363	142,220	TCVTSYCS			
3	Interrupt_PSW_of_Process	Q003.5	O	See Appendix E		TPCTPSW			
4	I/O_Sync_In_Progress_Indicator	Q001.50	O	305,363	142,280	TCVTIOS			
5	Program_Counter_2_Old_PSW	&001.71	I		142	TPSAC2OP			
6	PC1_Software_Portion	&001.86	I	141,142, 300	142	TPSAPC1			
7	PC2_Software_Portion	&001.87	I	300	142	TPSAPC2			
8	TQE_Type_Indicator	Q005.10	I	101,104, 107	140,142, 146,147				
9	Redundant_Set_Sync_Mask	Q001.47	I	305,363	See App. E	TCVTRCSM			
10	TQE_Time_of_Expiration	Q005.3		140,142, 145,167	See App. E	TTQETOXH TTQETOXL			
11	Time_of_Expiration_Half_Hour	Q005.4	O	140,142, 148, 167	See App. E	TTQETOXH			
12	Time_of_Expiration_Half_Hour_Multiples	Q005.5	O	140,142, 148,167	See App.	TTQETOXM			
13	Last_Expired_TQE_Time	#025			140,142, 144,149				
14	Last_TQE_Half_Hour_Portion	#025.1	O	142,149	140,142, 246	TFCMLTQH	V91W1999C	X	
15	Last_TQE_Half_Hour_Count	#025.2	O	142,149	140,144	TFCMLTQM	V91W1999C	X	
16	Active_PCT_Address	Q001.2	I	103	See App. E	TCVTOLL			
17	Next_To_Execute_PCT_Address	Q001.3	I	See Appendix F		TCVTNEX			
18	Top_TQE_Address	Q001.4	O	142,143, 166,305	140,142	TCVTTTQE			
19	TQE_Free_Pool_Address	Q001.11	O	142,143, 166,305	See App. E	TCVTTQEP			
20	TQE_Status	Q005.7	O	101,143	142				

DATA TABLE 3.1.2.3-1 (Cont'd)

NAME TQE_Expiration_Processor (FPMIHPC2)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
21	Null_TQE_Indicator	Q005.8	O	101,142	140				
22	Go/No_Go_Phase_Count	#011	O	142		TFCMGNGP			
23	Go/No_Go_Clock_Value	#006	I		142	TFCMGNGV			
24	Duty_Cycle_Computation_Phase_Count	#012	O	142		TFCMDCYP			
25	Duty_Cycle_Computation_Frequency	#010	I		142	TFCMDCYF			
26	Scheduled_Wait_Indicator	Q003.48	C	101,107, 142	See App. E				
27	Cycle_Overrun_Indicator	Q003.34	O	142					
28	PCT_Wait_Indicators	Q003.42	O	See Appendix E		TPCTWAIT			
29	PCT_Priority	Q003.2	I	101,105, 106	See App. E	TPCTPRI			
30	PCT_Flags	Q003.29	O	101, 142, 320	107,133, 140,225	TPCTFLGS			
31	TQE_Flags	Q005.6	O	104,107, 140,142	140,143 148,170	TTQEFLGS			
32	SSIP_PCT_Indicator	Q003.38	I	101	101,142				
33	Cancelled_PCT_Indicator	Q003.30	O	101,109, 142	See App. E				
34	Event_Used_Indicator	@007.1	I		See App. E				
35	Process_Event	Q002.1	I	See App. E	See App. E	TPDEVENT			
36	MTU_Update_Time	#022		148	140,142				
37	MTU_Update_Half_Hour_Portion	#022.1	I	148	140,142	TFCMMTUH			
38	MTU_Update_Half_Hour_Count	#022.2	I	148	140,142	TFCMMTUM			
39	ICC_Prime_Time	I610.06	I		142,149				
40	Internal_Time_Microseconds	I610.10	I	149	142,660, 665,955	TICCGMTH			

DATA TABLE 3.1.2.3-1 (Cont'd)

NAME TQE_Expiration_Processor (FPMIHPC2)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
41	Internal_Time_Half_Hours	I610.14	I	149	142,149, 660,665	TICCGMTM			
42	GPC_Software_Clock	Q001.86		141,142, 149,305	149,162 163	TCVTSWCH TCVTSWCM			
43	Software_Clock_30_Minute_Portion	Q001.87	O	141,142, 149,305	162,163	TCVTSWCH			
44	Software_Clock_Half_Hour_Counter	Q001.88	O	141,142, 149,305	149,162 242	TCVTSWCM			
45	MET_Reference_Time	Q001.89		142,149, 205,305	144	TCVTMRTH TCVTMRTH			
46	MET_Reference_30_Minute_Portion	Q001.90	O	142,149, 205,305	142,144, 149,205	TCVTMRTH			
47	MET_Reference_Half_Hour_Counter	Q001.91	O			TCVTMRTM			
48	Prime_GPC_Time_Milliseconds	I250	O	142,149 205		CZ2VMETM			
49	Prime_GPC_Time_Half_Hours	I260	O	142,149, 205		CZ2VMETH			
50	ICC_Status_Flags	I320	O	See App. E	See App. E	CZ2VIF1			
51	System_Reset_Emulation_Flag_Word	Q001.96	I	142,305	142,305	TCVTSREF			
52	System_Reset_PSW	&001.15	I	300	142,190	TPSASRP			
53	Duty_Cycle_Limit	Q001.43	I		142	TCVTDCLM			
54	GPC_Self_Status_Table_Address	Q001.58	I	300	See App. E	TCVTGST			
55	Timer_Code_Inverse_Sending_Pattern	Q001.70	I		142,280	TCVTTIMI			
56	GPC_Duty_Cycle	I010.01	O	142,800	See App. E	TGSTDUTY	See App. E	x	x
57	GST_Status_Flags	I010.14	O	See Appendix E		TGSTIND		x	x
58	GPC_Prime_ID	I050	I	351,700, 880,820	See App. E	CZ2VGPCP			
59	GPC_ID	#001	I	300	See App. E	TFCMID			
60	Go/No_Go_Processing_Frequency	#009	I		142	TFCMGNGF			

DATA TABLE 3.1.2.3-1 (Cont'd)

NAME TQE_Expiration_Processor (FPMIHPC2)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
61	PDE_Address	Q003.4	I	101	See App. E	TPCTPDE			
62	Next_TQE_Address	Q005.1	O	142,143, 166	142,143, 166	TTQENXT			
63	TQE_Parent_PCT_Address	Q005.2	I	See Appendix E		TTQEPCT			
64	ICC_Buffer_Address_Table	Ø005	I		142,149	FCMICCAD			
65	Event_State	@007.2	O	See Appendix E					
66	Idle_Time	@003	O	142,191	142,191	FPMIDTIM			
67	ICC_Duty_Cycle	I320.05	O		142				
68	FMPT_HDR_Bits	T070	O	See Appendix E		CDLBERRM			

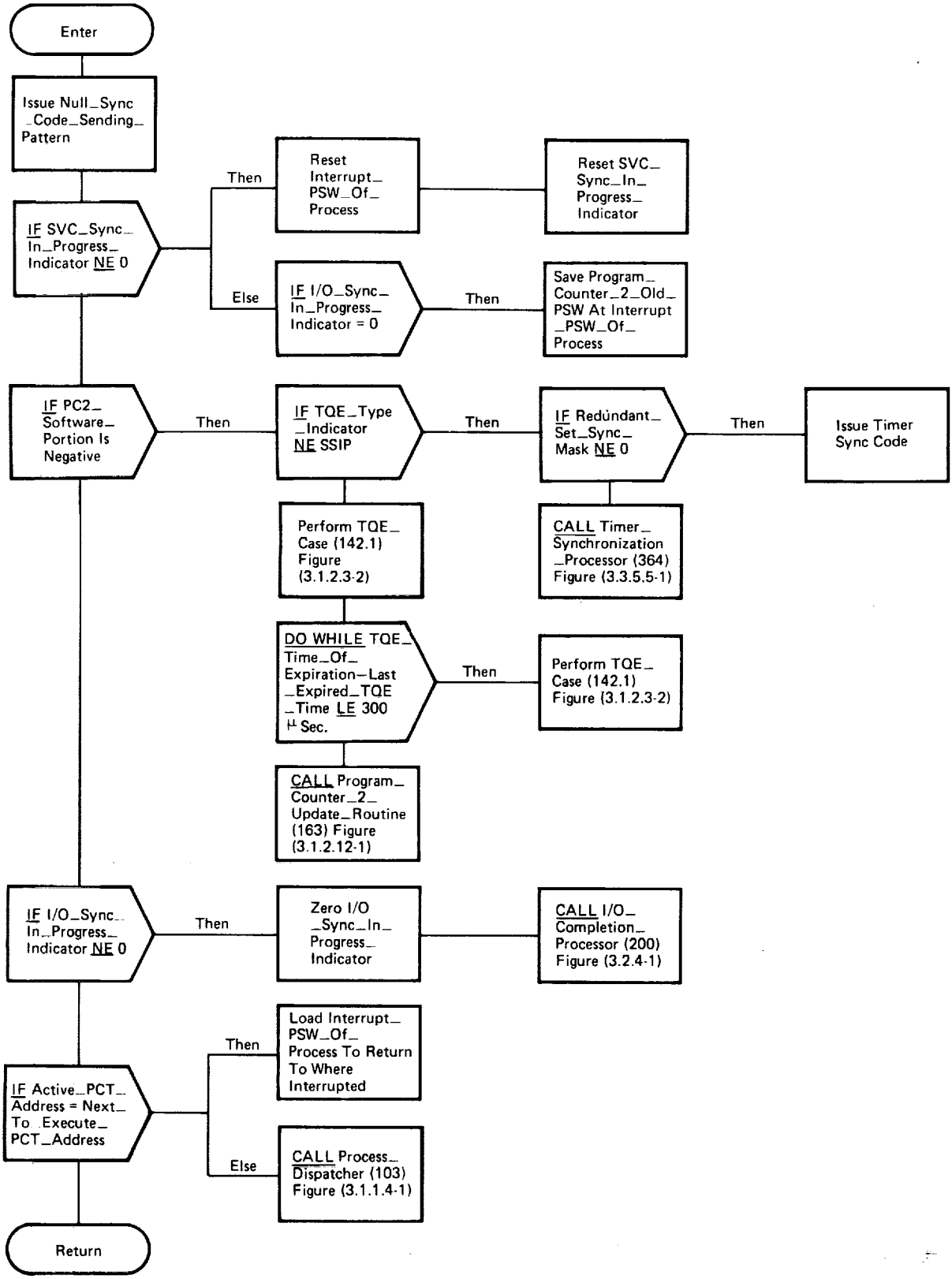


Figure 3.1.2.3-1. TQE_Expiration_Processor (FPMIHPC2)

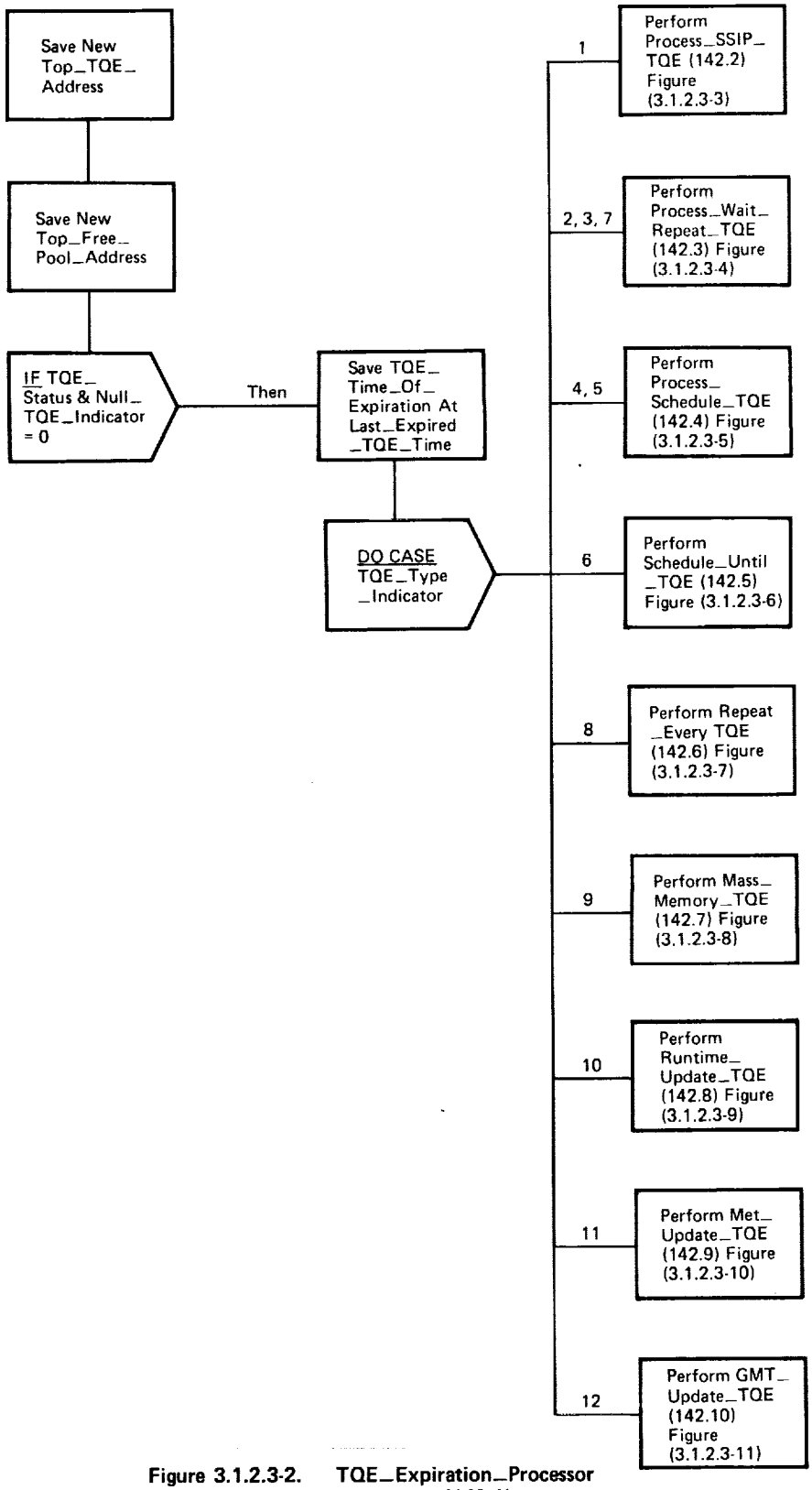


Figure 3.1.2.3-2. TQE_Expiration_Processor TQE_Case (142.1)

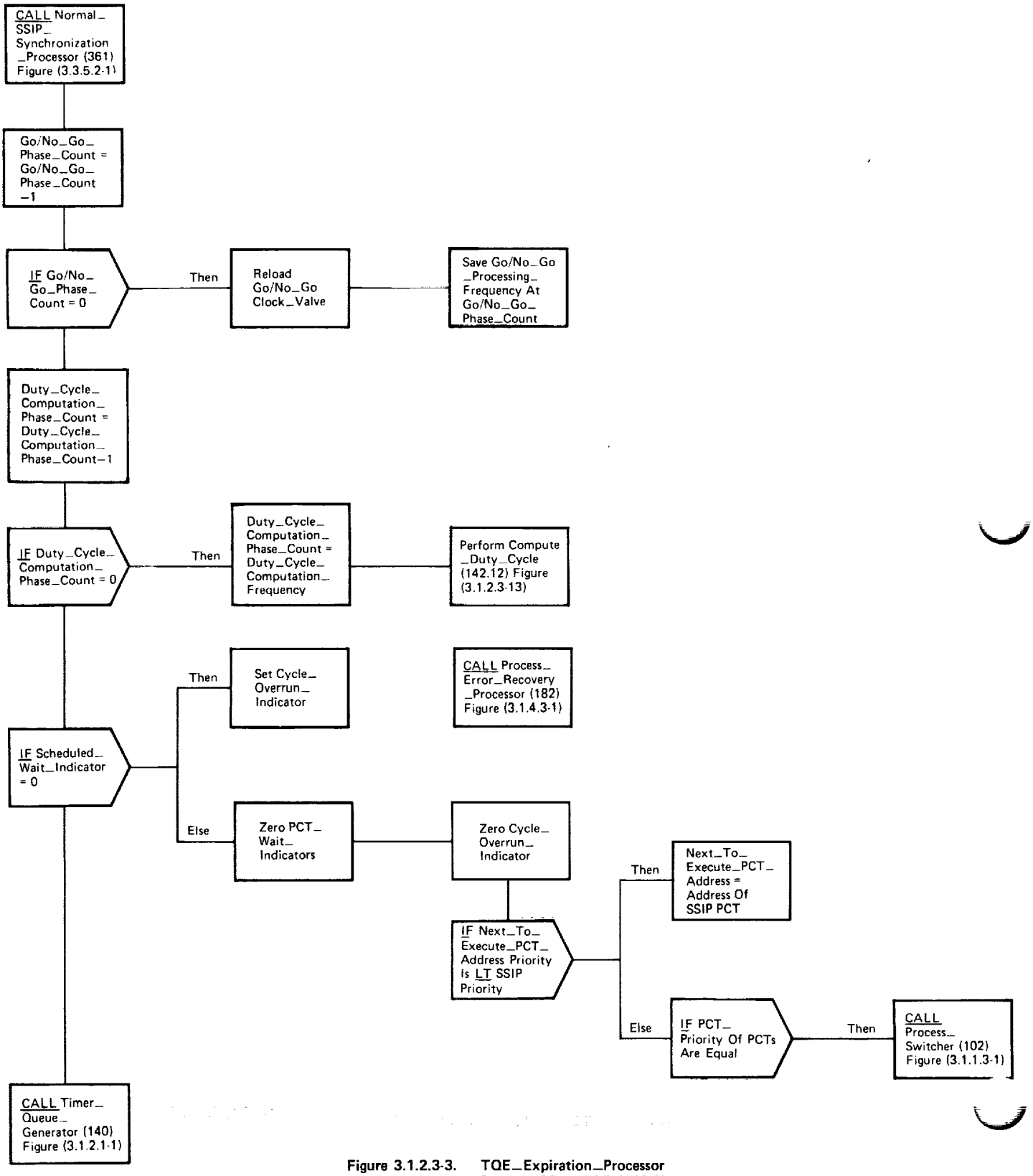


Figure 3.1.2.3-3. TQE_Expiration_Processor Process_SSIP_TQE (142.2)

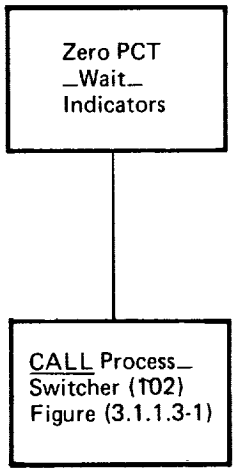


Figure 3.1.2.3-4. TQE_Expiration_Processor Process_Wait_Repeat_TQE (142.3)

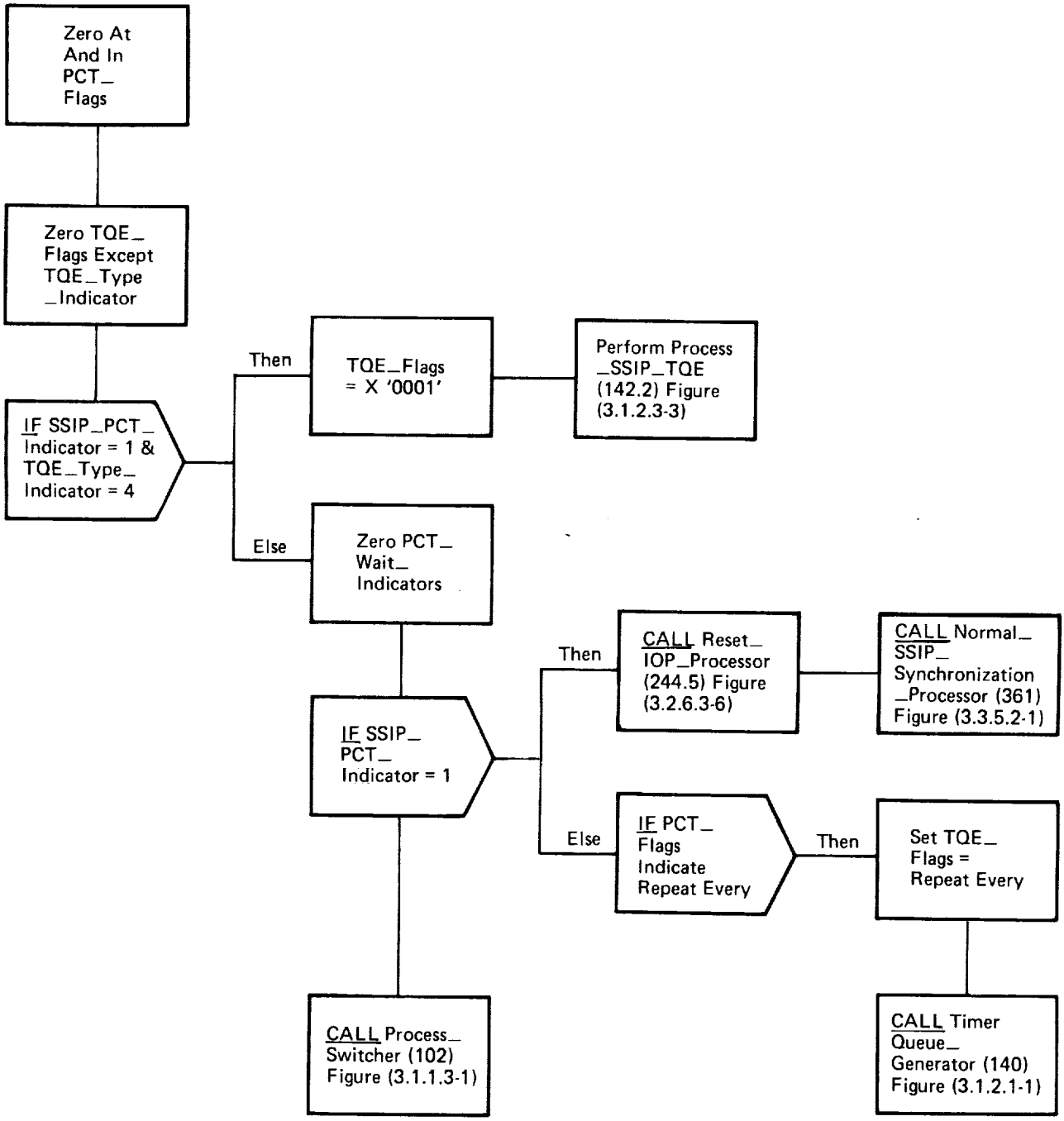


Figure 3.1.2.3-5. TQE.Expiration_Processor Process_Schedule_TQE (142.4)

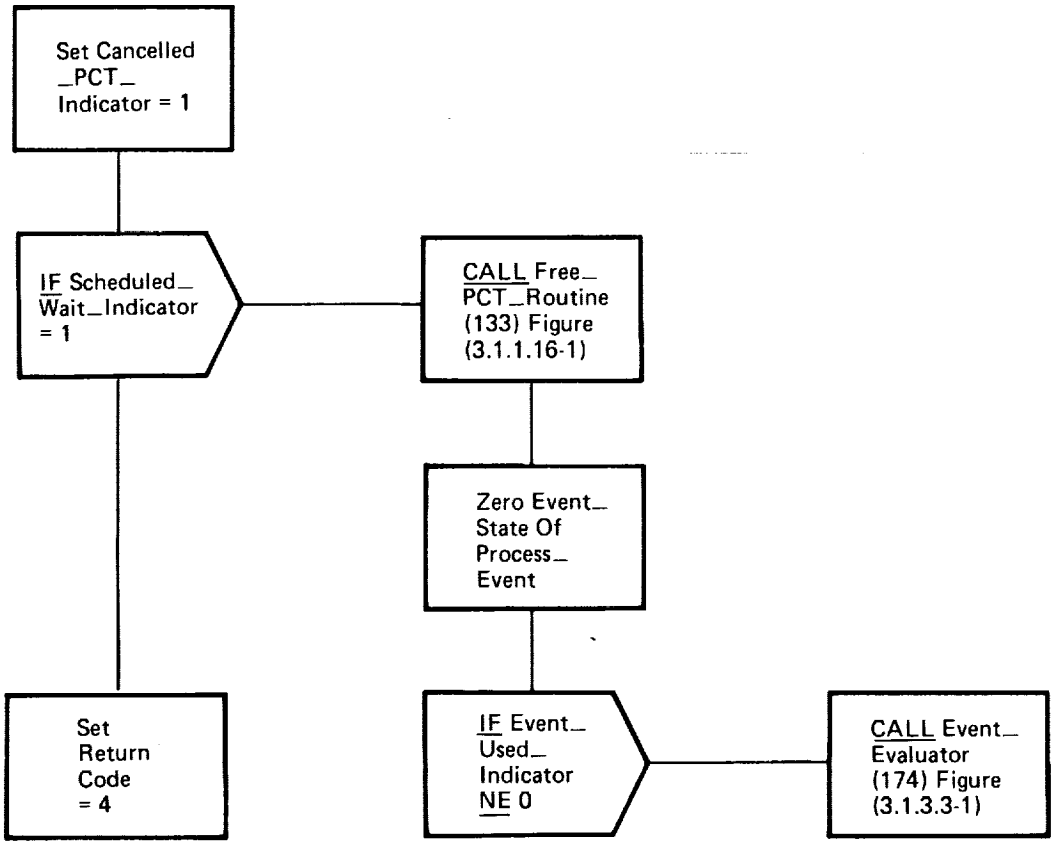


Figure 3.1.2.3-6. TQE_Expiration_Processor Schedule_Until_TQE (142.5)

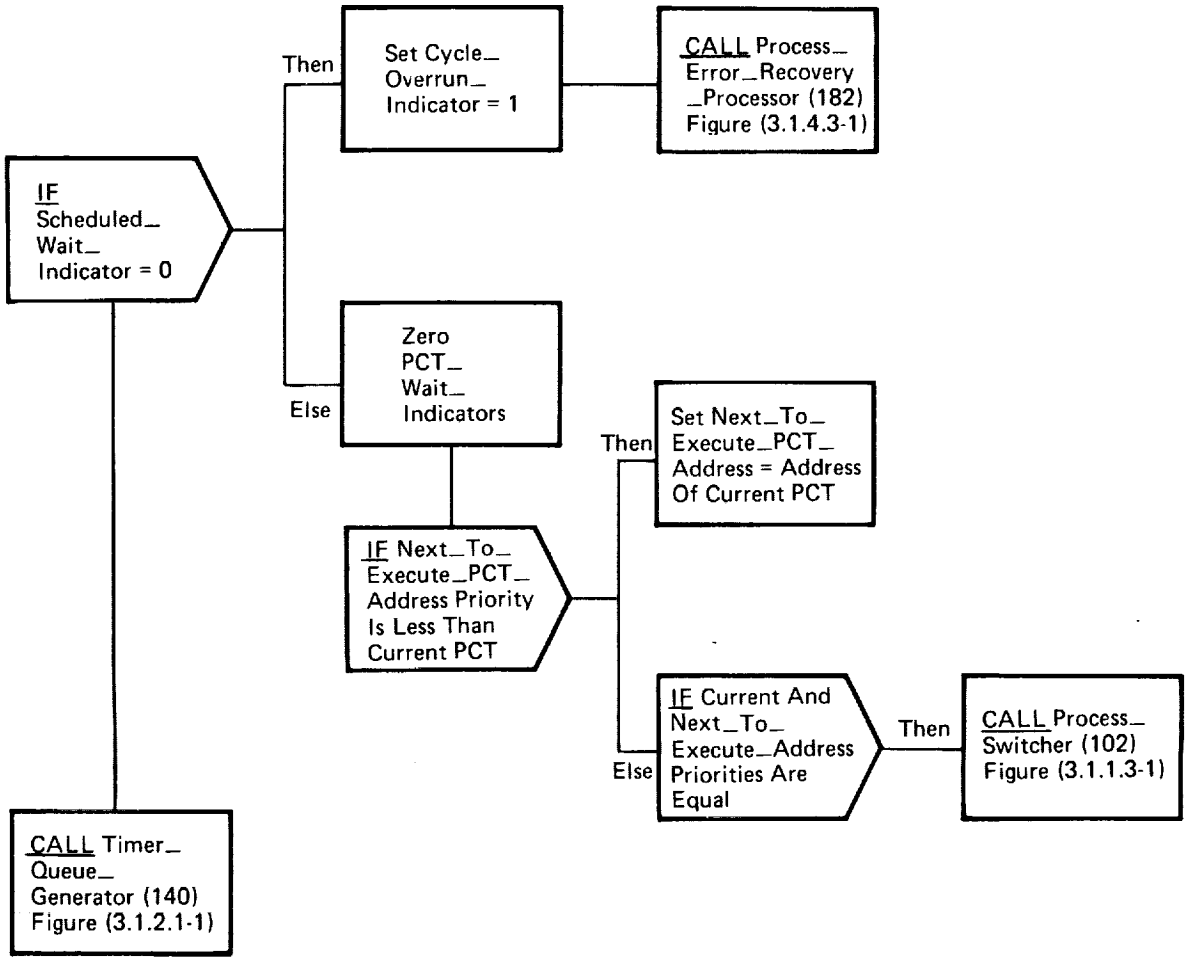


Figure 3.1.2.3-7. TQE_Expiration_Processor Repeat_Every_TQE (142.6)

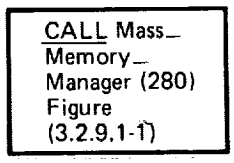


Figure 3.1.2.3-8. TQE_Expiration_Processor Mass_Memory_TQE (142.7)

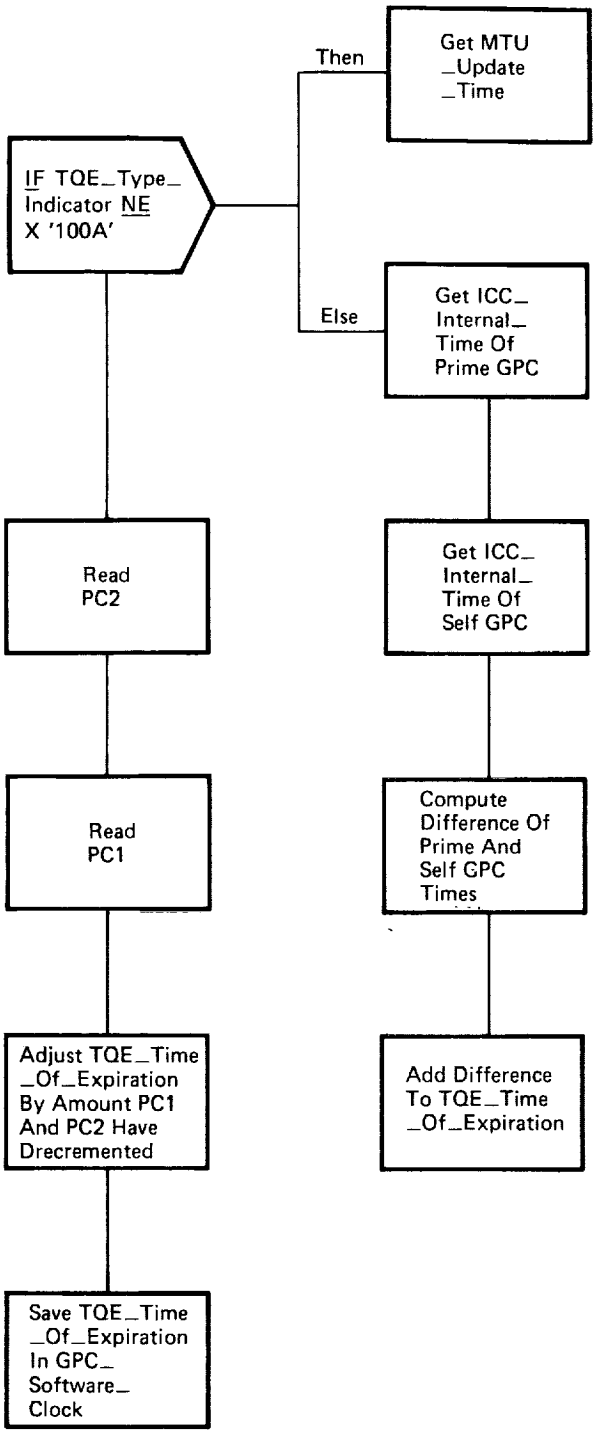


Figure 3.1.2.3-9. TQE_Expiration_Processor RUNTIME_Update_TQE (142.8)

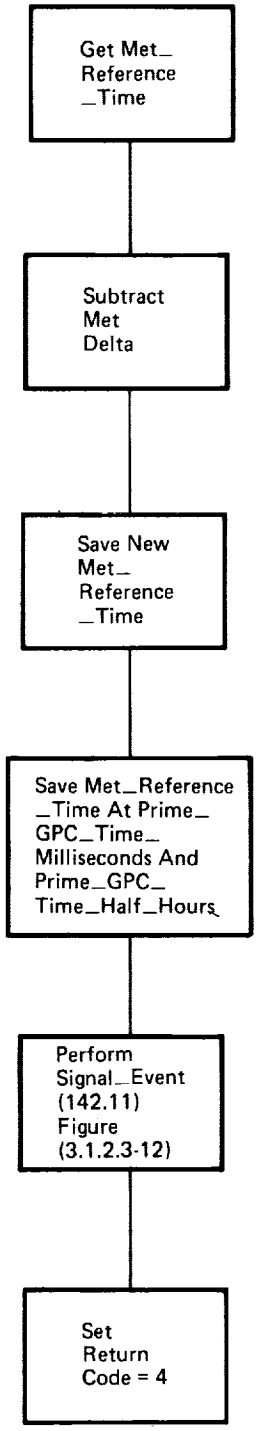


Figure 3.1.2.3-10. TQE_Expiration_Processor MET_Update_TQE (142.9)

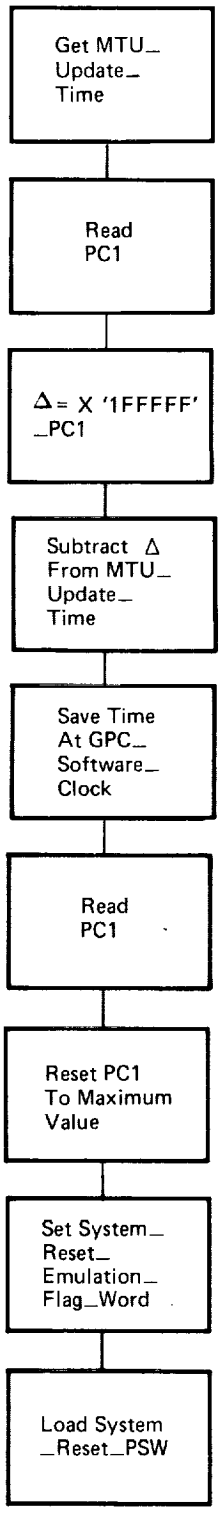


Figure 3.1.2.3-11. TQE_Expiration_Processor GMT_Update_TQE (142.10)

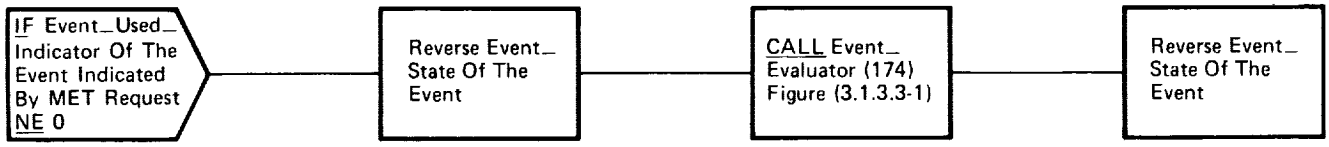


Figure 3.1.2.3-12. TQE_Expiration_Processor Signal_Event (142.11)

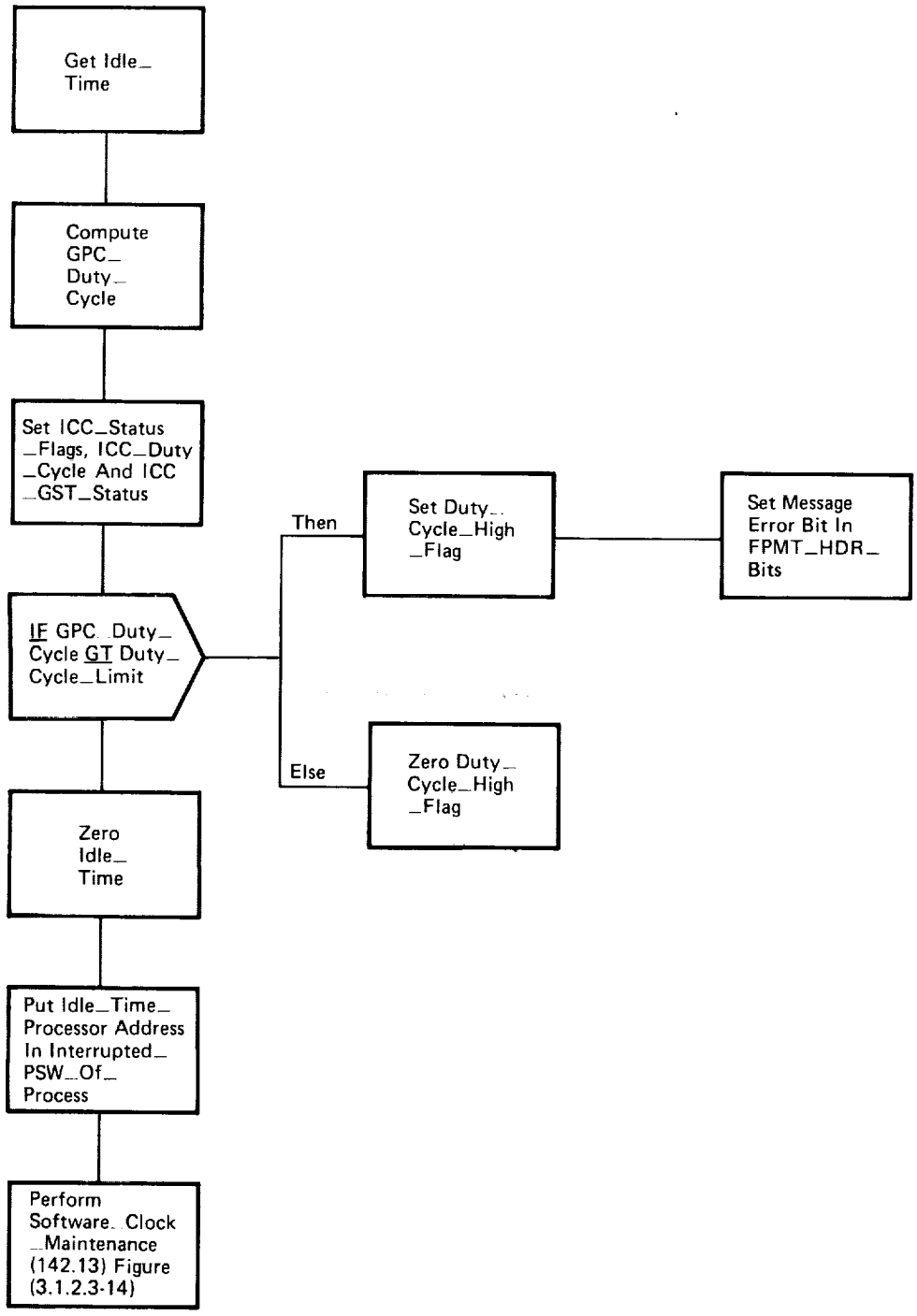


Figure 3.1.2.3-13. TQE_Expiration_Processor Compete_Duty_Cycle (142.12)

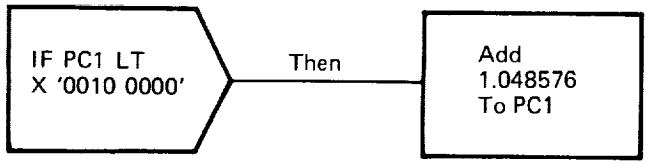


Figure 3.1.2.3-14. TQE_Expiration_Processor Software_Clock_Maintenance (142.13)

3.1.2.4 TQE_Dequeue_Processor (FPMTMDEQ) (143)

TQE_Dequeue_Processor scans the active TQE chain for any TQE's associated with the PCT being cancelled or terminated.

a. Control Interface -

1. CALLed by (133) Free_PCT_Routine (FPMFRPCT).

b. Input - Register 0 contains the address of the PCT being terminated. See table 3.1.2.4-1.

c. Process Description - Get Top_TQE_Address. Loop through the active TQE chain comparing TQE_Parent_PCT_Address with the PCT address in register 0 of the cancelled or terminating PCT. When an equal compare is found or a TQE belonging to the PCT being terminated or cancelled, the TQE is removed from the active chain. However, if the TQE at the Top_TQE_Address belongs to the PCT being terminated of cancelled then it is not removed from the chain. The TQE_Status bit is set to one in the TQE_Flags saying this TQE is cancelled.

The control flow for this module is shown in Figure 3.1.2.4-1.

d. Output - A revised active TQE chain if a TQE belonging to the cancelled or terminating PCT was found.

e. Module References - None

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.1.2.4-1

NAME TQE_Dequeue_Processor (FPMTMDEQ)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Top_TQE_Address	Q001.4	I	142,143, 166,305	140, 142	TCVTTTQE			
2	TQE_Free_Pool_Address	Q001.11	O	142,143, 166,305	See App. E	TCVTTQEP			
3	Timer_Queue_Element	Q005	I			TFTQE			
4	Next_TQE_Address	Q005.1	O	142, 143 166	142, 143 166	TTQENXT			
5	TQE_Parent_PCT_Address	Q005.2	I	See Appendix E		TTQEPCT			
6	TQE_Flags	Q005.6	O	104,107 140,142	140,143, 148,174	TTQEFLGS			
7	TQE_Status	Q005.7	O	101,143	142				

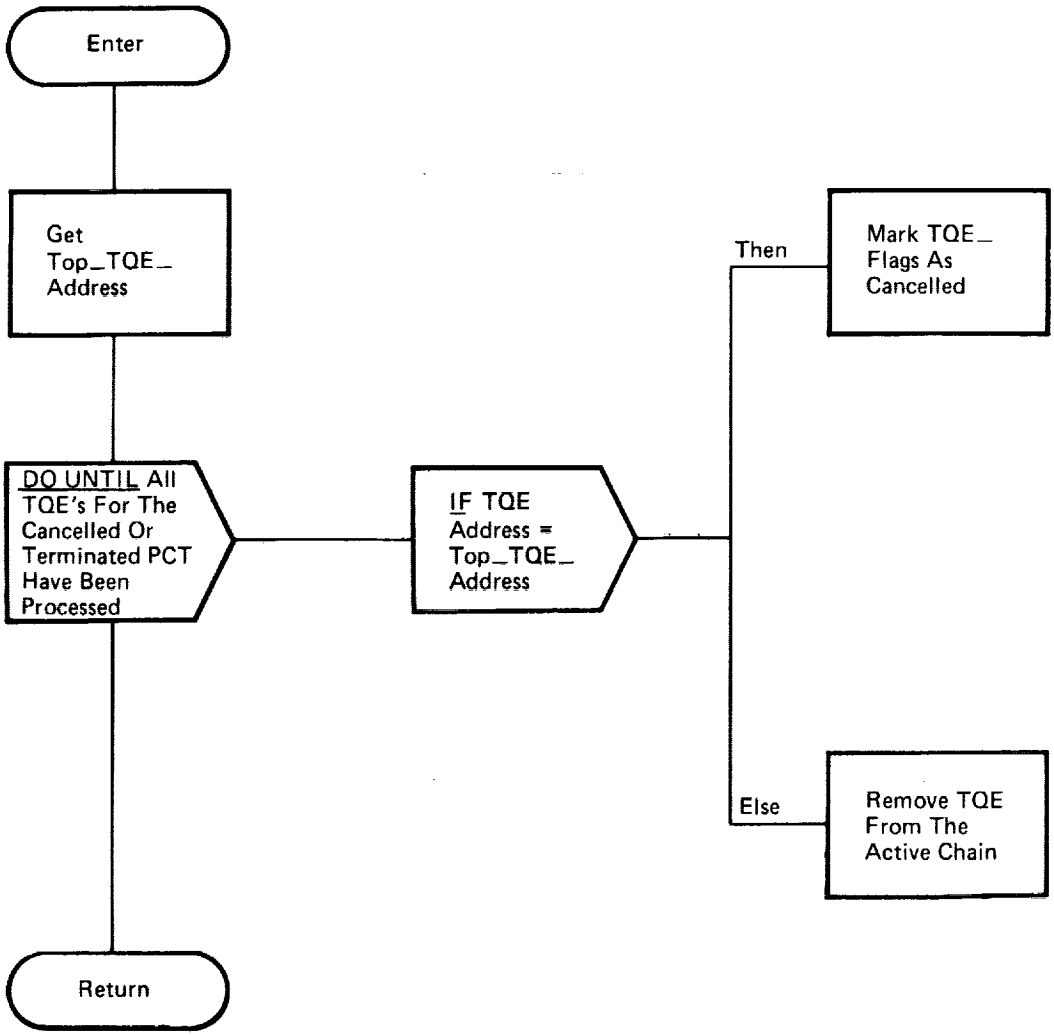


Figure 3.1.2.4-1. TQE_Dequeue_Processor (FPMTMDEQ)

3.1.2.5 Time/Date_Application_Requests_Processor (FPMTMHAL) (144)

Time/Date_Application_Requests_Processor services requests for RUNTIME, CLOCKTIME, DATE, and Mission Elapsed time (MET). SVC 22.

a. Control Interface -

1. CALLed by (100) SVC_Handler (FPMSVC)

b. Input - Bits 0-15 of register 0 contain the address of the SVC parameter list. See table 3.1.2.5-1.

c. Process Description - For a Time/Date_Request_Type of 0 compute RUNTIME. CALL Current_GMT_Routine to get current time and CALL Convert_To_Floating_Point_Routine to convert time to floating point format.

For a Time/Date_Request_Type of 1 compute CLOCKTIME. CALL SVC_Synchronization_Processor and get Last_Expired_TQE_Time. CALL Convert_To_Floating_Point_Routine to convert time to floating point format.

For a Time/Date_Request_Type of 2 compute the DATE. Execute procedure Date_Request which will get Last_TQE_Half_Hour_Count and subtract MET_Reference_Half_Hour_Counter. This result is converted to days and Day_of_Year_of_Lift_Off is added to it. The number of days is then checked to see if the year end has been exceeded. If so the number of days in the year is subtracted and the year incremented by one. The date is then placed in register 5 of the application register set.

For a Time/Date_Request_Type of other than 0, 1, or 2 compute the Mission Elapsed time. Get Last_Expired_TQE_Time and subtract the MET_Reference_Time. This result is then converted to floating point by CALLing Convert_To_Floating_Point_Routine and saved in MET_Buffer_Address passed in the SVC parameter list.

The control flow for this module is shown in Figures 3.1.2.5-1 and 3.1.2.5-2.

d. Output -

1. RUNTIME and CLOCKTIME (in seconds) are returned in floating point registers 0 and 1.

2. DATE is returned in general purpose register 5 in the following format.

Bits 0-15 - Year

Bits 16-31 - Day of the year

3. MET (in floating point seconds) is stored in the location specified in the SVC parameter list.

e. Module References -

1. (161) Convert_To_Floating_Point_Routine (FPMCVTFL) is CALLed.
2. (162) Current_GMT_Routine (FPMGMTLM) is CALLed.
3. (363) SVC_Synchronization_Processor (FCMSSYNC) is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.1.2.5-1

NAME Time/Date_Application_Requests_Processor (FPMTMHAL)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Time/Date_Request_Parameter_List	S022	I		144				
2	Time/Date_Request_Type	S022.1	I		144				
3	MET_Buffer_Address	S022.3	I		144				
4	Last_Expired_TQE_Time	#025	I		140,142 144,149		V91W1999C		
5	Last_TQE_Half_Hour_Count	#025.2	I	142,149	140,144	TFCMLTQM	V91W1999C	x	
6	MET_Reference_Time	Q001.89	I	142,149 205,305	144	TCVTMRTH			
						TCVTMRTM			
7	MET_Reference_Half_Hour_Counter	Q001.91	I	142,149 205,305	142,144, 149,205	TCVTMRTM			
8	Year_of_Lift_Off	Q001.84	I		144	TCVTLOY			
9	Day_of_Year_of_Lift_Off	Q001.85	I		144	TCVTLOD			

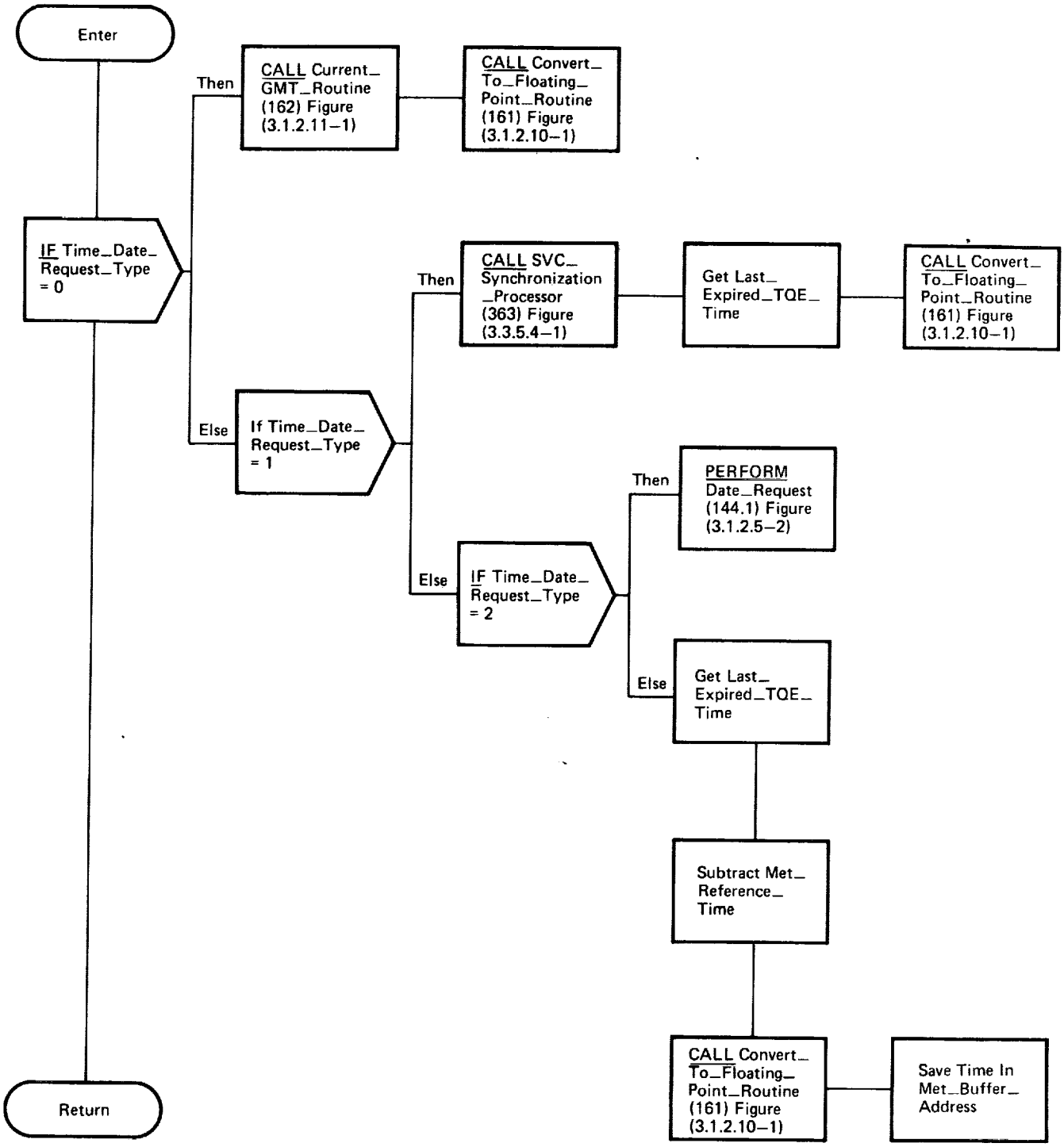


Figure 3.1.2.5-1. Time/Date_Application_Request_Processor (FPMTMHAL)

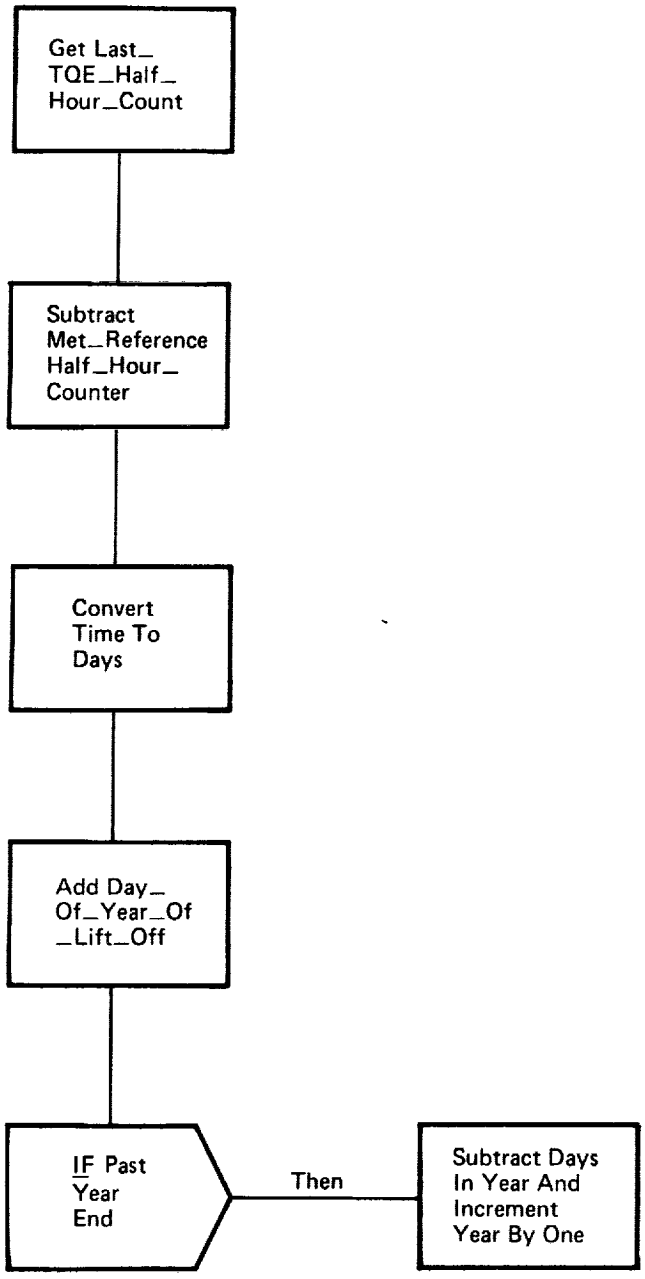


Figure 3.1.2.5-2. Time/Date_Application_Requests_Processor Date_Request (144.1)

3.1.2.6 MTU_Update_Processor (FPMUPMTU) (148)

MTU_Update_Processor supports user directed updates to the MTU and allows all time sources to be set equal to the source currently selected by the prime GPC. SVC 31.

a. Control Interface -

1. CALLed by (100) SVC_Handler (FPMSVC).

b. Input - Bits 0-15 of general purpose register 0 contain the address of the MTU_Update_Parameter_List.

c. Process Description - CALL SVC_Synchronization_Processor to sync on SVC31. Save MTU_Update_Completion_Event_Address and execute Determine_Bus_For_Update to decide which MTU accumulator to write to. Save the Device_ID_Number in the SVC 24 I/O_SVC_Parameter_List and execute the case macro to determine which MTU_Update_Request_Type is to be processed.

For a MTU_Update_Request_Type of 1 MTU_Update_Processor will perform MET_Reset. MET_Reset initializes the I/O_Buffer_Address, Operation_Code and the Event_Address in the I/O_SVC_Parameter_List. Then MET_Reset CALLs the I/O_SVC_Service_Processor to issue the I/O request. On return from the I/O_SVC_Service_Processor the Event_Address is zeroed.

For a MTU_Update_Request_Type of 2 perform MET_Update. MET_Update initializes the I/O_Buffer_Address and Operation_Code in the I/O_SVC_Parameter_List. Then Convert_To_BCD is executed to convert MTU_Update_Coincident_Time from minutes to BCD format. Next Common_MET/GMT_Update_Logic is executed to perform the common processing for a MET/GMT update.

Common_MET/GMT_Update_Logic gets the MTU_Update_Absolute_Time and CALLs Convert_To_Fixed_Point_Routine to convert MTU_Update_Absolute_Time to FCOS fixed point format. MTU_Update_Absolute_Time is then saved in MTU_Update_Time. The MTU_Update_Absolute_Time is then converted to MTU format by CALLing Fixed_To_MTU_Format_Conversion_Routine. Next I/O_SVC_Service_Processor is CALLed to process the I/O request.

On return from the I/O_SVC_Service_Processor a TQE is obtained from the TQE free pool at TQE_Free_Pool_Address. The MTU_Update_Coincident_Time is stored into TQE_Parent_PCT_Address of the TQE, and the MTU_Delta_Update_Time is obtained and converted to fixed point format by CALLing Convert_To_Fixed_Point_Routine. Then the TQE_Flags field is initialized with a TQE_Type_Indicator specifying either a MET or GMT update. Finally Timer_Queue_Generator is CALLed to enqueue the TQE.

For a MTU_Update_Request_Type of 3 and 4 perform a GMT_Update. GMT_Update initializes the I/O_Buffer_Address and Operation_Code in the I/O_SVC_Parameter_List. The MTU_Update_Coincident_Time is then converted to BCD format by CALLing Convert_To_BCD. Then the Common_MET/GMT_Update_Logic is executed. (See previous paragraph).

For a MTU_Update_Request_Type of 5 execute Accumulator_Sync. Accumulator_Sync starts by initializing the I/O_Buffer_Address in the I/O_SVC_Parameter_List and getting the GST_Status_Flags to find the selected time source.

If the selected time source is a MTU, then perform MTU-Time_Source. MTU_Time_Source gets the address of the selected MTU buffer and CALLs MTU_To_Fixed_Format_Conversion_Routine to convert the MTU time to fixed point format. MTU_Time_Source computes the coincident time by rounding the fixed point time up to the nearest minute and saves it in MTU_Update_Time. The coincident time is converted to BCD by Convert_To_BCD. Next a TQE is obtained from the TQE Free Pool by getting the TQE address from the TQE_Free_Pool_Address. The Coincident time computed by MTU_Time_Source is then saved in TQE_Parent_PCT_Address field of the TQE. Finally the TQE_Flags are set with an accumulator sync TQE_Type_Indicator.

If the selected time source is GPC prime time, then the I/O_Buffer_Address and Operation_Code are initialized in the I/O_SVC_Parameter_List. The I/O_SVC_Service_Processor is CALLed to execute the I/O request. On return from I/O_SVC_Service_Processor a TQE is obtained from the TQE_Pool_Address and Current_GMT_Routine is CALLed to get the current time. The current time is stored into the TQE_Parent_PCT_Address field and Update_Time is executed to add in the Software_Clock_Offset to the TQE_Time_of_Expiration. Update_Time is executed again to add one additional minute to the TQE_Time_of_Expiration. Finally the TQE_Flags are initialized with the accumulator sync TQE_Type_Indicator.

After handling the selected time source Accumulator_Sync continues by initializing the I/O_Buffer_Address and Operation_Code again. The Event_Address is also zeroed. Next the TQE_Time_of_Expiration is converted to MTU format by CALLing Fixed_To_MTU_Format_Conversion_Routine. Then I/O_SVC_Service_Processor is CALLed to process the I/O request followed by a CALL to Timer_Queue_Generator to enqueue the TQE.
The control flow for this module is shown in figure 3.1.2.6-1.

d. Output - See table 3.1.2.6-1. A MTU I/O request and a TQE for the update request.

e. Module References -

1. (140) Timer_Queue_Generator (FPMTMENQ) is CALLed.

2. (160) Convert_To_Fixed_Point_Routine (FPMCVTFX) is CALLed.

3. (162) Current_GMT_Routine (FPMGMTIM) is CALLed.

4. (164) Fixed_To_MTU_Format_Conversion_Routine (FPMFXMTU) is CALLed.

5. (165) MTU_To_Fixed_Format_Conversion_Routine (FPMMTUFX) is CALLed.

6. (200) I/O_SVC_Service_Processor (FIOSVC) is CALLed. Each time this routine is called it is entered at a secondary entry point FIOSVC1.

7. (363) SVC_Synchronization_Processor (FCMSSYNC) is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation -

1. If no bus was available, accumulator 1 is assumed. The I/O requests will be issued by MTU_Update_Processor but no I/O will be started by I/O management.

DATA TABLE 3.1.2.6-1

NAME MTU_Update_Processor

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	MTU_Update_Parameter_List	S031				TFMTUSPL			
2	MTU_Update_Request_Type	S031.1	I		148	TMTURSVC			
3	MTU_Update_Completion_Event_Address	S031.3	I		148	TMTUEVNT			
4	MTU_Update_Coincident_Time	S031.5	I		148	TMTUCTIM			
5	MTU_Update_Absolute_Time	S031.6	I		148	TMTUUTIM			
6	MTU_Delta_Update_Time	S031.7	I		148	TMTUDTIM			
7	I/O_SVC_Parameter_List	S024				TFIOS			
8	Device_Id_Number	S024.17	O	148	200	TIOSDVID			
9	Operation_Code	S024.18	O	148	200	TIOSOPCD			
10	I/O_Buffer_Address	S024.21	O	148	200	TIOSBUFA			
11	Event_Address	S024.22	O	148	200	TIOSEVNT			
12	Timer_Queue_Element	Q005				TFTQE			
13	TQE_Parent_PCT_Address	Q005.2	O	See Appendix E		TTQEPCT			
14	TQE_Time_of_Expiration	Q005.3	O	140,142, 148,167	See App. E	TTQETOXH			
						TTQETOXM			
15	Time_of_Expiration_Half_Hour	Q005.4	O	140,142, 148,167	See App. E	TTQETOXH			
16	Time_of_Expiration_Half_Hour_	Q005.5	O	140,142, 148,167	See App. E	TTQETOXM			
17	TQE_Flags	Q005.6	O	104,107, 140,142	140,143, 148,170	TTQEFLGS			

DATA TABLE 3.1.2.6-1 (Cont'd)

NAME MTU_Update_Processor

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
18	TQE_Type_Indicator	Q005.10	O	101,104, 107	140,142, 148,174				
19	IOP_Receiver_State	I010.08	I	355,355	See App. E	TGSTTCCR			
20	Bus_Masks	I010.09	I	355	See App. E	TGSTBMSK	See Appendix E	x	x
21	Data_Path_Masks	I010.11	I	244,290, 355	148,290, 665	TGSTDPM	See Appendix E	x	
22	GST_Status_Flags	I010.14	I	See Appendix E		TGSTIND		x	x
23	MTU_Update_Time	#022	O	148	140,142				
24	MTU_Update_Half_Hour_Portion	#022.1	O	148	140,142	TFCMMTUH			
25	MTU_Update_Half_Hour_Count	#022.2	O	148	140,142	TFCMMTUM			
26	Software_Clock_Offset	#023	I	148	140,148	TFCMOFST			
27	TQE_Free_Pool_Address	Q001.11	I	142,143, 166,305	See App. E	TCVTTQEP			

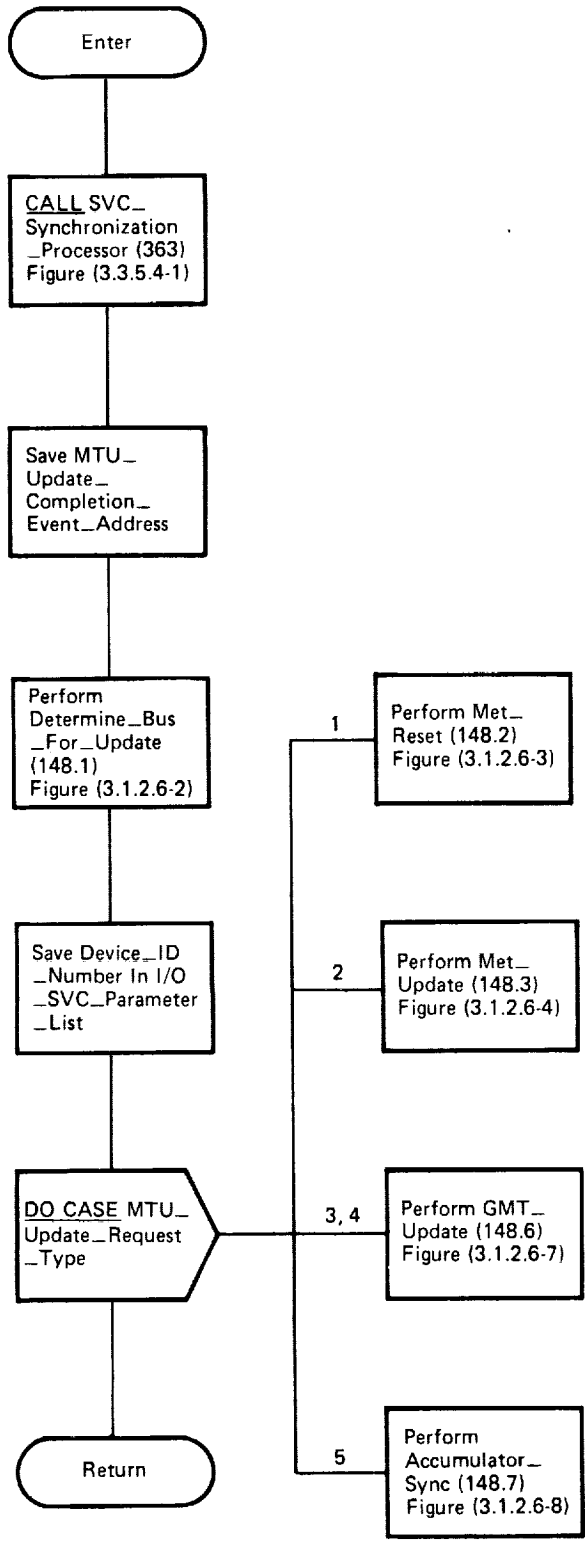


Figure 3.1.2.6-1. MTU_Update_Processor (FPMUPMTU)

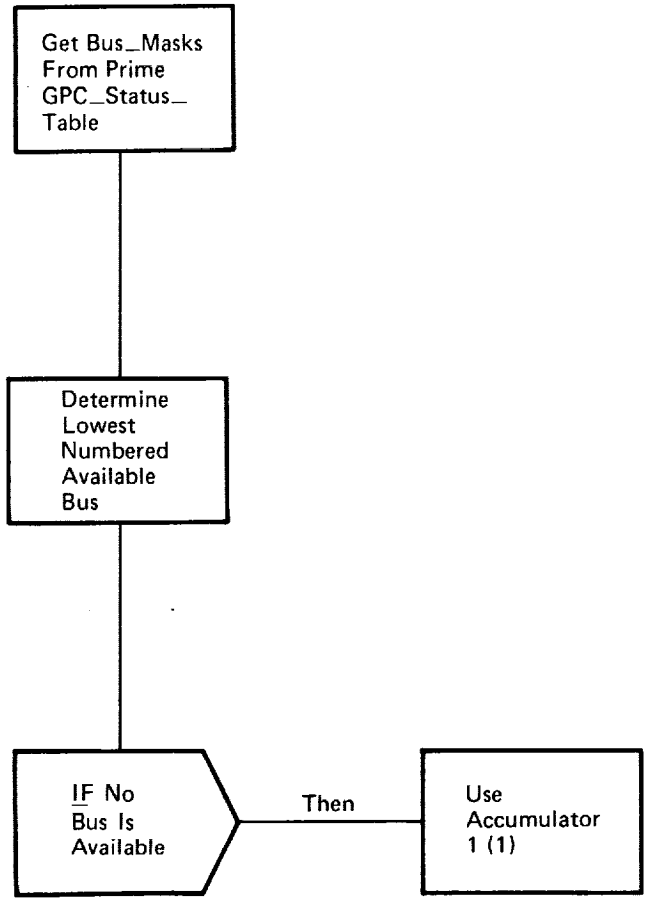


Figure 3.1.2.6-2. MTU_Update_Processor Determine_Bus_For_Update (148.1)

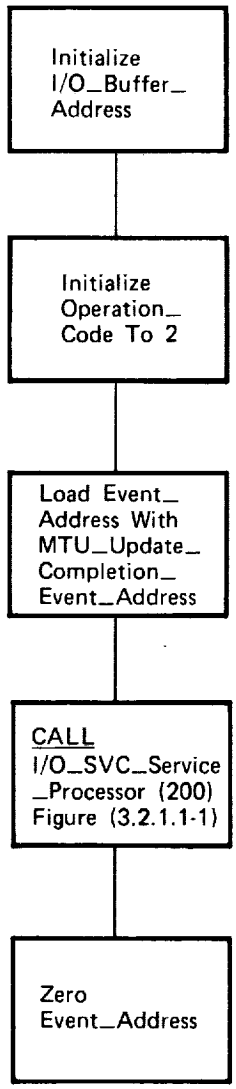


Figure 3.1.2.6-3. MTU_Update_Processor MET_Reset (148.2)

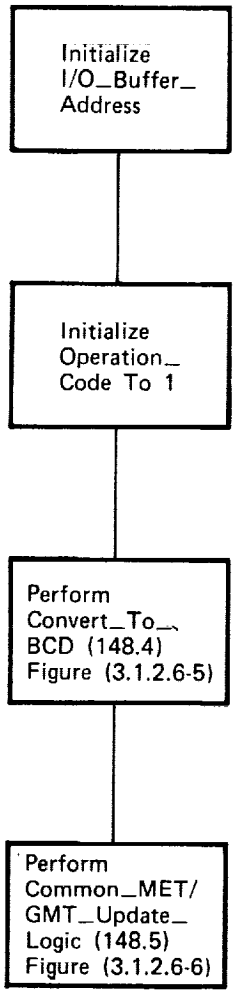


Figure 3.1.2.6-4. MTU_Update_Processor MET_Update (148.3)

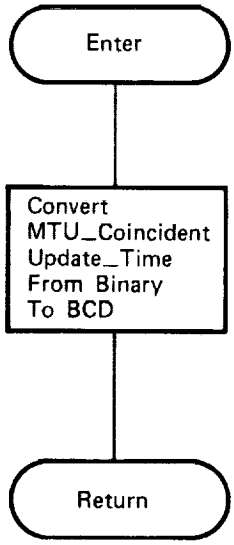


Figure 3.1.2.6-5. MTU_Update_Processor Convert_To_BCD (148.4)

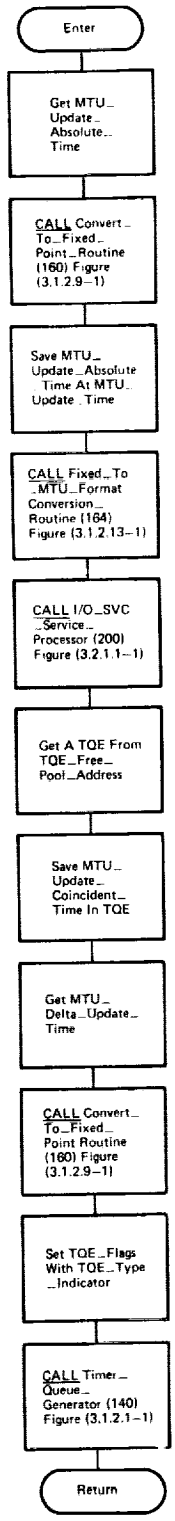


Figure 3.1.2.6-6. MTU_Update_Processor Common_Met/GMT_Update_Logic (148.5)

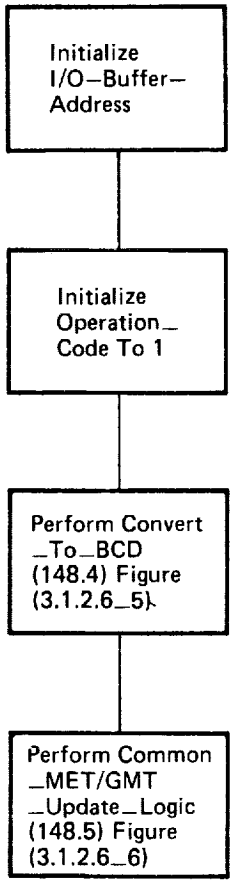


Figure 3.1.2.6-7. MTU_Update_Processor GMT_Update (148.6)

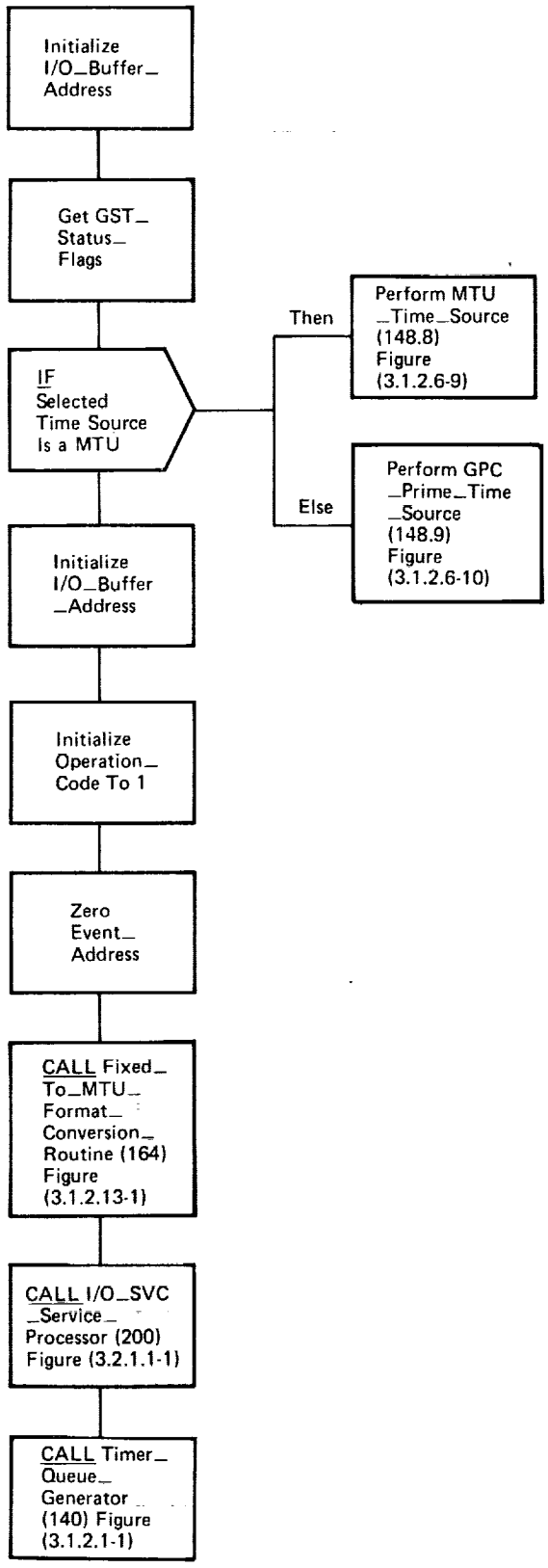


Figure 3.1.2.6-8. MTU_Update_Processor Accumulator_Sync (148.7)

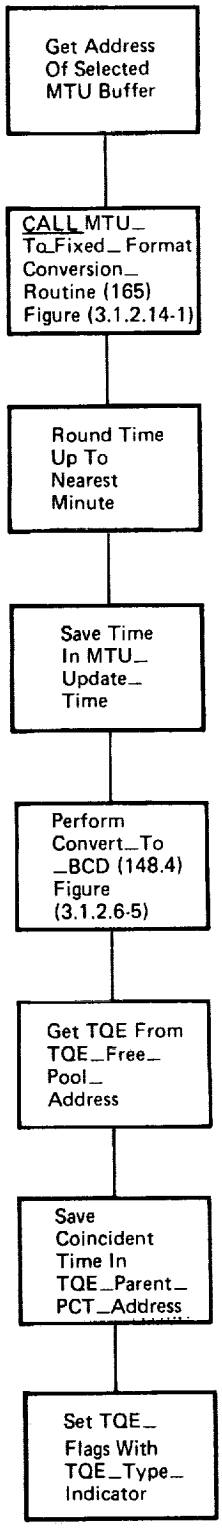


Figure 3.1.2.6-9. MTU_Update_Processor MTU_Time_Source (148.8)

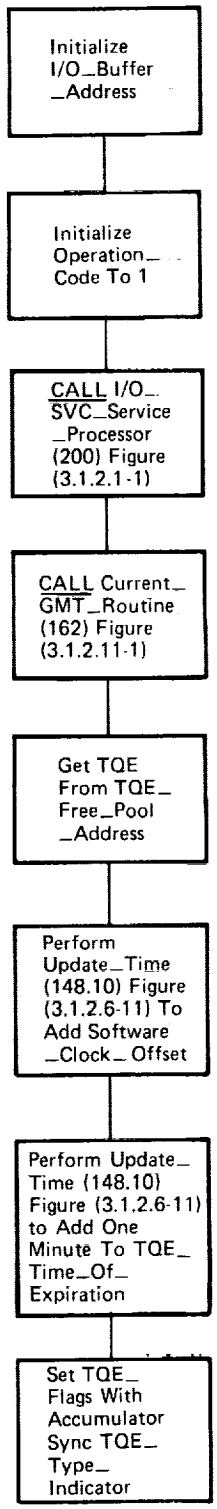


Figure 3.1.2.6-10. MTU_Update_Processor GPC_Prime_Time_Source (148.9)

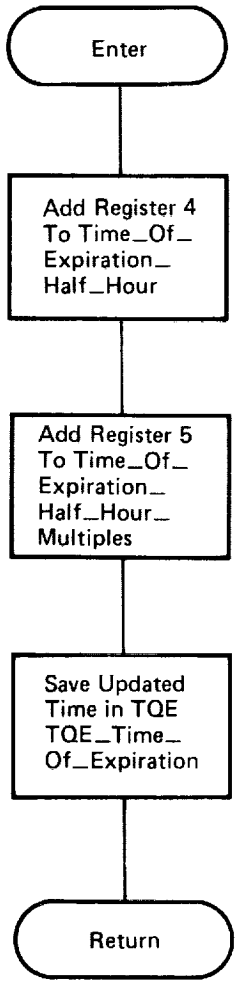


Figure 3.1.2.6-11. MTU_Update_Processor Update_Time (148.10)

3.1.2.7 MTU_Redundancy_Manager (FPMMTURM)(149)

The MTU_Redunadancy_Manager is responsible for keeping internal GMT synchronized with the MTU or with the prime GPC's internal time if the MTU is unavailable or out of tolerance. The MTU_Redundancy_Manager also supports internal GMT (RUNTIME) initialization, and time tagging of applications data.

a. Control Interface -

1. CALLed by (100) SVC_Handler (FPMSVC)
2. CALLed by (220) I/O_Completion_Processor (FIOCMPLT)

b. Input -

Bits 0-15 of general purpose register 0 contain the address of the SVC parameter list.

See Table 3.1.2.7-1.

c. Process Description - There are two entry points for the MTU_Redundancy_Manager. The second entry point, FPMMTUR1, bypasses a CALL to SVC_Synchronization_Processor. The MTU_RM_Request_Type is obtained and the case macro is executed.

For a MTU_RM_Request_Type of 1 the first GPC is initialized by Initialize_First_GPC. MTU_Comfault1_Word2 is obtained to determine the lowest valid MTU accumulator. If there is a valid MTU, then MTU_To_Fixed_Format_Conversion_Routine is CALLed to convert MTU time to fixed point format. The Bias time is added to the converted time and saved at Last_Expired_TQE_Time. PC1 is read and the amount of time PC1 has decremented is saved in three places, GPC_Software_Clock, ICC_Prime_Time, and MET_Reference_Time.

For a MTU_RM_Request_Type of 2 a secondary GPC is initialized by performing Initialize_Secondary_GPC. ICC_Prime_Time of the prime GPC is obtained and ICC_Prime_Time of the secondary GPC is subtracted and saved in the secondary GPC_Software_Clock. Next Current_GMT_Routine is CALLed to obtain the current time and saved in Last_Expired_TQE_TIME. Get the selected time source in from the primary GPC and save that time source in the secondary GPC_Status_ Flags. Finally Prime_GPC_Time_Milliseconds and Prime_GPC_Time_Half_Hours is obtained and saved in MET_Reference_Time.

For a MTU_RM_Request_Type of 3 Time_Management_Processing is performed. First, MTU_1_Bite_Word, MTU_2_Bite_Word, and MTU_3_Bite_Word are saved at MTU1_BTU_Bite, MTU2_BTU_Bite, and MTU3_BTU_Bite. Then the GST_Status_Flags are checked to see if a new time source has been selected.

If a new time source has been selected, the ICC update flag is set and the new time source is set in the GST_Status_Flags. If a new time source was not selected, then zero all bits except the selected time source in GST_Status_Flags.

If the selected time source is a MTU then perform Find_Valid_MTU. Find_Valid_MTU loops through the three MTUs or until a valid MTU is found. Each MTU is checked for an I/O error. If all MTUs had I/O errors, then save that status in MTU_Accumulator_GO/NO_GO_Status word. If a valid MTU is found then CALL MTU_To_Fixed_Format_Conversion_ Routine converts MTU time to fixed point format and Compute_Delta is performed to compute the difference between the MTU time and ICC_Prime_ Time. If this difference is less than or equal to K1 (580 μsec), then the MTU is considered good and this status is saved in the MTU_Accumulator_ Go/No_GO_Status word. If all MTUs fail the K1 and I/O error tests, then force select GPC prime time. Zero GPC prime time indicator in the MTU_ Accumulator_Go/No_Go_ Status word.

If the time source is GPC prime time, then get ICC_Prime_Time for the prime and current GPC and perform Compute_Delta. If the computed delta is greater than K1, then set the time source to GPC my time and set MTU_ Accumulator_Go/No_Go_Status to show GPC prime time failed.

Compute and save the Software_Clock_Offset. Add the Software_Clock_ Offset to the GPC_Software_Clock.

If a new time source has been selected, then save it in GST_Status_Flags. Set ICC update flag and annunication flag to indicate an automatic time source selection.

For a MTU_RM_Request_Type of 4 Time_Tagging is performed. Time is obtained from Last_Expired_TQE_Time and Convert_To_Floating_Point_ Routine is CALLed to convert time to floating point format. The time in floating point is then saved 8 halfwords before the Address_of_I/O_ Buffer. Also the time source ID is stored into the transaction status word.

d. Output - See Table 3.1.2.7-1. The following outputs are possible:

1. RUNTIME maybe initialized.
2. An application time tag.
3. A new OFST to be used in computing RUNTIME.
4. A change in time source selection.
5. An error message if the source selection changed.

e. Module References -

1. (161) Convert_To_Floating_Point_Routine is CALLed.
2. (162) Current_GMT_Routine is CALLed.
3. (165) MTU_To_Fixed_Format_Conversion_Routine is CALLed.
4. (174) Event_Evaluator is CALLed.
5. (363) SVC_Synchronization_Processor is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation -

1. SWC = Software Clock
   GPC_{PT} = GPC prime time
   GPC_{MY} = GPC my time

2. Δ = delta computed by Compute_Delta
   K1 = Bais time (580 μ sec)

DATA TABLE 3.1.2.7-1

NAME MTU_Redundancy_Management (FPMMTURM)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	GST_Status_Flags	I010.14	O	See Appendix 'E'		TGSTIND		x	x
2	OPS_Status_Flags	I040	I	700,660, 910,950	See App. E	CZ2BSTAT			
3	GPC_Prime_ID	I050	I	351,700, 880,820	See App. E	CZ2VGPCP			
4	MTU1_BTU__Bite	I100.16	O	149, 361	660, 665	CZ2BTMT1			
				820, 880					
5	MTU2_BTU_Bite	I100.17	O	149, 361		CZ2BTMT2			
				820, 880					
6	MTU3_BTU_Bite	I100.18	O	149,361, 820,880		CZ2BTMT3			
7	Prime_GPC_Time_Microseconds	I250	O	142, 149		CZ2VMETM			
				205					
8	Prime_GPC_Time_Half_Hours	I260	O	149,142, 205		CZ2VMETH			
9	ICC_Prime_Time	I610.06	I	205	149,142				
10	Internal_Time_Microseconds	I610.10	I	149	142,660	TICCGMTH			
					665,955				
11	Internal_Time_Half_Hours	I610.14	I	149	142,149	TICCGMTM			
					660, 665				
12	ICC_Status_Flags	I320	O	See Appendix E		CZ2VIF1			

DATA TABLE 3.1.2.7-1 (cont'd)

NAME MTU_Redundancy_Management (FPMMTURM)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
13	MET_Reference_Time	Q001.89	I	142,149, 205,305	144	TCVTMRTH TCVTMRTH			
14	MET_Reference_30_Minute_Portion	Q001.90	O	142,149, 205,305	144,205	TCVTMRTH			
15	MET_Reference_Half_Hour_Counter	Q001.91	O	142,149, 205,305	142,144, 149,205	TCVTMRTM			
16	GPC_Software_Clock	Q001.86	I	141,142, 149,305	149,162, 163	TCVTSWCH TCVTSWCM			
17	Software_Clock_30_Minute_Portion	Q001.87	O	141,142, 149,305	149,162, 242	TCVTSWCH			
18	Software_Clock_Half_Hour_Counter	Q001.88	O	141,142, 149,305	149,162, 242	TCVTSWCM			
19	Address_of_I/O_Buffer	Q007.5	I	200	149,205	TIOQBUFA			
20	GST_Address_Table	@004	I	368	See App. E	FPMGSTAD			
21	ICC_Buffer_Address_Table	0005	I		142, 149	FCMICCAD			
22	MTU_RM_Parameter_List	S032	I		149	TFSVCPL			
23	MTU_RM_Request_Type	S032.1	I		149	TSVCRTSN			
24	MTU_RM_Completion_Event	S032.3	I		149	TSVCEVNT			
25	MTU_Accumulator_Go/No_Go_Status	#013	O	149		TFCMMASB			
26	MTU_1_Bite_Word	#017.3	I	149	149, 955	TFCMM1BW	V75M3040P	x	x
27	MTU_2_Bite_Word	#018.3	I		149	TFCMM2BW	V75M3140P	x	x
28	MTU_3_Bite_Word	#019.3	I		149	TFCMM3BW	V75M3240P	x	x
29	Last_Expired_TQE_Time	#025	I		140, 142		V91W1999C		
					144, 149				

DATA TABLE 3.1.2.7-1 (cont'd)

NAME MTU_Redundancy_Manager (FPMMTURM)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
30	Last_TQE_Half_Hour_Portion	#025.1	O	142, 149	140, 144	TFCMLTQH	V91W1999C	x	
					246				
31	Last_TQE_Half_Hour_Count	#025.2	O	142, 149	140,	TFCMLTQM	V91W1999C	x	
					144				
32	MTU_Comfault 1_Word 2	#031	I	205	149	FIOBCSI2	V91M2303P	x	
33	Software_Clock_Offset	#023	O	148	148,	TFCMOFST			
					140				

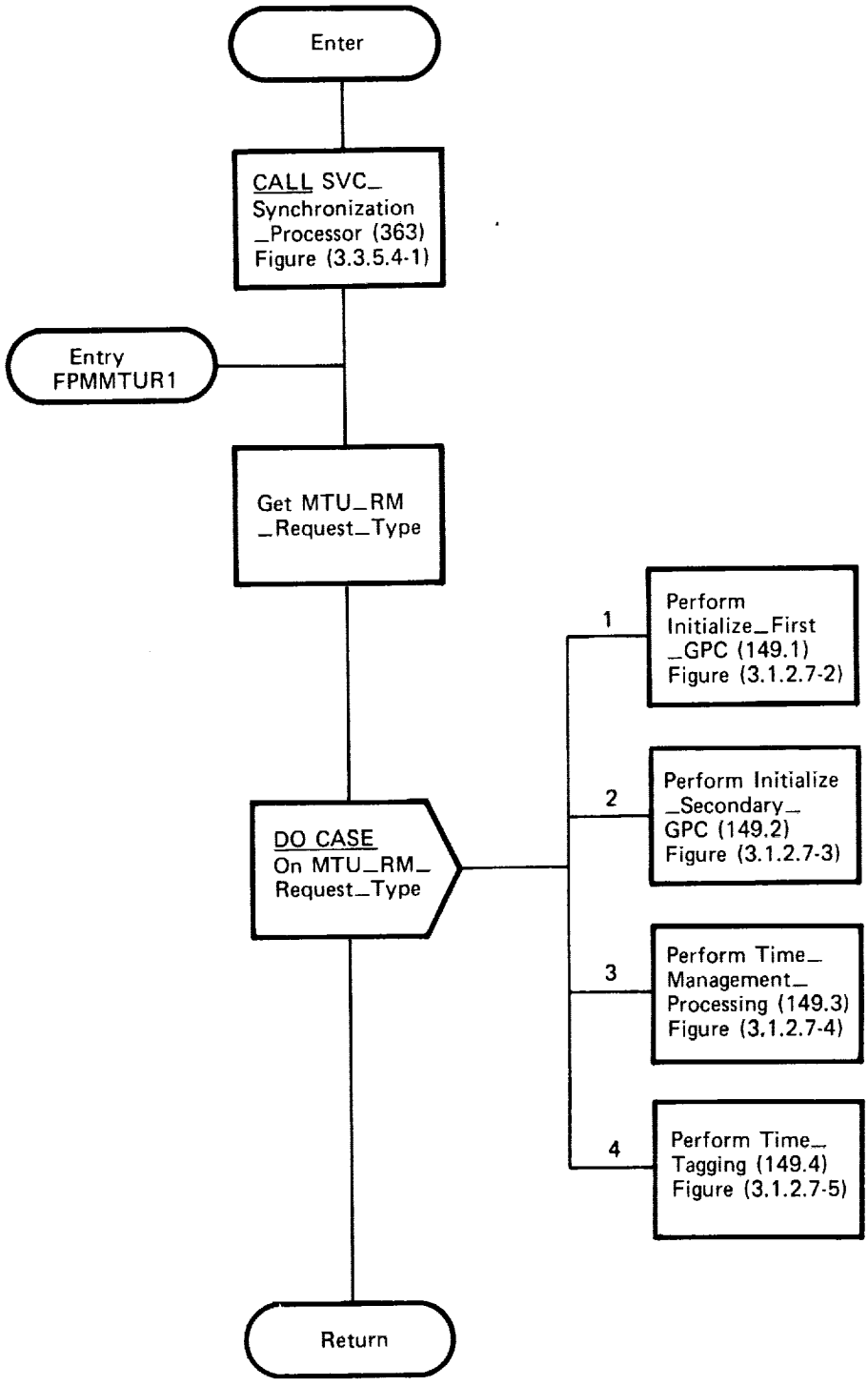


Figure 3.1.2.7-1. MTU_Redundancy_Manager (149.0)

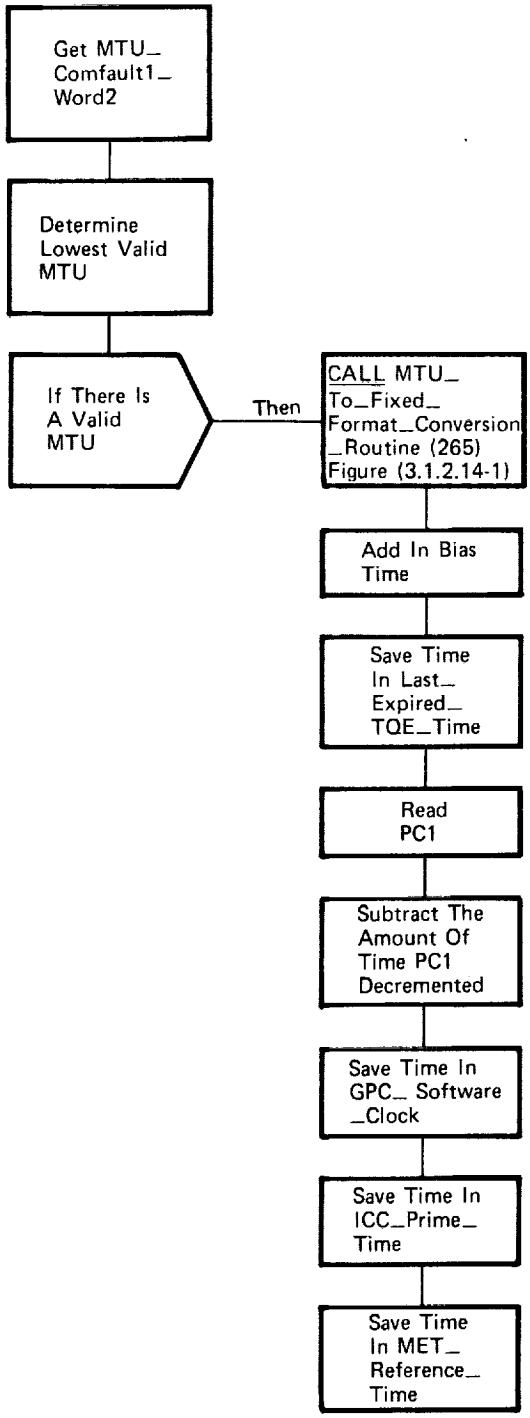


Figure 3.1.2.7-2. MTU_Redundancy_Manager Initialize_First_GPC (149.1)

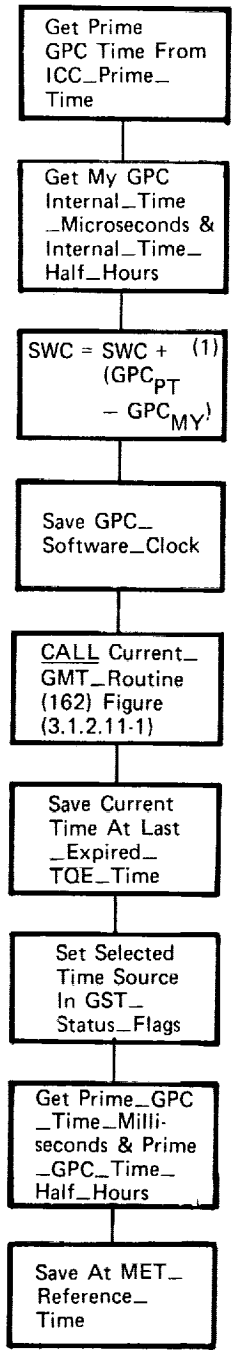


Figure 3.1.2.7-3. MTU_Redundancy_Manager Initialize_Secondary_GPC (149.2)

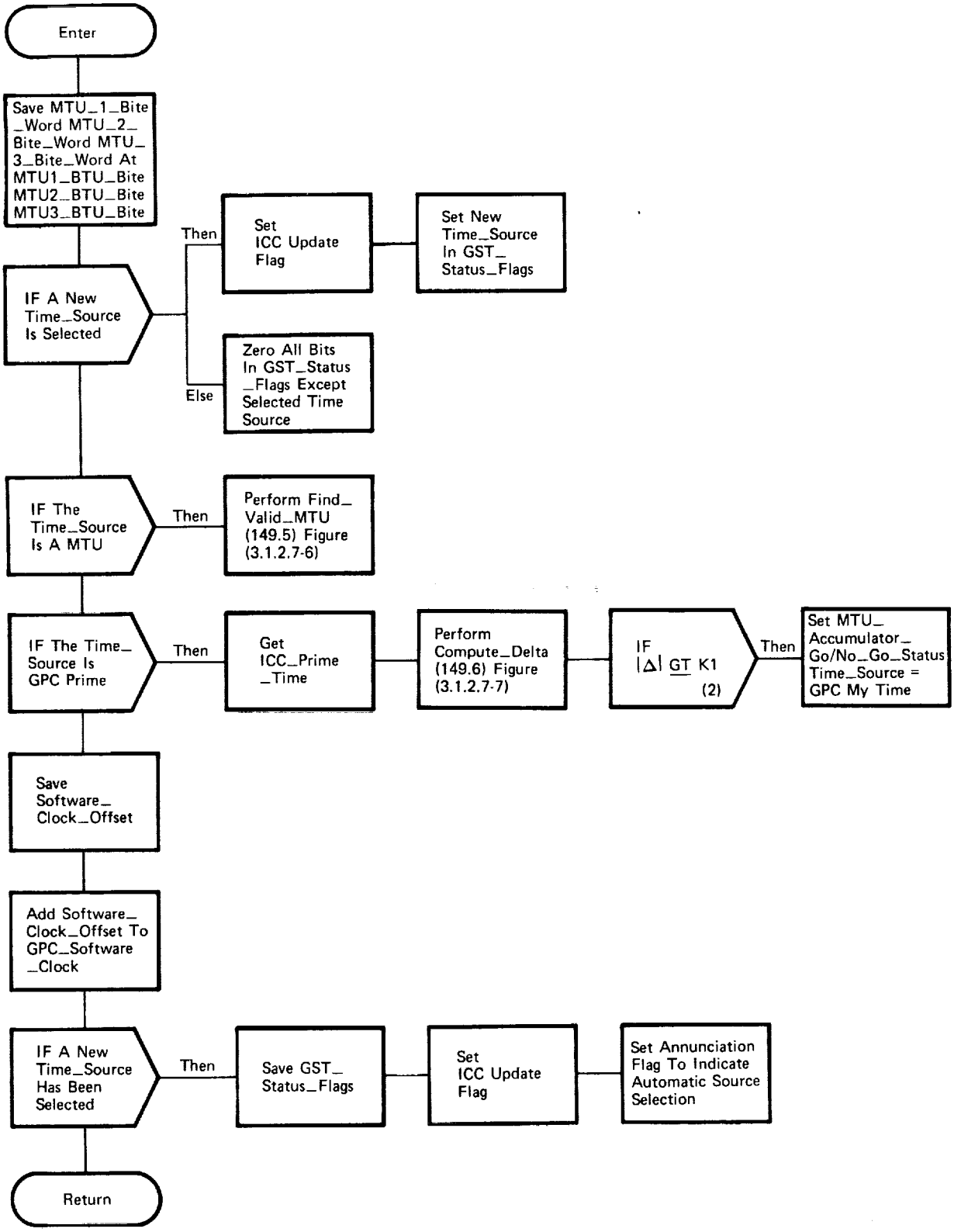


Figure 3.1.2.7-4. MTU_Redundancy_Manager Time_Management_Processing (149.3)

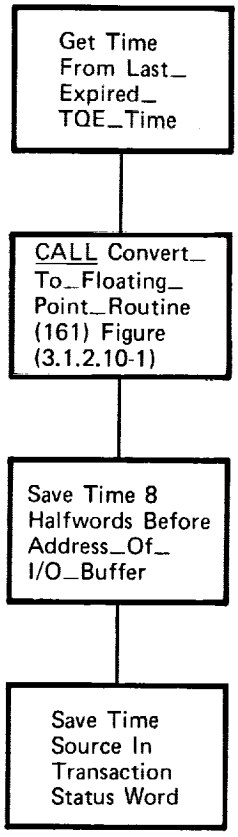


Figure 3.1.2.7-5. MTU_Redundancy_Manager Time_Tagging (149.4)

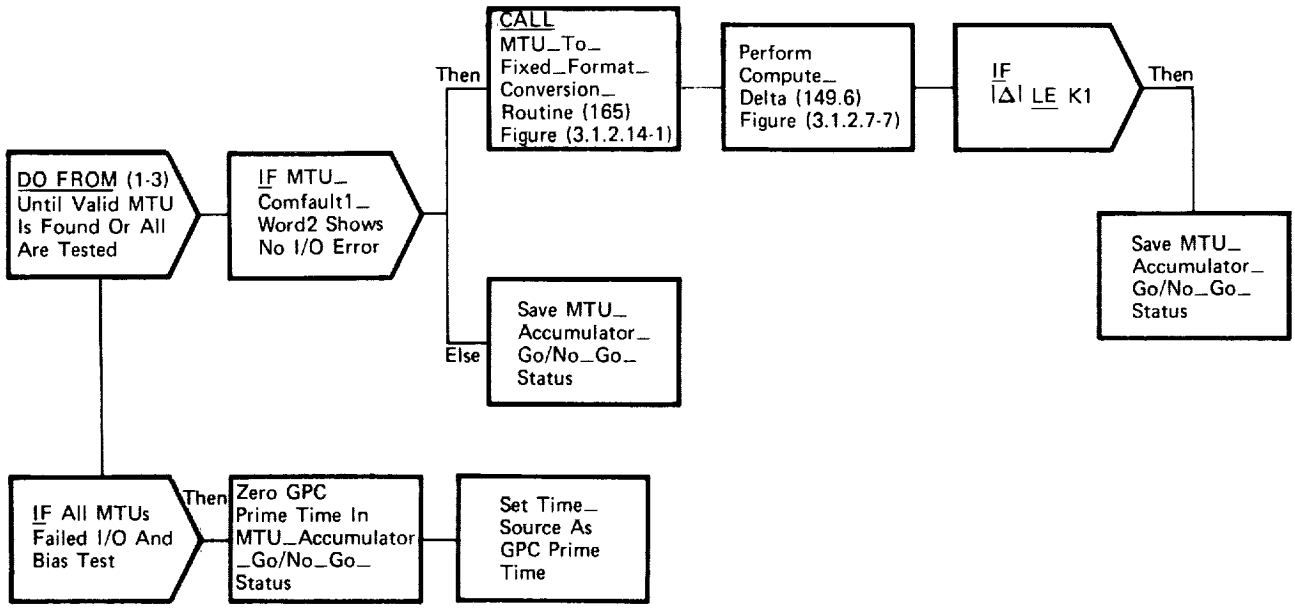


Figure 3.1.2.7-6. MTU_Redundancy_Manager Find_Valid_MTU (149.5)

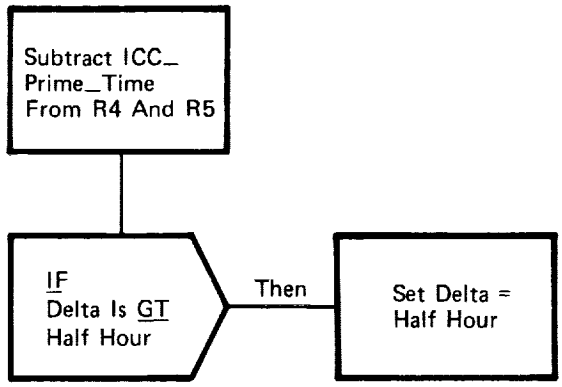


Figure 3.1.2.7-7. MTU_Redundancy_Manager Compute_Delta (149.6)

3.1.2.8 Time_Conversion_SVC_Services (FPMTMCVT) (150)

Time_Conversion_SVC_Services is a time conversion routine which will convert time in MTU format or fixed point format to floating point format and store the converted time at an address specified by the caller.

a. Control Interface

1. CALLed by (100) SVC_Handler (FPMSVC).

b. Input - Register 0 contains the address of the Time_Conversion_parameter_list. See table 3.1.2.8-1.

c. Process Description - Time_Conversion_SVC_Services checks the Time_Conversion_Request_Type.

For a Time_Conversion_Request_Type of 1, time is converted from MTU format to floating point format in the following manner. The Time_Conversion_Input_Address containing the time is obtained, and MTU-To_Fixed_Format_Conversion_Routine is CALLed to convert the time to fixed point. Next, the Convert_To_Floating_Point_Routine is CALLed to convert time to floating point.

For a Time_Conversion_Request_Type of other than 1, time is converted from fixed point to floating point in the following manner. The Time_Conversion_Input_Address containing the time is obtained, and the Convert_To_Floating_Point_Routine is CALLed to convert the time.

The time now in floating point format is now stored at the Time_Conversion_Output_Address specified in the SVC parameter list.

The control flow for this module is shown in figure 3.1.2.8-1.

d. Output - See table 3.1.2.8-1.

e. Module References -

1. (161) Convert_To_Floating_Point_Routine (FPMCVTFL) is CALLed.
2. (165) MTU-To_Fixed_Format_Conversion_Routine (FPMMTUFX) is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.1.2.8-1

NAME TIME_CONVERSION_SVC_SERVICES (FPMTMCVT)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	TIME_CONVERSTION_PARAMETER_LIST	S043	I		144,150				
2	TIME_CONVERSION_REQUEST_TYPE	S043.1	I		144,150				
3	TIME_CONVERSION_SVC_NUMBER	S043.2			100,150				
4	TIME_CONVERSION_INPUT_ADDRESS	S043.3	I		144,150				
5	TIME_CONVERSION_OUTPUT_ADDRESS	S043.4	O		144,150				

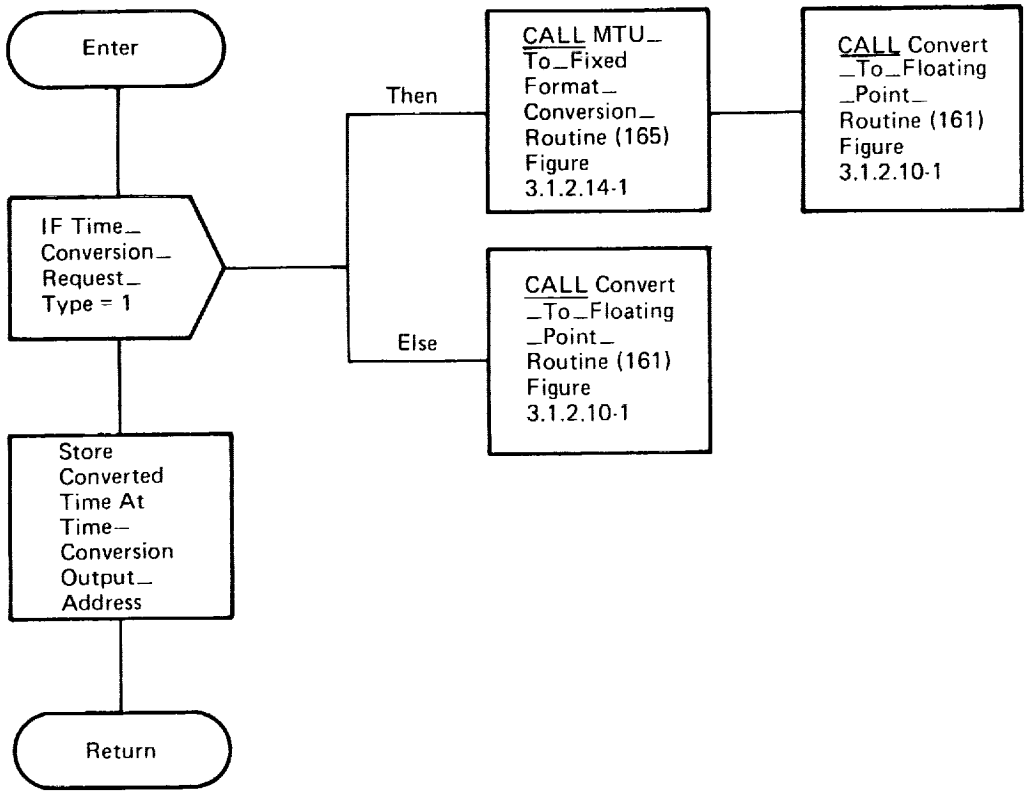


Figure 3.1.2.8-1. Time_Conversion_SVC_Services (FPMTMCVT)

3.1.2.9 Convert_To_Fixed_Point_Routine (FPMCVTFX) (160)

Convert_To_Fixed_Point_Routine converts time from floating point to fixed point format.

a. Control Interface -

1. CALLed by (140) Timer_Queue_Generator (FPMTMENQ).
2. CALLed by (148) MTU_Update_Processor (FPMUPMTU).
3. CALLed by (101) Process_Scheduler (FPMSCHED).

b. Input - Bits 0-15 of register 4 will contain the even register number of the even/odd pair of floating point registers which contain the time to be converted.

c. Process Description - Convert_To_Fixed_Point_Routine first checks register 4 to see which pair of floating point registers contains the time to be converted. If register 4 is invalid then floating point register 0 is zeroed and used. The valid settings for register 4 is 0, 2, 4 and 6.

Next the time is divided by a half hour to determine the number of half hour intervals. The number of half hour intervals is converted to fixed point.

Then the fractional half hour portion is computed and converted to fixed point.

The final output is, register 4 contains the half hour portion in microseconds and register 5 the number of half hour multipules.

The control flow for this module is shown in figure 3.1.2.9-1.

d. Output - General purpose register 4 will contain a fractional portion of the converted time, up to 30 minutes, in microseconds. General purpose register 5 will contain the number of half hour intervals.

e. Module References - None

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - It is the callers responsibility to initialize register 4 and load an even/odd pair of floating point registers prior to calling this routine.

j. Detailed Implementation - None

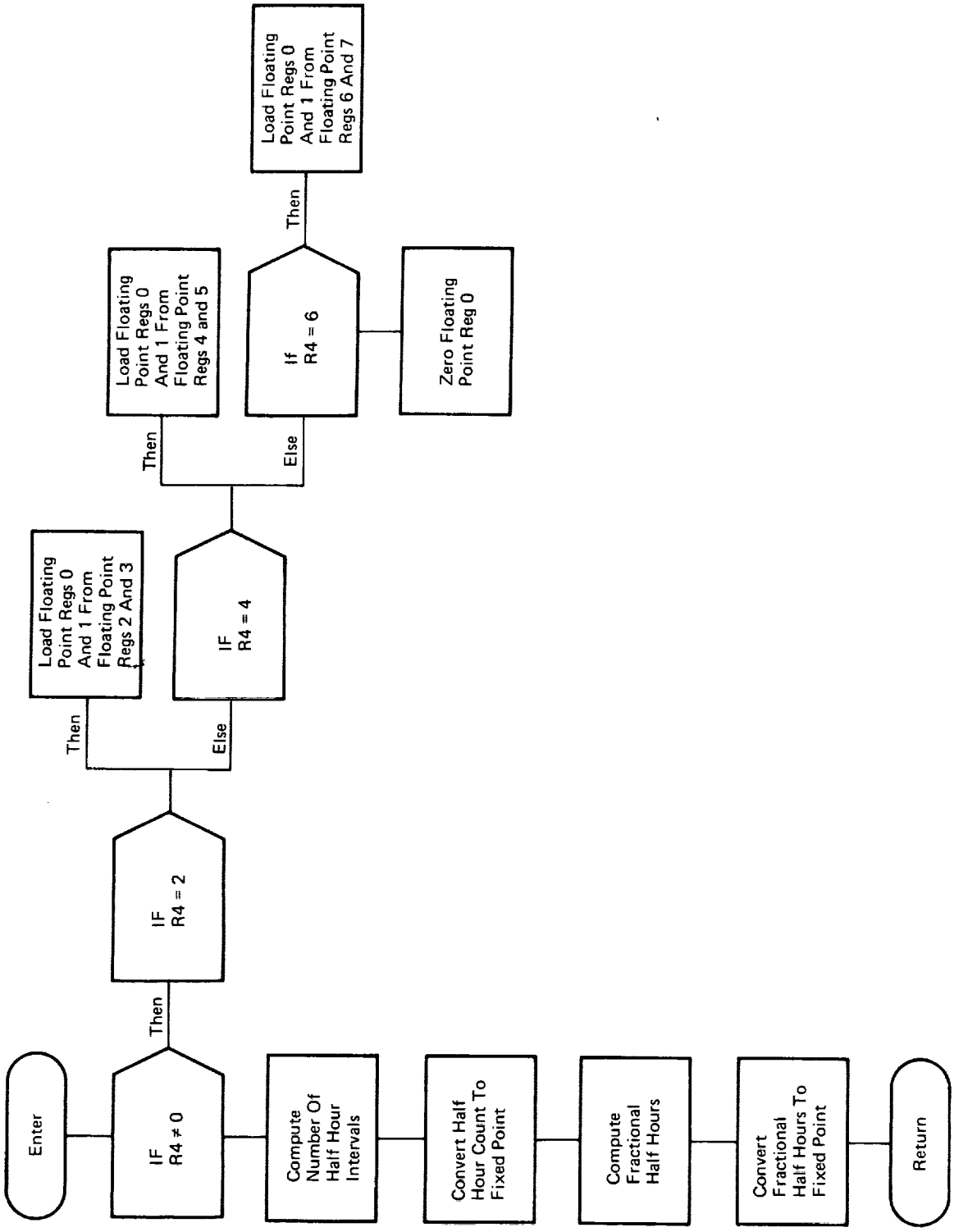


Figure 3.1.2.9-1. Convert_To_Fixed_Point_Routine (FPMCVTFX)

3.1.2.10 Convert_To_Floating_Point_Routine (FPMCVTFL) (161)

Convert_To_Floating_Point_Routine converts time in fixed point format to time in floating point format.

a. Control Interface -

1. CALLed by (140) Timer_Queue_Generator (FPMTMENQ).
2. CALLed by (144) Time/Date_Application_Requests_Processor (FPMTMHAL).
3. CALLed by (149) MTU_Redundancy_Manager (FPMMTURM).
4. CALLed by (150) Time_Conversion_SVC_Services (FPMTMCVT).

b. Input - General registers 4 and 5 contain the time to be converted to floating point format.

Register 4 has up to 30 minutes in microseconds.
Register 5 has the number of half hour multipules.

c. Process Description - Convert half hour multipules in register 5 to seconds. Convert microseconds in register 4 to seconds. Add the two results such that time will be in floating point register 0 in seconds.

The control flow for this module is shown in figure 3.1.2.10-1.

d. Outputs - Time in seconds in floating point register 0.

e. Module References - None

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumpations - General registers 4 and 5 are destroyed and not restored.

j. Detailed Implementation - None

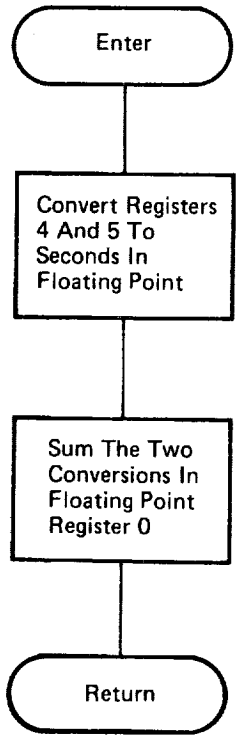


Figure 3.1.2.10-1. Convert_To_Floating_Point_Routine (FPMCVTFL)

3.1.2.11 Current_GMT_Routine (FPMGMTIM) (162)

Current_GMT_Routine computes the current mission elapsed time (MET).

a. Control Interface -

1. CALLed by (140) Timer_Queue_Generator (FPMTMENQ).
2. CALLed by (144) Time/Date_Application_Requests_Processor (FPMTMHAL).
3. CALLed by (148) MTU_Update_Processor (FPMUPMTU).
4. CALLed by (149) MTU_Redundancy_Manager (FPMMTURM).
5. CALLed by (205) IOP Dispatcher (FIOPDISP).
6. CALLed by (183) Process_Error_Logger (FPMERLOG).

b. Input - See table 3.1.2.11-1.

c. Process Description - Current_GMT_Routine reads PC1. The PC1 value is then subtracted from X'1FFFFF', the maximum value for PC1, to get a delta time. This delta time is then added to Software_Clock_30_Minute_Portion. If the Software_Clock_30_Minute_Portion then exceeds a half hour, then subtract a half hour and add one to the Software_Clock_Half_Hour_Counter. Now the current time is in registers 0 and 1 to be used by the caller.

The control flow for this module is shown in figure 3.1.2.11-1.

d. Output - Register 0 and 1 contain the current time. Register 0 has the 30 minute elapsed time in microseconds, and register 1 has the half hour multipules.

e. Module References - None

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.1.2.11-1

NAME Current_GMT_Routine (FPMGMTIM)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Software_Clock_30_Minute_Portion	Q001.87	I	141, 142	163, 162	TCVTSWCH			
				149, 305					
2	Software_Clock_Half_Hour_Counter	Q001.88	I	141, 142	149, 162	TCVTSWCM			
				149, 305	242				

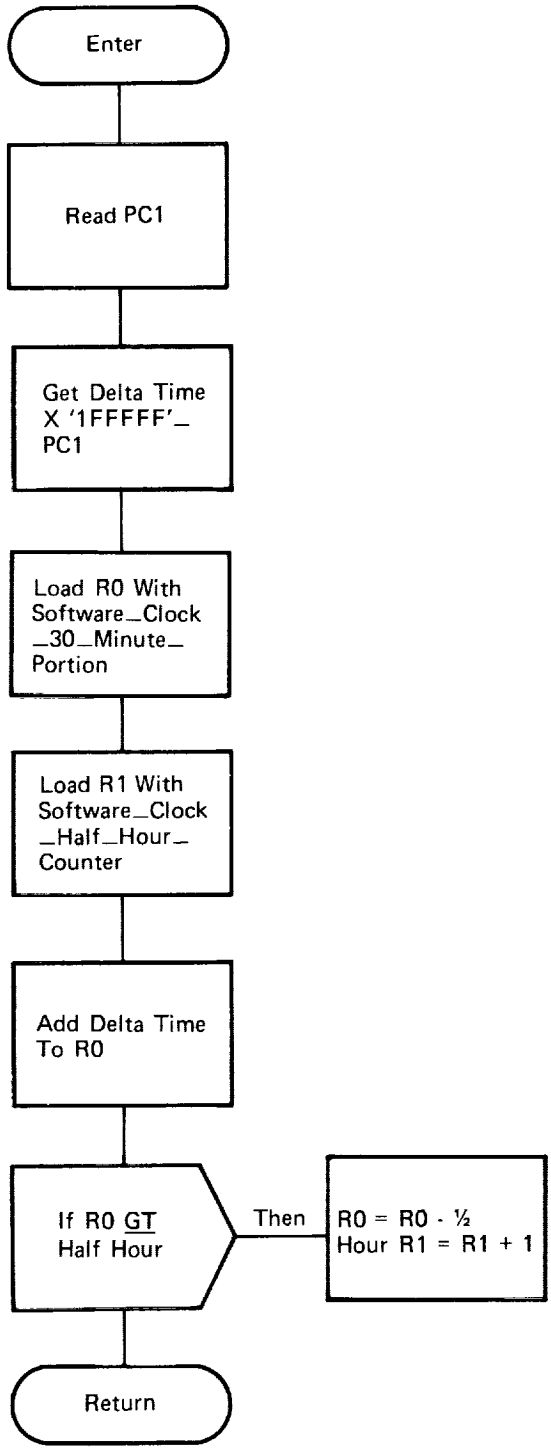


Figure 3.1.2.11-1. Current_GMT_Routine (FPMGMTIM)

3.1.2.12 Program_Counter_2_Update_Routine (FPMITUPD) (163)

Program_Counter_2_Update_Routine computes timer intervals and attempts to update Program Counter 2 (interval timer).

a. Control Interface -

1. CALLed by (140) Timer_Queue_Generator (FPMTMENQ).
2. CALLed by (142) TQE_Expiration_Processor (FPMIHPC2).

b. Input - Bits 0-15 of register 3 contain the address of the top active TQE.

See table 3.1.2.12-1.

c. Process Description - Read Program Counter 1 and compute the PC2 value, time difference, between PC1 and the Time_of_Expiration_Half_Hour of the TQE at Top_TQE_Address. If the PC2 value is greater than 15 minutes, then set PC2 value time to 10 microseconds and the return code to 4. Update PC2 with the computed PC2 value. The Control flow for this module is shown in figure 3.1.2.12-1.

d. Output - PC2 will be updated and buts 16-31 of register 7 will contain one of the following return codes.

0 - PC2 has been updated.
4 - Time has already passed.

e. Module References - None

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.1.2.12-1

NAME Program_Counter_2_Update_Routine (FPMIHPC2)

NO.	ITEM	ID	ACT	SOURCE	DESTINATION	ASSEMBLER NAME	MML	D	C
1	Time_Of_Expiration_Half_Hour	Q005.4	I	140,142, 148,167	See App. E	TTQETOXH			
2	Software_Clock_30_Minute_Portion	Q001.87	I	141, 142,	162, 163	TCVTSWCH			
				149, 305					
3	Top_TQE_Address	Q001.4	I	142,143	140, 142	TCVTTTQE			
				166,305					

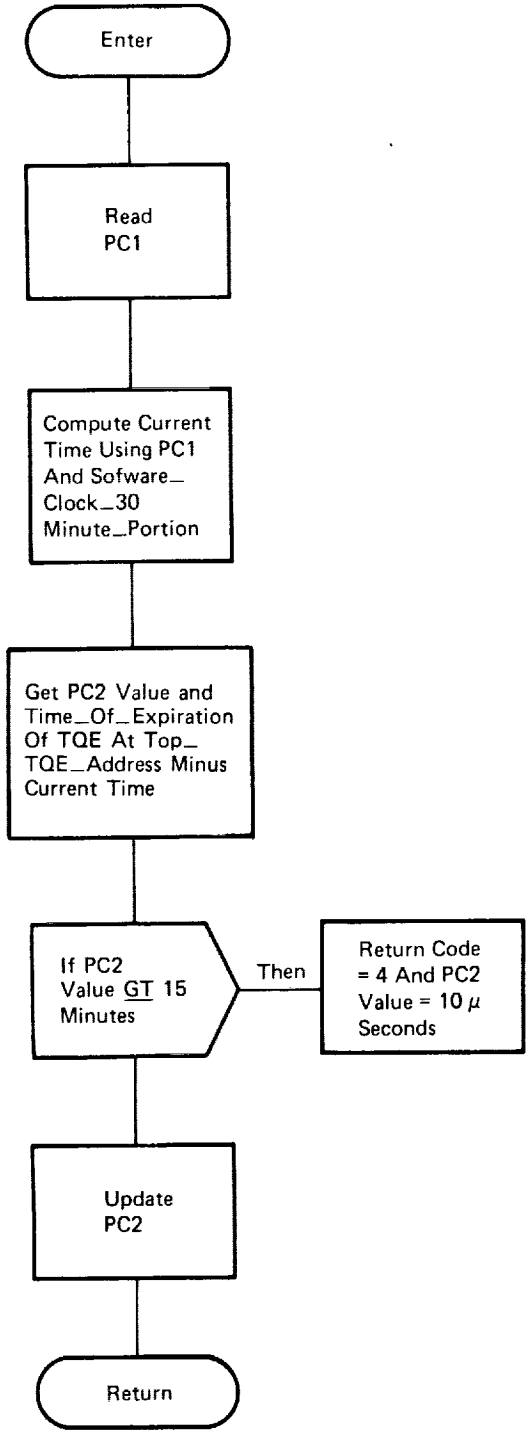


Figure 3.1.2.12-1. Program_Counter_2_Update_Routine (FPMITUPD)

3.1.2.13 Fixed_To_MTU_Format_Conversion_Routine (FPMFXMTU)(164)

Fixed_To_MTU_Format_Conversion_Routine converts time in fixed point to MTU format.

a. Control Interface -

1. CALLed by (148) MTU_Update_Processor (FPMUPMTU)

b. Input - General purpose register 0 contains the address of the location to receive the time in MTU format. General purpose registers 4 and 5 contain the time to be converted.

c. Process Description - Extract the number of days from fixed point time and convert to MTU format. Repeat this procedure for following units of time; hours, minutes, seconds, and milliseconds. After each of three half words of MTU time have been developed they are stored at the location specified by the address in general purpose register 0. For a format of MTU time see data descriptor table @006. The control flow for this module is shown in figure 3.1.2.13-1.

d. Output - Time in MTU format stored at the address passed in general purpose register 0.

e. Module References - None

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

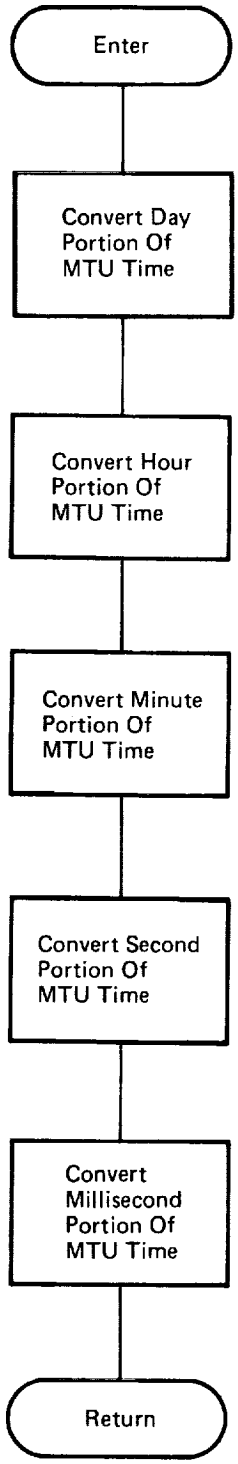


Figure 3.1.2.13-1. Fixed_To_MTU_Format_Conversion_Routine (FPMFXMTU)

3.1.2.14 MTU_To_Fixed_Format_Conversion_Routine (FPMMTUFX)(165)

MTU_To_Fixed_Format_Conversion_Routine converts time in MTU format to fixed point format used by FCOS.

a. Control Interface -

1. CALLed by (148) MTU_Update_Processor (FPMUPMTU)
2. CALLed by (149) MTU_Redundancy_Manager (FPMMTURM)
3. CALLed by (150) Time_Conversion_SVC_Processor (FPMTMCVT)

b. Input - Bits 0-15 of general purpose register 0 contain the address of the time in MTU format. See table 3.1.2.14-1.

c. Process Description - Convert MTU-DAYS and MTU_HOURS to number of half hours. Add number of half hours from MTU_HOURS and MTU_DAYS. Convert MTU_MINUTES to binary minutes. If the number of minutes is greater than 30 then, subtract 30 minutes and add one to the half hour count. Convert binary minutes to microseconds. Convert and add MTU_Milliseconds to microseconds.

The control flow for this module is shown in figure 3.1.2.14-1.

d. Output - On exit from MTU_To_Fixed_Format_Conversion_Routine register 4 will contain the microsecond portion of the converted time and register 5 contains the number of half hour multipules.

e. Module References - None

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.1.2.14-1

NAME MTU_To_Fixed_Format_Conversion_Routine (FPMMTUFX)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	MTU_DAYS_HOURS	@006.1	I	148	148, 165	TMTUDYHR			
2	MTU_DAYS	@006.2	I	148	148, 165				
3	MTU_HOURS	@006.3	I	148	148, 165				
4	MTU_MINUTES_SECONDS	@006.4	I	148	148, 165	TMTUMNSC			
5	MTU_MINUTES	@006.5	I	148	148, 165				
6	MTU_SECONDS	@006.6	I	148	148, 165				
7	MTU_ MILLISECONDS	@006.7	I	148	148, 165	TMTUMSEC			
8	MTU_.125_MILLISECONDS	@006.8	I	148	148, 165				

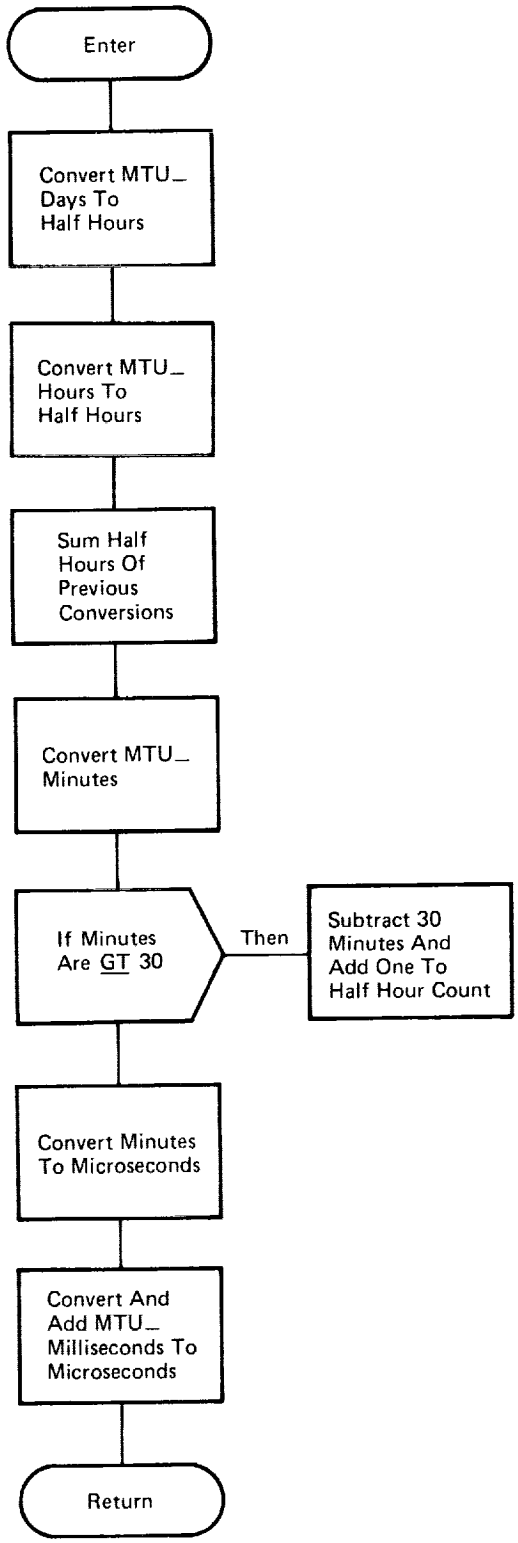


Figure 3.1.2.14-1. MTU_To_Fixed_Format_Conversion_Routine (FPMMTUFX)

3.1.2.15 Chain_TQE_Routine (FPMCHTQE)(166)

Chain_TQE_Routine inserts TQEs into the TQE chain in the proper position.

a. Control Interface -

1. CALLed by (140) Timer_Queue_Generator (FPMTMENQ)

b. Input -

Register 3 contains the address of the New TQE to be added to the chain.

Register 4 contains the microsecond portion of the time of expiration.

Register 5 contains the half hour multipules of the time of expiration.

See table 3.1.2.15-1.

c. Process Description - This module will add a TQE to the TQE chain in the proper place. If the time of expiration of the New TQE is less than that of the first TQE in the chain, then the New TQE will be added as the first TQE in the chain. Otherwise Chain_TQE_Routine will loop through the TQE chain until it finds a TQE whose time of expiration is greater than that of the New TQE. The New TQE will then be removed from the free pool, added to the chain, and the TQE_FREE_POOL_ADDRESS will be updated.

The control flow for this module is shown in figure 3.1.2.15-1.

d. Outputs - A new TQE in the TQE chain.

e. Module References - None

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation -

1. 'New TQE' is the TQE being added to the chain.

DATA TABLE 3.1.2.15-1

NAME CHAIN_TQE_ROUTINE (FPMCHTQE)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Top_TQE_Address	Q001.4	O	142, 143	140, 142	TCVTTTQE			
				166, 305					
2	TQE_Free_Pool_Address	Q001.11	O	142, 143	See App. E	TCVTTQEP			
				166, 305					
3	Next_TQE_Address	Q005.1	O	142,143, 166	142,143, 166	TTQENXT			
4	TQE_Time_Of_Expiration	Q005.3	I	140, 142	See App. E	TTQETOXH			
				148, 167		TTQETOXM			

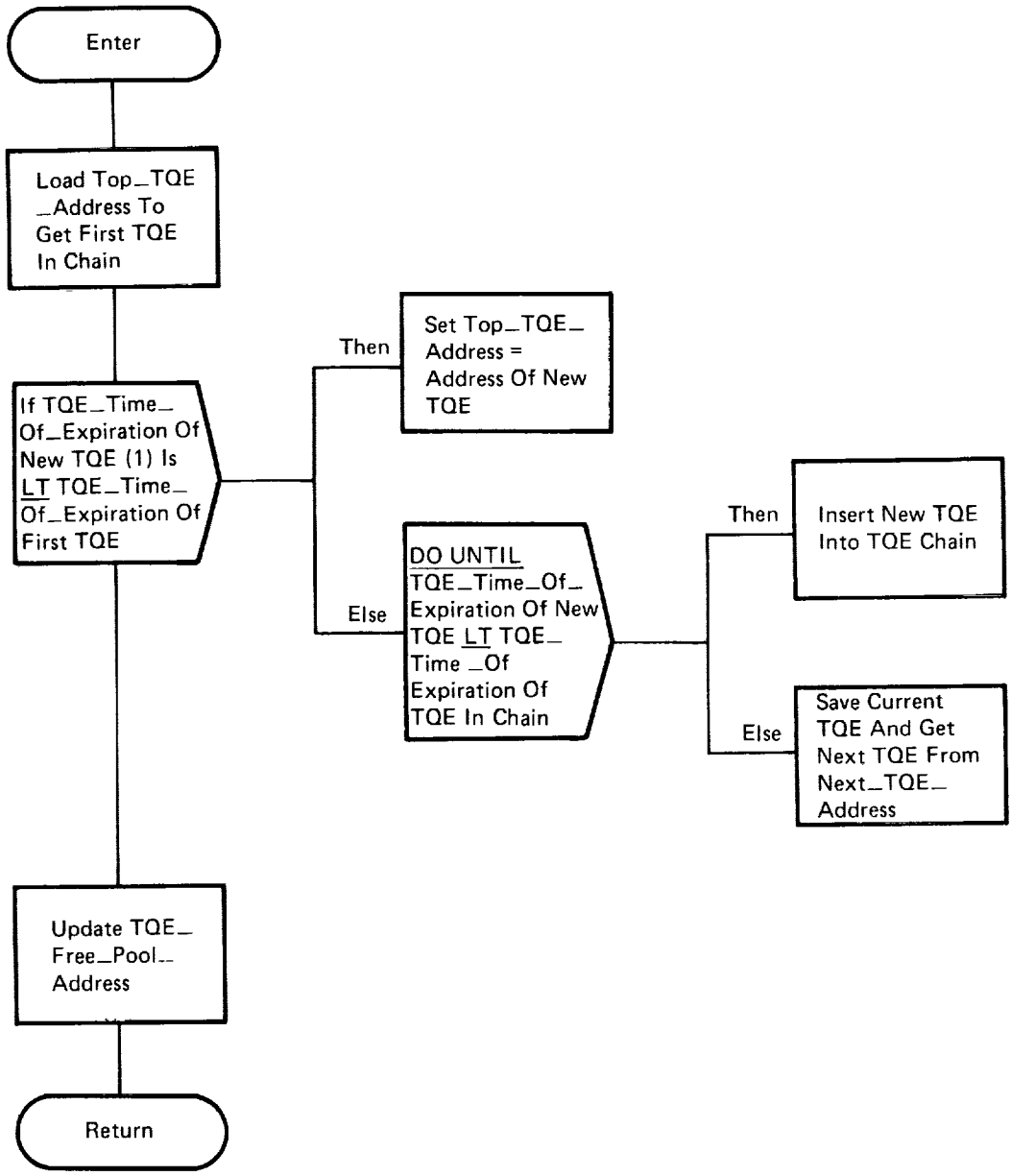


Figure 3.1.2.15-1. Chain_TQE_Routine (FPMCHTQE)

3.1.2.16 Expiration_Time_Update_Routine (FPMUPTOX)(167)

The Expiration_Time_Update_Routine updates the TQE_Time_of_Expiration fields in the Timer_Queue_Element.

a. Control Interface -

1. CALLed by (140) Timer_Queue_Generator (FPMTMENQ)

b. Input -

Register 3 address of Timer_Queue_Element to be updated.
Register 4 half hour portion of update time in microseconds.
Register 5 multipules of half hour update time.

See table 3.1.2.16-1.

c. Process Description - Add update time to TQE_Time_of_Expiration. If the Time_of_Expiration_Half_Hour exceeds 30 minutes then subtract 30 minutes from Time_of_Expiration_Half_Hour and add one to Time_of_Expiration_Half_Hour_Multipules. Store the updated TQE_Time_of_Expiration in the Timer_Queue_Element.

The control flow for this module is shown in figure 3.1.2.2-1.

d. Output - See table 3.1.2.16-1.

e. Module References - None

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.1.2.16-1

NAME Expiration_Time_Update_Routine (FPMUPTOX)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Timer_Queue_Element	Q005	I			TFTQE			
2	TQE_Time_of_Expiration	Q005.3	O	140,142	140,142	TTQETOXH			
				148,167	148,163	TTQETOXM			
					166,167				
3	Time_of_Expiration_Half_Hour	Q005.4	O	Same as	above	TTQETOXH			
4	Time_of_Expiration_Half_Hour_Multipules	Q005.5	O	Same as	above	TTQETOXM			

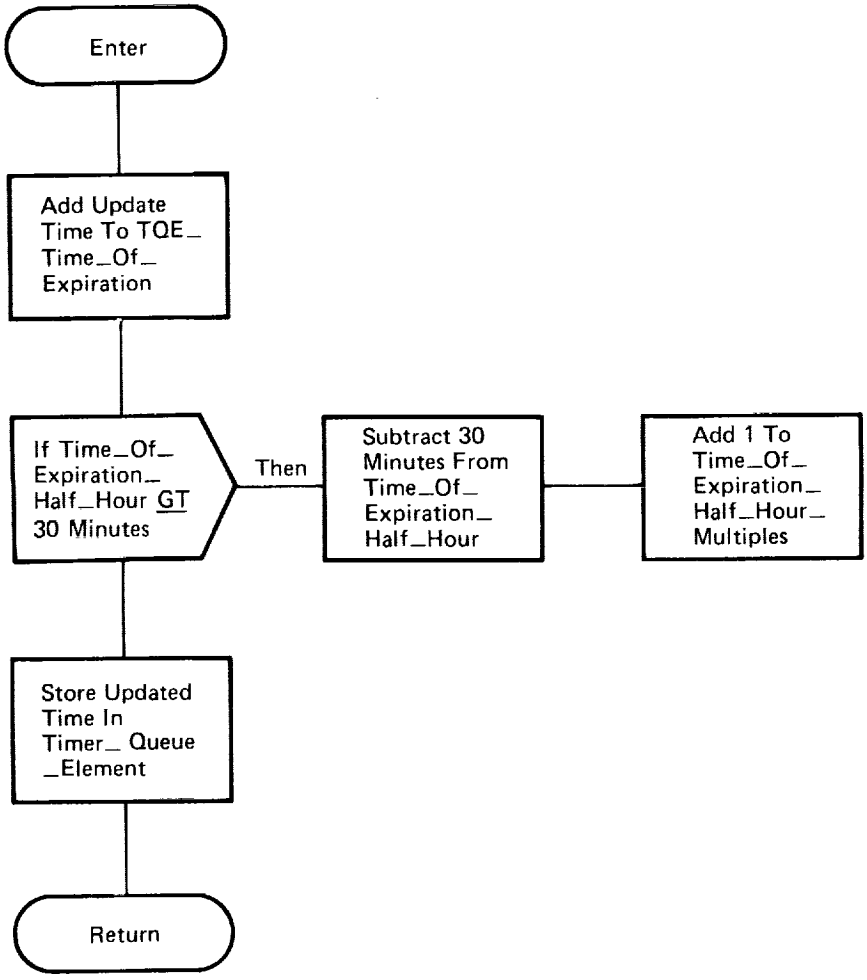


Figure 3.1.2.16-1. Expiration_Time_Update_Routine (FPMUPTOX)

3.1.3 Event Management

FCOS Event Management is responsible for evaluating event conditions used in application process' SCHEDULE and WAIT statements, for retaining the event conditions for future evaluation if they are not immediately satisfied, and for changing event variable states.

Event Management retains event conditions for future evaluation in table elements called Event_Queue_Elements (EQE's). An EQE contains flag bits to indicate the purpose of the EQE, a pointer to the Process_Control_Table (PCT) of the process to which the EQE applies, and the event condition to be evaluated. EQE's which are retained for future evaluation are said to be in the active EQE pool.

However, FCOS only supports the following types of event expressions:

1. Expressions containing a single Event_Variable with/without "NOT"

2. Expressions containing more than one Event_Variable connected by all "OR" operators.

3. Expressions containing more than one Event_Variable connected by all "AND" operators.

There are two types of events, normal events and process events. Normal events are declared and controlled by an application process. Process events are created automatically by the HAL/S compiler for every scheduleable process. A process event is true when the process is in the FCOS process run queue and false at all other times.

Figure 3.1.3-2 shows an interface flow of FCOS Event Management. Whenever a SCHEDULE or WAIT SVC is issued that has event options, the appropriate FCOS processor builds an EQE and calls the Event_Queue_Generator to place the EQE in the active EQE pool. The Event_Queue_Generator first calls the Event_Evaluator to determine if the event condition has been satisfied. If the event condition has been satisfied, the EQE is not added to the active EQE pool. If the event condition has not been satisfied, the EQE is added to the active pool for future evaluation.

The application process can change an event state by issuing a SET, RESET, or SIGNAL SVC, by requesting that an event be SET when an I/O request has completed, or by requesting that an event be SET, RESET, or SIGNALed when an error condition occurs. In each of the cases, the appropriate FCOS processor will perform the requested action and then will call the Event_Evaluator to evaluate any retained EQE which references the affected event variable.

Whenever a process is scheduled, the FCOS Process_Scheduler sets the process event for the process to TRUE and the Event_Evaluator is called to evaluate any retained EQE's which reference the event. Whenever a process is removed from the process run queue via a CLOSE, CANCEL, or TERMINATE SVC, or as a result of SCHEDULE cancellation conditions being met, the process event for the process is reset to FALSE, and the Event_Evaluator is called to evaluate any retained EQE's which reference the affected process event.

Anytime a retained EQE is evaluated and found to be satisfied or the process to which a retained EQE applies is removed from the process run queue, the EQE_Dequeue_Processor is called to remove the EQE from the active EQE pool.

The functional description of the Event Management presented in the following paragraphs is divided into the following areas: (Figure 3.1.3-1)

a. Set_Event_Processor - Sets event to true state.

b. Reset_Event_Processor - Resets event to false state.

c. Signal_Event_Processor - Signals event true.

d. Event_Queue_Generator - The process of adding an EQE to the active EQE pool.

e. Event_Evaluator - Evaluation of EQE's in the active EQE pool.

f. EQE_Dequeue_Processor - Removing a satisfied EQE from the active EQE pool.

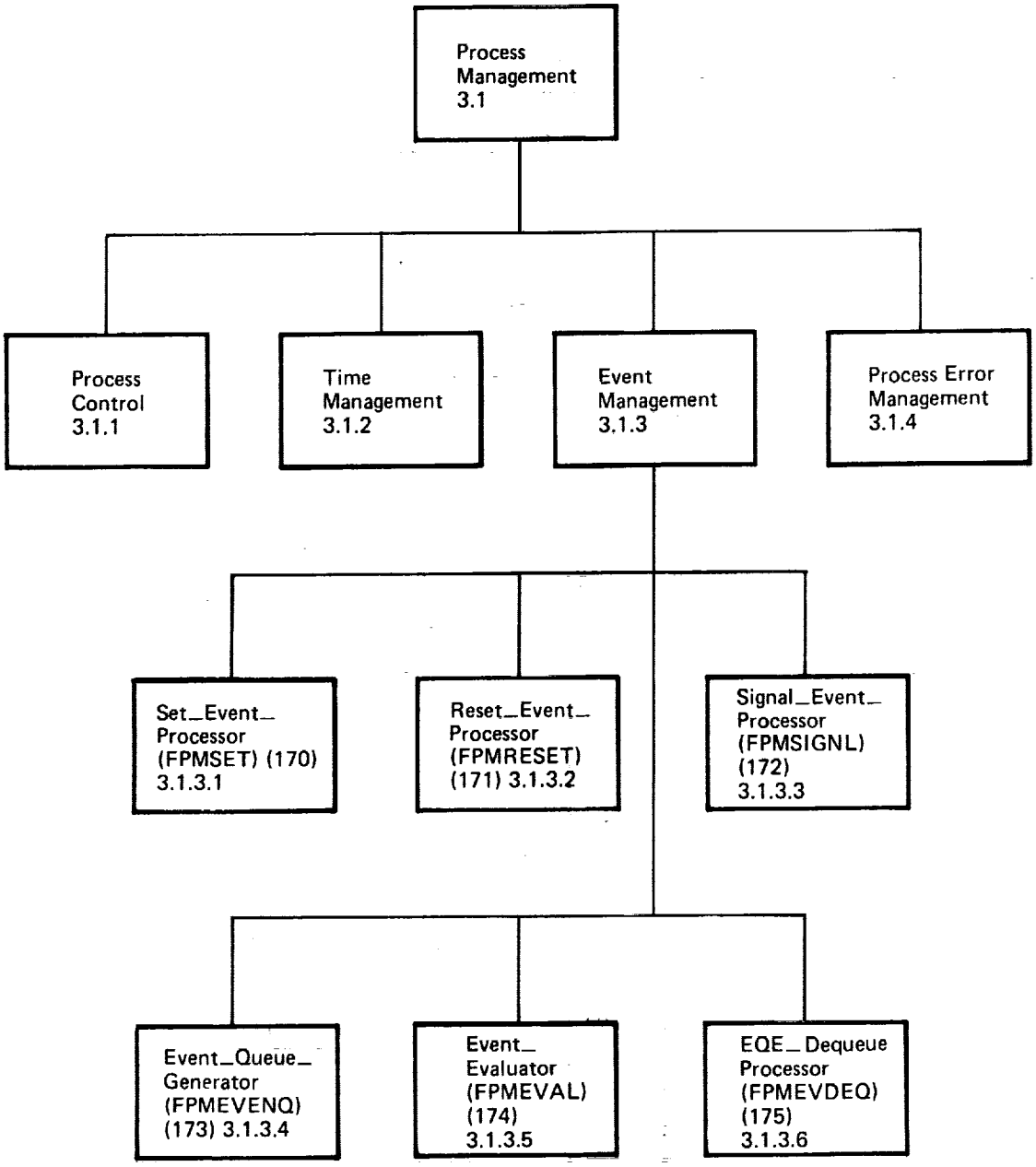


Figure 3.1.3-1. Event Management Hierarchy Diagram

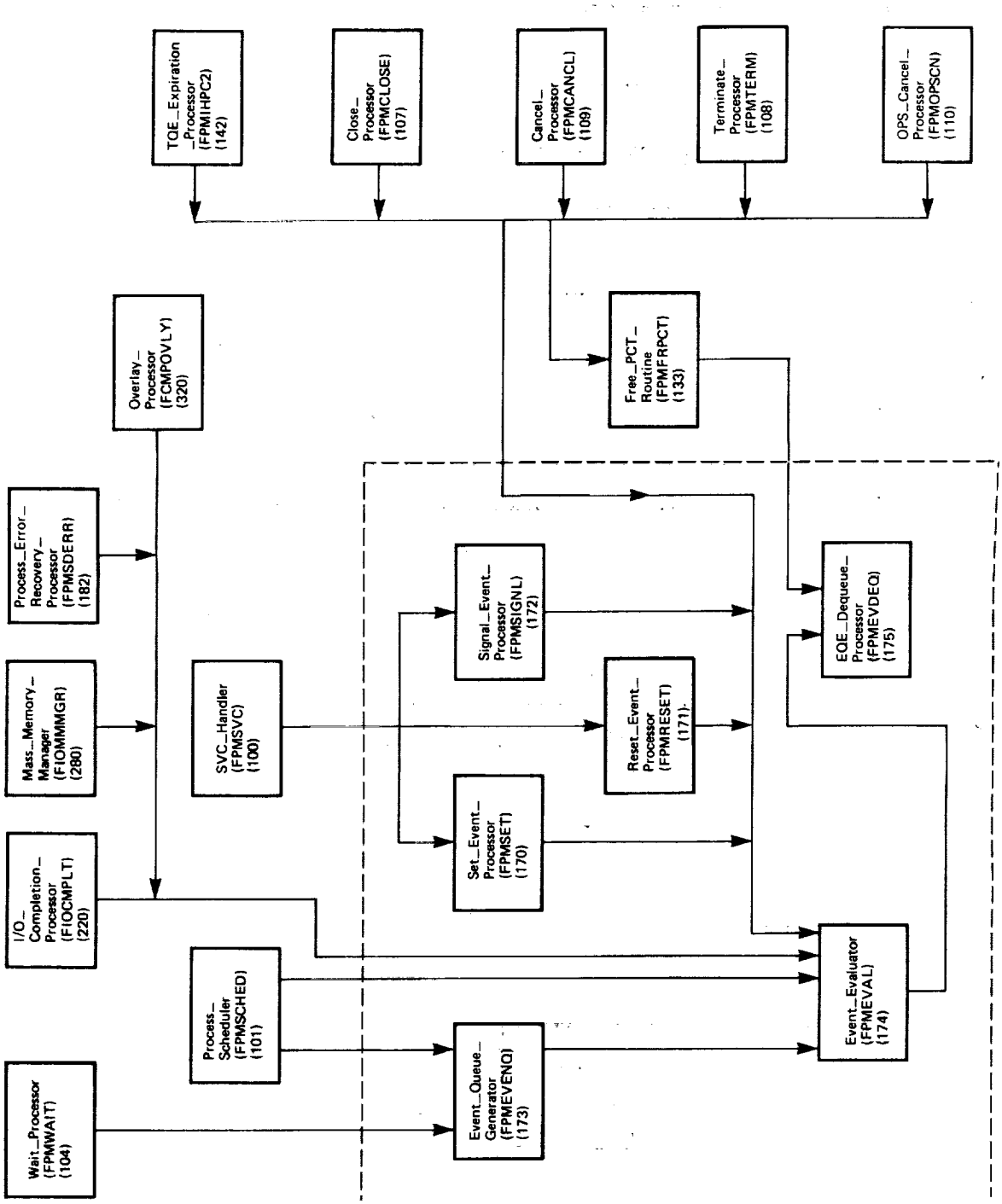


Figure 3.1.3-2. Event Management Control Flow

3.1.3.1 Set_Event_Processor (FPMSET)(170)

The Set_Event_Processor services the HAL/S SET instruction by changing an Event_Variable to the TRUE state.

a. Control Interface - CALLed by (100) SVC_Handler (FPMSVC).

b. Input - See Table 3.1.3.1-1.

c. Process Description - This program is CALLed by the SVC_Handler when an application process executes the SET instruction.

On entry, redundant set computers are synchronized by CALLing the SVC_Synchronization_Processor.

After synchronization, the Event_Variable is located, and, if TRUE, the Set_Event_Processor immediately returns to the SVC_Handler.

Otherwise, the Event_Variable is set TRUE and, if there is an application process whose execution depends on the TRUE state of the Event_Variable, the Event_Evaluator is CALLed.

The Set_Event_Processor then returns to the SVC_Handler.

The control flow for this module is shown in Figure 3.1.3.1-1.

d. Outputs - An Event_Variable may be set TRUE.

e. Module References -

1. (363) SVC_Synchronization_Processor (FCMSSYNC) is CALLed.
2. (174) Event_Evaluator (FPMEVAL) is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation -

1. An Event_Variable is set TRUE by changing its low order bit to '1'. No other bits are modified.

2. If a process' execution depends on the TRUE state of the Event_Variable, the high order 15 bits of the Event_Variable will be non zero.

DATA TABLE 3.1.3.1-1

NAME Set_Event_Processor (FP MSET)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Set_Event_Parameter_List	S013	I						
2	PSW_Interrupt_Code	&002.26	I		180,242, 320				
3	SVC_Old_PSW	&001.67	I		See App. E	TPSASOP			
4	Set_Event_Address	S013.2	I		170				
5	Event_Variable	@007	O	182	See App. E				
6	Event_State	@007.2	O	See APP	E				

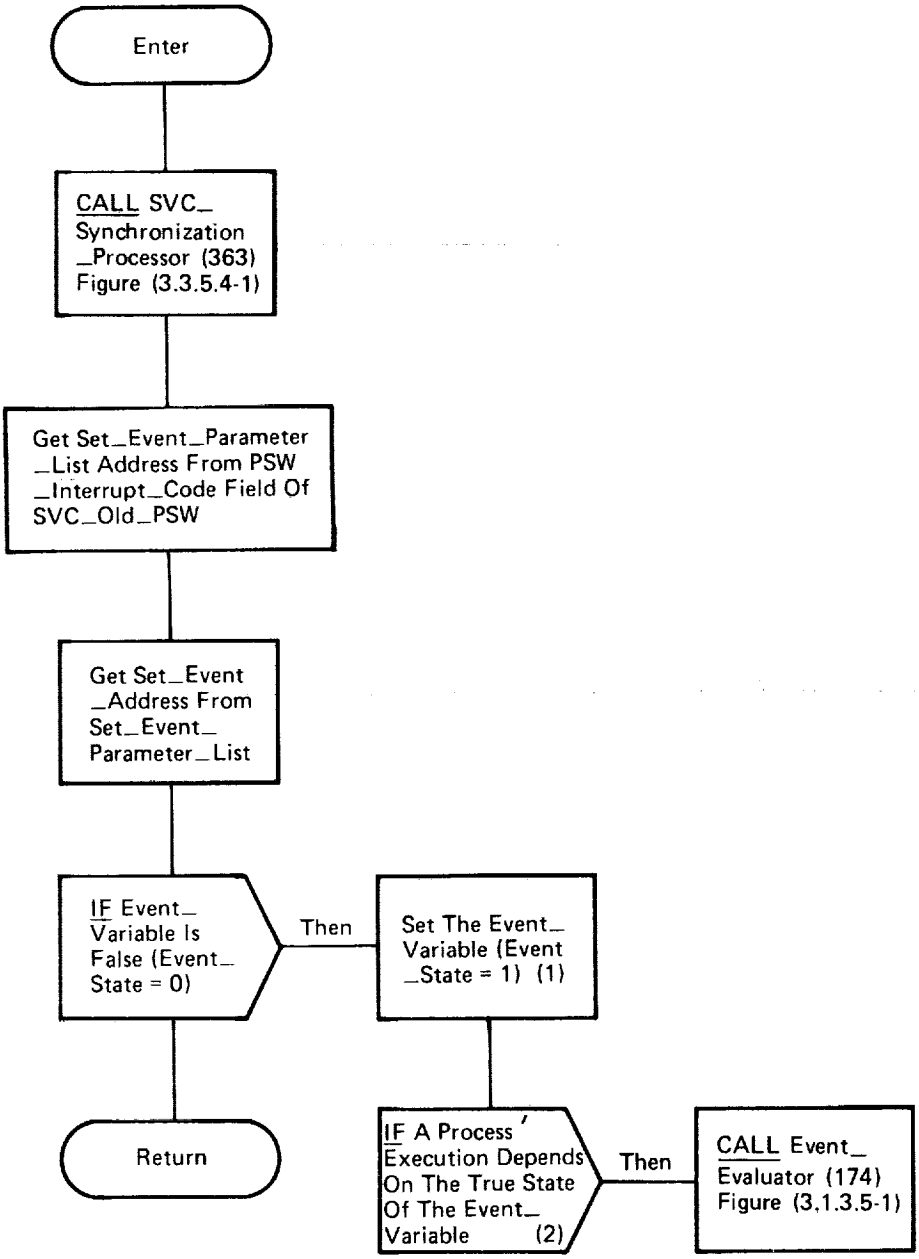


Figure 3.1.3.1-1. Set_Event_Processor (FPMSET)

3.1.3.2 Reset_Event_Processor (FPMRESET) (171)

The Reset_Event_Processor services the HAL/S RESET instruction by changing an Event_Variable to the FALSE state.

a. Control Interface - CALLed by (100)SVC_Handler (FPMSVC).

b. Input - See Table 3.1.3.2-1.

c. Process Description - This program is CALLed by the SVC_Handler when an application process executes the RESET instruction.

On entry, redundant set computers are synchronized by CALLing the SVC_Synchronization_Processor.

After synchronization, the Event_Variable is located, and, if FALSE, the Reset_Event_Processor immediately returns to the SVC_Handler.

Otherwise, the Event_Variable is reset to FALSE and, if there is an application process whose execution depends on the FALSE state of the Event_Variable, the Event_Evaluator is CALLed.

The Reset_Event_Processor then returns to the SVC_Handler.

The control flow for this module is shown is Figure 3.1.3.2-1.

d. Outputs - An Event-Variable may be reset FALSE.

e. Module References -

1. (363) SVC_Synchronization_Processor (FCMSSYNC) is CALLed.
2. (174) Event_Evaluator (FPMEVAL) is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation -

1. An Event_Variable is reset FALSE by zeroing the low order bit. No other bits are modified.
2. If a process' execution dependends on the FALSE state of the Event_Variable, the high order 15 bits of the Event_Variable will be non zero.

DATA TABLE 3.1.3.2-1

NAME Reset_Event_Processor (FPMRESET)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Reset_Event_Parameter_List	S014	I		171				
2	PSW_Interrupt_Code	&002.26			180, 242, 320				
3	SVC_Old_PSW	&001.67	I		See App. E	TPSASOP			
4	Reset_Event_Address	S014.2	I		171				
5	Event_Variable	@007	O	182	See App. E				
6	Event_State	@007.2	O	See APP. E					

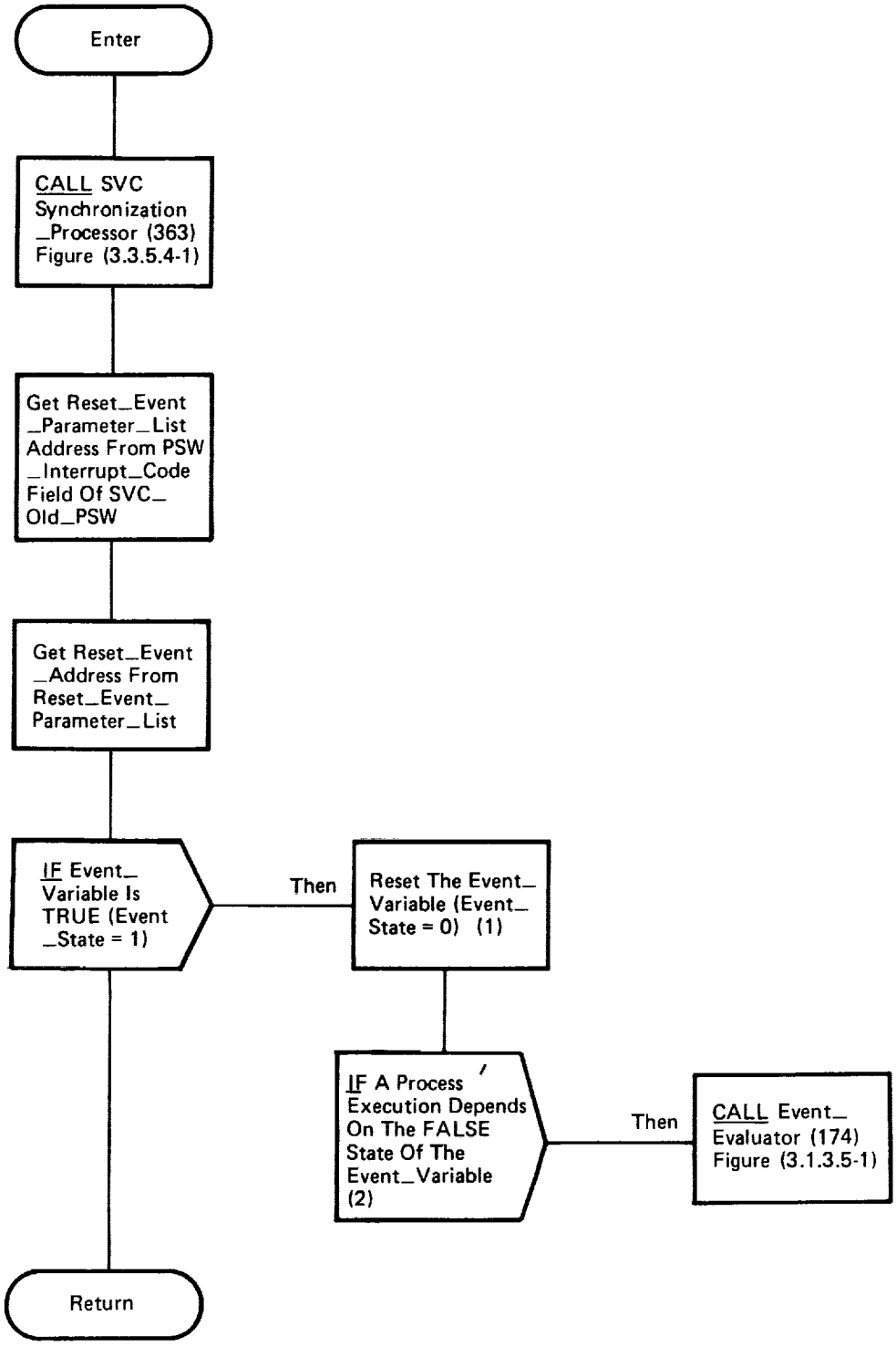


Figure 3.1.3.2-1. Reset_Event_Processor (FPMRESET)

3.1.3.3 Signal_Event_Processor (FPMSIGNL) (172)

The Signal_Event_Processor services the HAL/S SIGNAL instruction by momentarily reversing an Event_State.

a. Control Interface - CALLed by (100) SVC_Handler (FPMSVC)

b. Input - See Table 3.1.3.3-1.

c. Process Description -

This program is CALLed by the SVC_Handler when an application process executes the SIGNAL instruction.

On entry, redundant set computers are synchronized by CALLing the SVC_Synchronization_Processor.

Afer synchronization, the Event_Variable to be SIGNALed is examined. If there is no application process whose execution depends on the state (TRUE or FALSE) of the Event_Variable, the Signal_Event_Processor immediately returns to the SVC_Handler.

Otherwise, the Event_State is reversed (TRUE becomes FALSE, FALSE becomes TRUE) and the Event_Evaluator is CALLed.

On return from the Event_Evaluator the Event_State is restored to its original condition and the Signal_Event_Processor returns to the SVC_Handler.

The control flow for this module is shown in Figure 3.1.3.3-1.

d. Outputs - None

e. Module References -

1. (363) SVC_Synchronization_Processor (FCMSSYNC) is CALLed.
2. (174) Event_Evaluator (FPMEVAL) is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation -

1. If a process 'execution depends on the Event_Variable, the high order 16 bits of the Event_Variable will be non zero.

2. A TRUE Event_Variable will be reset.
A FALSE Event_Variable will be set.

DATA TABLE 3.1.3.3-1

NAME Signal_Event_Processor (FPMSIGNL)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Signal_Event_Parameter_List	S012	I		172				
2	PSW_Interrupt_Code	&002.26	I		180, 242, 320				
3	SVC_Old_PSW	&001.67	I		See App. E	TPSASOP			
4	Signal_Event_Address	S013.2	I		172				
5	Event_Variable	@007	O	182	See App. E				
6	Event_State	@007.2	O	See App.	E				

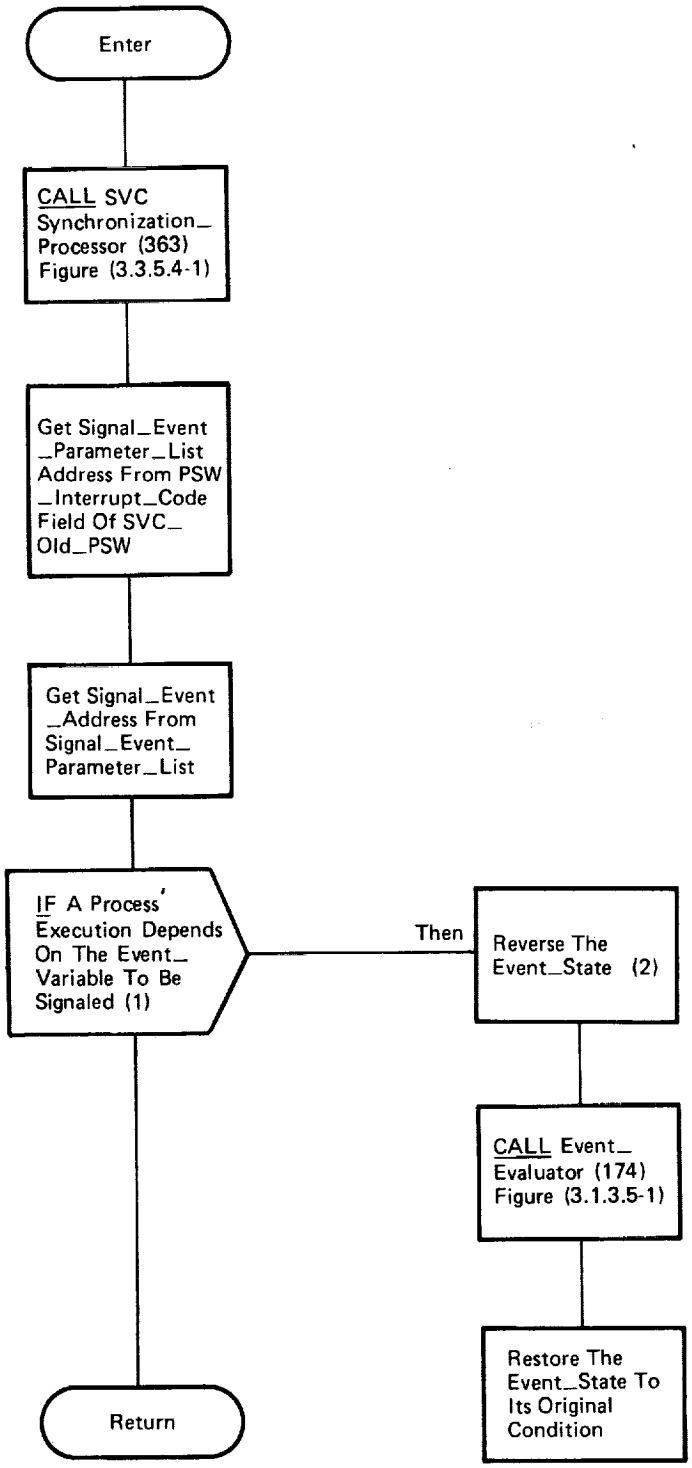


Figure 3.1.3.3-1. Signal_Event_Processor (FPMSIGNL)

3.1.3.4 Event_Queue_Generator (FPMEVENQ) (173)

The Event_Queue_Generator is called to add an EQE to the active EQE pool.

a. Control Interface -

1. CALLed by (101) Process_Schedule (FPMSCHED)
2. CALLed by (104) Wait_Processor (FPMWAIT)

b. Input - See Table 3.1.3.4-1.

c. Process Description -

The top free EQE is initialized by the Process_Scheduler when the ON, WHILE, or UNTIL event condition options of the HAL/S SCHEDULE statement are used, and by the Wait_Processor when the FOR event condition option of the HAL/S WAIT statement is used.

The Event_Queue_Generator is called to add the EQE to the active EQE pool. The event expression in the EQE will then be evaluated each time an Event_Variable in the expression changes states (TRUE to FALSE, FALSE to TRUE) until the expression is satisfied. At that time appropriate action based on the EQE_Type will be taken and the EQE will be returned to the EQE free pool.

Before adding the EQE to the active EQE pool, the event expression in the EQE is evaluated.

If the event expression contains only one Event_Variable, the Event_Queue_Generator evaluates the expression itself and sets flags in the EQE identifying the Expression_Type to facilitate future evaluation.

If there is more than one Event_Variable in the event expression, the Event_Evaluator is called to evaluate the EQE.

If the EQE was not satisfied and not flagged NULL, it is removed from the EQE free pool and becomes a member of the active EQE pool. An EQE is flagged NULL by the Process_Schedule when a non-cyclic process is scheduled with an UNTIL or WHILE event condition. The Process_Scheduler does not want the EQE added to the active active EQE pool, but wants to know if the EQE is satisfied.

Active EQE's are not chained to each other, but may be linked to the Event_Variables used in the EQE event expression.

When an EQE is added to the Active EQE pool, the Event_Used_Indicator portion of the Event_Variables referenced in the EQE is modified. The Event_Used_Indicator will be modified to either point to the EQE or contain a count of the EQE references to the Event_Variable.

After adding the EQE to the active EQE pool or if the EQE was satisfied or NULL, the Event_Queue_Generator return to the calling process with a return code indicating its evaluation of the EQE.

The control flow for this module is shown in Figure 3.1.3.4-1.

d. Outputs - See Table 3.1.3.4-1.

e. Module References -

1. (174) Event_Evaluator (FPMEVAL) is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions -

The Event_Queue_Generator in the FCOS is limited to the following types of event expressions:

1. Expressions containing a single Event_Variable with or without the "NOT" operator,

2. Event expressions which contain more than one Event_Variable connected by all "OR" operators,

3. Event expressions which contain more than one Event_Variable connected by all "AND" operators.

j. Detailed Implementation -

1. The calling process has initialized the top free pool EQE for input to the Event_Queue_Generator.

2. When an EQE is initialized it is assumed that there are five Event_Variables in the event expression. The Event_Queue_Generator zeroes the first unused Event_Variable pointer in the expression to simplify later event expression evaluation.

3. All EQE's except SCHEDULE WHILE EQE's are satisfied when the event condition is satisfied. SCHEDULE WHILE EQE's are satisfied when the event condition is not satisfied.

4. A SCHEDULE WHILE EQE is satisfied when the event condition it contains is not satisfied. Therefore, a SCHEDULE WHILE EQE containing only one event will have its TRUE/FALSE single Event_Variable flags set to indicated the EQE is satisfied when its event expression is not satisfied.

5. When there are multiple Event_Variables in an event expression, the number of Event_Variables in the expression can be determined by dividing the Count_of_Operators plue one by two. See the constraints and assumptions above.

6. An EQE is flagged NULL when only an evaluation of the EQE is desired. An EQE is flagged NULL by the Process_Scheduler when a non-cyclic process is scheduled with WHILE or UNTIL event condition cancellation criteria.

7. One process is dependent on the event variable if the Event_Used_Indicator contains an address rather than a count (Event_Used_Indicator > 64).

DATA TABLE 3.1.3.4-1

NAME Event_Queue_Generator (FPMEVENQ)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	EQE_Free_Pool_Address	Q001.10	I,O	173 175	101, 104, 174	TCVTEQEP			
2	Event_Variable	@007	I,O	182	See App. E				
3	Count_Of_Operators	Q006.4	I	101, 104	173				
4	Event_Variable_2	Q006.7	I,O	101,104, 173	174	TEQEVAR2			
5	Event_Variable_1	Q006.6	I,O	101,104, 173,175	173 174, 175	TEQEVAR1			
6	Expression_Type	Q006.13	O	104, 173	173, 174				
7	Event_State	@007.2	I	See App.	E				
8	Option_Event_Type	Q006.12	I	101, 104	173, 174				
9	Event_Variable_3	Q006.8	I,O	101,104	173 174	TEQEVAR3			
10	Event_Variable_4	Q006.9	I,O	101,104	173 174	TEQEVAR4			
11	Event_Variable_5	Q006.10	I,O	101,104	173 174	TEQEVAR5			
12	Next_EQE_Address	Q006.1	I	173, 174 175	173 174	TEQENXT			
13	EQE_Parent_PCT_Address	Q006.2	I	101,104 175	173, 174 175	TEQEPCT			
14	Outstanding_EQE-Indicators	Q003.32	O	133, 173, 174, 175	See APP. E				
15	Event_Used_Indicator	@007.1	O		See App.E				

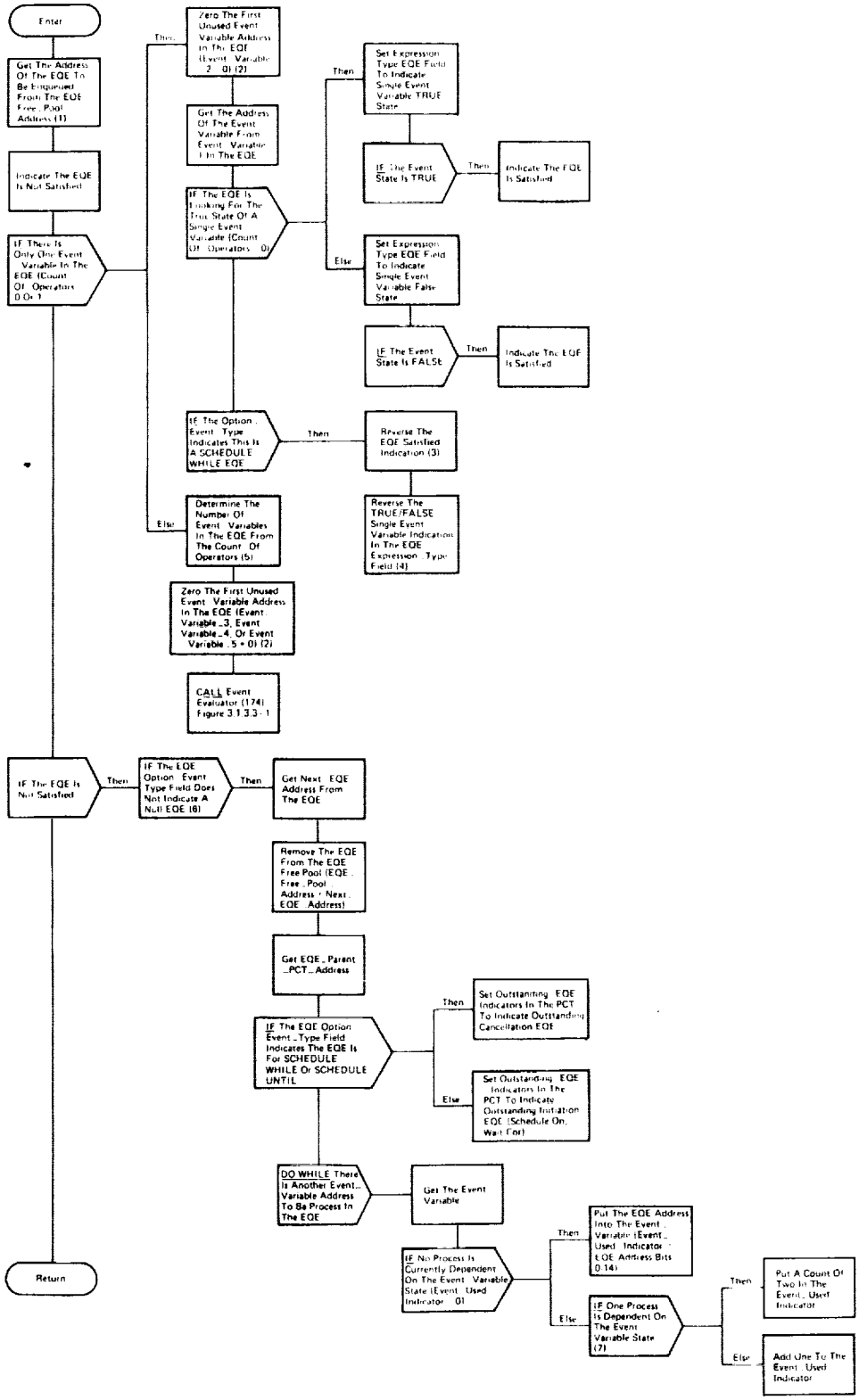


Figure 3.1.3.4-1. Event_Queue_Generator (FPMEVENQ)

3.1.3.5 Event_Evaluator (FPMEVAL)(174)

The Event_Evaluator is called to evaluate active EQE's when an Event_Variable changes states and initiates the appropriate action based on the EQE type when it finds the EQE is satisfied.

The Event_Evaluator is also called by the Event_Queue_Generator to evaluate EQE's which have more than one Event_Variable in an event expression.

a. Control Interface -

1. CALLed by (101) Process_Scheduler (FPMSCHED)
2. CALLed by (107) Close_Processor (FPMCLOSE)
3. CALLed by (108) Terminate_Processor (FPMTERM)
4. CALLed by (109) Cancel_Processor (FPMCANCL)
5. CALLed by (110) OPS_Cancel_Processor (FPMOPSCN)
6. CALLed by (142) TQE_Expiration_Processor (FPMIHPC2)
7. CALLed by (170) Set_Event_Processor (FPMSET)
8. CALLed by (171) Reset_Event_Processor (FPMRESET)
9. CALLed by (172) Signal_Event_Processor (FPMSIGNL)
10. CALLed by (173) Event_Queue_Generator (FPMEVENQ)
11. CALLed by (182) Process_Error_Recovery_Processor (FPMSDERR)
12. CALLed by (220) I/O_Completion_Processor (FIOCMPLT)
13. CALLed by (280) Mass_Memory_Manager (FIOMMMGR)
14. CALLed by (320) Overlay_Processor (FCMPOVLY)
15. CALLed by (149) MTU_Redundancy_Manager (FPMMTURM)

b. Input - Register 0 bits 0-15 are zero if called by the Event_Queue_Generator. Otherwise Register 0 bits 0-15 contain an Event_Variable address. Also, see Table 3.1.3.5-1.

c. Process Description - When called to evaluate a specific EQE, the Event_Evaluator does the evaluation and returns to the calling program with a return code. When passed an Event_Variable address, the Event_Evaluator locates all EQE's which use the Event_Variable and evaluates them. Finding an EQE which is satisfied will result in action indicated by the EQE_Type. The following is a summary of the action taken and the processes called for each type of EQE when it is found to be satisfied:

1. SCHEDULE UNTIL or SCHEDULE WHILE - If the process to which the EQE applies is in an execution cycle, set the Canceled_PCT_Indicator in the process' PCT and dequeue the EQE. If the process to which the EQE applies is not in an exection cycle, remove the PCT from the run queue, turn off its Process_Event, evaluate any EQE's which use the Process_Event, and take appropriate action and dequeue the EQE.

   Evaluation of EQE's which use the Process_Event is delayed. The Event_Evaluator processes all EQE's for a given Event_Variable and then checks for any Process_Events which have changed. It then processes each Process_Event in the same manner.

2. SCHEDULE ON - The PCT_Wait_Indicators are turned off. If the PCT is a REPEAT EVERY PCT, a REPEAT EVERY TQE is created and added to the TQE chain by calling the Timer_Queue_Generator program. The EQE is removed from the active EQE pool by calling the EQE_Dequeue_Processor. The Process_Switcher is then called.

3. WAIT FOR - The PCT_Wait_Indicators are turned off and the EQE is removed from the active EQE pool. The Process_Switcher is then called.

d. Outputs - See Table 3.1.3.5-1. A return code is passed to the Event_Queue_Generator to indicate if the EQE was satisfied.

e. Module References -

1. (140) Timer_Queue_Generator (FPMTMENQ) is CALLed.
2. (175) EQE_Dequeue_Processor (FPMEVDEQ) is CALLed.
3. (102) Process_Switcher (FPMSWTCH) is CALLed.
4. (133) Free_PCT-Routine (FPMFRPCT) is CALLed.

f. Module Attributes - Program.

g. Template References - N/A

h. Error Handling - None.

i. Constraints and Assumptions - The Event_Evaluator in the FCOS is limited to the following types of event expressions:

1. Expressions containing a single Event_Variable with or without the "NOT" operator,

2. Event expressions which contain more than one Event_Variable connected by all "OR" operators,

3. Event expressions which contain more than one Event_Variable connected by all "AND" operators.

j. Detailed Implementation -

1. The top free EQE has been initialized by the Schedule_Processor or the Wait_Processor.

2. This loop processes all EQE's which reference the input Event_Variable as well as all EQE's which reference any Process_Events which may be reset. Process_Events are reset when a process is cancelled.

3. An Event_Variable points to the EQE which references it unless there is more than one EQE reference. In that case the Event_Variable contains a count of EQE references. Active EQE's are not chained to each other so the Event_Evaluator will search through all EQE's until it finds all EQE references to the Event_Variable.

4. If a SCHEDULE WHILE or SCHEDULE UNTIL EQE is satisfied a process may be cancelled. When a process is removed from the run queue its Process_Event is reset. If any EQE references the Process_Event, it will be evaluated after all EQE's for the current Event_Variable have been evaluated.

   Process_Events are remembered by the Event_Evaluator by saving the EQE which caused the Process_Event to be reset in a temporary EQE chain. The Process-Event is located using the EQE_Parent_PCT_Address.

5. A SCHEDULE WHILE EQE is satisfied when the event expression it contains is not satisfied. Therefore, the return code is changed to reflect the opposite of the evaluation performed.

6. The Process_Event is the first halfword of a PDE. Therefore, getting the PDE_Address is equivalent to getting the Process_Event address.

DATA TABLE 3.1.3.5-1

NAME Event_Evaluator (FPMEVAL)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	EQE_Free_Pool_Address	Q001.10	I	173,175	101,104, 174	TCVTEQEP			
2	Event_Variable	@007	I	182	See App. E				
3	Event_Used_Indicator	@007.1	I		See App. E				
4	Top_EQE_Address	Q001.14	I		174 175	TCVTTEQE			
5	EQE_Type	Q006.11	I	101,104, 173, 175	173,175 174	TEQETYPE			
6	Option_Event_Type	Q006.12	I	101,104	173,174				
7	Temporary_Process_Event_Chain_Address	Q001.15	I,O	174	174	TCVTPECH			
8	Expression_Type	Q006.13	I	104 173	173,174				
9	Event_State	@007.2	I	See App. E	See App. E				
10	EQE_Parent_PCT_Address	Q006.2	I	101,104, 175	173,174, 175	TEQEPCT			
11	Repeat_Options	Q003.36	I	101	101,107, 133,174				
12	TQE_Free_Pool_Address	Q001.11	I	142,143, 166,305	See App. E	TCVTTQEP			
13	TQE_Parent_PCT_Address	Q005.2	O	See App.	E	TTQEPCT			
14	TQE_Flags	Q005.6	O	104,107, 140,142	140,143, 148,174	TTQEFLGS			
15	TQE_Type_Indicator	Q005.10	O	101,104, 107	140,142, 148,174				
16	PCT_Wait_Indicators	Q003.42	I,O	See App.	E	TPCTWAIT			
17	Outstanding_EQE_Indicators	Q003.32	O	133,173, 174,175	See App. E				
18	Cancelled_PCT_Indicator	Q003.30	O	101,109, 142	See App. E				
19	Initial_Conditions	Q003.39	O,I	101,174	101,174				
20	Process_Directory_Entry	Q002	I		174	TFPDE			

DATA TABLE 3.1.3.5-1 (Cont'd.)

NAME Event_Evaluator (FPMEVAL)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
21	Process_Event	Q002.1	I,O	See APP. E		TPDEVENT			
22	Next_EQE_Address	Q006.1	O	173,174, 175	173,174	TEQENXT			
23	PDE_Address	Q003.4	I	101	See App. E	TPCTPDE			
24	Scheduled_Wait_Indicator	Q003.48	I	101,107 142	See App. E				
25	Event_Variable_1	Q006.6	I	101,104, 173,175	173,174, 175	TEQEVAR1			
26	Event_Variable_2	Q006.7	I	101,104, 173	174	TEQEVAR2			
27	Event_Variable_3	Q006.8	I	101,104	173,174	TEQEVAR3			
28	Event_Variable_4	Q006.9	I	101,104	173,174	TEQEVAR4			
29	Event_Variable_5	Q006.10	I	101,104	173,174	TEQEVAR5			

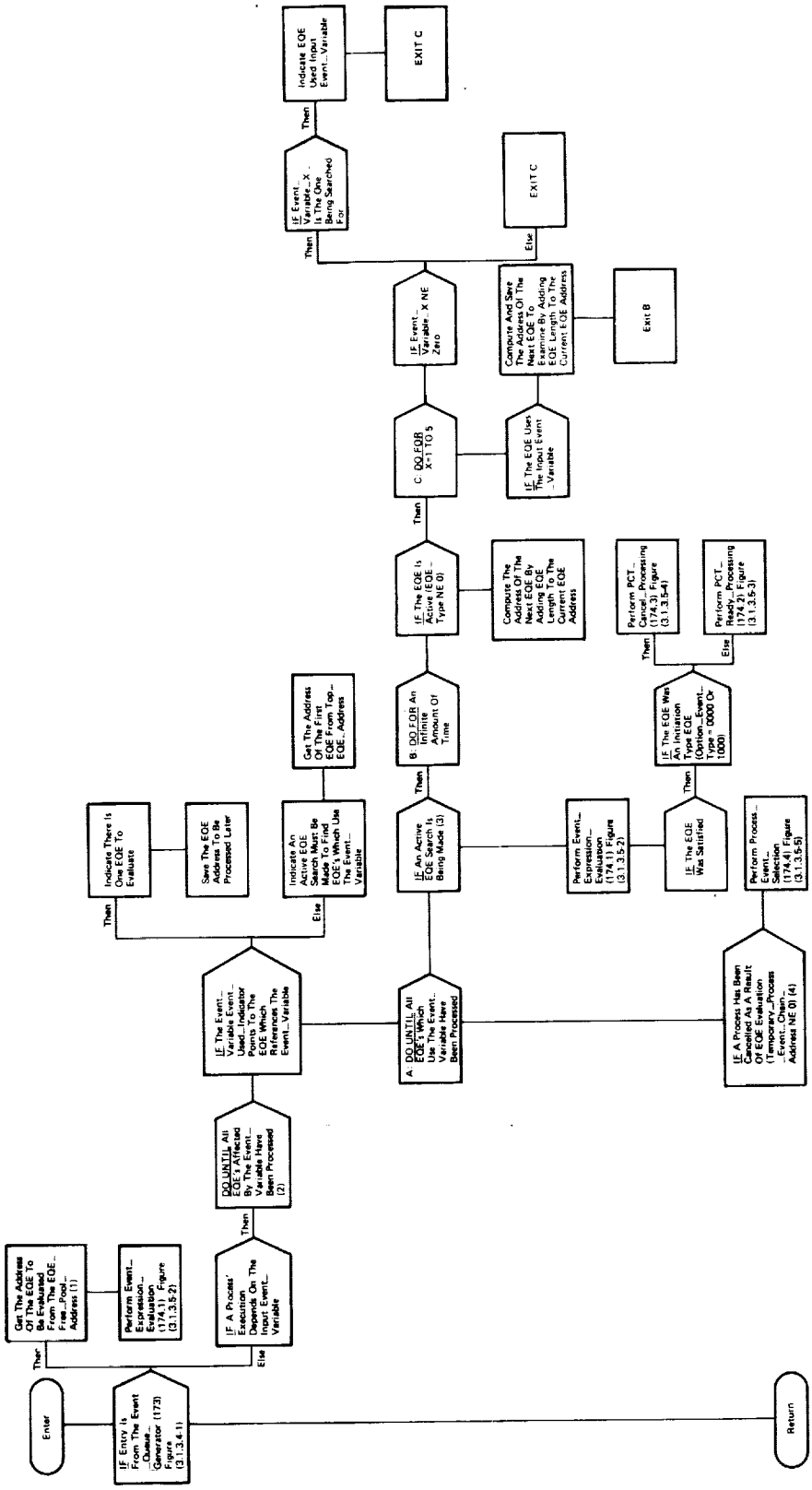


Figure 3.1.3.5-1. Event_Evaluator (FPMEVAL)

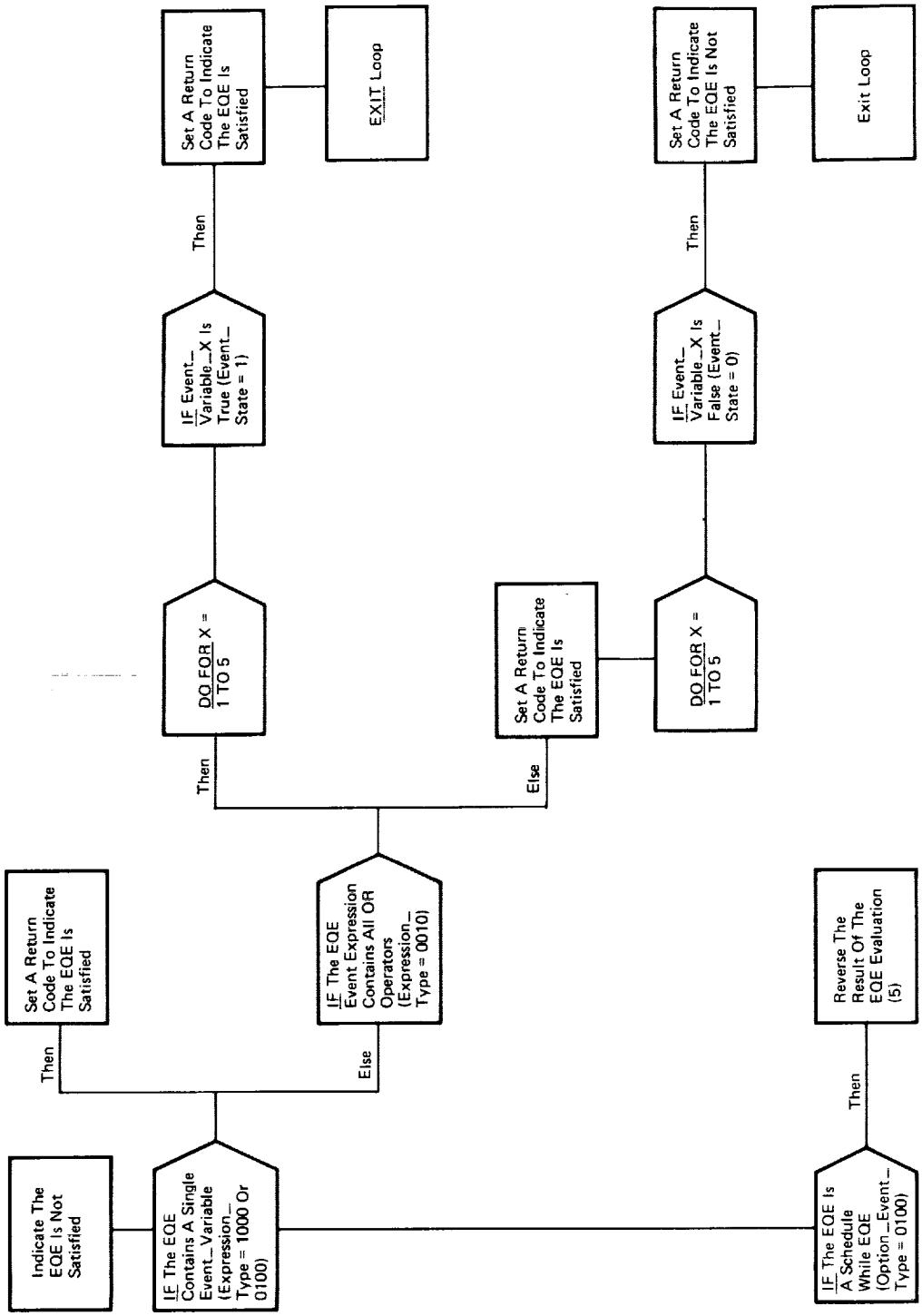


Figure 3.1.3.5-2. Event_Evaluator Event_Expression_Evaluation (174.1)

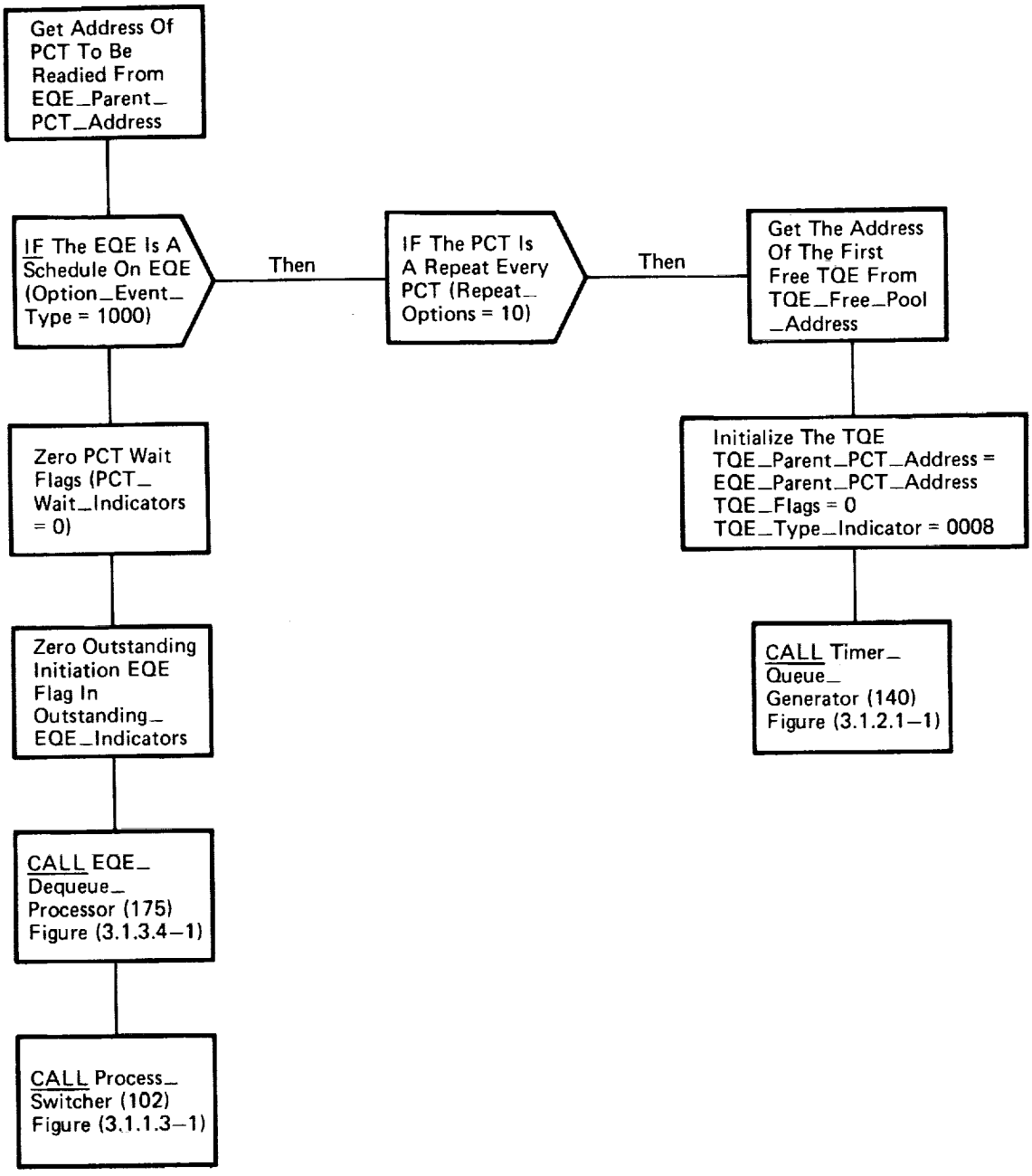


Figure 3.1.3.5-3. Event_Evaluator PCT_Ready_Processing (174.2)

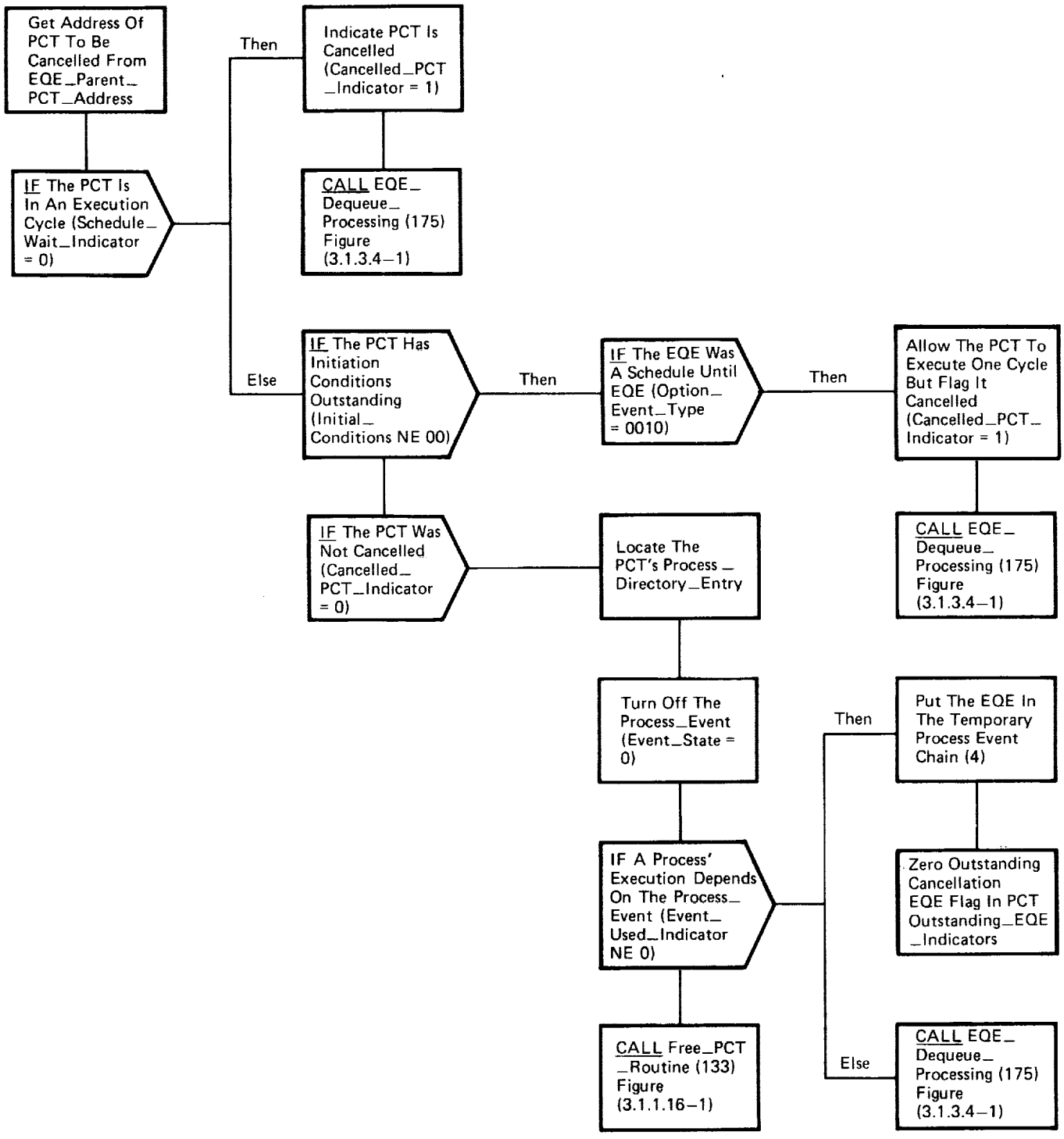


Figure 3.1.3.5-4. Event_Evaluator PCT_Cancel_Processing (174.3)

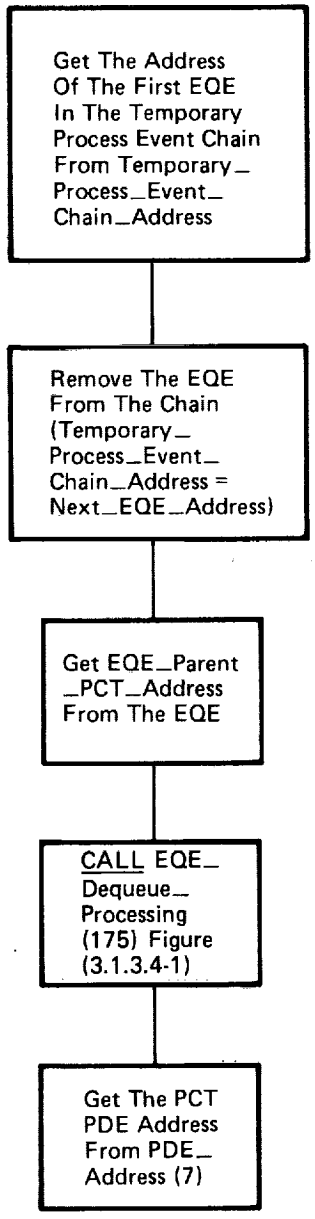


Figure 3.1.3.5-5. Event_Evaluator Process_Event_Selection (174.4)

3.1.3.6 EQE_Dequeue_Processor (FPMEVDEQ) (175)

The EQU_Dequeue_Processor is CALLed to remove an EQE from the active EQE pool and return it to the EQE free pool.

a. Control Interface -

1. CALLed by (174) Event_Evaluator (FPMEVAL)
2. CALLed by (133) Free_PCT_Routine (FPMFRPCT)

b. Input -

1. When CALLed by the Event_Evaluator Register 0 bits 0-15 contain the address of the EQE to be dequeued. Register 0 bits 16-31 are zero.

2. When CALLed by the Free_PCT_Routine Register 0 bits 0-15 contain the address of the PCT whose EQE's are to be dequeued. Register 0 bits 16-30 are zero while bit 31 is set to one.

c. Process Description - When CALLed by the Event_Evaluator the EQE_Dequeue_Processor immediately starts dequeueing the input EQE.

When CALLed by the Free_PCT_Routine the EQE_Dequeue _Processor determines the number of outstanding EQE's from the input PCT and examines all active EQE's until the outstanding EQE's are found and dequeued.

To dequeue an EQE, each Event_Variable referenced by the EQE is updated to reflect the dequeueing of the EQE. The Outstanding_EQE_Indicators in the input PCT are updated, the EQE is indicated to be free, and the EQE is added to the top of the EQE free pool.

Return is to the calling program. The control flow for this module is presented in Figure 3.1.3.6-1.

d. Outputs - See Table 3.1.3.4-1

e. Module References - None

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation -

1. Bit 31 of the Register 0 is examined. See Input description given above.

DATA TABLE 3.1.3.6-1

NAME EQE_Dequeue_Processor (FPMEVDEQ)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	EQE_Parent_PCT_Address	Q006.2	I,O	101,104 175	173,174 175	TEQEPCT			
2	Outstanding_EQE_Indicators	Q003.32	I,O	133,173 174,175	See Appendix E				
3	Top_EQE_Address	Q001.14	I		175 174	TCVTTEQE			
4	Event_Variable_1	Q006.6	I,O	101,104 173,175	173,174 175	TEQEVAR1			
5	Event_Variable_2	Q006.7	I	101,104 175	174 175	TEQEVAR2			
6	Event_Variable_3	Q006.8	I	101 174	173,174 175	TEQEVAR3			
7	Event_Variable_4	Q006.9	I	101 104	173,174 175	TEQEVAR4			
8	Event_Variable_5	Q006.10	I	101 104	173,174 175	TEQEVAR5			
9	Event_Variable	@007	I	182	See APP. E				
10	Event_Used_Indicator	@007.1	I,O		See APP. E				
11	Option_Event_Type	Q006.12	I	101 104	173,174				
12	EQE_Type	Q006.11	I,O	101,104 173,175	173,174 175	TEQETYPE			
13	EQE_Free_Pool_Address	Q001.10	O	173 175	101,104 174	TCVTEQEP			
14	Next_EQE_Address	Q006.1	O	173,174 175	173 174	TEQENXT			

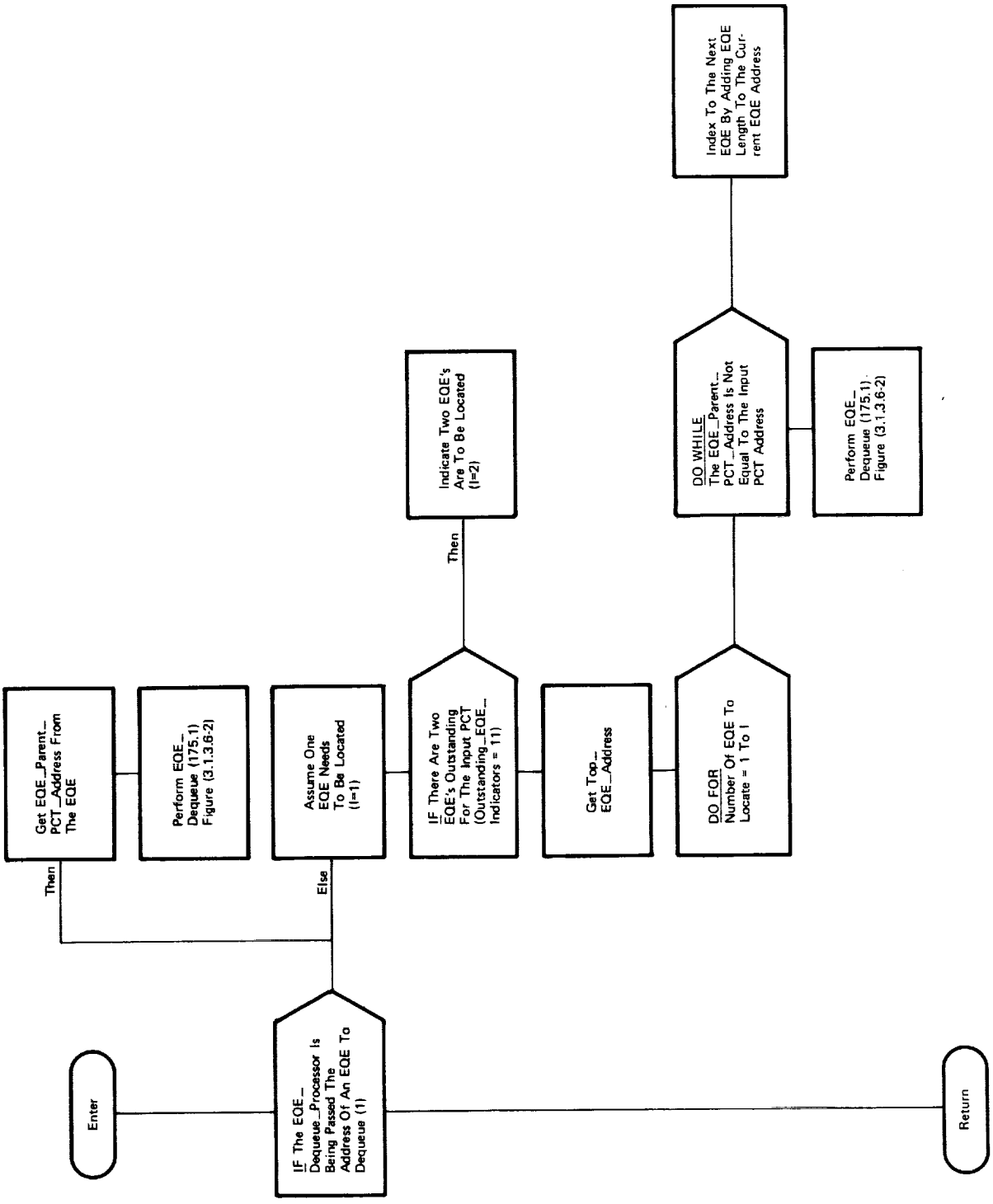


Figure 3.1.3.6-1. EQE_Dequeue_Processor (FPMEVDEQ)

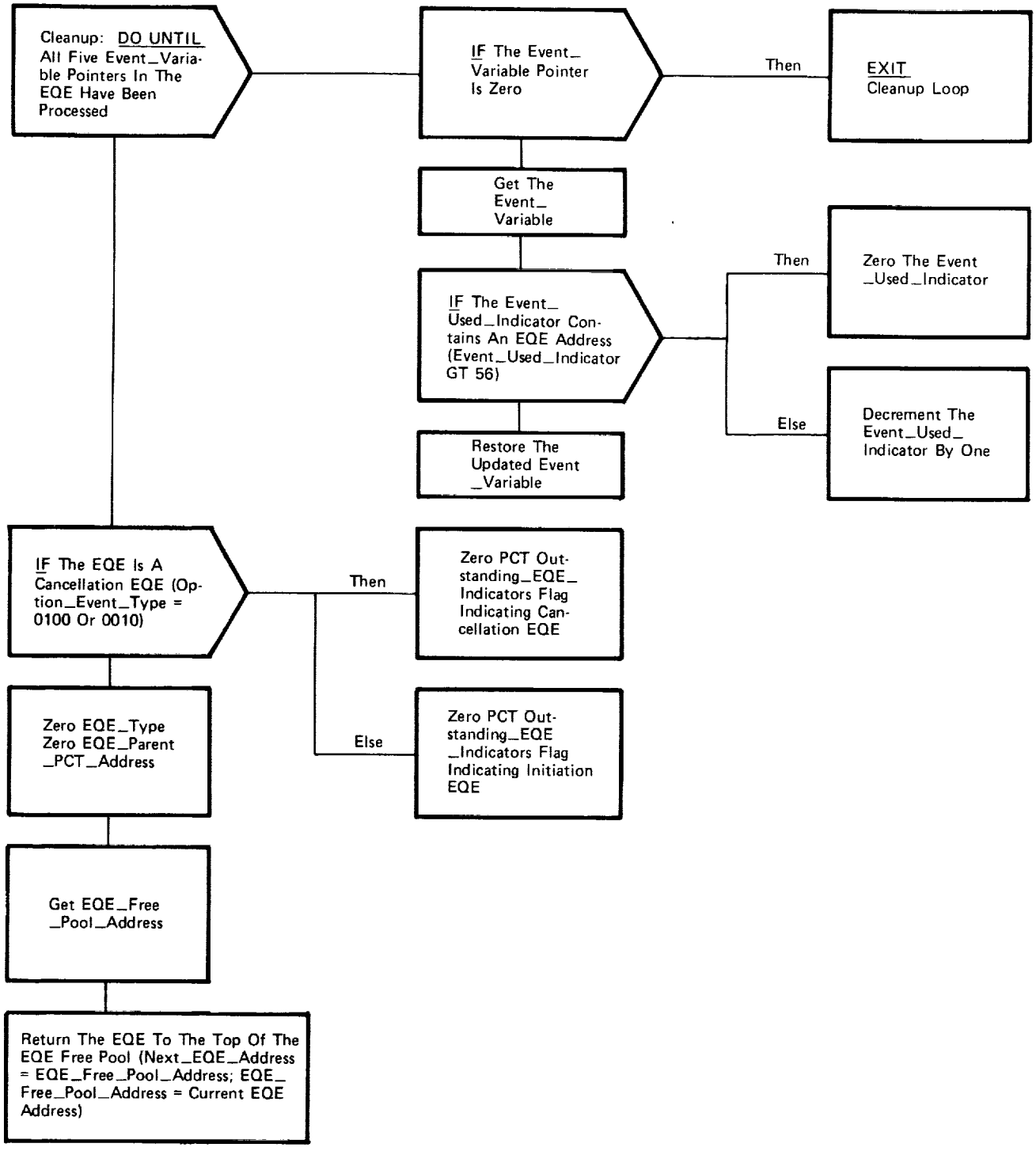


Figure 3.1.3.6-2. EQE_Dequeue_Processor EQE_Dequeue (175.1)

3.1.4 Process Error Management

Process Error Management provides FCOS support for the handling of error conditions which occur during FCOS or application execution.

Errors that occur are categorized under the following group numbers:

- Group 2 for FCOS detected software errors
- Group 3 for Program Interrupts
- Group 4 for HAL/S - FC defined errors
- Group 6 for application defined errors.

All errors are assigned codes. Errors within group 2 are assigned codes by FCOS. Group 3 errors are defined per AP-101 interrupt code. Group 4 errors are HAL/S defined. Group 6 are defined by application software via ON ERROR statements.

The application programmer has the capability of overriding the FCOS system default action via the ON ERROR statement. The compiler processes these statements and stores the appropriate information in the error environment of the process which resides in the stack area of the process. FCOS examines this error environment when processing errors.

If the application programmer has not specified otherwise, FCOS performs system defined default actions when errors occur. These system default actions have no relation to the group assignments but are based on severity of the error and information available to FCOS at time of the error. The FCOS ground rule in performing default actions is always to attempt to continue execution within the constraints which exist at the time of error. System default recovery is not unique for each error but consists of four basic approaches:

a. Ignore the error and continue processing.

b. Force close the application process and continue processing.

c. Ignore the error and continue processing until the number of errors encountered in this process reaches the maximum allowed. If this maximum is reached, the process is force closed.

d. Discontinue current FCOS processing and dispatch the highest priority ready application process.

FCOS error annunciation is accomplished by setting indicators which are associated with cases in which FCOS needs error annunciation performed. These indicators are interrogated by the System Control GPC_Switch_Monitor (Vol. 2, Part 3, Section 3.2.1), and appropriate error annunciation is performed.

The interface flow for application related errors is shown in Figure 3.1.4-2, Application-related errors include machine checks, instruction monitor interrupts and program interrupts occurring during application execution, and application errors detected by FCOS. The FCOS routines that detect these errors (in the case of error interrupts, this is the FCOS routine which gets control as a result of the interrupt) call the Process_Error_Recovery_Processor to perform the appropriate actions based on the error environment of the process in error. Event Management services, Process Control services, and the Process_Error_Logger are called as necessary by the Process_Error_Recovery_Processor. This routine returns to the calling FCOS routine after the error has been processed.

FCOS errors are handled within the Program_Interrupt_Handler and the Instruction_Monitor_Interrupt_Handler. In this case, the Process_Error_Logger is called directly by these programs. FCOS queue overflow results in a protection interrupt during FCOS execution and is treated as a severe error by the Program_Interrupt_Handler.

The FCOS does not process machine check error interrupts. The Machine Check New PSW is set to force the GPC into an uninterruptable wait state on the occurence of a machine check.

The remainder of the detailed description of Process Error Management is divided into the major areas (See Figure 3.1.4-1).

a. Program_Interrupt_Handler - The fielding of Program Monitor interrupts and error recovery of FCOS errors.

b. Instruction_Monitor_Interrupt_Hander - The fielding of Instruction Monitor Interrupts.

c. Process_Error_Recovery_Processor - The recovery processing of application related errors.

d. Process_Error_Logger - The recording of error conditions which occur during system execution.

e. Forced_Close_Processor - The clean up processing necessary when a process encounters certain errors conditions.

f. Application_Error_Number_Request_Processor - FCOS support for the HAL/S ERRGRP and ERRNUM functions.

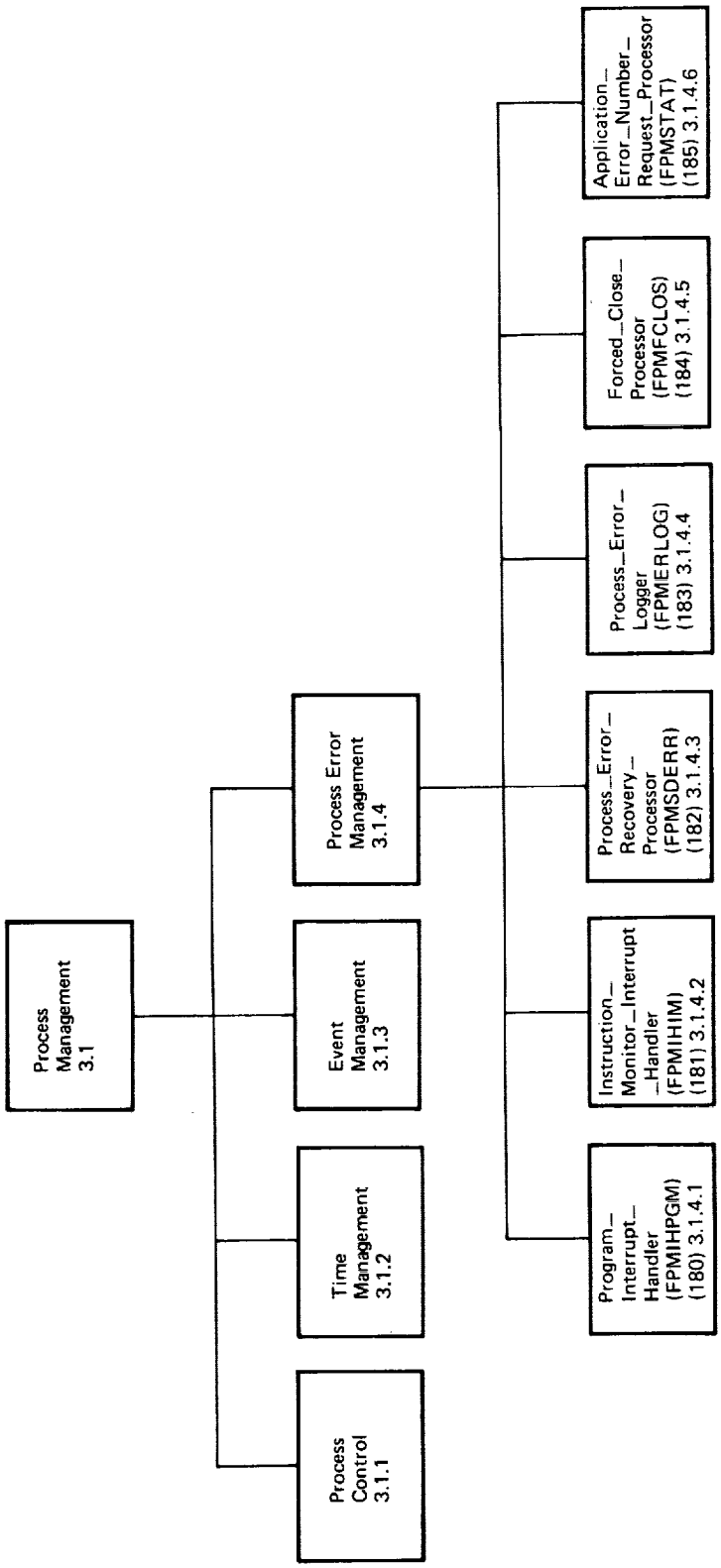


Figure 3.1.4-1. Process Error Management Hierarchy Diagram

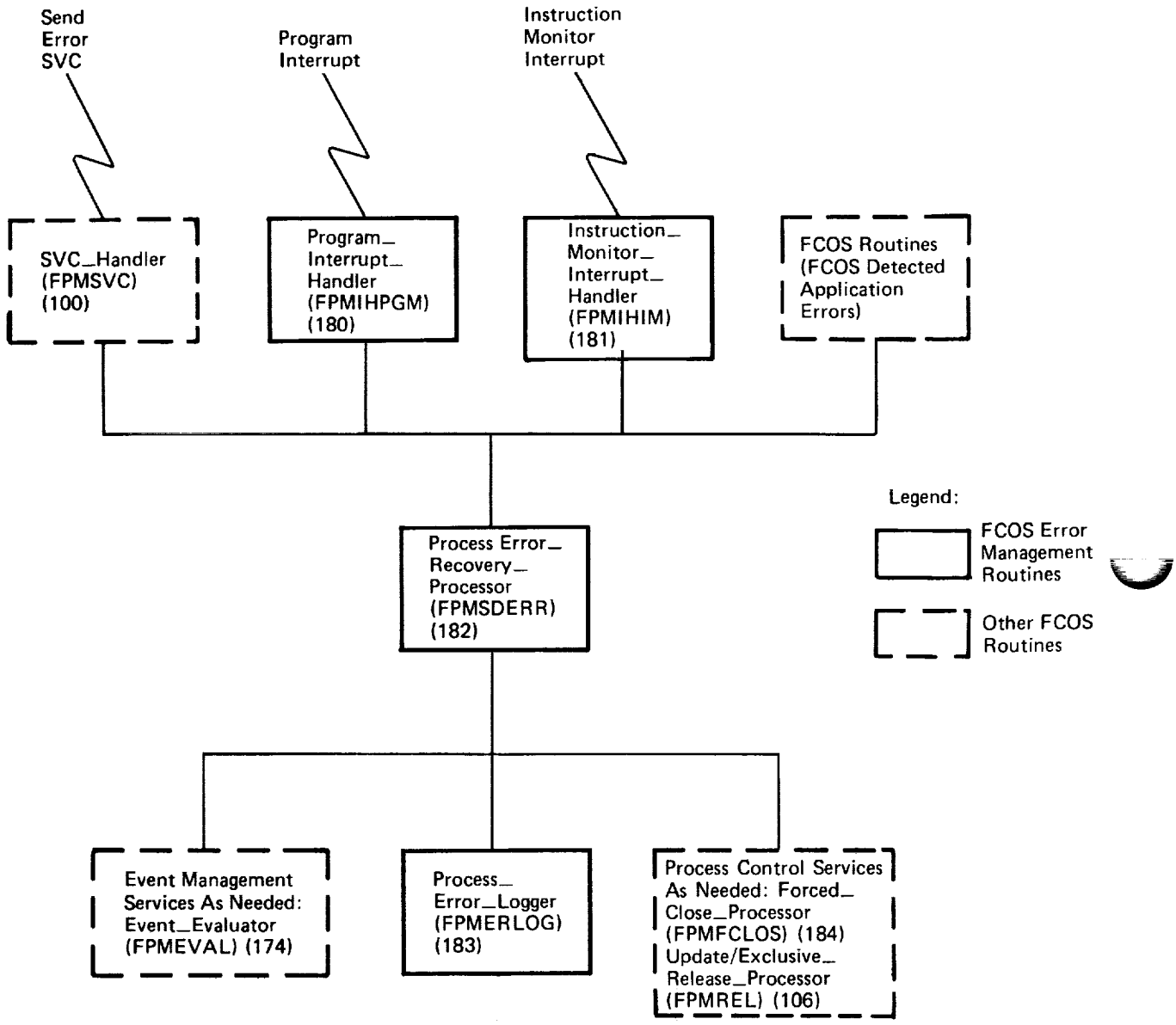


Figure 3.1.4-2. Error Management Flow For Application Errors

3.1.4.1 Program_Interrupt_Handler (FPMIHPGM) (180)

FPMIHPGM fields program interrupts and either processes the error (if the error occurred during FCOS execution) or invokes the Process_Error_Recovery_Processor to process the error (if the error occurred during application execution).

a. Control Interface - Hardware PSW swap because of program interrupt.

b. Input - See Table 3.1.4.1-1.

c. Process Description - The Program_Interrupt_Handler forms the error group code identifying the error by appending to the program interrupt group number (3), the PSW_Interrupt_Code of the Program_Interrupt_Old_PSW.

If the PSW_Run_State_Control indicator of the Program_Interrupt_Old_PSW indicates problem state, the error has occurred during application execution and the Process_Error_Recovery_Processor is invoked to process the error according to the error environment of the process in error. After the error has been processed, control is either returned to the process in error or to another process depending on how the error was processed.

If the PSW_Run_State_Control of the Program_Interrupt_Old_PSW indicates supervisor state, the error has occurred during FCOS execution and the error is processed in the following manner. The Process_Error_Logger is invoked to log the error. The appropriate indicator for program interrupts is set in the FMPT_HDR_Bits to cause annunciation of the error. Either control is returned to the FCOS program which was in error or control is given to the highest priority ready application process depending on the severity of the error. (See Appendix D).

The control flow for this module is presented in Figure 3.1.4.1-1.

d. Output - See Table 3.1.4.1-1.

e. Module References

1. (182) Process_Error_Recovery_Processor (FPMSDERR) is CALLed.

2. (183) Process_Error_Logger (FPMERLOG) is CALLed.

3. (103) Process_Dispatcher (FPMDISP) is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - An error annunciation indicator is set for FCOS program interrupts so that an error message is displayed (OS040).

i. Constraints and Assumptions - The PSW_Run_State_Control is assumed to be 0 (Supervisor state) for all FCOS execution and 1 (Problem State) for all application execution.

j. Detailed Implementation

1. Control is passed via a PSW load of the Interrupt_PSW_Of_Process.

2. There is no return from CALL to Process_Dispatcher.

3. Control is passed via a PSW load of the Program_Interrupt_Old_PSW.

DATA TABLE 3.1.4.1-1

NAME Program_Interrupt_Handler (FPMIHPGM)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Program_Interrupt_Old_PSW	&001.57	I		180	TPSAPIOP			
2	PSW_Interrupt_Code	&002.26	I		180				
3	PSW_Run_State_Control	&002.25	I		180,181, 320				
4	Active_PCT_Address	Q001.2	I	103	See App. E	TCVTOLD			
5	Interrupt_PSW_of_Process	Q003.5	O	See App. E	See App. E	TPCTPSW			
6	Next_To_Execute_PCT_Address	Q001.3	I	See App. E	See App. E	TCVTNEW			
7	FMPT_HDR_Bits	T070	O	See App. E	See App. E	CDLBERRM			

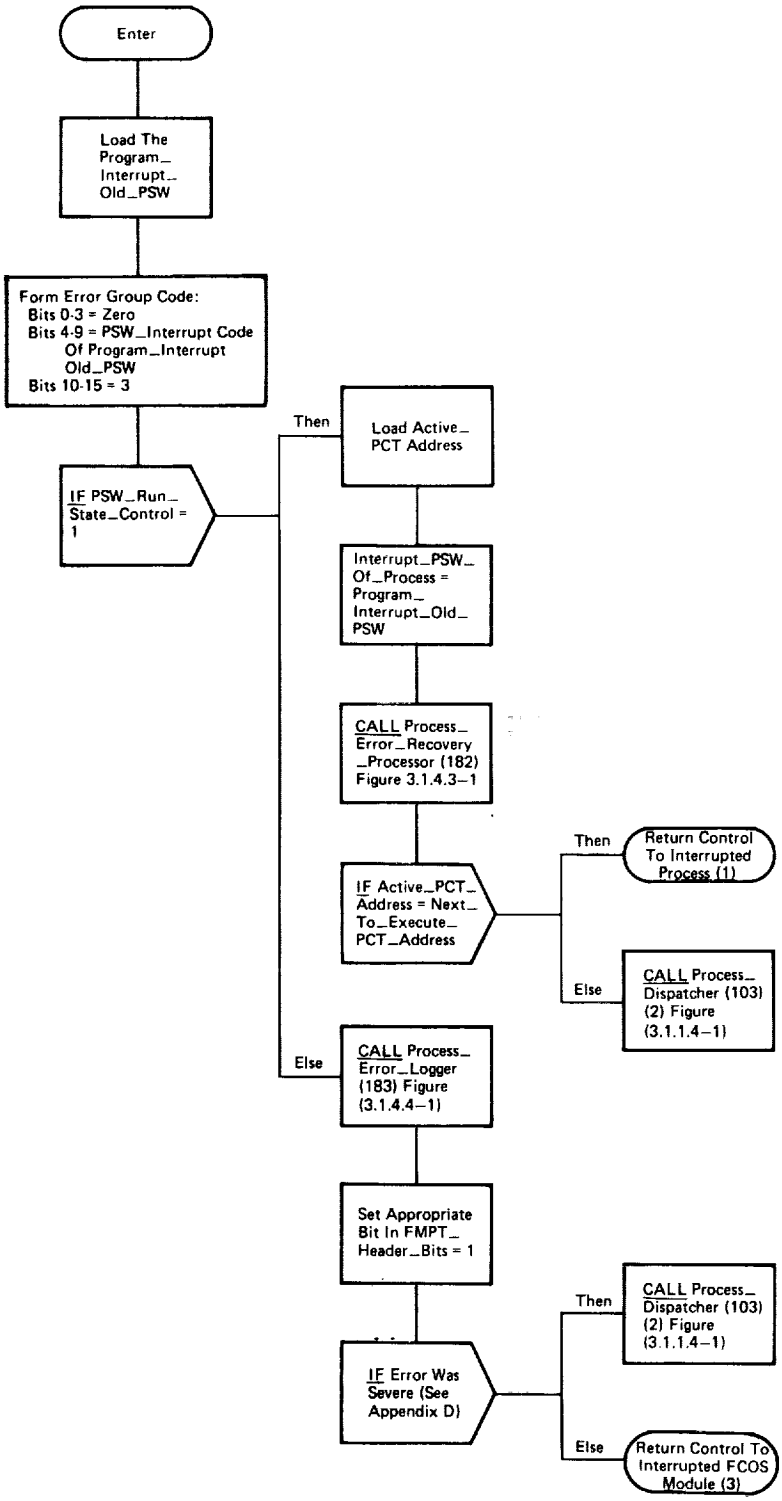


Figure 3.1.4.1-1. Program_Interrupt_Handler (FPMIHPGM)

3.1.4.2 Instruction_Monitor_Interrupt_Handler (FPMIHIM) (181)

FPMIHIM fields instruction monitor interrupts and either processes the error (if it occurred during FCOS execution) or invokes the Process_Error_Recovery_Processor to process the error (if the error occurred during application execution).

a. Control Interface - Hardware PSW swap because of instruction monitor interrupt.

b. Input - See Table 3.1.4.2-1.

c. Process Description - The Instruction_Monitor_Interrupt_Handler forms the error group-code identifying the error using a hardcoded error group-code of 3-20.

If the PSW_Run_State_Control of the Instruction_Monitor_Old_PSW indicates problem state, the error has occurred during application execution and the Process_Error_Recovery_Processor is invoked to process the error according to the error environment of the process in error. After the error has been processed, control is either returned to the process in error or to another process depending on how the error was processed.

If the PSW_Run_State_Control of the Instruction_Monitor_Old_PSW indicates supervisor state, the error occurred during FCOS execution and the error is processed in the following manner. The Process_Error_Logger is invoked to log the error. The appropriate indicator for instruction monitor interrupts is set in the FMPT_HDR_Bits to cause annunication of the error. Since instruction monitor interrupts are severe interrupts, control is always passed to the highest priority ready application process following an instruction monitor interrupt during FCOS execution.

The control flow for this module is presented in Figure 3.1.4.2-1.

d. Output - See Table 3.1.4.2-1.

e. Module References -

1. (182) Process_Error_Recovery_Processor (FPMSDERR) is CALLed.
2. (183) Process_Error_Logger (FPMERLOG) is CALLed.
3. (103) Process_Dispatcher (FPMDISP) is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - An annunication indicator is set for FCOS instruction monitor interrupts so that an errcr message is displayed (OS040).

i. Constraints and Assumptions - The PSW_Run_State_Control is assumed to be 0 (Supervisor state) for all FCOS execution and 1 (Problem State) State) for all application execution.

j. Detailed Implementation -

1. Control is passed via a PSW load of the Interrupt_PSW_Of_Process.
2. There is no return from the CALL to the Process_Dispatcher.

DATA TABLE 3.1.4.2-1

NAME Instruction_Monitor_Interrupt_Handler(FPMIHIM)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Instruction_Monitor_Old_PSW	&001.73	I		181	TPSAIMOP			
2	PSW_Run_State_Control	&002.25	I		180,181, 320				
3	Active_PCT_Address	Q001.2	I	103	See App. E	TCVTOLD			
4	Interrupt_PSW_of_Process	Q003.5	O	See App. E	See App. E	TPCTPSW			
5	Next_to_Execute_PCT_Address	Q001.3	I	See App. E	See App. E	TCVTNEW			
6	FMPT_HDR_Bits	T070	O	See App. E	See App. E	CDLBERRM			

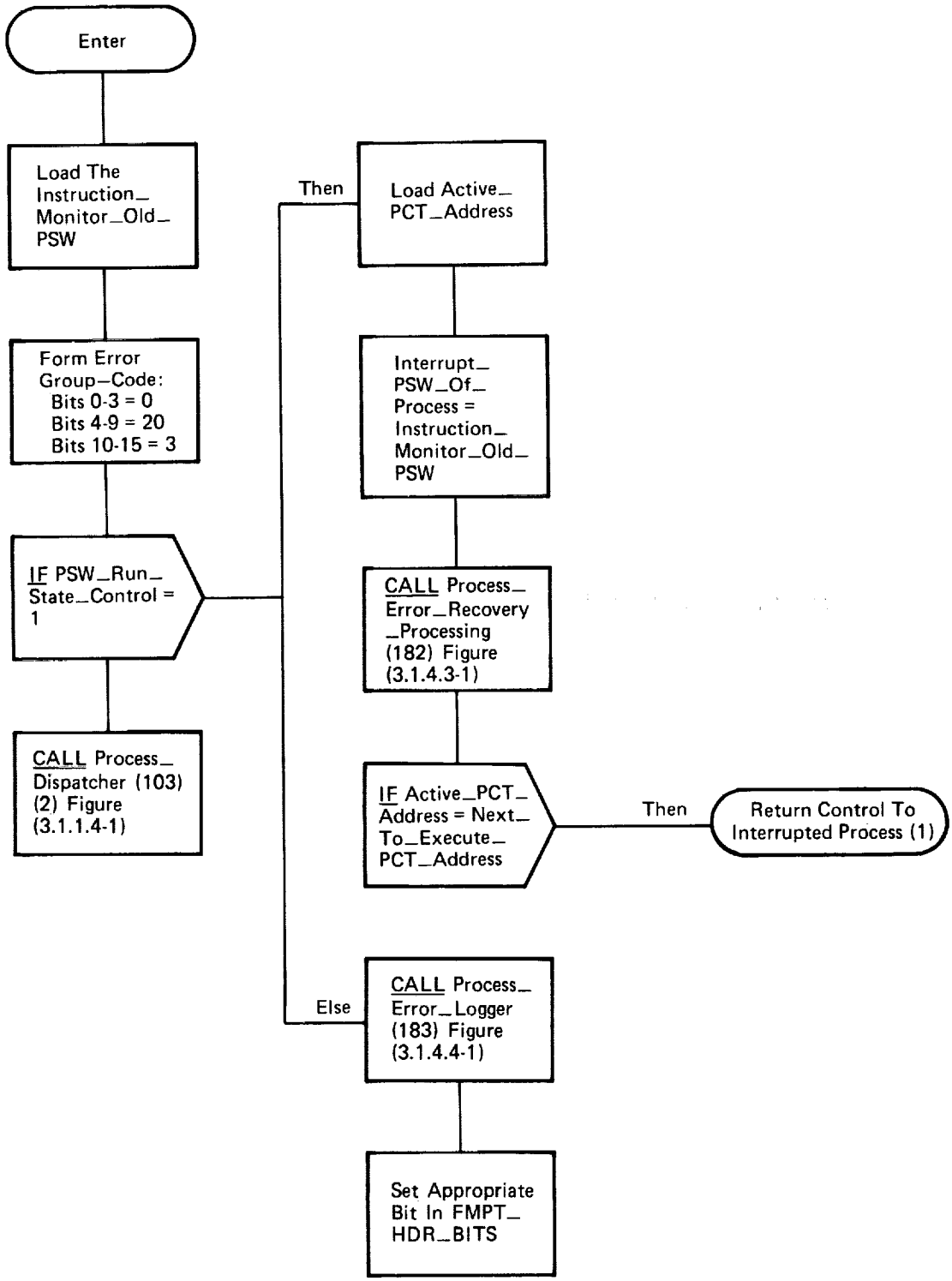


Figure 3.1.4.2-1. Instruction_Monitor_Interrupt_Handler (FPMIHIM)

3.1.4.3 Process_Error_Recovery_Processor (FPMSDERR) (182)

FPMSDERR processes errors according to the error environment defined by the application programs via the ON-ERROR statement.

a. Control Interface -

1. Called by the (100)SVC_Handler when a supervisor call 20 is executed.
2. Called by the (180) Program_Interrupt_Handler to process program interrupts.
3. Called by the (181) Instruction_Monitor_Interrupt_Handler to process instruction monitor interrupts.
4. Called by the (142) TQE_Expiration_Processor to process cycle overrun errors.

b. Input -

1. Input to Process_Error_Recovery_Processor when entry is via an SVC.
   Register 0 - bits 0-15 address of SVC parameter list
   Register 7 - bits 0-15 return address
2. Input to Process_Error_Recovery_Processor when entry is via a CALL
   Register 0 - bits 0-19 are zero
   Register 0 - bits 20-25 error code
   Register 0 - bits 26-31 error group code
   Register 1 - bits 0-15 address of PCT for process in error
   Register 7 - bits 0-15 return address
3. Other Inputs - See Table 3.1.4.3-1.

c. Process Description - The Process_Error_Recovery_Processor searches the Error_Vectors of the process which had the error. If an Error_Vector is found which defines what to do for this error, the Process_Error_Recovery_Processor performes the recovery defined in the Error_Vector, otherwise, it performs the system default action.

The application program via the ON ERROR statement has the capability of specifying one of the following actions in the Error_Vectors:

1. SET/RESET/SIGNAL an Event_Variable and ignore the error.

2. SET/RESET/SIGNAL an Event_Variable and perform the system default action for the error.

3. GO TO a specified location when return is made to the process in error.

4. Perform the system default action for the error.

5. Ignore the error.

If action 2, 3, or 4 is to be performed, the Process_Error_Recovery_Processor calls the Process_Error_Logger. The Event_Evaluator is invoked for the event processing associated with actions 1 and 2.

Table 3.1.4.3-2 lists the errors processed by the Process_Error_Recovery_Processor along with the system default action for each.

The control flow for this module is presented in Figures 3.1.4.3-1 through 3.1.4.3-7.

d. Output - See Table 3.1.4.3-1.

e. Module References -

1. (183) Process_Error_Logger (FPMLOGER) is called.
2. (106) UPDATE/EXCLUSIVE_Release_Processor (FPMREL) is called.
3. (174) Event_Evaluator (FPMEVAL) is called.
4. (184) Forced_Close_Processor (FPMFCLOS) is called.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - See Table 3.1.4.3-2.

i. Constraints and Assumptions - None

j. Detailed Implementation -

1. Machine check errors are no longer processed by the Process_Error_Recovery_Processor.

2. Currently the only Send_Error_Code in the Send_Error_Group_Number two is 13 (cycle wrap).

DATA TABLE 3.1.4.3-1

NAME Process_Error_Recovery_Processor (FPMSDERR)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Send_Error_Group_Number	S020.2	I		182				
2	Send_Error_Code_Number	S020.3	I		182				
3	Error_Vectors	@002.15	I		182				
4	Stack_Frame	@002	I	101	182,290	TFTSA			
5	Active_PCT_Address	Q001.2	I	103	See App. E	TCVTOLD			
6	FCOS_Work_Area	&001.88	O	See APP. E		TPSAWORK			
7	Process_Interrupt_Register_0	Q003.6	IO	101,103 107,182	103,108 182	TPCTGPRO			
8	Error_Action_Code	@002.16	I		182				
9	PSW_Register_Set	&002.22	O	182	320				
10	Local_Data_Area_Address	@002.9	I	101	108,182, 184,290	TTSALCDA			
11	Previous_Stack_Frame_Address	@002.3	I	101	108,182 184	TTSAPREV			
12	Number-of-Error_Vectors	@001.4	I		182				
13	Update/Exclusive_Indicator	@001.3	I		108, 182,184				
14	Release_SVC_Parameter_List	@001.7	I		108,182, 184	TLDARELP			
15	Error_Group	@002.18	I		182				
16	PSW_Instruction_Address	&002.1	O	182					
17	Interrupt_PSW_of_Process	Q003.5	O	See Appendix E		TPCTPSW			
18	Error_Variable_Field	@002.19	I		182				
19	PSW_Program_Mask	&002.2	O		182				
20	PSW_Program_Mask_Field	@002.2	I	101	182	TTSAPGMM			

DATA TABLE 3.1.4.3-1 (Cont'd)

NAME Process_Error_Recovery_Processor (FPMSDERR)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
21	Program_Level_Data_Address	@002.6	I		182	TTSAPLDA			
22	Process_Interrupt_Register_1	Q003.7	O	103,182 320	103,320	TPCTGPR1			
23	Process_Interrupt_Register_3	Q003.9	O	103,182 320	103,320	TPCTGPR3			
24	Event_State	@007.2	O	See Appendix E					
25	Event_Variable	@007	IO	182	See App.E				
26	Event_Used_Indicator	@007.1	I		See App.E				
27	FMPT_HDR_Bits	T070	O	See Appendix E		CDLBERRM			
28	Program_Interrupt-Count	Q003.28	IO	101,107	182	TPCTECNT			
29	Maximum_Program_Checks_Per_Process	Q001.21	I		182	TCVTIMAX			

ERROR Group-Code		ERROR CONDITION	SYSTEM DEFAULT ACTION (Application Errors)
02	0D	Attempt to start cycle of cyclic process still in previous cycle	Skip cycle (ignore timer interrupt) and continue (OS032)
03	00*	Illegal Operation Code	FORCE CLOSE process and continue (OS034)
	01*	Privileged Instruction	FORCE CLOSE process and continue (OS034)
	03*	CPU Address Specification	FORCE CLOSE process and continue (OS034)
	04	Fixed-Point Overflow	See action for 0309-030C
	05	Significance	See action for 0309-030C
	07*	CPU Protection Violation	FORCE CLOSE process and continue (OS034)
	09	Exponent Underflow (Floating Point)	Bump count of program interrupts for this process
	0A	Overflow (convert)	and
	0B	Exponent Overflow (Floating Point)	Ignore if count max or
	0C	Divide (Floating - Point)	FORCE CLOSE process if count=max (OS034)
	20*	Instruction Monitor	FORCE CLOSE process and continue (OS034)

* Application Override specifying IGNORE is not allowed for these errors.

Table 3.1.4.3-2 Errors Processed by Process_Error_Recovery_Processor

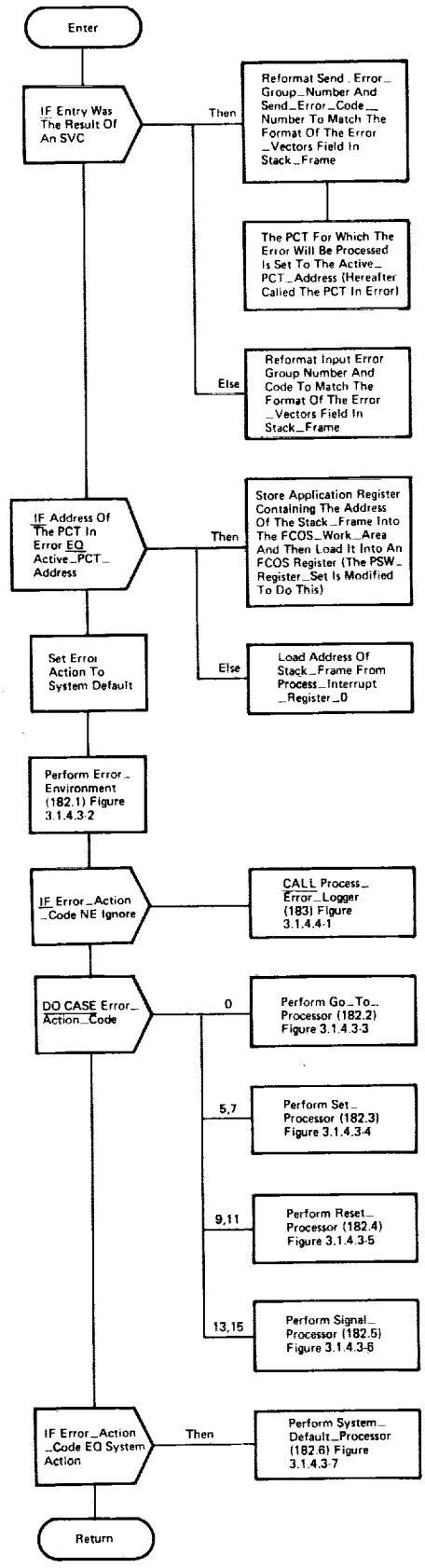


Figure 3.1.4.3-1. Process_Error_Recovery_Processor (FPMSDERR)

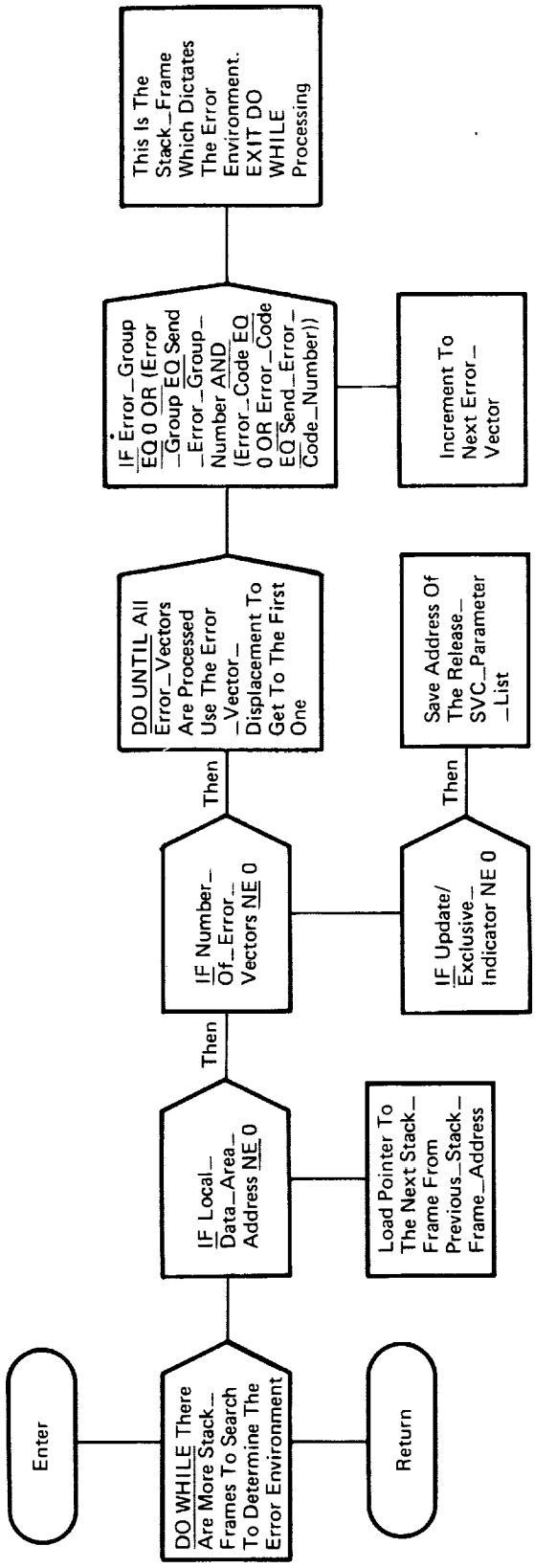


Figure 3.1.4.3-2. Process_Error_Recovery_Processor Error_Environment (182.1)

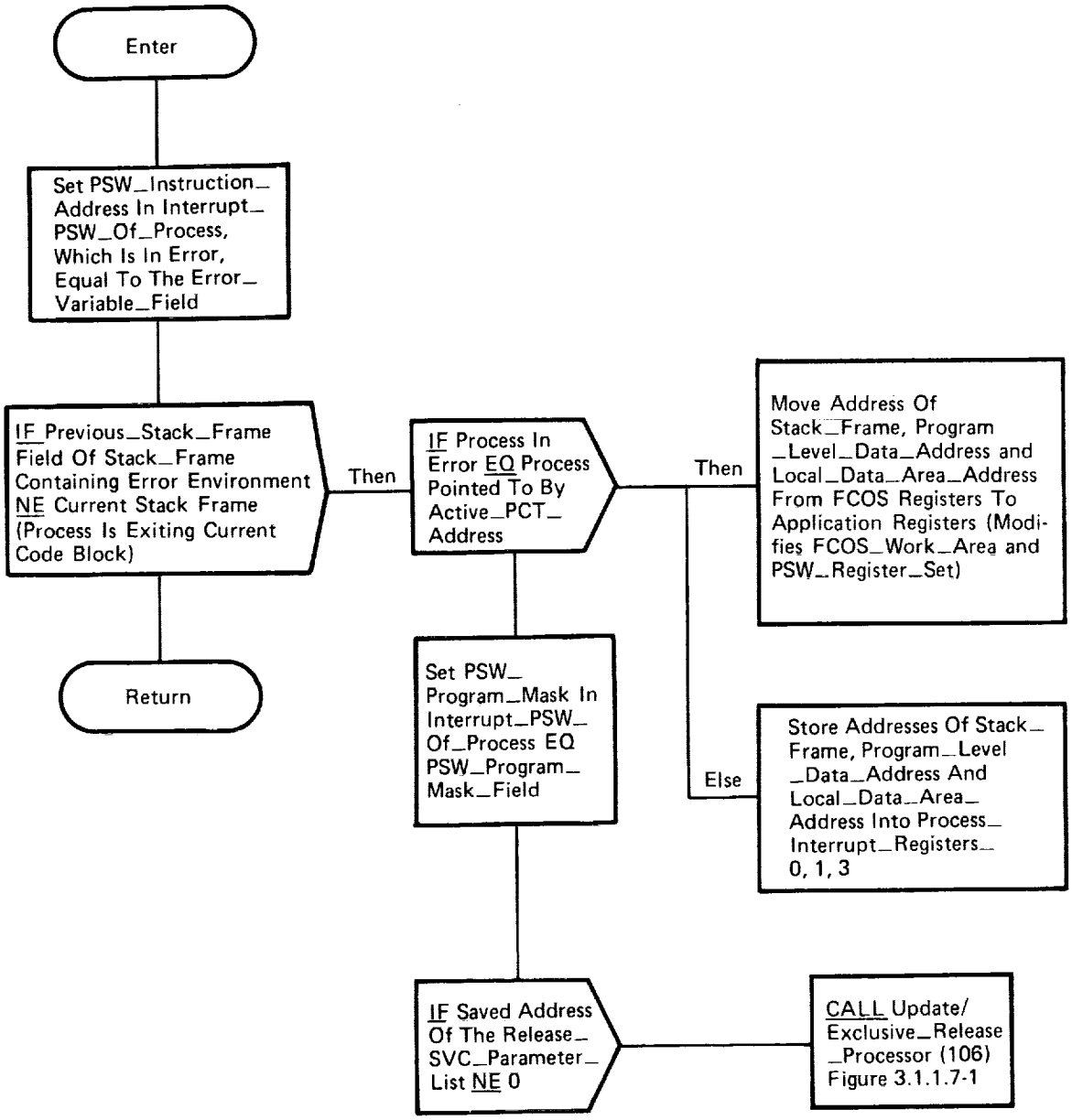


Figure 3.1.4.3-3. Process_Error_Recovery_Processor Go_To_Processor (182.2)

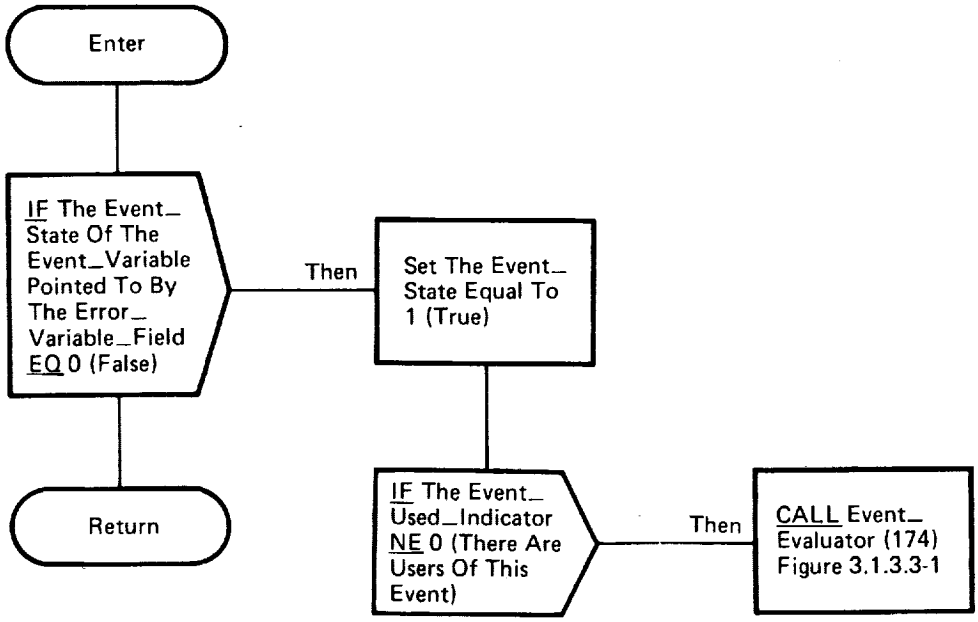


Figure 3.1.4.3-4. Process_Error_Recovery_Processor Set_Processor (182.3)

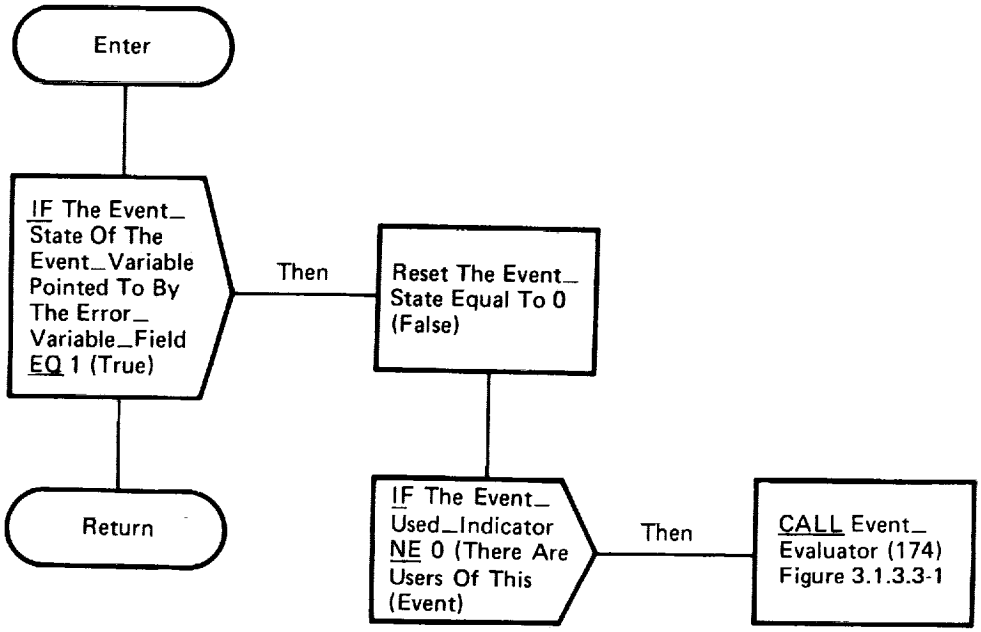


Figure 3.1.4.3-5. Process_Error_Recovery_Processor Reset_Processor (182.4)

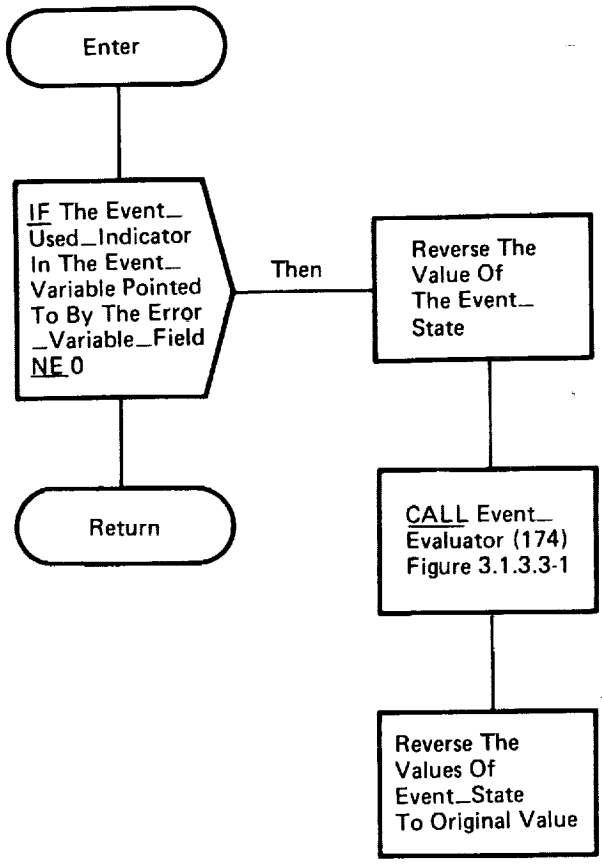


Figure 3.1.4.3-6. Process_Error_Recovery_Processor Signal_Processor (182.5)

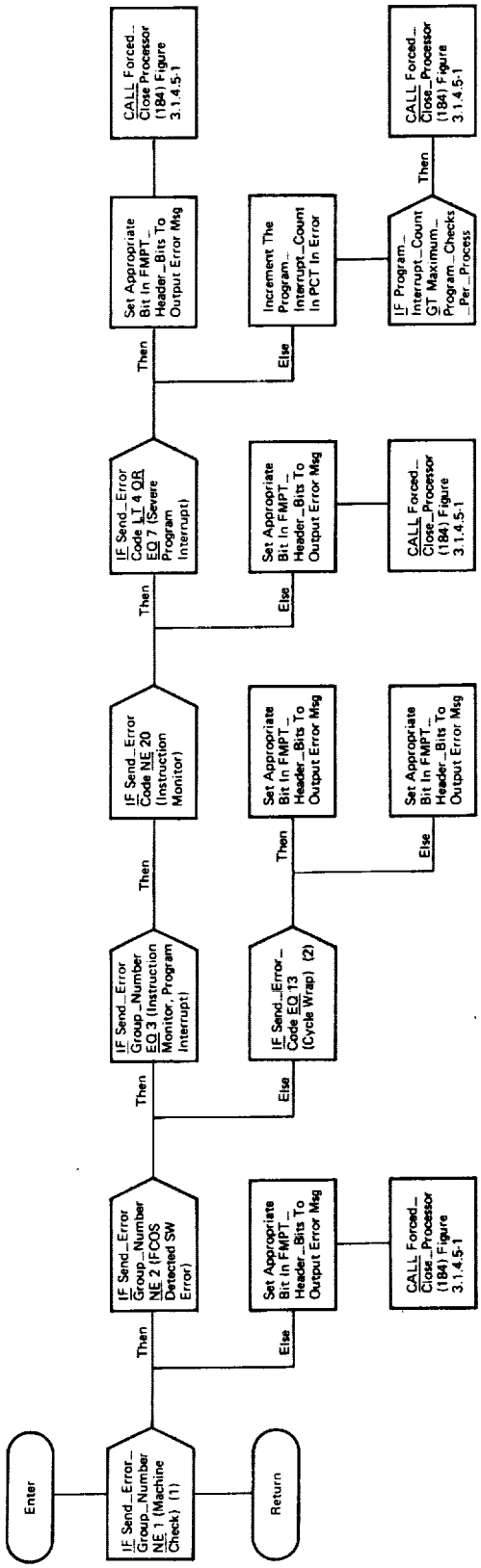


Figure 3.1.4.3-7. Process_Error_Recovery_Processor System_Default_Processor (182.6)

3.1.4.4 Process_Error_Logger (FPMERLOG) (183)

FPMERLOG logs GPC error conditions which have occurred.

a. Control Interface -

1. CALLed by (180) Program_Interrupt_Handler (FPMIHPGM)
2. CALLed by (181) Instruction_Monitor_Interrupt_Handler (FPMIHIM).
3. CALLed by (182) Process_Error_Recovery_Processor (FPMSDERR).

b. Input -

1. Register 0 contains the error group-code in the following format:

0	0	0	0	6-bit code	6-bit group

2. Register 1 contains the address of PCT of the process in error (Zero if FCOS error).

3. Register 3 contains first word of the appropriate interrupt old PSW if FCOS error.

c. Process Description - The Process_Error_Logger updates the GPC_Error_Log with information regarding the error condition which has occurred. This information consists of the following:

1. Time of error.

2. Error group-code.

3. Address from the PSW in effect at the time of the error or the PSW associated with the process in error (cycle overrun errors).

4. Sector information clarifying the address.

This information is placed in two locations: The appropriate entry in the GPC_Self_Circular_GPC_Error_Log and the correct entry in Latest_GPC_Error_Entries for this GPC. The GPC_X_Error_Count is incremented by 1 for this GPC. The ICC_Status_Flags word 2 is updated to cause ICC transfer of the Latest_GPC_Error_Entries field for this GPC. The Circular_GPC_Error_Log_Index is then updated to index to the next available entry into the GPC_Self_Circular_GPC_Error_Log.

The control flow for this module is presented in Figure 3.1.4.4-1.

d. Output - See Table 3.1.4.4-1.

e. Module References -

(162) Current_GMT_Routine (FPMGMTIM) is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - N/A

DATA TABLE 3.1.4.4-1

NAME Process_Error_Logger (FPMERLOG)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	GPC_Error_Log	I290			183,485, 660,665	CZ2VERLG			
2	GPC_ID	#001	I	300,	See App. E	TFCMID			
3	Latest_GPC_Error_Entries	I290.9	0	183	485,660, 665	TLOGLSTE			
4	GPC_X_Error_Count	I290.10	0	183		TLOGLSTN	V92Q4288C V92Q4296C	X	
							V92Q4306C V92Q4314C		
5	Last_Error_Group_Code	Q003.27	0	101,107, 183	185	TPCTERR			
6	Interrupt_PSW_of_Process	Q003.5	I	See APP. E		TPCTPSW			
7	Circular_GPC_Error_Log_Index	I290.1	I,O	183	183,660, 665	TLOGINDX	V92Q4250C	X	
8	GPC_Self_Circular_GPC_Error_Log	I290.2	0	183	660,665	TLOGNTRY		X	
9	GPC_Error_Time_Tag	I290.3	0	183		TLOGSLFT	V92W4256C V92W4264C	X	
							V92W4270C V92W4276C		
							V92W4282C		
10	GPC_X_Error_Time_Tag	I290.11	0	183		TLOGLSTT	V92W4286C V92W4294C	X	
							V92W4304C V92W4312C		
11	PSW_Branch_Sector_Register	&002.10	I		183				
12	GPC_Error_Identification	I290.4	0	183		TLOGSLFC	V92U4252C V92U4258C	X	
							V92U4266C V92U4272C		
							V92U4278C		
13	GPC_X_Error_Identification	I290.12	0	183		TLOGLSTC	V92U4284C V92U4290C	X	
							V92U4300C V92U4308C		

DATA TABLE 3.1.4.4-1 (Cont'd)

NAME Process_Error_Logger (FPMERLOG)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
14	GPC_Error_Address	I290.8	O	183		TLOGSLFA	V92U4254C V92U4260C	X	
							V92U4268C V92U4274C		
							V92U4280C		
15	GPC_X_Error_Address	I290.13	O	183		TLOGSTA	V92U4285C V92U4292C	X	
							V92U4302C V92U4310C		
16	PSW_Instruction_Address	&002.1	I						
17	ICC_GPC_ERR_GPC1	I320.27	O	183		CZ2VIF2			
18	ICC_GPC_ERR_GPC2	I320.28	O	183		CZ2VIF2			
19	ICC_GPC_ERR_GPC3	I320.29	O	183		CZ2VIF2			
20	ICC_GPC_ERR_GPC4	I320.30	O	183		CZ2VIF2			
21	GPC_ERROR_LOG_ICC_Indicator_Mask	Q001.57	I	300	183	TCVTGERM			
22	Error_Log_Table_Maximum_Index	Q001.20	I		183	TCVTLMAX			

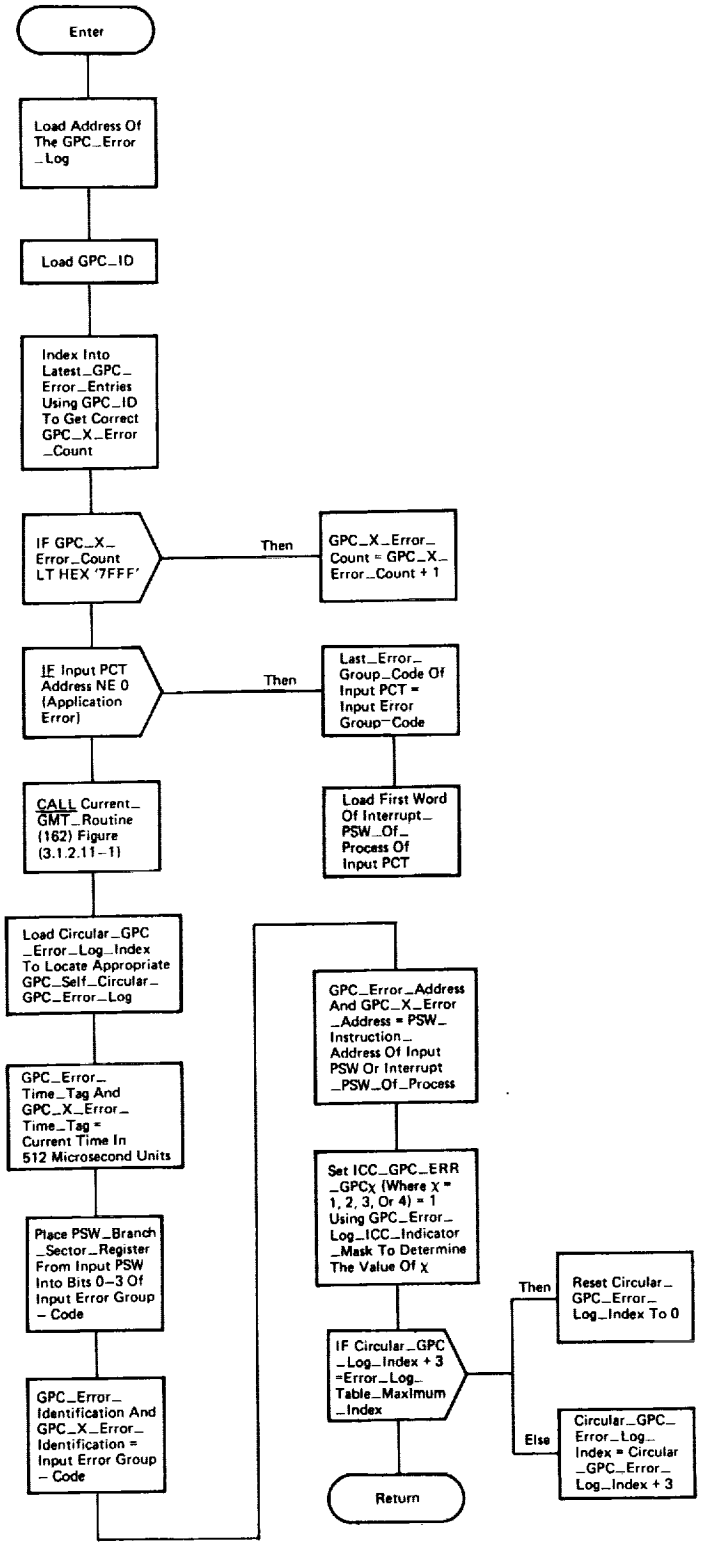


Figure 3.1.4.4-1. Process_Error_Logger (FPMERLOG)

3.1.4.5 Forced_Close_Processor (FPMFCLØS)(184)

FPMFCLØS performs the clean up and/or reinitialization functions necessary when a process encounters certain error conditions.

a. Control Interface -

CALLed by (182) Process_Error_Recovery_Processor(FPMSDERR)

b. Input - Register 0 contains the address of the PCT to be closed. See Table 3.1.4.5-1 for other input.

c. Process Description - The Forced_Close_Processor CALLs the Exclusive_Release_Processor if the process in error was within an UPDATE block or EXCLUSIVE procedure (Update/Exclusive_Indicator is on). The Interrupt_PSW_of_Process is modified to point to a dummy CLOSE SVC in case the Close_Processor must re-issue a CLOSE SVC. Then, the Close_Processor is CALLed. The control flow for this module is shown in Figure 3.1.4.5-1.

d. Output - See Table 3.1.4.5-1.

e. Module References -

1. (106) Update/Exclusive_Release_Processor(FPMREL) is CALLed.
2. (107) Close_Processor(FPMCLØSE) is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - The process to be forced closed is the current process.

j. Detailed Implementation -

1. The Close_Processor is CALLed to clean-up and/or reinitialize the process (cyclic processes only); therefore, there is no return from this call.
2. A CLØSE SVC may need to be reissued out of the Close_Processor after timer or I/O interrupts are enabled during sync processing.

DATA TABLE 3.1.4.5-1

NAME Forced_Close_Processor (FPMFCLOS)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Previous_Stack_Frame_Address	@002.3	I	101	180,182, 184	TTSAPREV			
2	Local_Data_Area_Address	@002.9	I	101	108,182 184,290	TTSALCDA			
3	Update/Exclusive_Indicator	@001.3	I		108,182 184	TLDAEINF			
4	Release_SVC_Parameter_List	@001.7	I		108,182 184	TLDARELP			
5	Interrupt_PSW_of_Process	Q003.5	O	See App. E	See App. E	TPCTPSW			

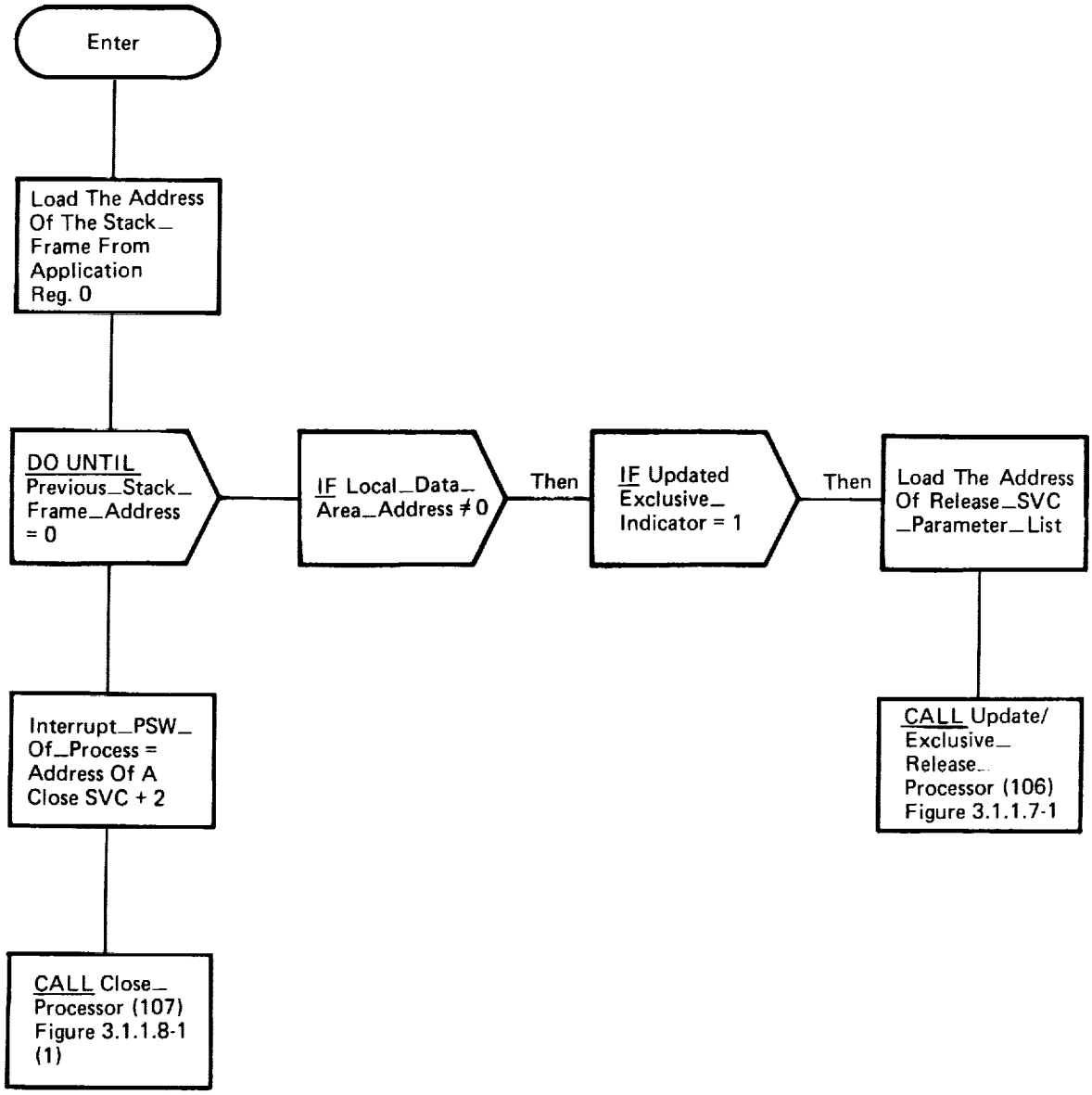


Figure 3.1.4.5–1. Forced_Close_Processor (FPMFCLOS)

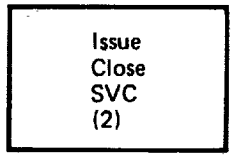


Figure 3.1.4.5-2. Forced_Close_Processor

3.1.4.6 Application_Error-Number_Request_Processor (FPMSTAT) (185)

FPMSTAT processes HAL/S ERRNUM/ERRGRP functions.

a. Control Interface - CALLed by (100) SVC_Handler (FPMSVC).

b. Input - Register 0 contains the address of the Process_Status_Parameter_List. See Table 3.1.4.6-1.

c. Process Description - The Application_Error_Number_Request_Processor examines the Status_Request_Type to determine which function has been requested.

   If ERRGRP has been requested, the error group of the last error which has occurred during execution of the requesting process is extracted from the Last_Error_Group_Code field of the active PCT.

   If ERRNUM has been requested, the error code is extracted from the Last_Error_Group_Code field of the active PCT. The requested information is placed in the application general purpose register 5 for return to the application. The control flow for this module is presented in Figure 3.1.4.6-1.

d. Ouptput - Application register 5 contains the requested information.

e. Module References - None

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - N/A

DATA TABLE 3.1.4.6-1

NAME Application_Error_Number_Request_Processor (FPMSTAT)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Active_PCT_Address	Q001.2	I	103	See App. E	TCVTOLD			
2	Status_Request_Type	S023.1	I		185				
3	Last_Error_Group_Code	Q003.27	I	101,107 183	185	TPCTERR			

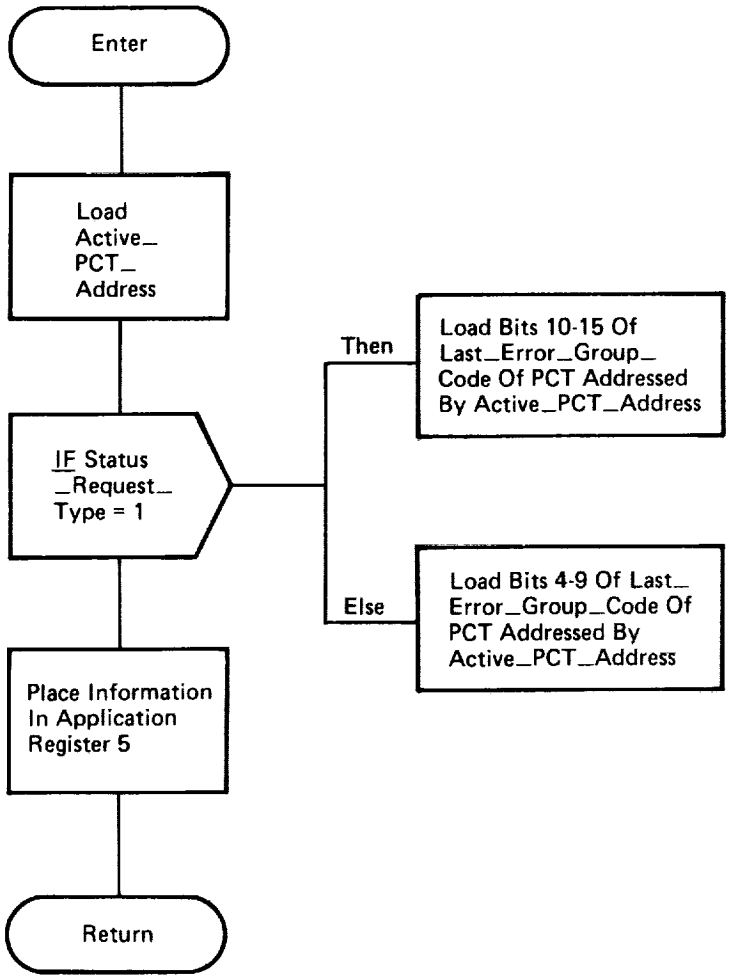


Figure 3.1.4.6-1. Application_Error_Number_Request_Processor (FPMSTAT)

3.1.5 Idle Time Processor (FPMIDLE) (191)

FPMIDLE receives control when there is no processing to be done. It approximates the total amount of idle time which occurs between computations of CPU duty cycle.

a. Control Interface - Pre-scheduled (priority 0).

b. Input - Floating point register 1 has delta time of Idle_Time_Processor. See Table 3.1.5-1.

c. Process Description - The Idle_Time_Processor is an infinite loop which consists of loading Idle_Time, adding to it a value which is the approximate time of the Idle_Time_Processor, and storing this new value as Idle_Time. This Idle_Time is used by TQE_Expiration_Processor to compute duty cycle. The Idle_Time_Processor is a pre-scheduled process of priority 0 and is always on the PCT run queue. It receives control through regular processing of the FCOS Process_Dispatcher when no other process is ready to execute. The control flow for this module is presented in Figure 3.1.5-1.

d. Output - See Table 3.1.5-1.

e. Module References - None

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - N/A

DATA TABLE 3.1.5-1

NAME Idle_Time_Processor (FPMIDLE)

NO.	ITEM	ID	ACT	SOURCE	DESTINATION	ASSEMBLER NAME	MML	D	C
1	Idle_Time	@003	I O	142 191	142 191	FPMIDTIM			

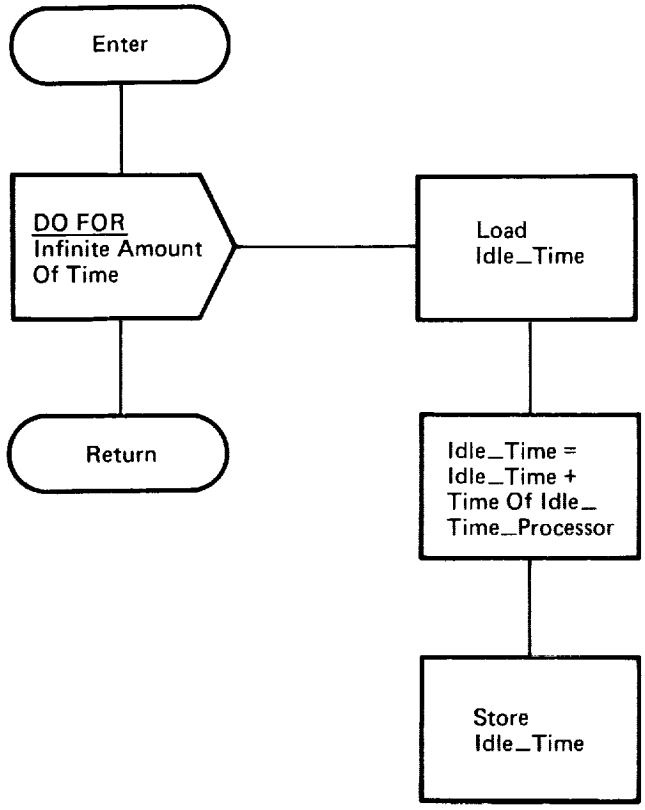


Figure 3.1.5. Idle_Time_Processor (FPMIDLE)

3.2 I/O Management

To Be Provided

3.2.1 I/O SVC Servicing

To Be Provided

3.2.1.1 I/O_SVC_Service_Processor (FIOSVC) (200)

The I/O_SVC_Service_Processor is called to initialize a free IOQE and to either pass that IOQE to the IOP_Dispatcher or Mass_Memory_Manager to start the I/O operation or, if necessary resources are not available, add the IOQE to the I/O wait queue.

a. Control Interface -

1. CALLed by (100) SVC_Handler (FPMSVC) at entry point FIOSVC.

2. CALLed by (148) MTU_Update_Processor (FPMUPMTU) at entry point FIOSVC1.

3. CALLed by (361) Common_Set_Synchronization_Processor (FCMCSYNC) at entry point FIOSVC1.

4. CALLed by (220) I/O_Completion_Processing (FIOCMPLT) at entry point FIOSVC1.

b. Input - See Table 3.2.1.1-1.

c. Process Description - The SVC_Synchronization_Processor is called to synchronize redundant set GPC's. An alternate entry point (FIOSVC1) is provided for use when synchronization is not necessary.

The top free IOQE is then initialized from the I/O_SVC_Parameter_List. The mask of buses needed to service the I/O request is obtained from the BCE_Device_Mask_Table and put into the IOQE. If an Event_Variable is to be set on I/O completion the Event_Variable is reset. The requesting PCT's Outstanding_I/O_Count is incremented by one.

After initialization the IOQE is removed from the I/O free pool.

If the requesting process wants to wait for the I/O to complete its PCT is flagged waiting for I/O.

At this point Mass Memory I/O requests are given to the Mass_Memory_Manager while all other requests are checked to see if they are dispatchable. An I/O request is dispatchable if the buses used in the I/O are not marked busy in the BCE_Busy_Idle_Indicators. A dispatchable request is given to the IOP_Dispatcher but requests that are not dispatchable are added to the I/O wait queue. The I/O wait queue is ordered by IOQE_Priority with the highest priority waiting IOQE being first in the queue.

The I/O_SVC_Service_Processor returns to the program which called it. The control flow for this module is presented in Figure 3.2.1.1-1.

d. Outputs - See Table 3.2.1.1-1.

e. Module References -

1. (280) Mass_Memory_Manager (FIOMMMGR) is CALLed.
2. (205) IOP_Dispatcher (FIOPDISP) is CALLed.
3. (363) SVC_Synchronization_Processor (FCMSSYNC) is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation -

1. This alternate entry point is provided for use when GPC's are already synchronized. To use this entry point Register 2 bits 0-15 must contain the requestor's PCT address and Register 3 bits 0-15 must contain the I/O_SVC_Parameter_List address. Bits 16-31 of both regsiters must be zero.

2. Note that no check is made to determine if a process' execution depends on the FALSE state of the Event_Variable.

3. FCOS uses the alternate entry point (FIOSVC1). Register 2 (PCT address) will be zero for FCOS I/O requests.

DATA TABLE 3.2.1.1-1

NAME I/O_SVC_Service_Processor (FIOSVC)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Active_PCT_Address	Q001.2	I	103	See Appendix E	TCVTOLD			
2	PSW_Interrupt_Code	&002.26	I		180,242, 320				
3	SVC_Old_PSW	&001.67	I		See Appendix F	TPSASOP			
4	I/O_Free_Pool_Address	Q001.12	I,O	See Appendix E	225	TCVTIOFP			
5	Address_Of_Next_IOQE_In_Chain	Q007.1	O	200,201 225	205 225	TIOQNXT			
6	Wait_For_I/O	Q007.33	I,O	200	200 201	TIOQWAIT			
7	I/O_Wait_Indicator	Q003.45	O	200 201	103 105				
8	Next_To_Execute_PCT_Address	Q001.3	O	See Appendix E	See Appendix E	TCVTNEW			
9	Mask_Of_Buses_To_Monitor	Q007.4	I,O	200,201 351	200,205 251	TIOQMNTM			
10	BCE_Busy/Idle_Indicators	Q001.17	I	See Appendix E	See Appendix E	TCVTBCEB			
11	I/O_SVC_Parameter_List	S024	I			TFIOS			
12	IOQE_Flags_Field_2	Q007.23	O,I	200	200 201	TIOQFLG2			
13	I/O_SVC_Parameter_List_Flags	S024.2	I		200	TIOSFLGS			
14	IOQE_Flags_Field_1	Q007.6	O	200,201 225	201 225	TIOQFLG1			
15	IOQE_Operation_Code	Q007.44	O	200	205	TIOQOPCD			
16	Operation_Code	S024.18	I	148	200	TIOSOPCD			
17	IOQE_Word_Count	Q007.45	O	200	205	TIOQWDCD			
18	Word_Count_Number	S024.20	I		200	TIOSWDCT			
19	IOQE_Device_Id	Q007.43	O	200	205,251 351	TIOQDVID			
20	Device_Id_Number	S024.17	I	148	200	TIOSDVID			

DATA TABLE 3.2.1.1-1 (cont'd)

NAME I/O_SVC_Service_Processor (FIOSVC)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
21	BCE_Device_Mask_Table	Y003	I	351 390	200,201 220,244	FIOBCD			
22	Address_Of_I/O_Buffer	Q007.5	O	200	149,205	TIOQBUFA			
23	I/O_Buffer_Address	S024.21	I	148	200	TIOSBUFA			
24	IOQE_Completion_Event_Address	Q007.42	O	200	201	TIOQEVNT			
25	Event_Address	S024.22	I	148	200	TIOSEVNT			
26	Event_Variable	@007	I	182	See App. E				
27	Event_State	@007.2	O	See App. E					
28	IOQE_Starting_Address	Q007.46	O	200	205	TIOQSTAD			
29	Miscellaneous_Communication	S024.23	I		200	TIOSSTAD			
30	IOQE_Priority	Q007.40	O,I	200 205	200,201 205	TIOQPRI			
31	I/O_Priority	S024.19	I		200	TIOSPRI			
32	Outstanding_I/O_Count	Q003.25	O	101,200 201	133	TPCTIORQ			
33	I/O_Wait_Queue_Address	Q001.6	O	See Appendix E	See Appendix E	TCVTIOW			
34	Address_Of_Next_IOQE_In_Chain	Q007.1	O	200,201 225	205,225	TIOQNXT			

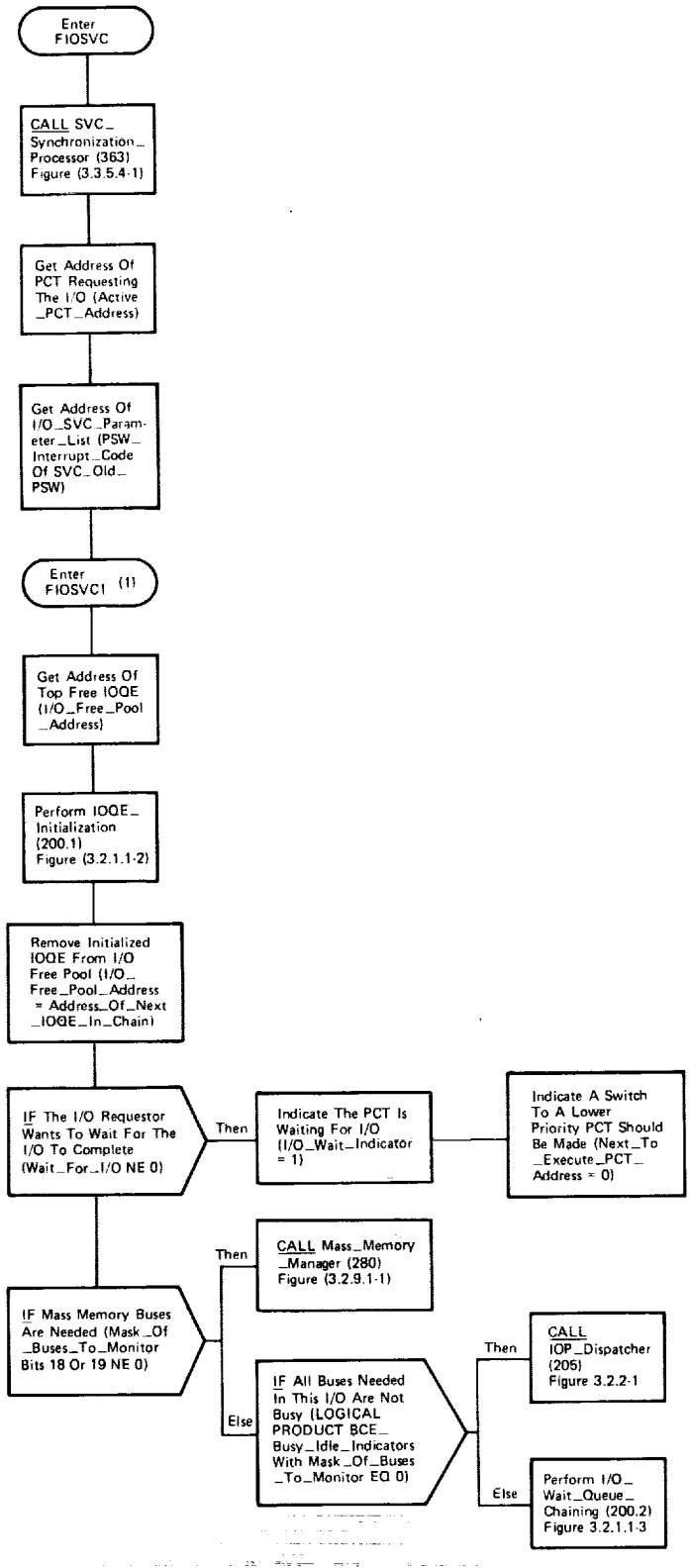


Figure 3.2.1.1-1. I/O_SVC_Service_Processor (FIOSVC)

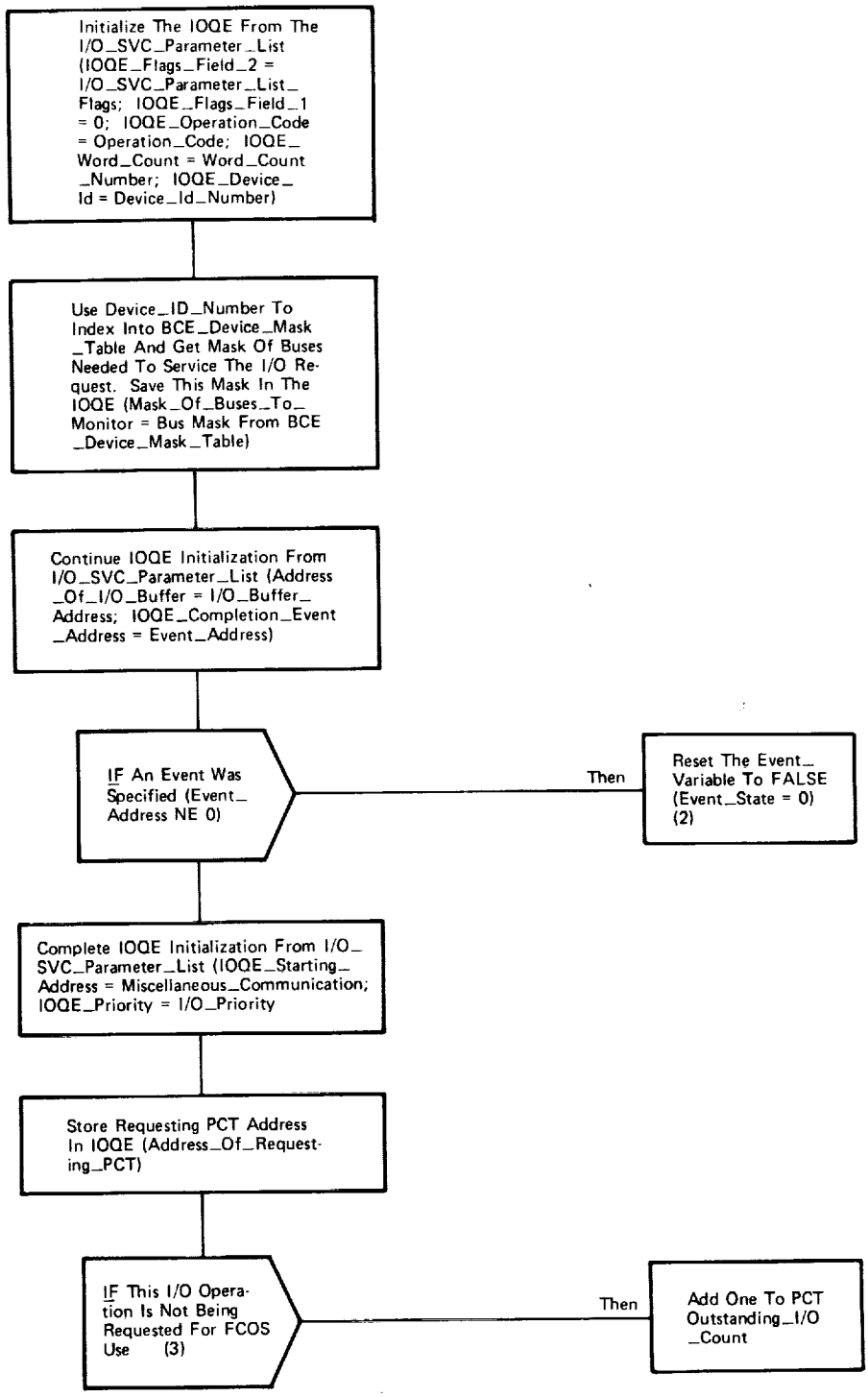


Figure 3.2.1.1-2. I/O_SVC_Service_Processor IOQE_Initialization (200.1)

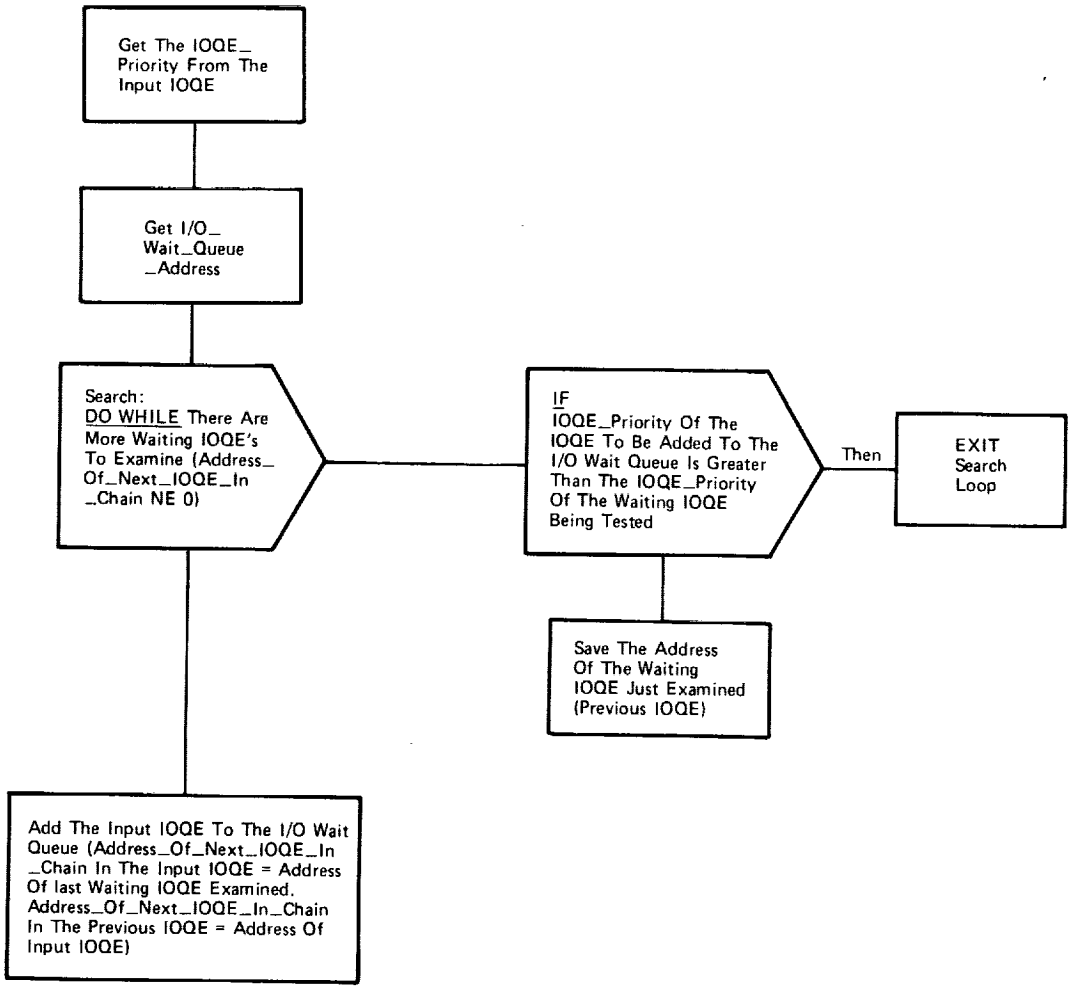


Figure 3.2.1.1-3. I/O_SVC_Service_Processor I/O_Wait_Queue_Chaining (200.2)

3.2.1.2 Pre-Initialized_I/O_SVC_Processor (FIOSVCP) (201)

The Pre-Initialized_I/O_SVC_Processor is CALLed to complete the initialization of a preinitialized IOQE and to either pass the IOQE to the IOP_Dispatcher to start the I/O operation or, if necessary resources are not available, add the IOQE to the I/O wait queue.

a. Control Interface -

1. CALLed by (100) SVC_Handler (FPMSVC) at entry point FIOSVCP.
2. CALLed by (140) Timer_Queue_Generator (FPMTMENQ) at entry point FIOSVCP1.

b. Inputs - See Table 3.2.1.2-1.

c. Process Description - The SVC_Synchronization_Processor is CALLed to synchronize redundant set GPC's. An alternate entry point (FIOSVCP1) is provided for use when synchronization is not necessary.

The preinitialized IOQE is then located and, if it is not active, is initialized with the requesting process' PCT address and the mask of buses needed to perform the I/O operation.

The requesting process' PCT Outstanding_I/O_Count is incremented by one and the PCT is flagged waiting for I/O if this is desired.

If an I/O completion event was specified, the Event_Variable is reset to FALSE.

The IOQE is flagged active. The I/O request is checked to determine if it is dispatchable. An I/O request is dispatchable if the buses used in the I/O operation are not marked busy in the BCE_Busy_Idle_Indicators. Dispatchable requests are given to the IOP_Dispatcher while those that are not dispatchable are added to the I/O wait queue. The I/O wait queue is ordered by IOQE_Priority with the highest priority waiting IOQE being first in the queue.

The Pre-Initialized_I/O_SVC_Processor returns to the program which called it. The control flow for this module is presented in Figure 3.2.1.2-1.

d. Outputs - See Table 3.2.1.2-1.

e. Module References -

1. (205) IOP_Dispatcher (FIOPDISP) is CALLed.
2. (363) SVC_Synchronization_Processor (FCMSSYNC) is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation -

1. This alternate entry point is provided for use when GPC's are already synchronized. To use this entry point Register 2 bits 0-15 must contain the requestor's PCT address and Register 3 bits 0-15 must contain the Preinitialized_I/O_SVC_Parameter_List address. Bits 16-31 of both registers should be zero.

2. Note that if the IOQE is in use (active), the program returns immediately to the calling program. There is no error indication returned.

3. Note that no check is made to determine if a process' execution depends on the FALSE state of the Event_Variable.

DATA TABLE 3.2.1.2-1

NAME Pre-Initialized_I/O_SVC_Processor (FIOSVCP)

Table 3.2.1.2-1

NO.	ITEM	ID	ACT	SOURCE	DESTINATION	ASSEMBLER NAME	MML	D	C
1	Active_PCT_Address	Q001.2	I	103	See Appendix E	TCVTOLD			
2	Preinitialized_I/O_SVC_Parameter_List	S040	I		201				
3	PSW_Interrupt_Code	&002.26	I		180,242, 320				
4	SVC_Old_PSW	&001.67	I		See Appendix E	TPSASOP			
5	Preinitialized_IOQE_Number	S040.2	I		201				
6	Preinitialized_IOQE_Address_Table	Y001	I		201	FIOIAT			
7	IOQE_Flags_Field_1	Q007.6	I,O	200,201 225	201 225	TIOQFLG1			
8	Pre-initialized_IOQE	Q007.14	O	200,201	225	TIOQPREI			
9	Address_Of_Requesting_Process_PCT	Q007.2	O	200 201	225	TIOQPCT			
10	Wait_For_I/O	Q007.33	I	200	200 201	TIOQWAIT			
11	I/O_Wait_Indicator	Q003.45	O	200 201	103 105				
12	Outstanding_I/O_Count	Q003.25	O	101,200 201	133	TPCTIORQ			
13	Next_To_Execute_PCT_Address	Q001.3	O	See Appendix E	See App. E	TCVTNEW			
14	IOQE_Completion_Event_Address	Q007.42	I	200	201	TIOQEVNT			
15	Event_Variable	@007	I	182	See App. E				
16	Event_State	@007.2	O	See App. E	See App. E				
17	IOQE_Device_Id	Q007.43	I	200	205 251,351	TIOQDVID			
18	BCE_Device_Mask_Table	Y003	I	351 390	200,201, 220,244	FIOBCD			
19	Mask_Of_Buses_To_Monitor	Q007.4	O	200,201 351	200,205 251	TIOQMNTM			
20	BCE_Busy/Idle_Indicators	Q001.17	I	See Appendix E	See Appendix E	TCVTBCEB			

DATA TABLE 3.2.1.2-1 (Cont'd)

NAME Pre-Initialized_I/O_SVC_Processor Table 3.2.1.2-1 (cont'd)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
21	IOQE_Priority	Q007.40	I	200 205	200,201, 205	TIOQPRI			
22	I/O_Wait_Queue_Address	Q001.6	I,O	See Appendix E	See Appendix E	TCVTIOW			
23	Address_Of_Next_IOQE_In_Chain	Q007.1	O	200,201 225	205 225	TIOQNXT			

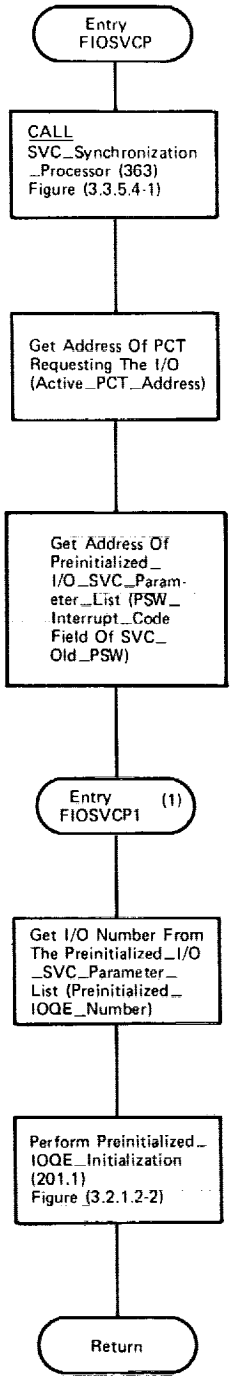


Figure 3.2.1.2-1. Pre-Initialized_I/O_SVC_Processor (FIOSVCP)

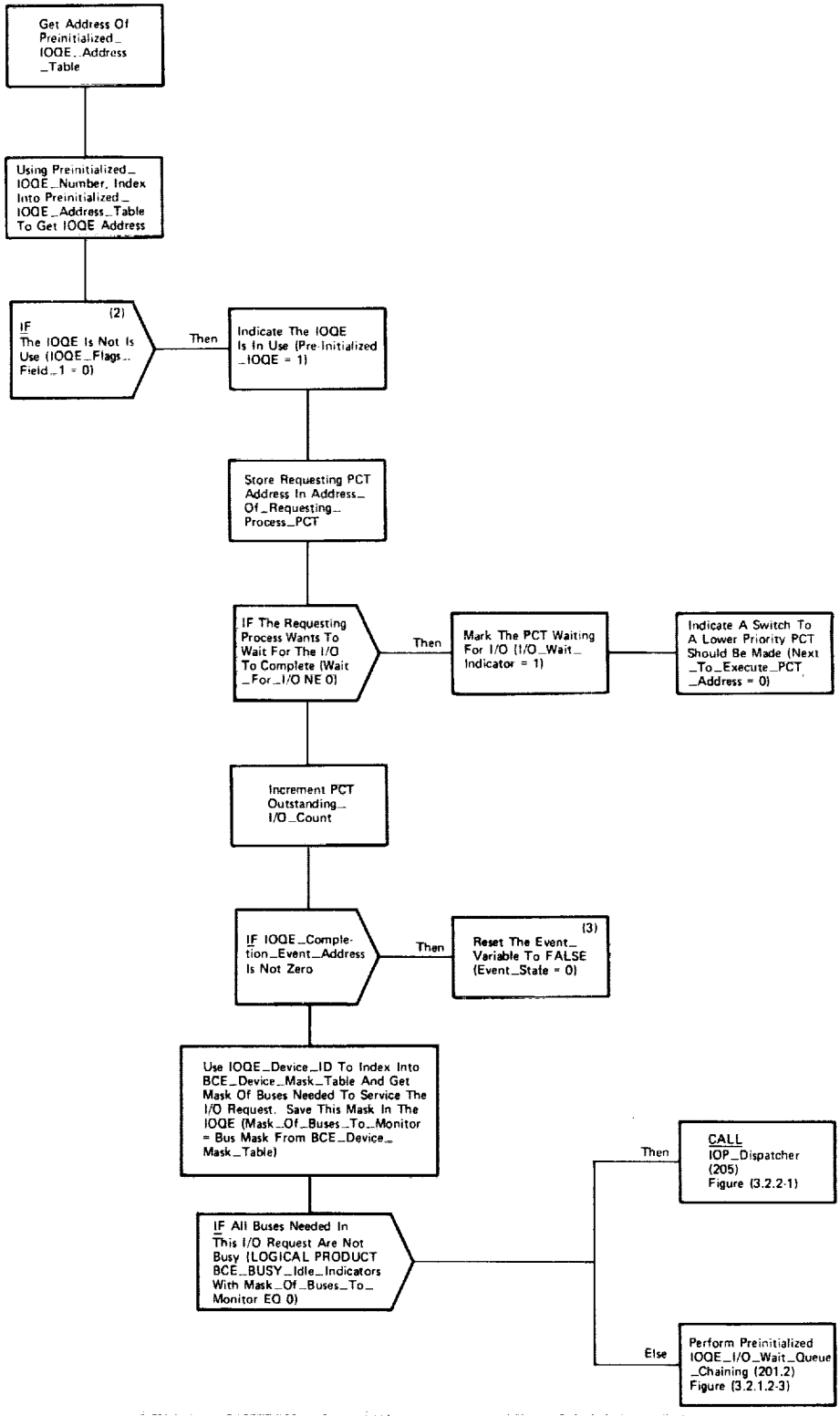


Figure 3.2.1.2-2. Pre-Initialized_I/O_SVC_Processor Preinitialized_IOQE_Initialization (201.1)

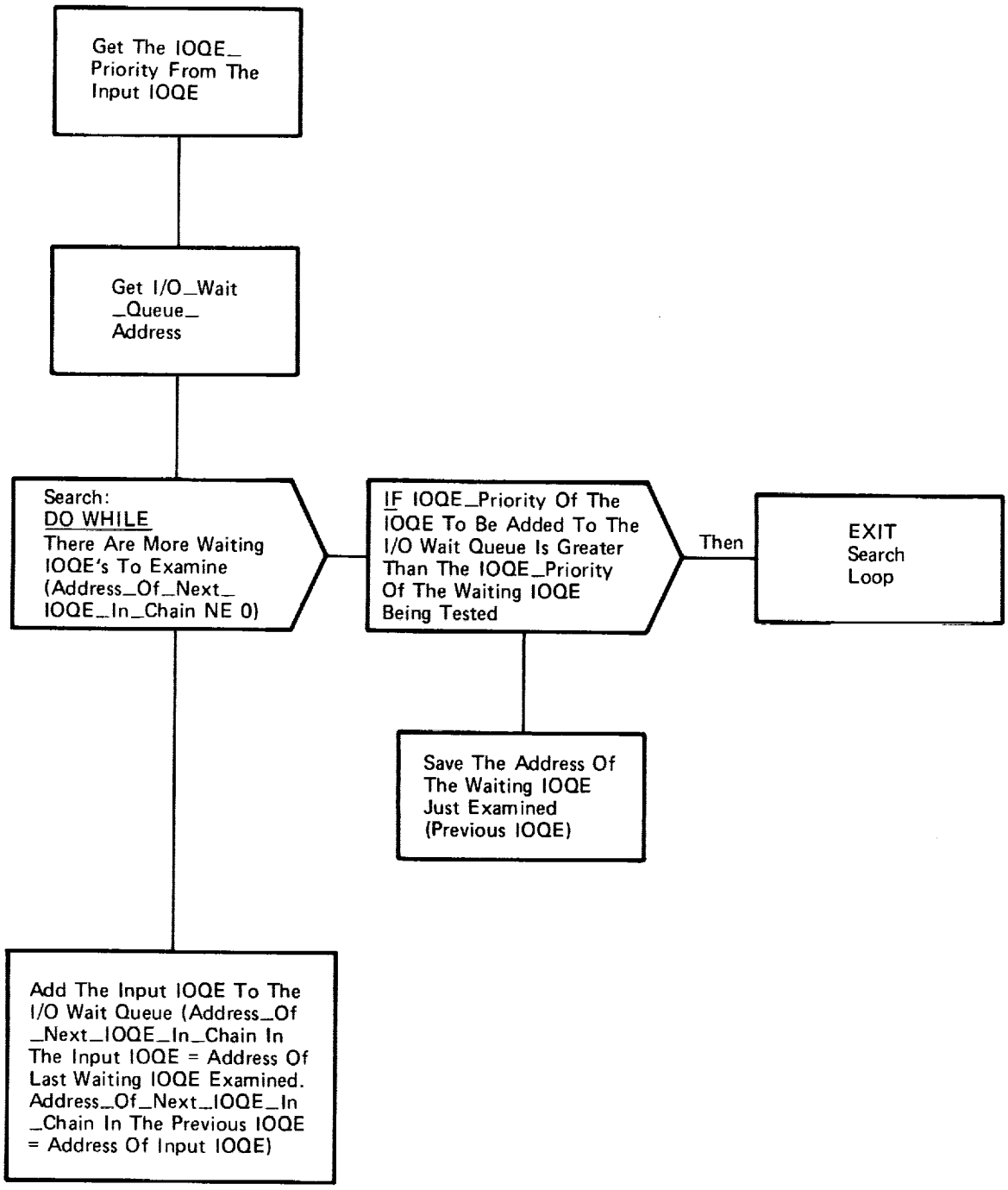


Figure 3.2.1.2-3. Pre-Initialized_I/O_SVC_Processor Preinitialized_IOQE_I/O_Wait_Queue_Chaining (201.2)

3.2.2 IOP_Dispatcher (FIOPDISP) (205)

The IOP_Dispatcher places I/O_Queue_Element's on the I/O active queue, modifies IOP accessible control tables with information from the I/O_Queue_Element, and initiates the execution of IOP programs to accomplish the requested I/O service.

a. Control Interface -

1. CALLed by (200) I/O_SVC_Service_Processor (FIOSVC)
2. CALLed by (201) Pre-Initialized_I/O_SVC_Processor (FIOSVCP)
3. CALLed by (220) I/O_Completion_Processor (FIOCMPLT)

b. Inputs - See Table 3.2.2-1.

c. Process Description - The IOP_Dispatcher chains the incoming IOQE into the I/O active queue at its proper position based upon IOQE_Priority. It then uses the Mask_Of-Buses_To_Be_Monitored in the IOQE to update BCE_Busy/Idle_Indicators. If BTU_ Port_Masks indicate that any of the desired data paths have been masked then the applicable buses are removed from Mask_Of_Buses_To_Be_Monitored.

Mask_Of_Buses_To_Monitor is used by the IOP_Dispatcher to modify various control tables which the MSC and the BCE's use to accomplish the requested I/O service. The tables contain such information as the number of words to be transferred, the location of the data buffer, and the starting address of the programs to be executed by the BCE's. Many I/O requests make use of BCE programs which have the values normally found in these tables hard-coded in the BCE program itself. The IOP_Dispatcher will bypass any table initialization not needed by these self-initializing BCE programs. The IOP_Dispatcher does any device-dependent processing required for each device type and determines which program counters are required for the BCE's which will be started by the request.

After the required control tables are initialized, Requested_I/O_Transmitter_Mask is used to remove listen mode buses from Mask_Of_Buses_To_Be_Monitored leaving a mask that is used by the IOP_Dispatcher to load the BCE program counter value for each bus into BCE_Program_Counter in MSC_Local_Store. Finally the I/O request is initiated by calling the Start_MSC_Processor.

The control flow for this module is shown in Figure 3.2.2-1.

d. Output - See Table 3.2.2-1.

e. Module References -

1. (210) Start_MSC_Processor (FIOSTMSC) is CALLed.

2. (162) Current_GMT_Routine (FPMGMTIM) is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation -

1. The Program-Controlled I/O (PC) instruction is used to write the BCE program counter value to BCE_Program_Counter in MSC_Local_Store.

2. When the IUA is greater than 31, it indicates that the IUA value is the address of a table of IUA's.

3. For PMU write computer data ram requests, the application passes all the BCE control information in a special table (Write_Computer_Data_Ram_Request_Table) created by the application. The information in this user supplied table is moved into special PMU BCE control tables that are used only by the PMU BCE programs. These special PMU BCE control tables are extentions of the standard BCE control tables.

4. PMU SM OI/PL data ram read requests have a special I/O interface. As many as 15 individual ram read requests can be initiated by one application I/O request. The control information for each ram read and the number of reads is contained in a user created Data_Acquisition_Request_Table. The address of this table is passed to FCOS in the I/O parameter list. The information in this table is used to fill in the appropriate entries in the special PMU BCE control tables described in (3).

DATA TABLE 3.2.2-1

NAME IOP_Dispatcher (FIOPDISP)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	COMFAULT_Status_Needed	Q007.30	I	200	205	TIOQCOMF			
2	BCE_Program_Counter_Pointer_Table	Y004	I		205	FIOPCP			
3	IOQE_Device_Id	Q007.43	I	200	205,251, 351	TIOQDVID			
4	Requested_I/O_Transmitter_Mask	#003	I	351,355	205,351, 250,350	TFCMXMSK			
5	Mask_Of_Buses_To_ Monitor	Q007.4	I	200,201, 351	200, 205,251	TIOQMNTM			
6	Iterations_Count_For_Request	Q007.41	O	205,210	205	TIOQICNT			
7	Initial_Iteration_Count_Table	Y016	O	205		FIOIIC			
8	MSC_Iteration_Count_Bias	Y065	I		205	FIODELTA			
9	FC_Buses_Comfault_Word	J150	O	205		FIOBCESI			
10	MTU_Comfault1_Word2	#031	O	205	149	FIOBCSI2	V91M2303P	X	
11	Bus_Masks	I010.09	I	355	See App. E	TGSTBMSK	See Appendix E	X	X
12	BCE_Busy/Idle_Indicators	Q001.17	I	See Appendix E		TCVTBCEB			
13	IUA_Table	Y002	I		205	FIOIUATB			
14	BTU_Port_Masks	#024	I	290	205,290, 355,368	TFCMBPM	See Appendix E	X	X
15	Error_In_Transaction	Q007.15	I	200	205	TIOQTRER			
16	I/O_Active_Queue_Address	Q001.5	I	205,220, 305	210,225, 242,361	TCVTIOA			
17	Address_Of_Next_IOQE_In_Chain	Q007.1	O	200,201, 225	205, 225	TIOQNXT			
18	IOQE_Priority	Q007.40	I	200,205	200,201 205	TIOQPRI			
19	RS_IPR_Program_Counter	Y033	I		205	FIOIPRRS			
20	CS_IPR_Program_Counter	Y034	I		205	FIOIPRCS			

DATA TABLE 3.2.2-1 (Cont'd)

NAME IOP_Dispatcher (FIOPDISP)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
21	8_Word_SSIP_ICC_Program_Counter	Y041	I		205	FIOSIP08			
22	16_Word_SSIP_ICC_Program_Counter	Y042	I		205	FIOSIP16			
23	32_Word_SSIP_ICC_Program_Counter	Y043	I		205	FIOSIP32			
24	64_Word_SSIP_ICC_Program_Counter	Y044	I		205	FIOSIP64			
25	128_Word_SSIP_ICC_Program_Counter	Y045	I		205	FIOSP128			
26	IOQE_Word_Count	Q007.45	I	200	205	TIOQWDCD			
27	ICC_Program_Counter_Table	Y032	O	205		FIOICCPC			
28	IOQE_Operation_Code	Q007.44	I	200	205	TIOQOPCD			
29	DEU_Fill_Command_Word	Y086	I		205	FIOPDFLC			
30	BCE_Command_Word_Table	Y007	O	205,280	264,282, 283	FIOCWE			
31	DEU_Fill_Program_Counter_Table	Y035	I		205	FIOPDFLP			
32	DEU_Poll_Program_Counter_Table	Y036	I		205	FIOPDPOL			
33	DEU_Keyboard_Request_Program_Counter_Table	Y039	I		205	FIOPDKEY			
34	DEU_Dump_Program_Counter_Table	Y037	I		205	FIOPDDPP			
35	DEU_Bite_Status_Program_Counter_Table	Y040	I		205	FIOPDRBS			
36	DEU_Reset_Scratch_Pad_Line_Program_Counter_Table	Y038	I		205	FIOPDRSP			
37	DEU_Initial_Iteration_Count_Table	Y018	I	205		FIODEUIC			
38	LDB_Interrogate_W/O_Data_Program_Counter_Table	Y046	I		205	FIOLIBCE			
39	IOQE_Starting_Address	Q007.46	I	200	205	TIOQSTAD			
40	LDB_Interrogate_W/Data_Program_Counter_Table	Y047	I		205	FIOLDBPC			

DATA TABLE 3.2.2-1 (Cont'd)

NAME IOP_Dispatcher (FIOPDISP)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
41	LDB_Status_Request_Program_Counter_Table	Y049	I		205	FIOLRBCE			
42	LDB_Status_Program_Counter_Table	Y050	I		205	FIOLCBCE			
43	LDB_Initial_Iteration_Count_Table	Y019	I	205		FIOLDBIC			
44	LDB_Transmission_Enable_Program_Counter_Table	Y048	I		205	FIOLDBTC			
45	PMU_Read_Bite_Program_Counter_Table	Y058	I		205	FIOPCBIT			
46	PMU_Initial_Iteration_Count_Table	Y017	I	205		FIOPMUIC			
47	PMU_Command_Word_Table	Y010	O	205		FIOPCWE			
48	Computer_Data_Ram_Command_Word	Y064.01	I		205				
49	Computer_Data_Ram_Buffer_Address	Y064.02	I		205				
50	PMU_Base_Register_Table	Y008	O	205		FIOPBRE			
51	PMU_Write_Computer_Data_Ram_Program_Counter_Table	Y054	I		205	FIOPCWCD			
52	PMU_Word_Count_Table	Y009	O	205		FIOPWCE			
53	Computer_Data_Ram_Word_Count	Y064.3	I		205				
54	BCE_Base_Register_Table	Y005	I,O	205,280	See App. E	FIOBRE			
55	PMU_Write_128/64_KBPS_Program_Counter_Table	Y055	I		205	FIOPCWDF			
56	PMU_Read_128/64_KBPS_Program_Counter_Table	Y056	I		205	FIOPCRDF			
57	PMU_Format_Select_Program_Counter_Table	Y057	I		205	FIOPCWFR			
58	PMU_TCS_OI/PL_Data_Ram_Read_Program_Counter_Table	Y060	I		205	FIOPCOI1			
59	Address_Of_I/O_Buffer	Q007.5	I	200	149,205	TIOQBUFA			
60	DART_Buffer_Displacement	Y063.04	I		205				

DATA TABLE 3.2.2-1 (Cont'd)

NAME IOP_Dispatcher (FIOPDISP)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
61	DART_Word_Count	Y063.03	I		205				
62	DART_Ram_Address	Y063.02	I		205				
63	PMU_SM_OI/PL_Data_Ram_Read_Program_Counter_Table	Y059	I		205	FIOPCOID			
64	MTU_Write_Program_Counter_Table	Y061	I		205	FIOMTUWT			
65	MET_Reference_Time	Q001.89	O	142,149, 205,305	144	TCVTMRTH/TCVTMRTM			
66	Prime_GPC_Time_Microseconds	I250	O	142,149, 205		CZ2VMETM			
67	Prime_GPC_Time_Half_Hours	I260	O	142,149, 205		CZ2VMETH			
68	MTU_Read_BCE_Jitter_Bias	Y069	I		205	FPMKBIAS			
69	ICC_Prime_Time	I610.06	O	205	142,149	CZ2B_GPC_MT / CZ2B_GPC_MT2			
70	MTU_Read_Program_Counter_Table	Y062	I		205	FIOMTURD			
71	TCS_Standard_Control_BCE_Program_Counter_Table	Y051	I		205	FIOACBCE			
72	TCS_Standard_Read_BCE_Program_Counter_Table	Y052	I		205	FIOARBCE			
73	TCS_Standard_Write_BCE_Program_Counter_Table	Y053	I		205	FIOAWBCE			
74	BCE_Word_Count_Table	Y006	O	205	261,263, 264,282	FIOWCE			
75	SYNC_Timeout_Save_Area	Y066	O	205		FIOSYNCT			
76	SSIP_Sync_Timeout_Counter	Y067	I		205	FIOSIPTO			
77	Sync_Timeout_Counter	I580.02	O	205	363,364	CZ2VSYNC			

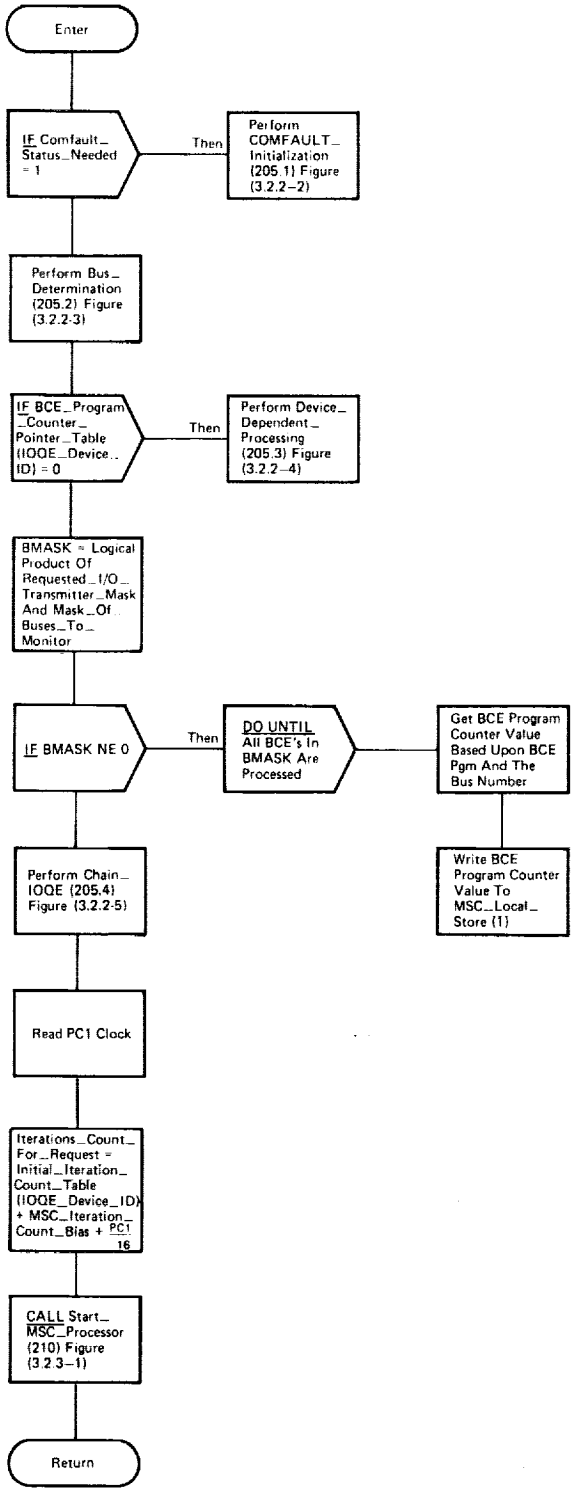


Figure 3.2.2-1. IOP_Dispatcher

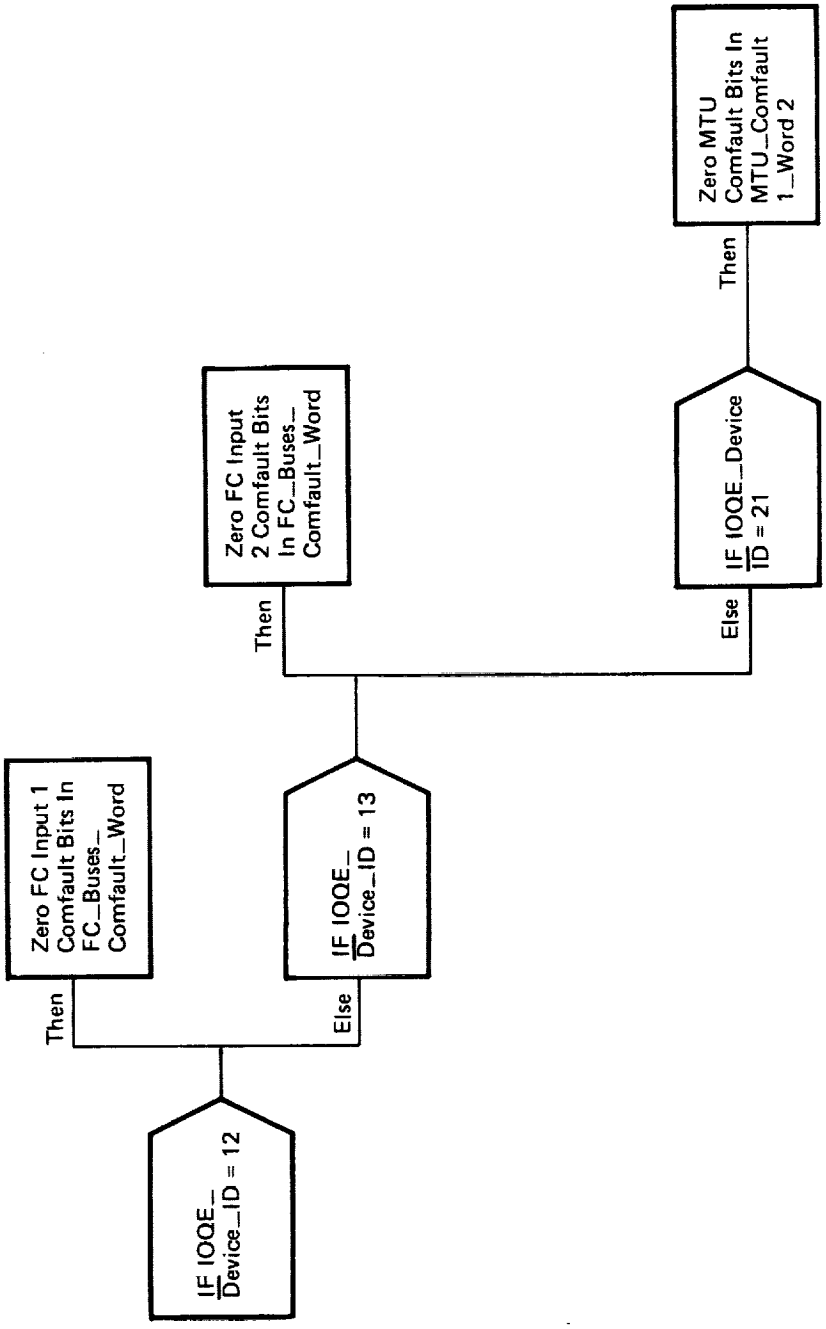


Figure 3.2.2-2. COMFAULT_Initialization (205.1)

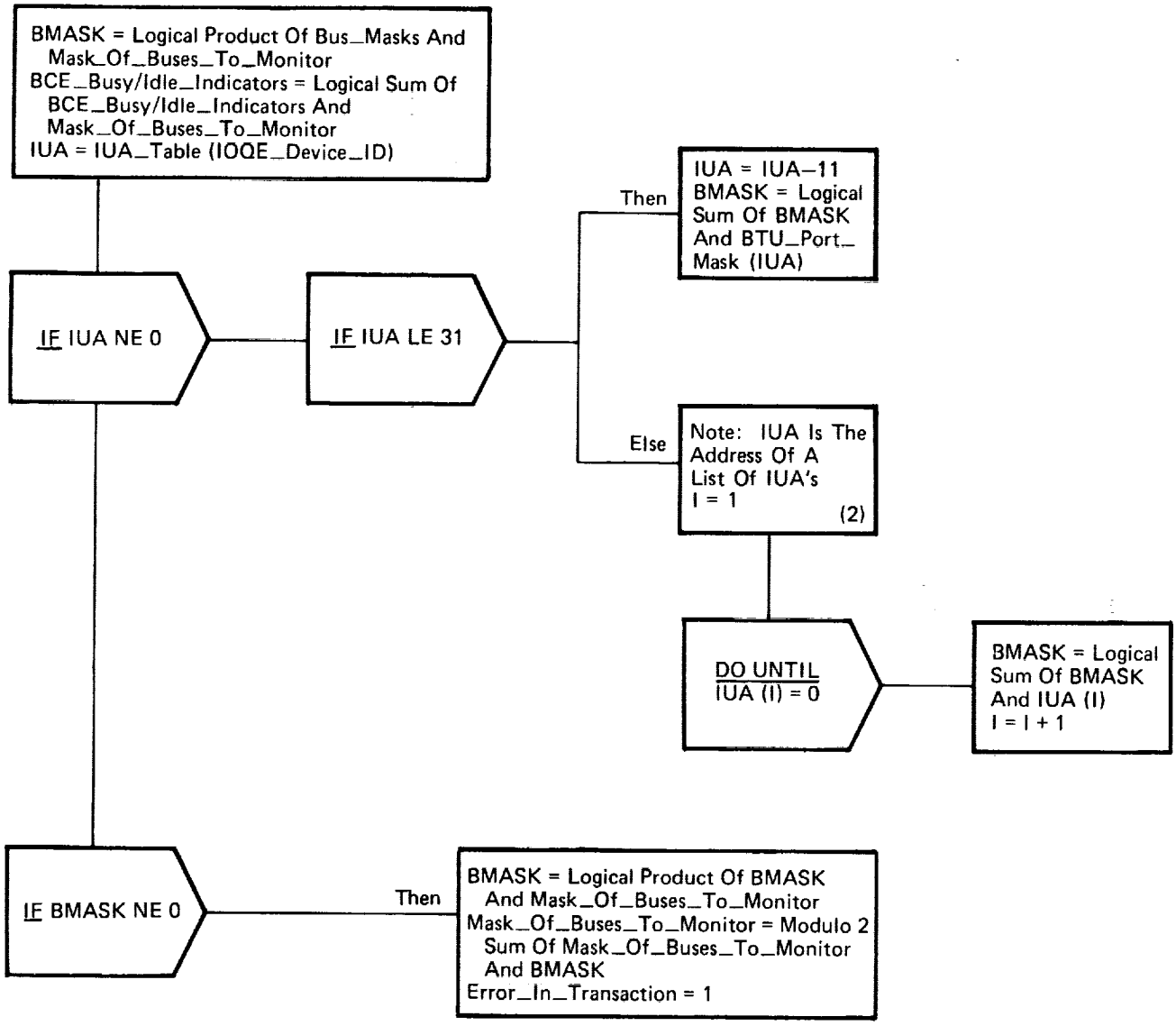


Figure 3.2.2-3. IOP_Dispatcher Bus_Determination (205.2)

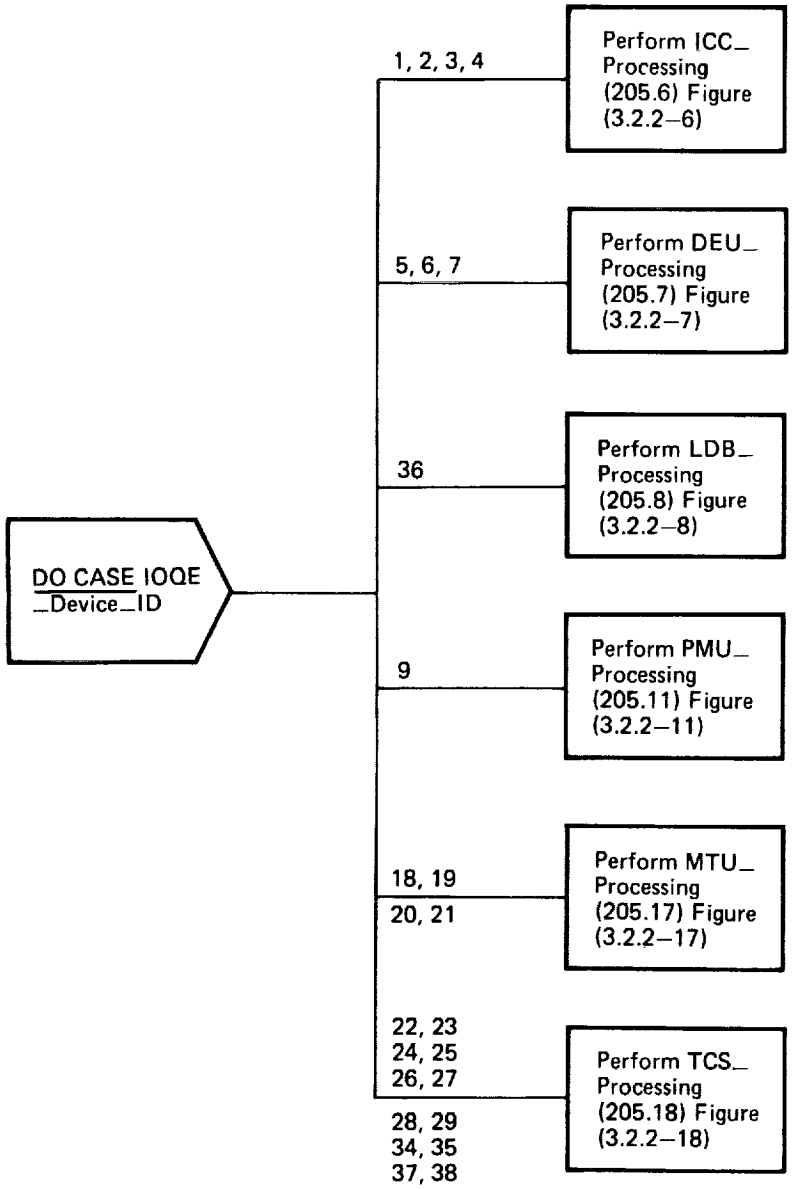


Figure 3.2.2-4. IOP_Dispatcher Device_Dependent_Processing (205.3)

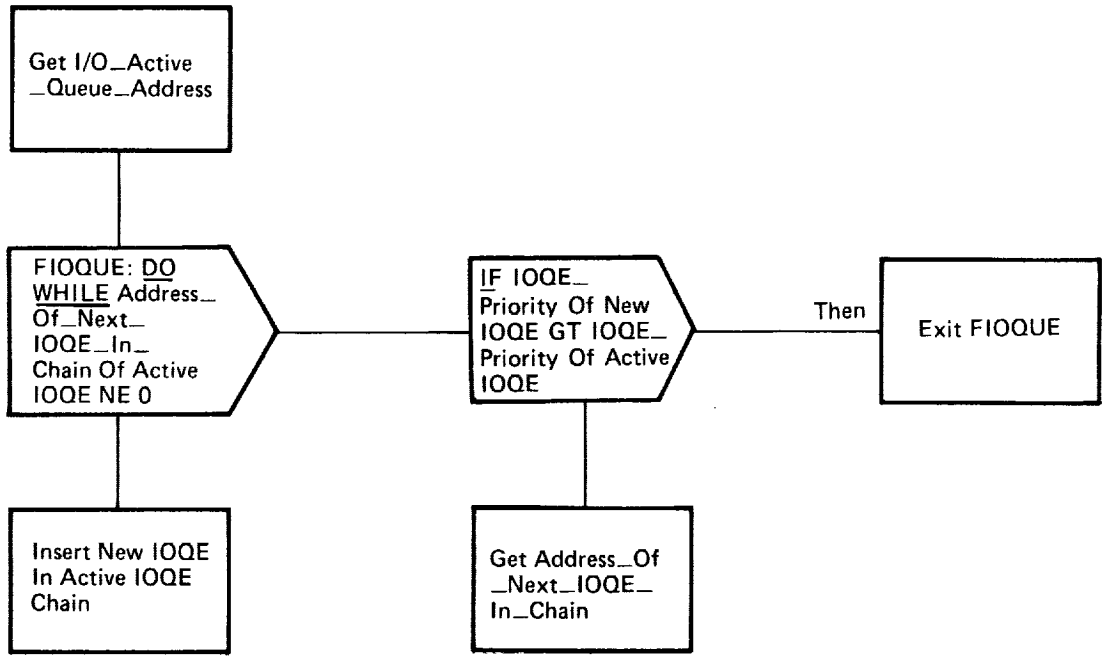


Figure 3.2.2-5. IOP_Dispatcher Chain_IOQE (205.4)

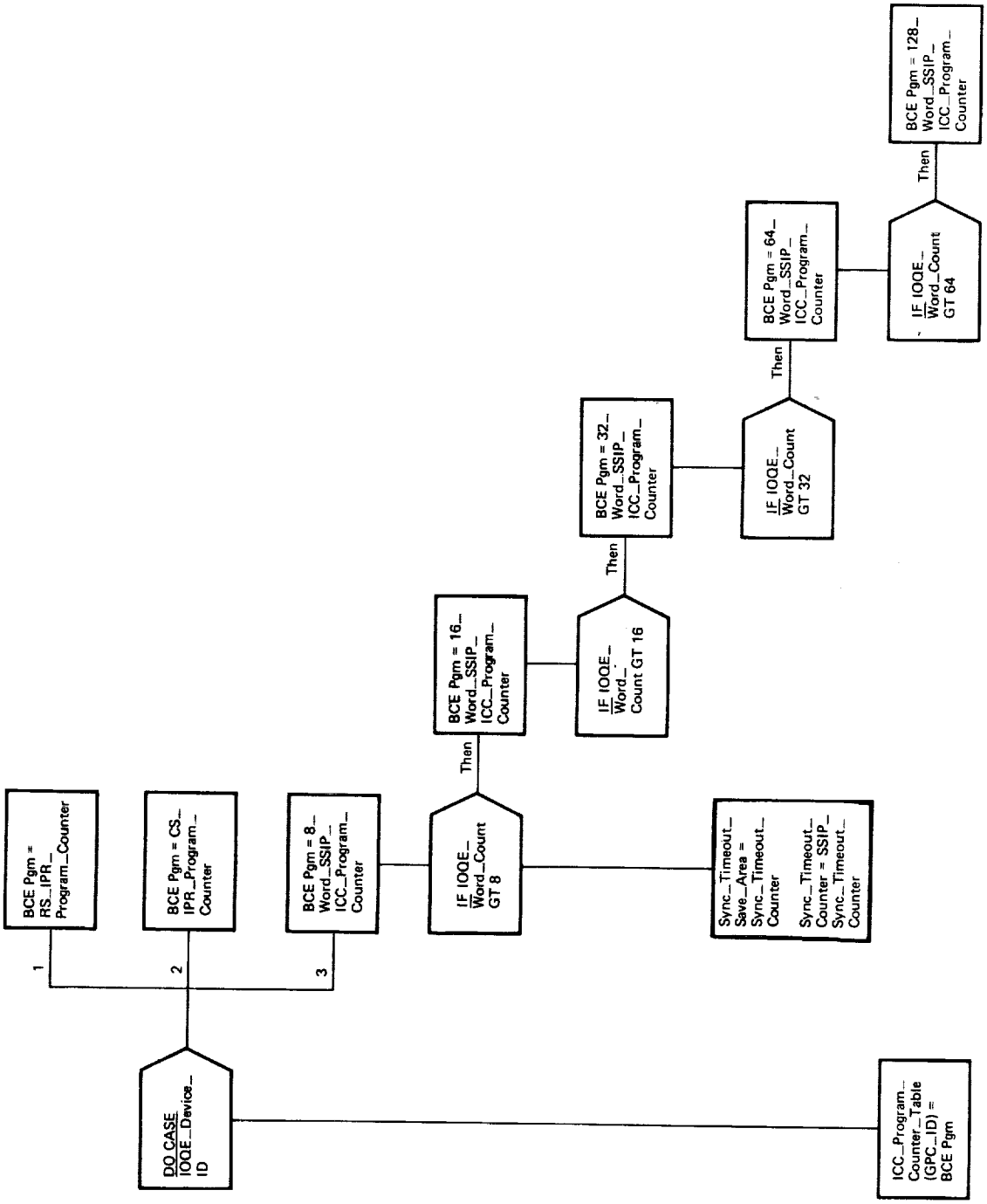


Figure 3.2.2-6. IOP_Dispatcher ICC Processing (205.6)

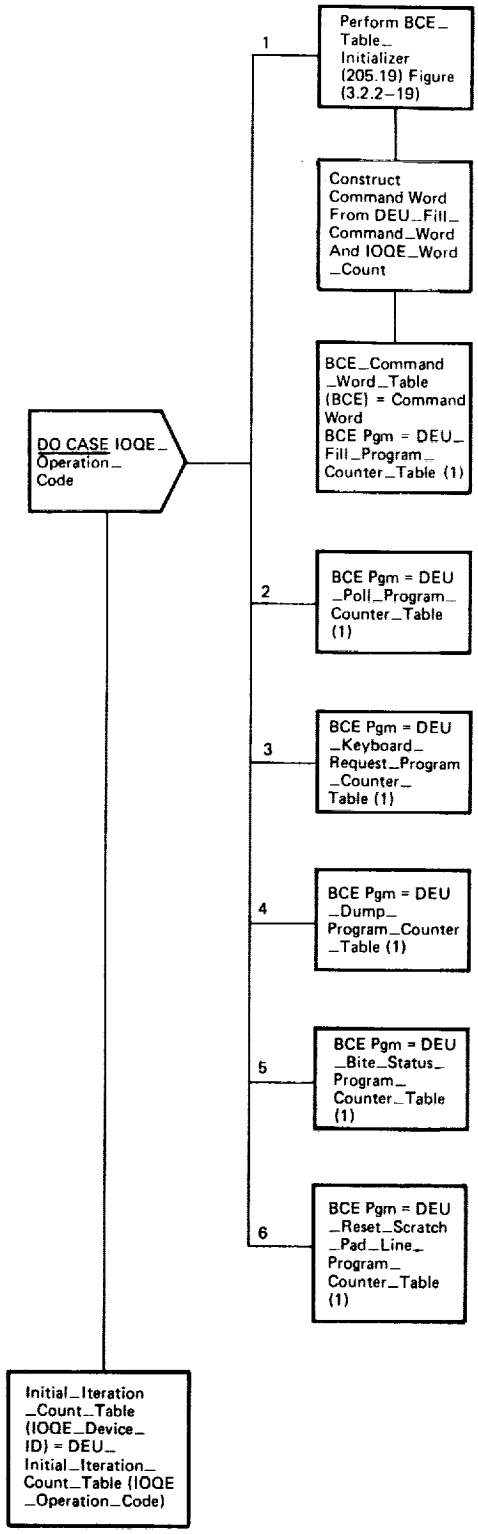


Figure 3.2.2-7. IOP_Dispatcher DEU_Processing (205.7)

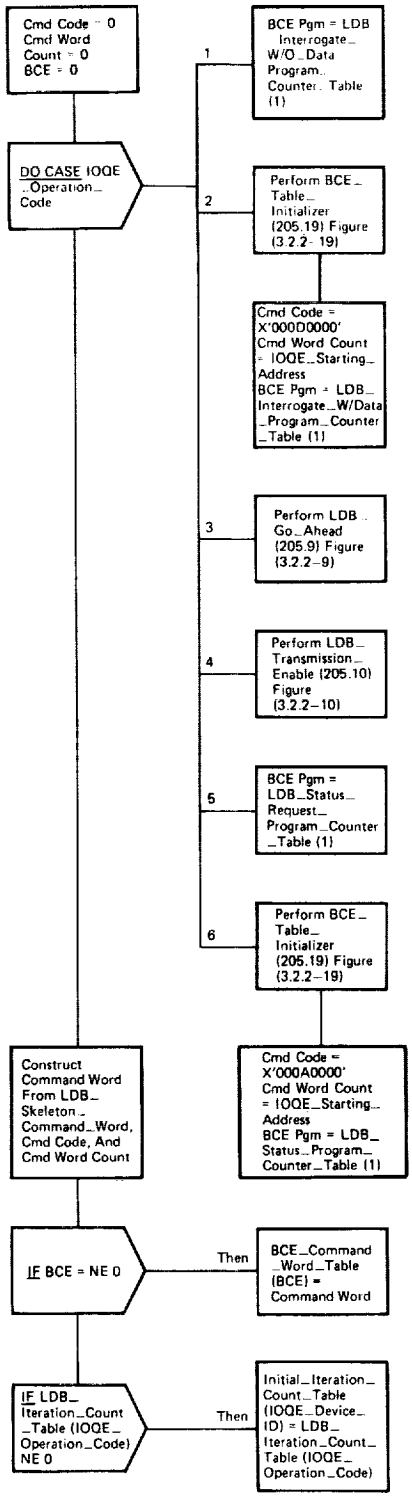


Figure 3.2.2-8. IOP_Dispatcher LDB_Processing (205.8)

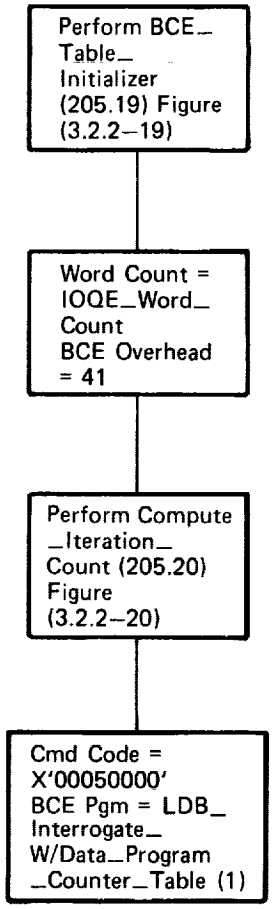


Figure 3.2.2-9. IOP_Dispatcher LDB_Go_Ahead (205.9)

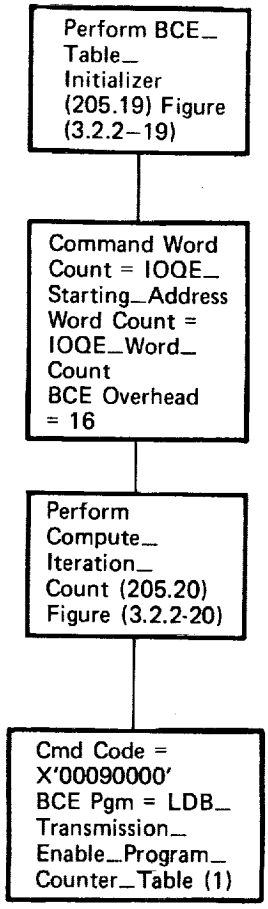


Figure 3.2.2-10. IOP_Dispatcher LDB_Transmission_Enable (205.10)

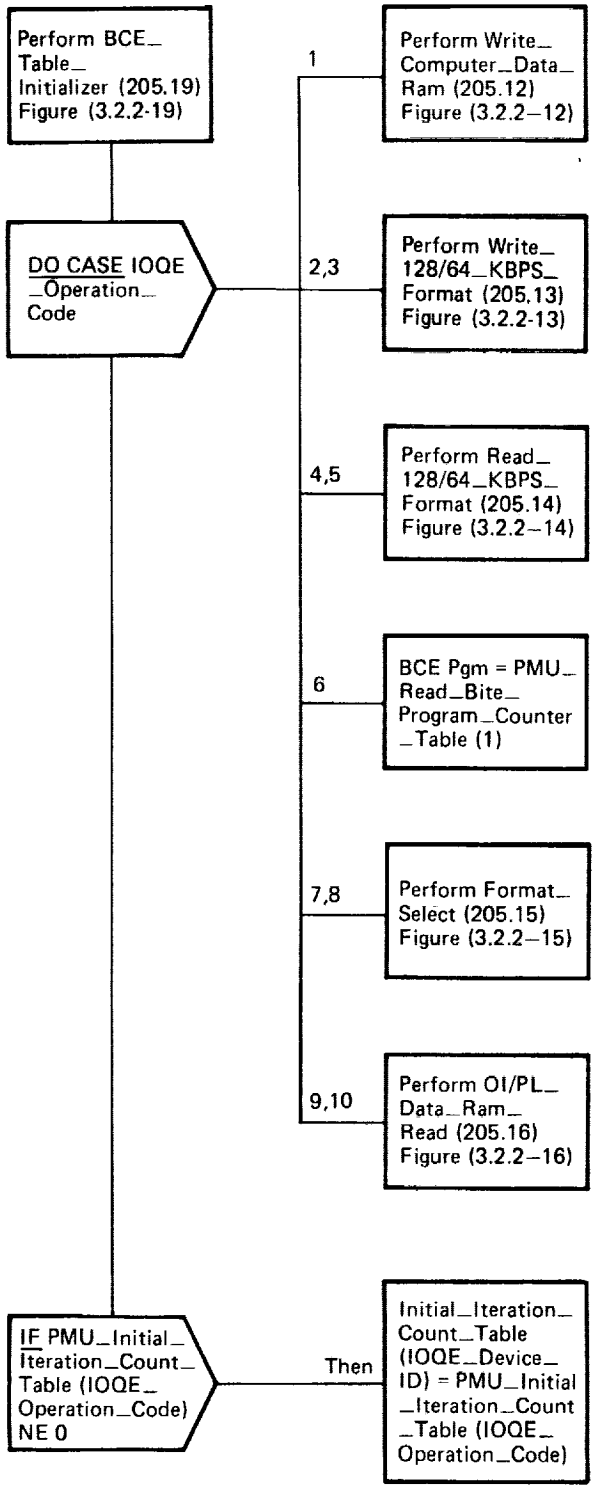


Figure 3.2.2-11. IOP_Dispatcher PMU_Processing (205.11)

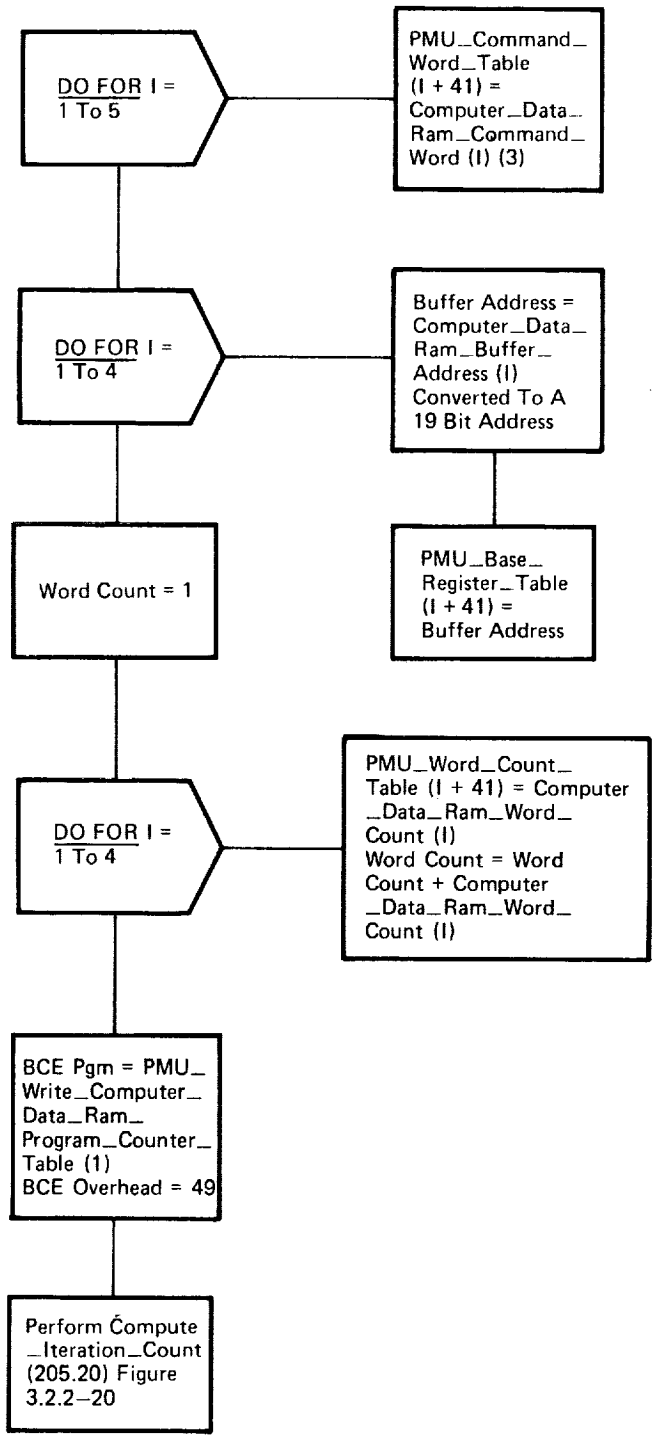


Figure 3.2.2-12. IOP_Dispatcher Write_Computer_Data_Ram (205.12)

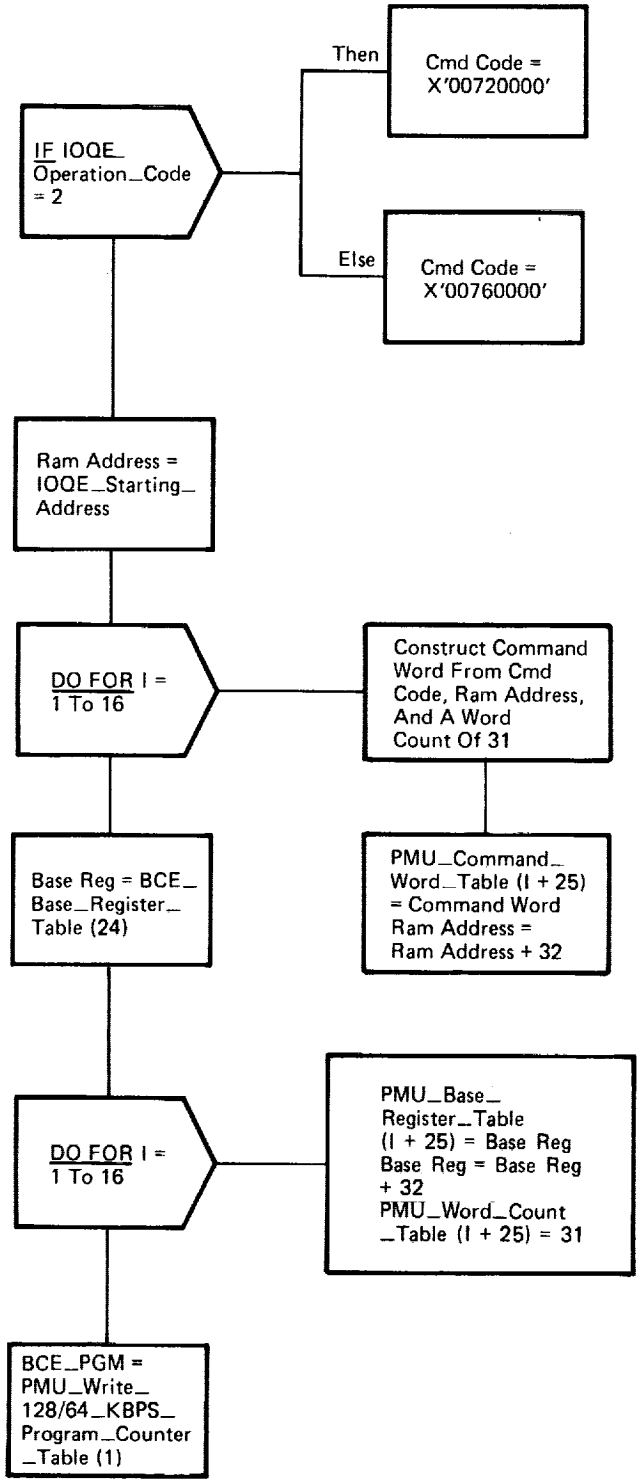


Figure 3.2.2-13. IOP_Dispatcher Write_128/64_KBPS_Format (205.13)

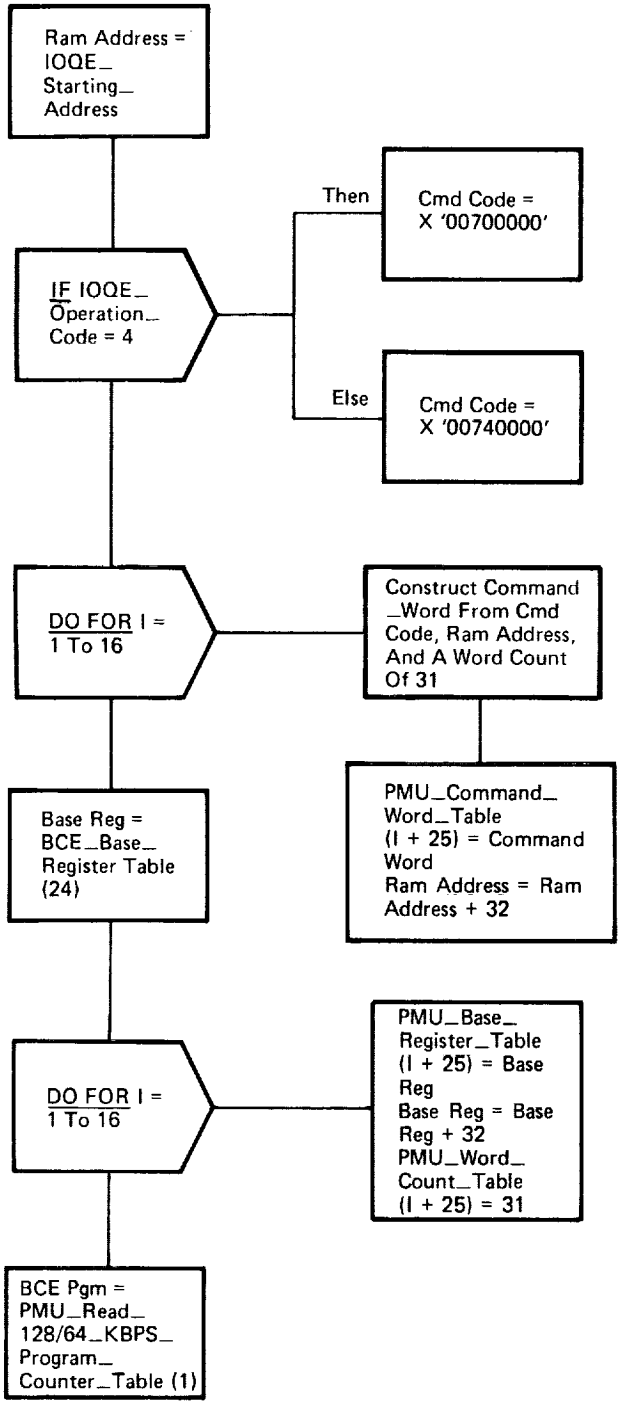


Figure 3.2.2-14. IOP_Dispatcher Read_128/64_KBPS_Format (205.14)

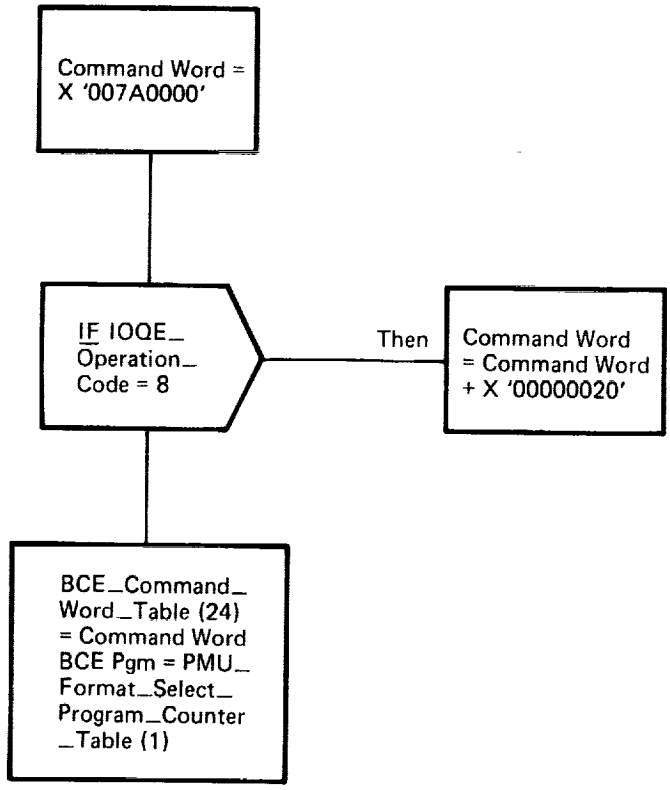


Figure 3.2.2-15. IOP_Dispatcher Format_Select (205.15)

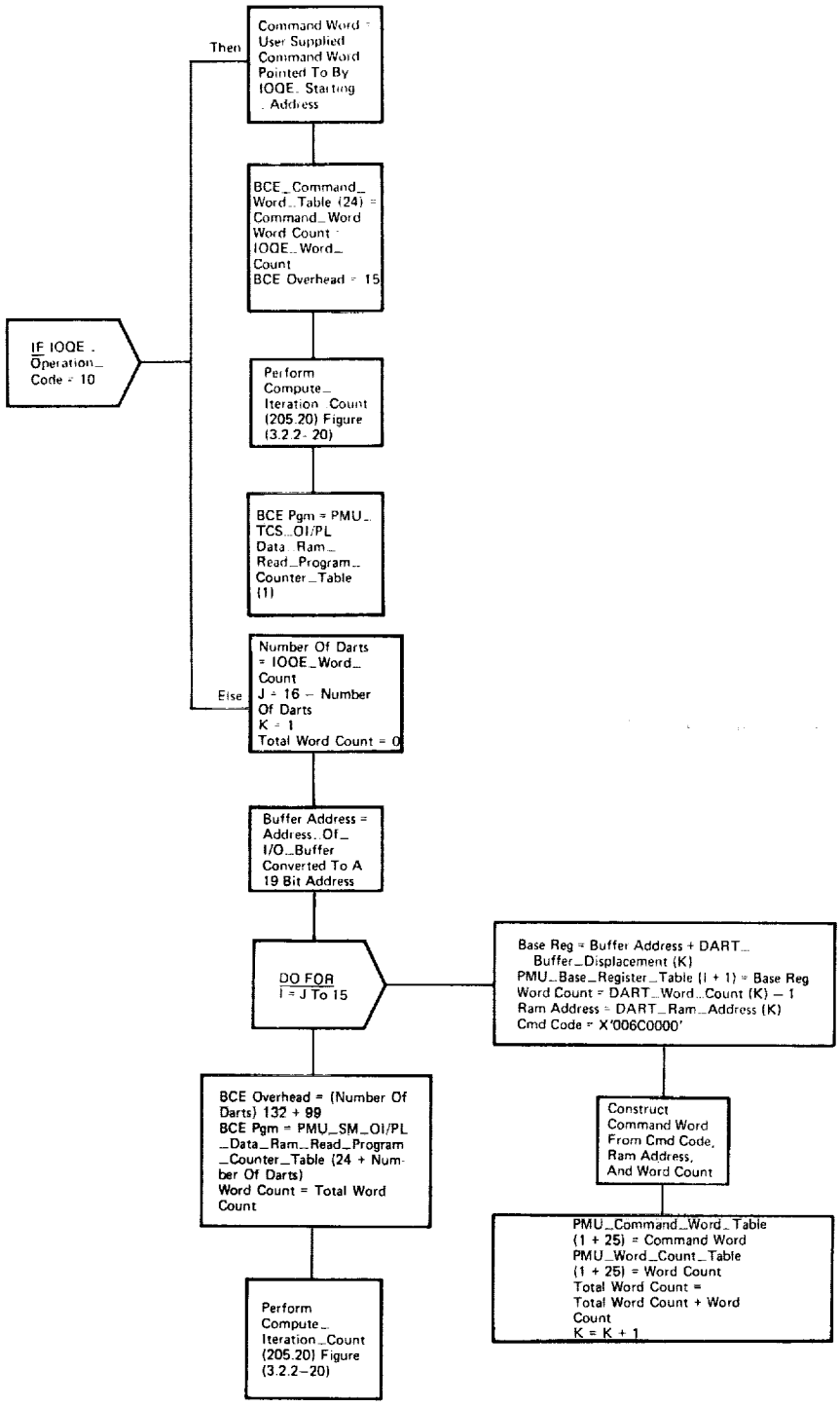


Figure 3.2.2-16. IOP_Dispatcher
OI/PL_Data_Ram_Read (205.16)

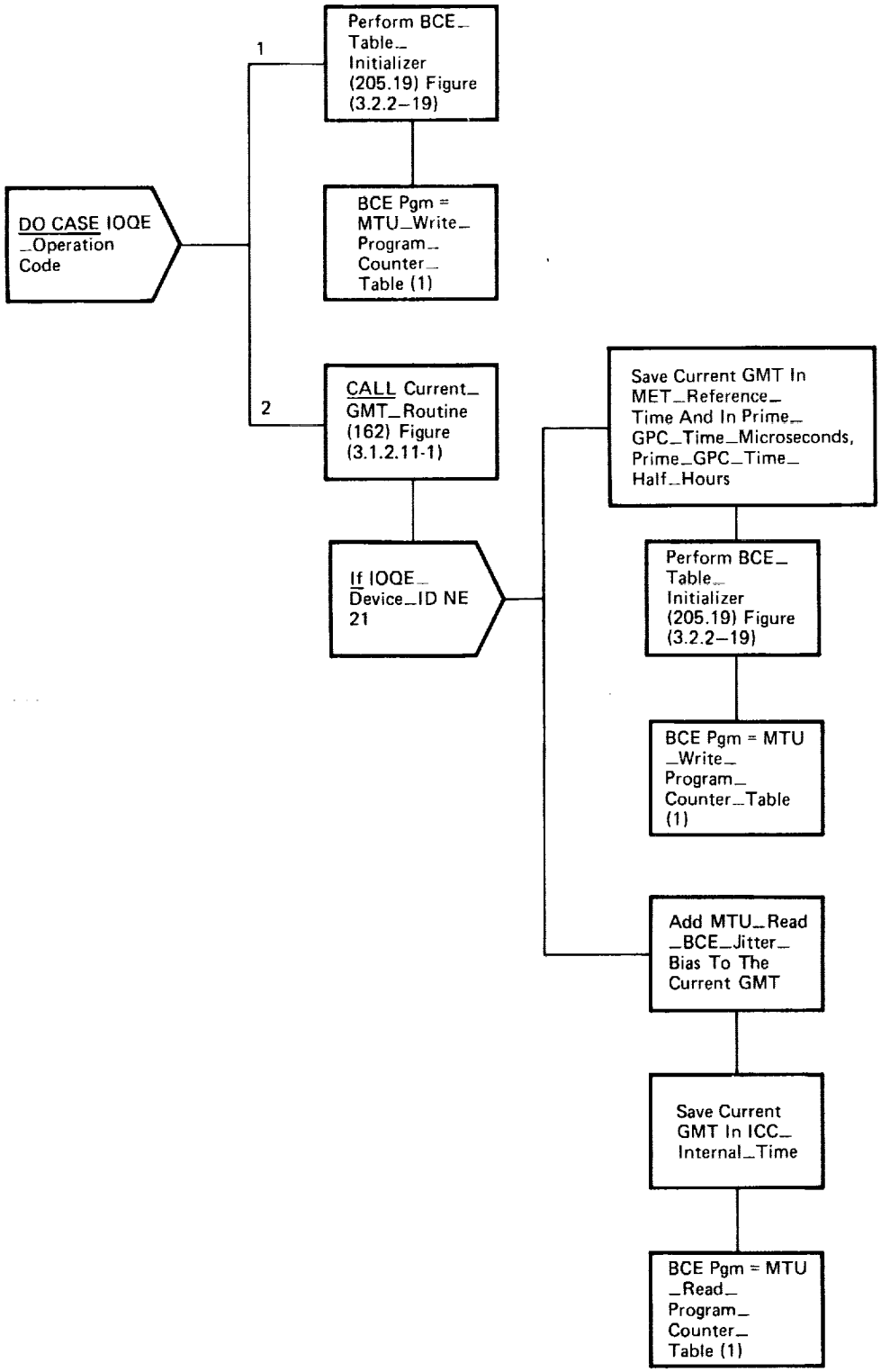


Figure 3.2.2-17. IOP_Dispatcher MTU_Processing (205.17)

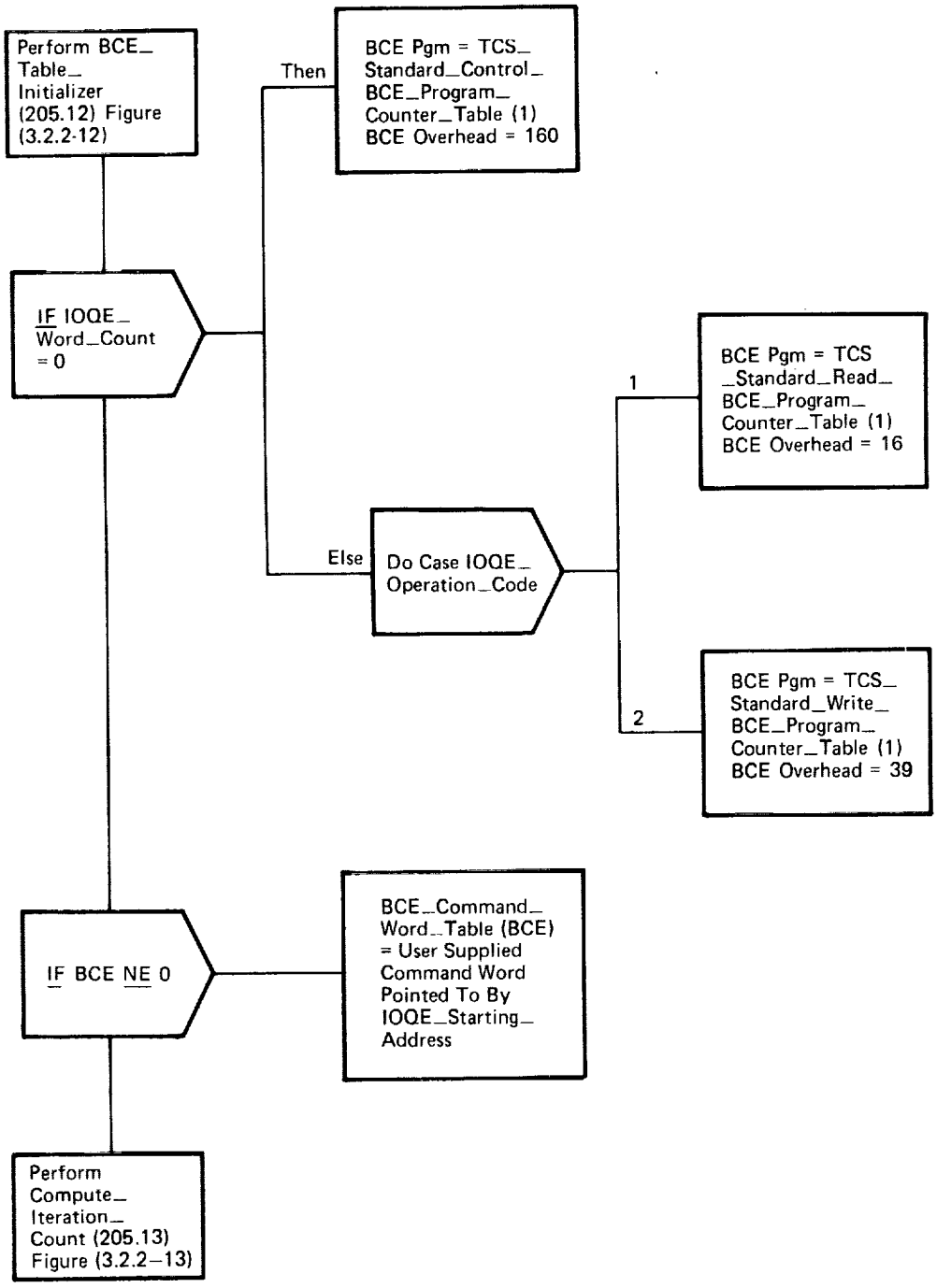


Figure 3.2.2-18. IOP_Dispatcher ICS_Processing (205.18)

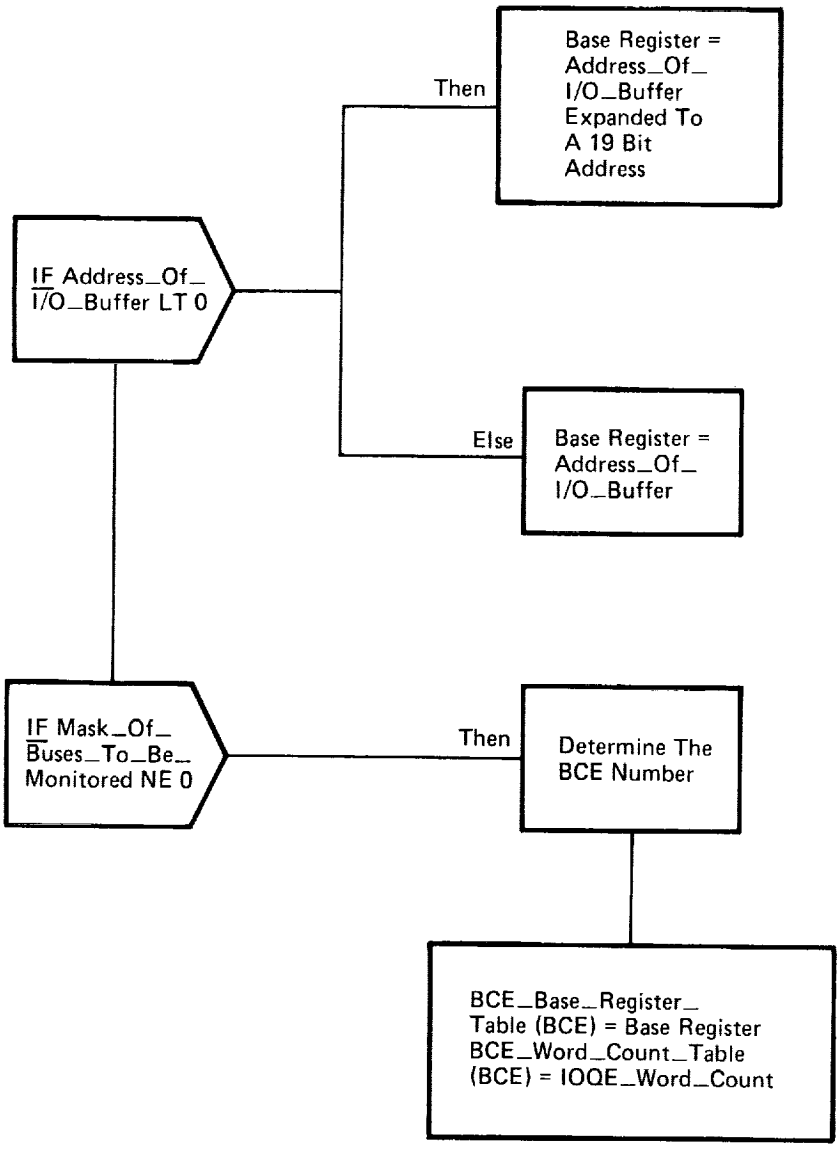


Figure 3.2.2-19. IOP_Dispatcher BCE_Table_Initializer (205.19)

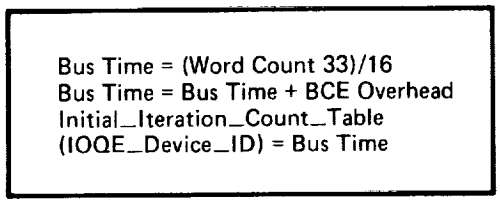


Figure 3.2.2-20. IOP_Dispatcher Compute_Iteration_Count (205.20)

3.2.3 Start_MSC_Processor (FIOSTMSC) (210)

The Start_MSC_Processor is called to start the MSC at a program address specified by the calling program. If required, it will halt then re-enable the MSC prior to loading the MSC's Program Counter (PC) with the specified program address. Also, if required, it will compute for the top active IOQE the time remaining before a MSC timeout is declared.

a. Control Interface -

1. Called by (205) IOP_Dispatcher (FIOPDISP)
2. Called by (220) I/O_Completion_Processor (FIOCMPLT)
3. Called by (244) Level_C_I/O_Error_Interrupt_HANDLER (FIOERRLC)
4. Called by (280) Mass_Memory_Manager (FIOMMMGR)
5. Called by (368) Sync_Fail_Processor (FCMSFAIL)
6. Called by (375) Fault_Detection_Identification (FCMFDI)
7. Called by (340) Miscellaneous_CM_Request_Processor (FCMSVC).

b. Input - See Table 3.2.3-1. General purpose register 0 contains a specified MSC program address.

c. Process Description - The Start_MSC_Processor is called whenever the MSC is required to work. The MSC is normally called to start or monitor an I/O operation.

The Start_MSC_Processor when entered will determine if the MSC is busy by testing the MSC_Status_Indicator for a nonzero value. If the MSC is busy, the Start_MSC_Processor will set the CPU_To_MSC_Status with a positive value indicating a CPU MSC request. Next, the Start_MSC_Processor will determine if the MSC is interruptable by testing the MSC_Status_Indicator for a non-positive value. If the MSC is non interruptable, the Start_MSC_Processor will enter a DO UNTIL LOOP that continually tests the MSC_Status_Indicator until it is non-positive. If or when the MSC is interruptable, the Start_MSC_Processor will halt the MSC, reset the CPU_To_MSC_Status to a non-positive value indicating that there are no requests for the MSC, and enable the MSC for work.

If or when the MSC is idle, the Start_MSC_processor will determine if the I/O for the top active IOQE is to be monitored for completion. If the I/O is not to be monitored, the MSC_I/O_Monitor _Flag will be set to zeroed. If the I/O time remaining is to be computed, the MSC_I/O_Monitor_Flag will be set with the I/O_Active_Queue_Address, the MSC iteration count in the Iteration_Count_For_Request of the top active IOQE will be selected and used to compute the time remaining, and the time remaining will be stored in the Time_Remaining_For_Top_IOQE.

Finally, the Start_MSC_Processor will load the MSC_Program_Counter with a specified program address, reset the MSC_Status_Indicator with a non-positive value indicating that the MSC is not interruptable, set the MSC busy, and return to the caller.

d. Output - See Table 3.2.3-1.

e. Module References -

1. (250) MSC_Control_Routine (FIOMCNTL) is started by loading the MSC_Program_Counter.

2. (251) MSC_I/O_Monitor (FIOMNTR) is started by loading the MSC_Program_Counter.

3. (252) MSC_BCE_Reset_Routine (FIOMSETB) is started by loading the MSC_Program_Counter.

4. (281) Mass_Memory_MSC_Processor (FIOMMMSC) is started by loading the MSC_Program_Counter.

f. Module Attributes - Program

g. Template References - None

h. Error Checks - None

i. Constraints and Assumptions - None

j. Detail Implementation - None

DATA TABLE 3.2.3-1

NAME Start_MSC_Processor (FIOSTMSC)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	MSC_Status_Indicator	Q001.97	I/O	210, 251,310	210	TCVTMSC			
2	I/O_Active_Queue_Address	Q001.5	I	205,220, 305	210,225 242,361	TCVTIOA			
3	I/O_Complete_IOQE_Address	Q001.78	I	220,250, 305	210,220	TCVTIOQ			
4	Iterations_Count_For_Request	Q007.41	I	205,210	205	TIOQICNT			
5	CPU_To_MSC_Status	Q001.98	O	210,310	251	TCVTCPU			
6	MSC_I/O_Monitor_Flag	Q001.93	O	210	251	TCVTMNTR			
7	Time_Remaining_For_Top_IOQE	Q001.92	O	210	251	TCVTMTTG			
8	MSC_Program_Counter	&007.02	O	210,300					

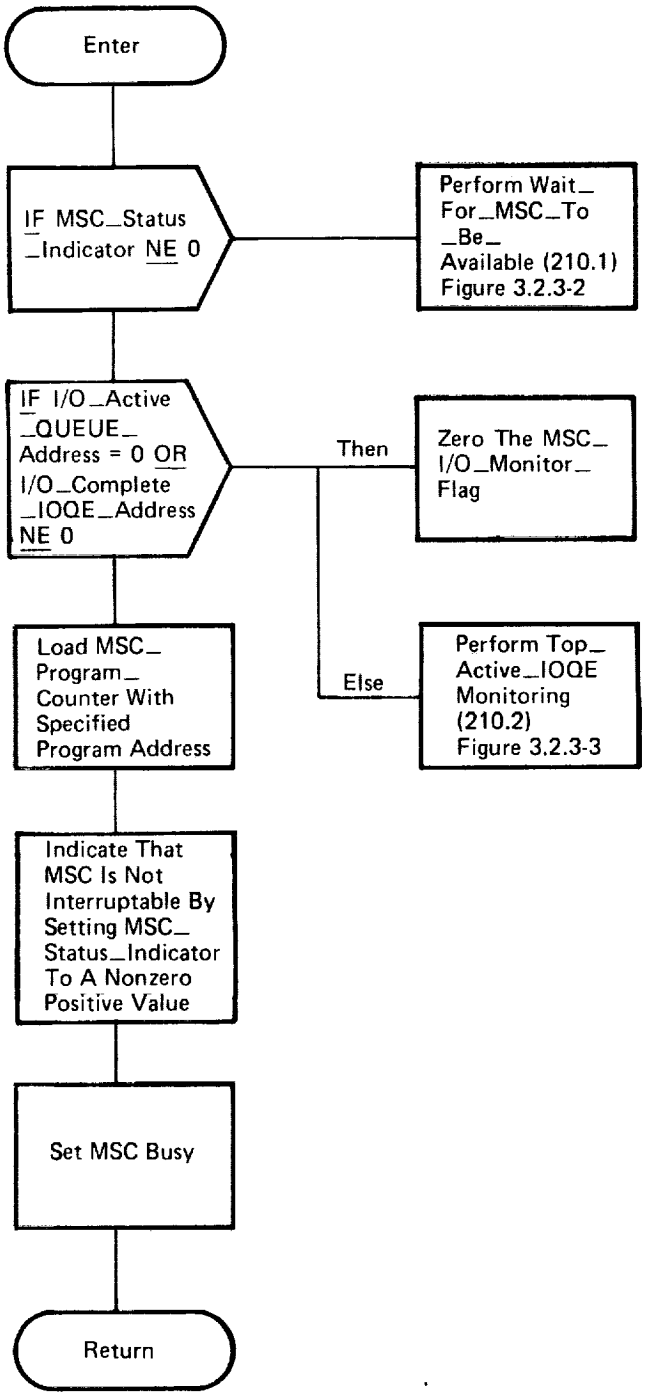


Figure 3.2.3-1. Start_MSC_Processor (FIOSTMSC)

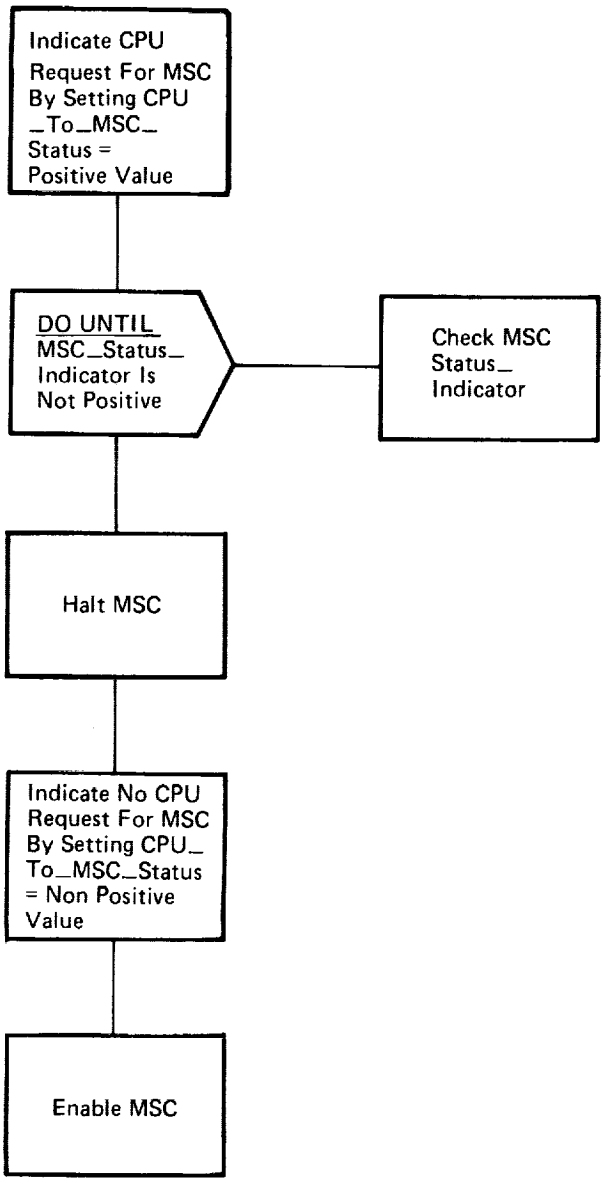


Figure 3.2.3-2. Start_MSC_Processor Wait_For_MSC_To_Be_Available (210.1)

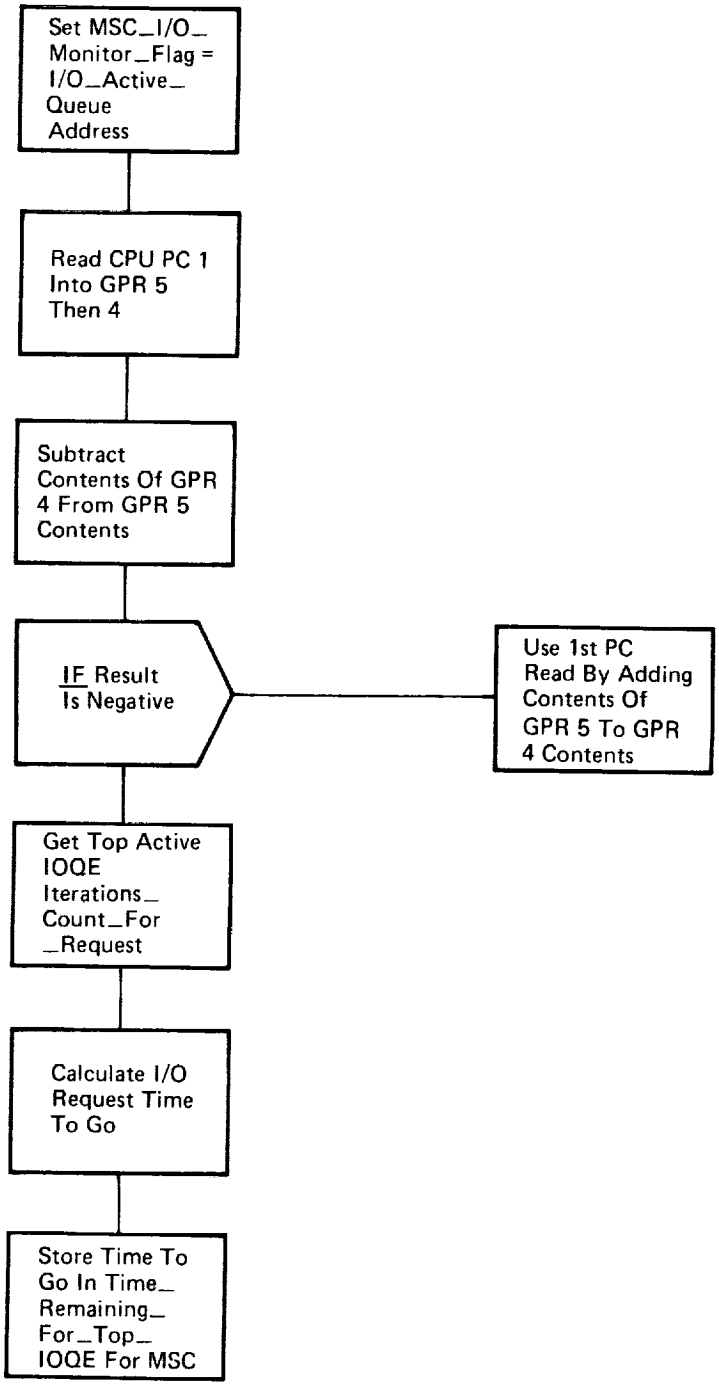


Figure 3.2.3-3. Start_MSC_Processor Top_Active_IOQE_Monitoring (210.2)

3.2.4 I/O_Completion_Processor (FIOCMPLT)(220)

To Be Provided

3.2.5 I/O_Termination_Processor (FIOPURGE) (225)

FIOPURGE dequeues IOQE's from the I/O WAIT Queue, and flags IOQE's from the I/O ACTIVE, I/O SUSPEND, and MM ACTIVE Queues with Parent_PCT_Terminated status. The flagged I/O ACTIVE and SUSPEND Queue IOQE's are dequeued by FIOCMPLT while the MM ACTIVE IOQE's are dequeued by FIOMMMGR. Also, FIOPURGE flags IOQE's that have invalid PCT's with Invalid_Parent_PCT status in support of FCOS CLOSE and FORCED CLOSE Processing.

a. Control Interfaces - CALLed by (133) Free_PCT_Routine (FPMFRPCT).

b. Input - General Purpose Register 0 bits 0-15 contain a PCT address. See Table 3.2.5-1 for other input to this module.

c. Process Description - The I/O_Termination_Processor when entered will reset an internal I/O WAIT Queue Indicator. The processor will then determine in succession if the I/O ACTIVE, I/O SUSPEND, MM ACTIVE, and I/O WAIT Queues are active by testing the I/O_Active_Queue_Address, IPR_IOQE_Address, Mass_Memory_Active_Queue_Address, and I/O_Wait_Queue_Address in the Communications_Vector_Table for nonzero values.

   After determining that a queue is active, the processor will set the internal I/O WAIT Queue indicator if the queue is the I/O WAIT Queue. Next, the processor will search the queue for an IOQE associated with the input PCT (Address_Of_Requesting_PCT_=PCT Address). If the search is successful, the processor will determine if the PCT Terminated_PCT_Indicator is set. If the Terminated_PCT_Indicator is not set, the IOQE Invalid_Parent_PCT will be set and the search will resume with the next IOQE in the queue.

   If the Terminated_PCT_Indicator is set, the processor will determine if the IOQE being processed is in the I/O WAIT Queue. If it is, the processor will dequeue the IOQE, and if the IOQE is pre-initialized, (Pre_Initialized_IOQE set) the IOQE will be returned to the IOQE Free Pool and IOQE_Flags_Field_1 reset. The IOQE search then resumes with the next IOQE in the queue.

   If the IOQE is not in the I/O WAIT Queue, the Parent_PCT_Terminated will be set. The IOQE search resumes with the next IOQE in the queue.

d. Output - See Table 3.2.5-1.

e. Module References - None

f. Module Attributes - Program

g. Template References - None

h. Error Checks - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.2.5-1

NAME I/O_Termination_Processor (FIOPURGE)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	I/O_Active_Queue_Address	Q001.5	I	205,220 305	210,225, 242,361	TCVTIOA			
2	IPR_IOQE_Address	Q001.40	I	220,305	220,225	TCVTIOIC			
3	Mass_Memory_Active_Queue_Address	Q001.16	I	280,305	225,280	TCVTMMA			
4	I/O_Wait_Queue_Address	Q001.6	I	See App. E	See App. E	TCVTIOW			
5	Address_Of_Next_IOQE_In_Chain	Q007.1	I/O	200,201 225	205,225	TIOQNXT			
6	Address_Of_Requesting_Process_PCT	Q007.2	I	200,201	225	TIOQPCT			
7	Terminated_PCT_Indicator	Q003.33	I	108,142	225	TPCTFLGS			
8	IOQE_Flags_Field_1	Q007.6	I/O	200,201, 225	201, 225	TIOQFLG1			
9	Pre_Initialized_IOQE	Q007.14	I	200,201	225	TIOQPREI			
10	Parent_PCT_Terminated	Q007.10	I	200	225	TIOQTRMM			
11	Invalid_Parent_PCT	Q007.16	I	200	225	TIOQIPCT			
12	I/O_Free_Pool_Address	Q001.12	I/O	See App. E	225	TCVTIOFP			

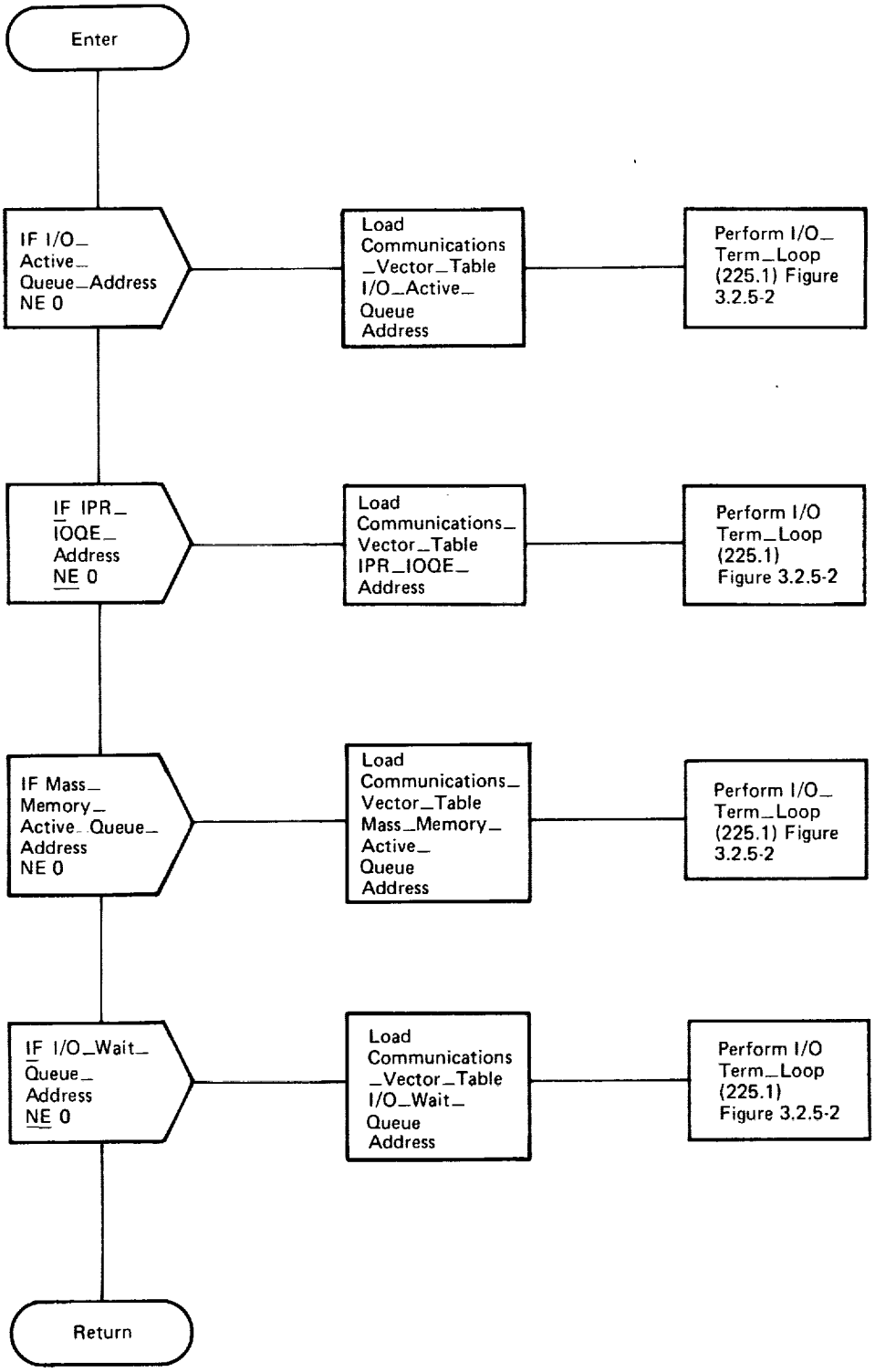


Figure 3.2.5-1. I/O_Termination_Processor (FIOPURGE)

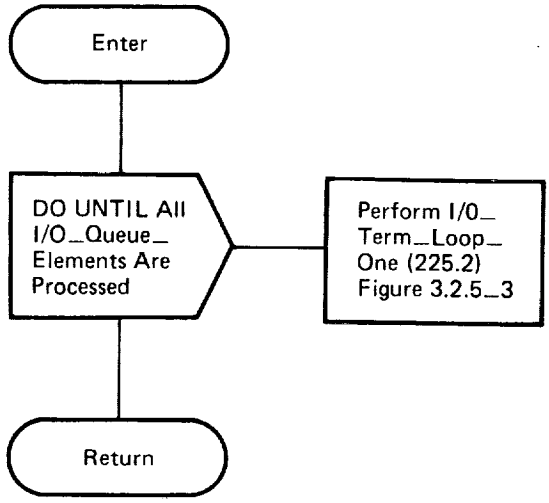


Figure 3.2.5-2. I/O_Termination_Processor I/O_Term_Loop (225.1)

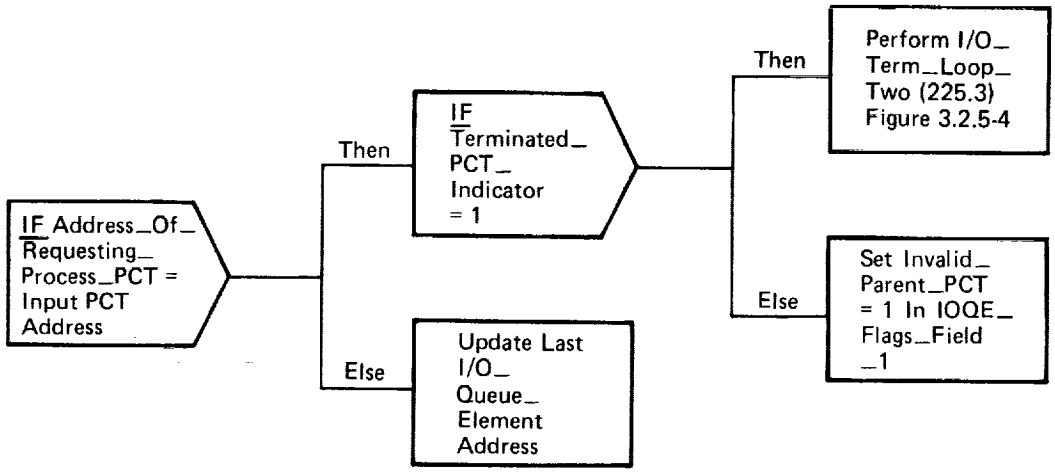


Figure 3.2.5-3. I/O_Termination_Processor I/O_Term_Loop_One (225.2)

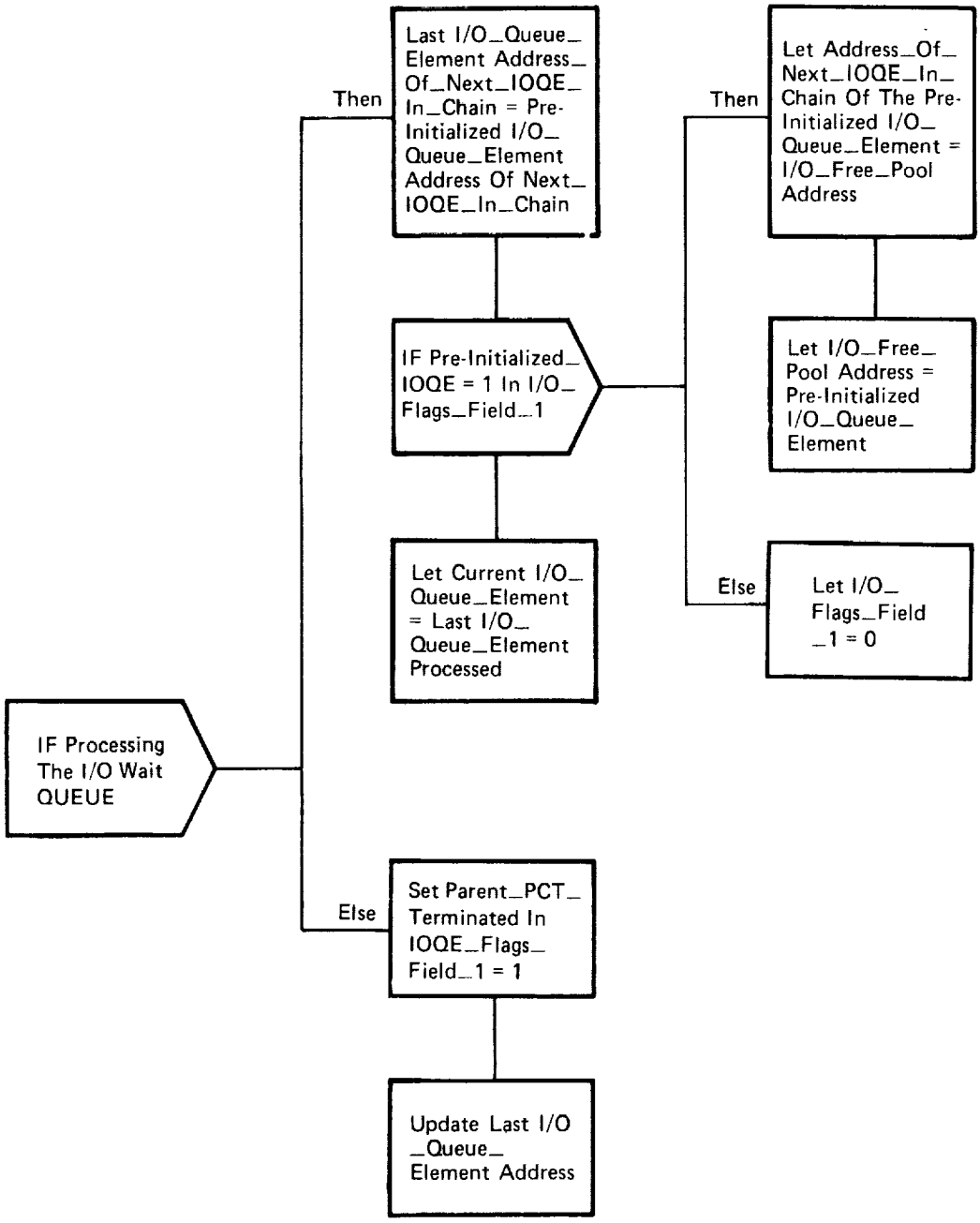


Figure 3.2.5-4. I/O_Termination_Processor I/O_Term_Loop_Two (225.3)

3.2.6.1 Level_A_I/O_Error_Interrupt_Handler (FIOERRLA) (240)

The Level_A_I/O_Error_Interrupt_Handler processes level A I/O interrupts generated by the IOP.

a. Control Interface - Hardware PSW swap because of level A I/O interrupt.

b. Inputs - See Table 3.2.6.1-1.

c. Process Description - The Interrupt_Register_A is read and the I/O_Error_Log_Routine is called. Then the type of error is determined and the appropriate action is taken. The error types and the action taken are listed below:

1. GO/NO-GO Timer Timeout - If the Timer_Termination_Control_Latch is one (set), then enter a wait state with all interrupts disabled. Otherwise, the error is ignored.

2. IOP Fail Latch - If the Voter_Termination_Control_Latch is one (set), then enter a wait state with all interrupts disabled. Otherwise, the error is ignored.

3. ROS Parity Error or IOP Fault - Reset the IOP and enter a wait state with all interrupts disabled.

If the machine is not placed into a wait state with all interrupts disabled, control is returned to the interrupted process.

The control flow for this module is shown in Figure 3.2.6.1-1.

d. Output - See Table 3.2.6.1-1.

e. Module References - (246) I/O_Error_Log_Routine is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation -

1. The Set System Mask (SSM) instruction is used to update PSW_System_Mask of current PSW to enter a wait state with all interrupts disabled.

2. Control is returned to the interrupted process via a Load PSW (LPS) instruction from External_Interrupt_0_Old_PSW.

DATA TABLE 3.2.6.1-1

NAME Level_A_I/O_Interrupt_Handler (FIOERRLA)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Interrupt_Register_A	&006.10	R		240				
2	GO/NO-GO_Timer_Interrupt	&006.11	R		240				
3	Timer_Termination_Control_Latch	#005.1	Z		240				
4	IOP_Fail_Latch_Interrupt	&006.12	R		240				
5	Voter_Termination_Control_Latch	#005.2	Z		240				
6	ROS_Parity_Error_Interrupt	&006.14	R		240				
7	IOP_Fault_Interrupt	&006.15	R		240				
8	External_Interrupt_0_Old_PSW	&001.75	I		240	TPSAEOP			

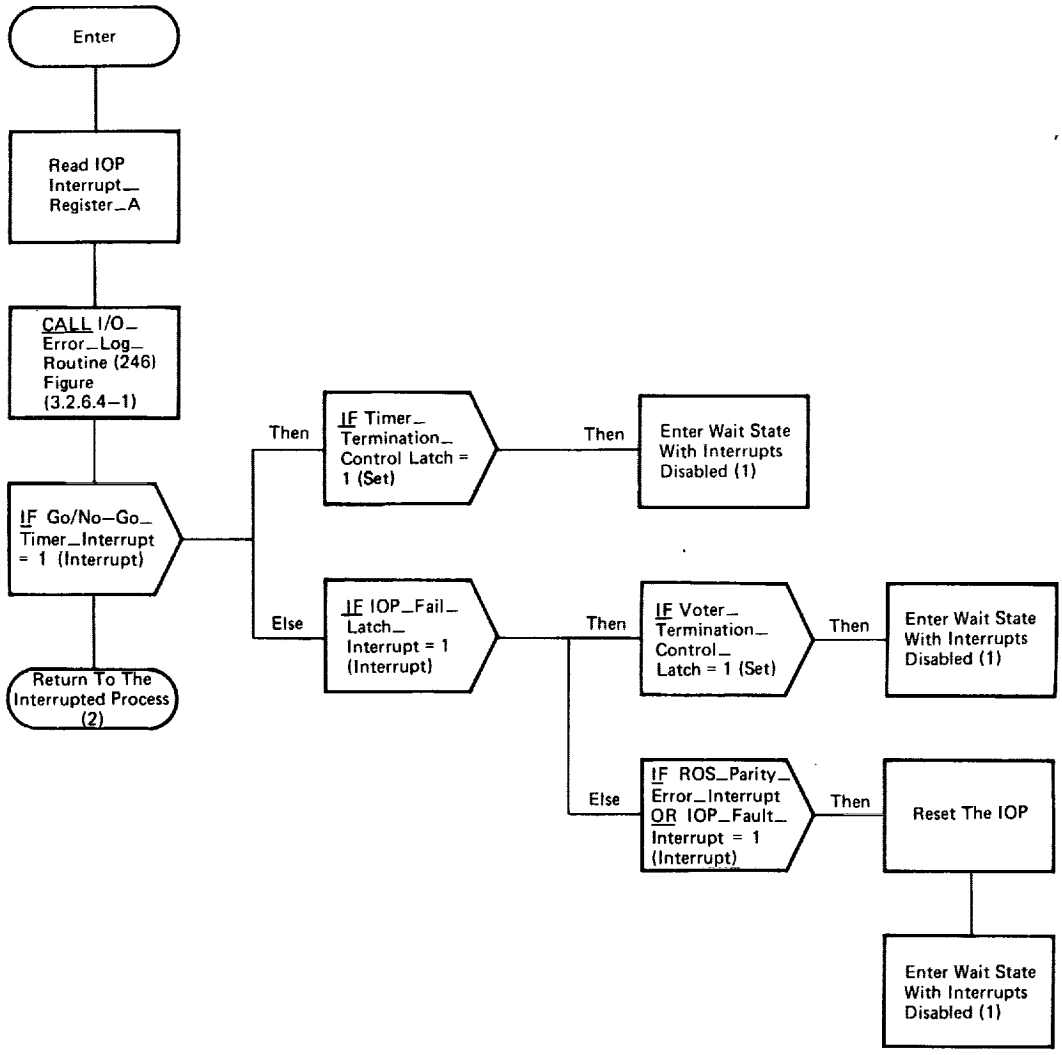


Figure 3.2.6.1-1. Level_A_I/O_Error_Interrupt_Handler (FIOERRLA)

3.2.6.2 Level_B_I/O_Error_Interrupt_Handler (FIOERRLB) (242)

The Level_B_I/O_Error_Interrupt_Handler processes level B I/O interrupts generated by the IOP.

a. Control Interface - Hardware PSW Swap because of level B I/O interrupt.

b. Inputs - See Table 3.2.6.2-1.

c. Process Description - Upon receiving control, the PSW_Interrupt_Code of the External_Interrupt_1_Old_PSW is examined to determine if it or the IOP Interrupt_Register_B contains the error information. If the PSW_Interrupt_Code is zero, the IOP Interrupt_Register_B is read, otherwise, the PSW_Interrupt_Code is used to determine the type of error. Figure 3.2.6.2-2 shows the various errors and the response made by FCOS. The I/O_Error_Log_Routine is called and control is either returned to the interrupted process or the machine is placed in a wait state with all interrupts disabled.

   The control flow for this module is shown in Figure 3.2.6.2-1.

d. Output - See Table 3.2.6.2-1.

e. Module References - (246) I/O_Error_Log_Routine (FIOLGERR) is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation -

   1. The Program-Controlled I/O (PC) instruction is used to read the IOP Interrupt_Register_B.

   2. The Program-Controlled I/O (PC) instruction is used to update the GO/NO-GO counter. The timeout value loaded is a data word of all zeros.

   3. The Program-Controlled I/O (PC) instruction is used to halt and enable the MSC.

4. This DO UNTIL loop is an infinite loop that is not terminated until GPC_AGE_Restart_Flag is reset to zero by an external manual intervention (i.e., the GPC is halted by the Microprogrammed Test Set, the GPC_AGE_Restart_Flag zeroed, and the GPC restarted). The purpose of this code is to define a Q-point for Simulation Stops/ Restarts.

5. Control is returned to the interrupted process via a Load PSW (LPS) instruction from External_Interrupt_1_Old_PSW.

6. The Program-Controlled I/O (PC) instruction is used to reset the IOP.

7. The Set System Mask (SSM) is used to update PSW_System_Mask of the current PSW to enter a wait state with all interrupts disabled.

DATA TABLE 3.2.6.2-1

NAME Level_B_I/O_Interrupt_Handler (FIOERRLB)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	PSW_Interrupt_Code	&002.26			180,242, 320				
2	External_Interrupt_1_Old_PSW	&001.77	I		242	TPSAE10P			
3	Interrupt_Register_B	&006.20	R		242				
4	I/O Active Queue Address	Q001.5	I	205,220 305	210,225 242,361	TCVTIOA			
5	AGE_Interrupt_GMT	Y027	O	242		FIOAGEFG			
6	Software_Clock_Half_Hour_Counter	Q001.88	I	141,142 149,305	149,162 242	TCVTSWCM			
7	GPC_AGE_Restart_Flag	Y026	I,O	242	242	FIORSUME			
8	AGE_Interrupt_Q_Point_Flag	Y028	I,O	242	242	FIOAGFL2			

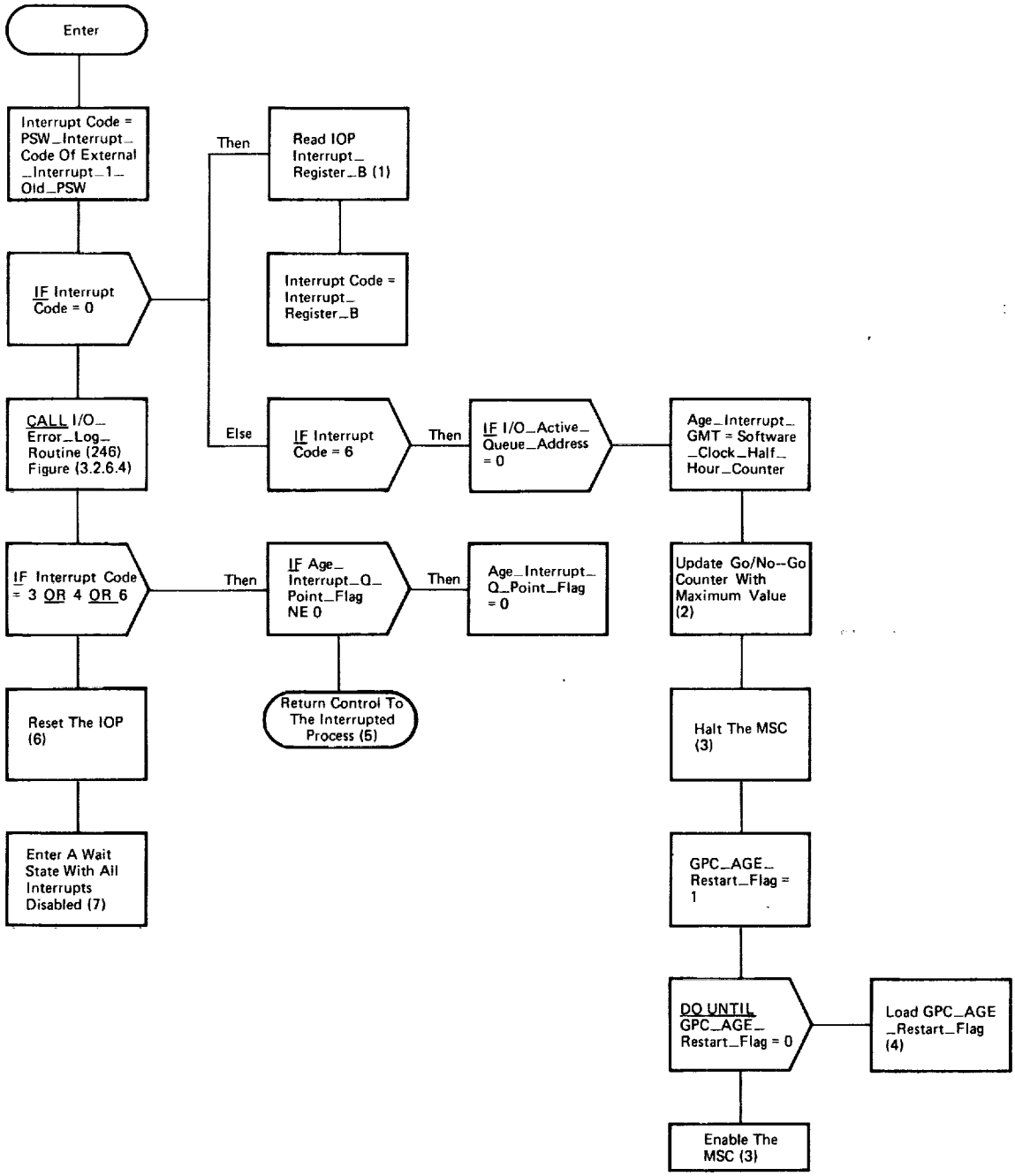


Figure 3.2.6.2-1. Level_B_I/O_Error_Interrupt_Handler (FIOERRLB)

Error Description	FCOS Response	Source
PCI/PCO Parity	Reset IOP, Enter WAIT State With Interrupts Disabled	Interrupt_Register B
DMA Instruction Parity	Reset IOP, Enter WAIT State With Interrupts Disabled	Interrupt_Register B
DMA Data Parity	Reset IOP, Enter WAIT State With Interrupts Disabled	Interrupt_Register B
Burst DMA Error	Reset IOP, Enter WAIT State With Interrupts Disabled	Interrupt_Register B
DMA Q Overflow	Reset IOP, Enter WAIT State With Interrupts Disabled	Interrupt_Register B
DMA Timeout	Reset IOP, Enter WAIT State With Interrupts Disabled	Interrupt_Register B
I/O Address Specification	Ignore	PSW_Interrupt_Code = 3
I/O Store Protect	Ignore	PSW_Interrupt_Code = 4
I/O Write Parity	Reset IOP, Enter WAIT State With Interrupts Disabled	PSW_Interrupt_Code = 2
PCI Data Parity	Reset IOP, Enter WAIT State With Interrupts Disabled	PSW_Interrupt_Code = 1
I/O Address Parity	Reset IOP, Enter WAIT State With Interrupts Disabled	PSW_Interrupt_Code = 5
AGE Interrupt	If The I/O–Active–Queue–Address Is Zero, Reload The Go/No–Go Counter With Its Maximum Value, Halt The MSC, Set GPC–AGE–Restart–Flag To A–1, And Loop Until GPC–AGE–Restart–Flag Is Set To Zero By Some External Manual Intervention. After GPC–AGE–Restart–Flag Has Been Zeroed, Restart The MSC And Continue.	PSW_Interrupt_Code = 6

Note: All Of The Above Errors Are Logged.

Figure 3.2.6.2-2. Level B I/O Error Response Description

3.2.6.3 Level_C_I/O_Error_Interrupt_Handler(FIOERRLC) (244)

To Be Provided

3.2.6.4 I/O_Error_Log_Routine (FIOLGERR) (246)

I/O_Error_Log_Routine records information pertinent to Level A, B, or C I/O error interrupts.

a. Control Interface -

1. CALLed by (240) Level_A_I/O_Error_Interrupt_Handler (FIOERRLA)
2. CALLed by (242) Level_B_I/O_Error_Interrupt_Handler (FIOERRLB)
3. CALLed by (244) Level_C_I/O_Error_Interrupt_Handler (FIOERRLC)

b. Input -

Bits 0-15 of register 2 contain BCE Element Number.
Bits 0-15 of register 3 contain Operation Code.
Bits 16-31 of register 3 contain Device ID.
Bits 0-15 of register 4 contain BCE Number.
Bits 16-31 of register 4 contain Residual Word Count.
Register 6 contains IOP Status Register.

c. Process Description - The I/O_Error_Log_Routine updates the I/O_Error_Log_Index to point to the oldest entry in the I/O_Error_Paramter of the I/O_Error_Log. The oldest entry is the entry to be overlaid. There is a separate entry in the I/O_Error_Log for storing the GPC error information for later transmission via ICC to the other active GPC's.

The BCE Element Number, Operation Code, BCE Number and Residual Word Count are stored in the cyclic Stack (I/O_Error_Parameters) and the GPC entry (GPC_X_I/O_Error_Parameter). The IOP Status Register is saved ir the I/O_Error_IOP_Status_Register and GPC_X_I/O_Error_IOP_Status_Register.

The GMT is obtained from the Last_TQE_Half_Hour_Portion COMPOOL parameter, converted from one microsecond resolution to 512 microsecond resolution, and saved in both the stack portion (I/O_Error_Time_Tag) and GPC entry (GPC_X_I/O_Error_Time_Tag).

The GPC_X_I/O_Error_Count is checked to see if it is less than or equal to the maximum number. If not it is incremented and stored. A flag is turned on in ICC_IO_Err_GPCx. GPCx is specified in I/O_Error_Log_ICC_Indicator_Mask. This flag will indicate the GPC portion of I/O_Error_Log to be transmitted via ICC to the other active GPC's.

The control flow for this module is shown in Figure 3.2.6.4-1.

d. Output - See Table 3.2.6.4-1.

e. Module References - None

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation -

1. Value of X to designate number of GPC is obtained from I/O_Error_Log_Table_Address.

2. Value of X to designate number of GPC is obtained from I/O_Error_Log_ICC_Indicator_Mask.

DATA TABLE 3.2.6.4-1

NAME I/O_Error_Log_Routine (FIOLGERR)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	I/O_Error_Log	I280	I	246	246,485 660,665	CZ2VIOER			
2	I/O_Error_Log_Index	I280.10	I/O	246	246	TIOEINDX, CZ2BERRX	V92U4320C	X	
3	GPC_Self_I/O_Error_Log	I280.1	I	246	246,660 665	(TIOESTAK)		X	
4	I/O_Error_Log_Table_Address	Q001.53	I	300	246	TCVTIERA			
5	Latest_I/O_Error_Entries	I280.11	O	246	485,660 665	TIOELIST			
6	I/O_Error_Parameters	I280.2	O	246		TIOEPRMS			
7	I/O_Error_BCE_Number	I280.3	O	246		TIOEPRMS			
8	I/O_Error_BCE_Element_Number	I280.4	O	246		TIOEPRMS			
9	I/O_Error_Device_ID	I280.5	O	246		TIOEPRMS			
10	I/O_Error_Op_Code	I280.6	O	246		TIOEPRMS			
11	I/O_Error_Residual_Word_Count	I280.7	O	246		TIOEPRMS			
12	GPC_X_I/O_Error_Parameters	I280.12	O	246		TIOEPRML	See Appendix E	X	
13	I/O_Error_IOP_Status_Register	I280.8	O	246		TIOESRGS	See Appendix E	X	
14	GPC_X_I/O_Error_IOP_Status_Register	I280.13	O	246		TIOESRGL	See Appendix E	X	
15	Last_TQE_Half_Hour_Portion	#025.1	I	142,149	140,144 246	TFCMLTQH	V91W1999C	X	
16	I/O_Error_Time_Tag	I280.9	O	246		TIOEGMTS	See Appendix E	X	
17	GPC_X_I/O_Error_Time_Tag	I280.14	O	246		TIOEGMTL	V92W4768C	X	
							V92W4845C		
							V92W4918C		

DATA TABLE 3.2.6.4-1 (Cont'd)

NAME I/O_Error_Log_Routine (FIOLGERR)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
							V92W5005C		
18	GPC_X_I/O_Error_Count	I280.15	I/O	246	246	TIOERCNT	V92Q4770C	X	
							V92Q4847C		
							V92Q4720C		
							V92Q5007C		
19	ICC_IO_ERR_GPC1	I320.22	O	246		CZ2VIF2			
20	ICC_IO_ERR_GPC2	I320.23	O	246		CZ2VIF2			
21	ICC_IO_ERR_GPC3	I320.24	O	246		CZ2VIF2			
22	ICC_IO_ERR_GPC4	I320.25	O	246		CZ2VIF2			
23	ICC_IO_ERR_GPC5	I320.26	O	246		CZ2VIF2			
24	I/O_Error_Log_ICC_Indicator_Mask	Q001.56	I	300	246	TCVTIERM			

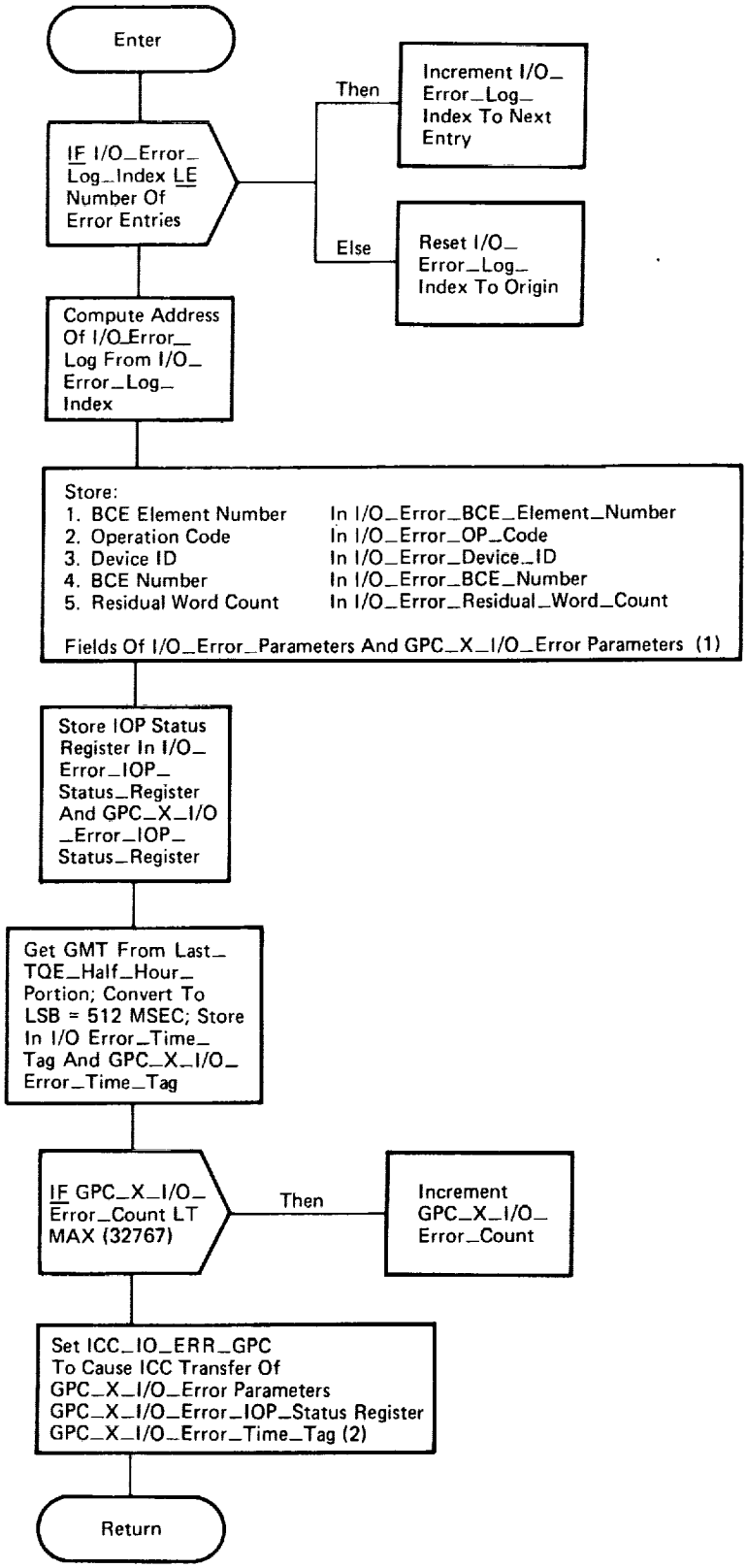


Figure 3.2.6.4-1. I/O_Error_Log_Routine (FIOLGERR)

3.2.7 MSC Programs

The MSC programs are executed by the MSC in the IOP. One of their functions is to start the command mode Bus Control Elements (BCE) which the CPU has designated. One or more buses can be started by the MSC in a single request by the CPU. When the buses are started the CPU indicates to the MSC which buses are to be monitored (Both listen and command mode buses are monitored). The MSC, monitors the designated buses and interrupts the CPU upon their completion; it also notes to the CPU whether or not an error has occurred on one of the monitored buses. The completed BCE programs for listen mode buses are then reset to a wait for index (WIX) instruction.

The following MSC programs are documented in this section (see Figure 3.2.7 -1):

a. MSC_Control_Routine (250) - This program is used to start BCE programs other than Mass Memory BCE programs.

b. MSC_I/O_Monitor (251) - This program monitors BCE's (other than mass memory) for I/O Completion.

c. MSC_BCE_Reset_Routine (252) - This program causes BCE's to be placed on the standard system wait for index (WIX) instruction.

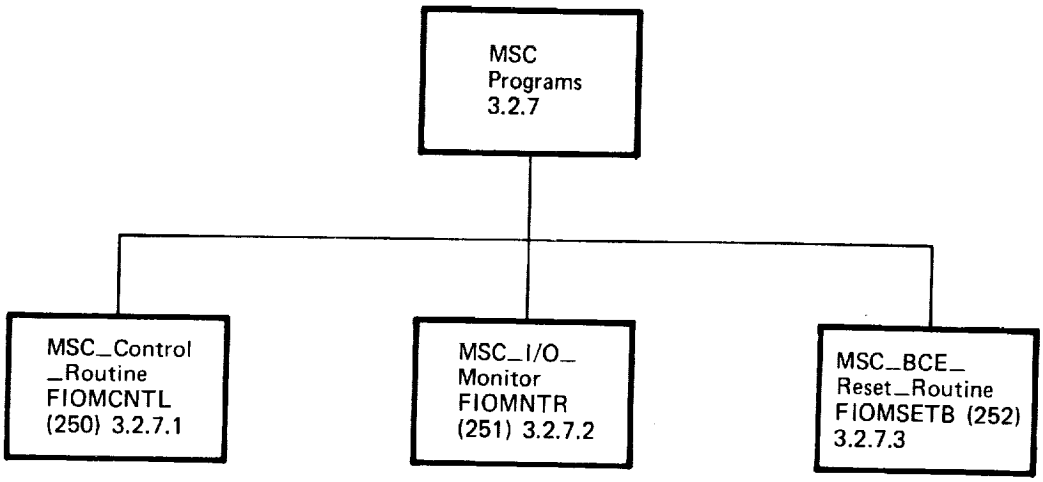


Figure 3.2.7-1. MSC Programs

3.2.7.1 MSC_Control_Routine (FIOMCNTL) (250)

FIOMCNTL is a program which is executed by the MSC to initiate BCE execution.

a. Control Interface - Invoked by (210) Start_MSC_Processor (FIOSTMSC) when the MSC_Program_Counter is loaded.

b. Input - See Table 3.2.7.1-1.

c. Process Description - The MSC_Control_Routine takes the Start_I/O_BCE_Mask which indicates which buses are to be started and forms the logical product with the Requested_I/O_Transmitter_Mask. The logical product indicates the buses which the MSC_Control_Routine then starts. The control flow for this module is presented in Figure 3.2.7.1-1.

d. Output - See Table 3.2.7.1-1.

e. Module References -

1. (251) MSC_I/O_Monitor (FIOMNTR) routine is called.
2. (260) DDU_BCE_Processor (FICDDUPG) is started by an SIO instruction.
3. (261) DEU_BCE_Processor (FIODEUPG) is started by an SIO instruction.
4. (262) ICC_BCE_Processor (FIOICCPG) is started by an SIO instruction.
5. (263) LDB_BCE_Processor (FIOLDBPG) is started by an SIO instruction.
6. (264) MDM_BCE_Processor (FIOMDMPG) is started by an SIO instruction.
7. (265) Payload_Discrete_BCE_Processor (FIOPDSPG) is started by an SIO instruction.
8. (266) PMU_BCE_Processor (FIOPMUPG) is started by an SIO instruction.
9. (267) PROM_BCE_Processor (FIOPRMPG) is started by an SIO instruction.

f. Module Attributes - MSC Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.2.7.1-1

NAME MSC_Control_Routine (FIOMCNTL)

NO.	ITEM	ID	ACT	SOURCE	DESTINATION	ASSEMBLER NAME	MML	D	C
1	Start_I/O_BCE_Mask	Q001.18	I	205 280,305	250	TCVTSIOM			
2	Requested_I/O_Transmitter_Mask	#003	I	351,355	205,250, 350,351	TFCMXMSK			
3	Busy_Wait_Register	&006.3	W	250,251 252,280	251,280, 281				

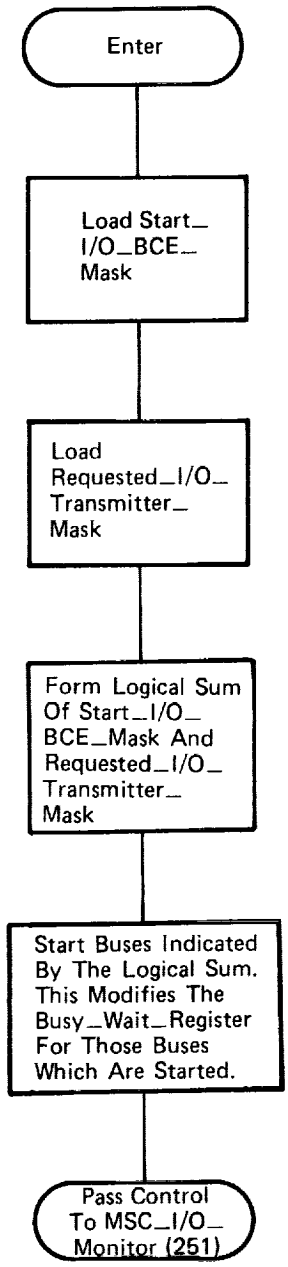


Figure 3.2.7.1-1. MSC_Control_Routine (FIOMCNTL)

3.2.7.2 MSC_I/O_Monitor (FIOMNTR) (251)

FIOMNTR monitors the BCE's specified in the Top_Active_IOQE for I/O completion. When the BCE's have completed, FIOMNTR generates an interrupt to the CPU.

a. Control Interface -

1. Invoked by (210) Start_MSC_Processor (FIOSTMSC) when the MSC_Program_Counter is loaded.
2. Called by (250) MSC_Control_Routine (FIOMCNTL)
3. Called by (252) MSC_BCE_Reset_Routine (FIOMSETB)
4. Called by (281) Mass_Memory_MSC_Processor (FIOMMMSC)

b. Input - See Table 3.2.7.2-1.

c. Process Description - The MSC_I/O_Monitor receives control whenever BCE programs might have to be monitored for completion. It checks the MSC_I/O_Monitor_Flag to determine if it has I/O to monitor. If it does not, it puts the MSC into a wait state until it is restarted by the CPU. (The MSC_I/O_Monitor never returns to its caller.)

If there were BCE's to monitor it obtains the Mask_of_Buses_to_Monitor. If the Mask_of_Buses_to_Monitor is negative the I/O has already been marked complete by the CPU, or there are no buses to monitor. In either case, the MSC_I/O_Monitor stores the completed IOQE address in the I/O_Complete_IOQE_Address, interrupts the CPU and then places the MSC in a wait state.

If there were BCE's to monitor the program calculates how long it is until the interrupt will occur. If the time is too far into the future the MSC_I/O_Monitor delays until it is about the time for the interrupt. It then waits for the BCE's to complete. If the BCE's do not complete in time or an error is indicated in the Program_Exception_Register, an error flag is set in the MSC_Completed_I/O_Request. If an error was not detected, the monitored buses are restarted. The address of the completed IOQE is moved from the MSC_Completed_I/O_Request to the I/O_Complete_IOQE address and the CPU is interrupted. The MSC is then placed in a wait state until restarted by the CPU.

The control flow for this module is presented in Figure 3.2.7.2-1.

d. Output - See Table 3.2.7.2-1.

e. Module References -

1. (268) Common_BCE_Processing (#PFIOECT) is started by an @SIO MSC instruction.

f. Module Attributes - MSC Program

g. Template References - N/A

h. Error Handling -

1. MSC Timeout is detected and communicated to the CPU via the I/O_Complete_IOQE_Address. The buses associated with the error are not restarted to position them at a WIX instruction.

2. If an error is indicated for one of the monitored buses in the Program_Exception_Register, the MSC_I/O_Monitor communicates this to the CPU via the I/O_Complete_IOQE_Address. The buses associated with this I/O request are not restarted to position them at a WIX instruction.

i. Constraints and Assumptions - None

j. Detailed Implementation -

1. This block is never executed since the MSC has gone into a wait state prior to this point. When the MSC is restarted by the CPU it will start execution where the CPU specifies, not at the instruction after the wait.

2. The MSC_Monitor_Count is a fictitious constant used in the flowchart for readability. In the actual code the value resides in a register.

DATA TABLE 3.2.7.2-1

NAME MSC_I/O_Monitor (FIOMNTR)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	MSC_I/O_Monitor_Flag	Q001.93	I	210	251	TCVTMNTR			
2	MSC_Status_Indicator	Q001.97	O	210,251, 310	210	TCVTMSC			
3	Busy_Wait_Register	&006.3	WR	250,251 252,280	251,280, 281				
4	Top_Active_IOQE_18_Bit_Address	Q001.7	I	205,220 305	250,251	TCVTMIOA			
5	MSC_Completed_I/O Request	Y023.1	L	251	251	FIOMIOQE			
6	Mask_of_Buses_to_Monitor	Q007.4	I	200,201, 351	200,205, 251	TIOQMNTM			
7	CPU_to_MSC_Status	Q001.98	I	210,310	251	TCVTCPU			
8	Interrupt_Register_C	&006.30	W	251					
9	IOQE_Device_ID	Q007.43	I	200	205,251, 351	TIOQDVID			
10	MSC_SSIP_ICC_Device_Id	Y023.2	L		251	FIOSSIP			
11	MSC_Monitor_Count	Y023.4	L	251	251				
12	MSC_I/O_Monitor_Window	Y023.3	L		251	FIOMWNDS FIOMWNDO			
13	Program_Exception_Register	&006.2	R		251				
14	Time_Remaining_For_Top_IOQE	Q001.92	I	210	251	TCVTMTTG			
15	I/O_Complete_IOQE_Address	Q001.78	O	220,250, 305	210,220, 251				

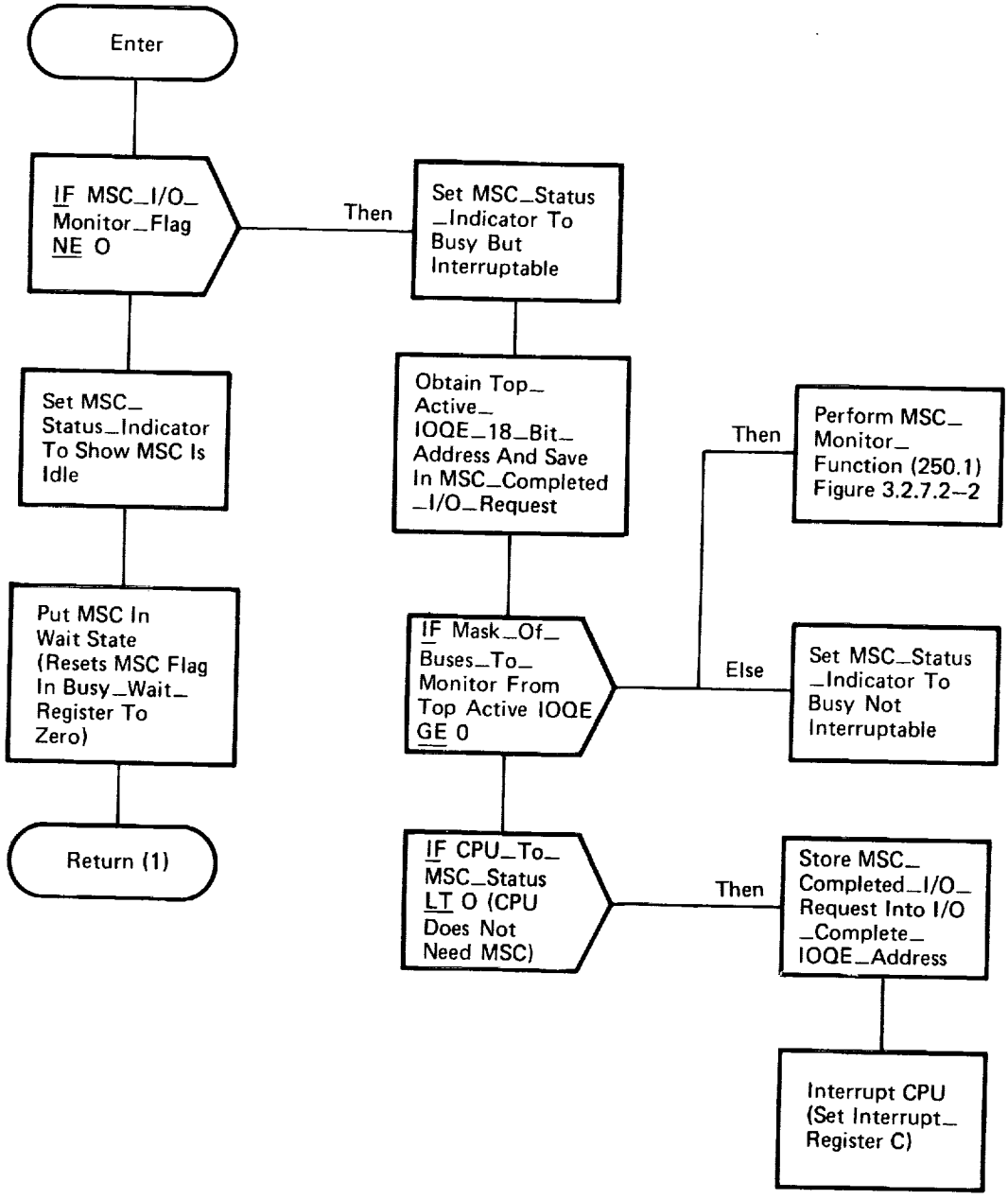


Figure 3.2.7.2-1. MSC_I/O_Monitor (FIOMNTR)

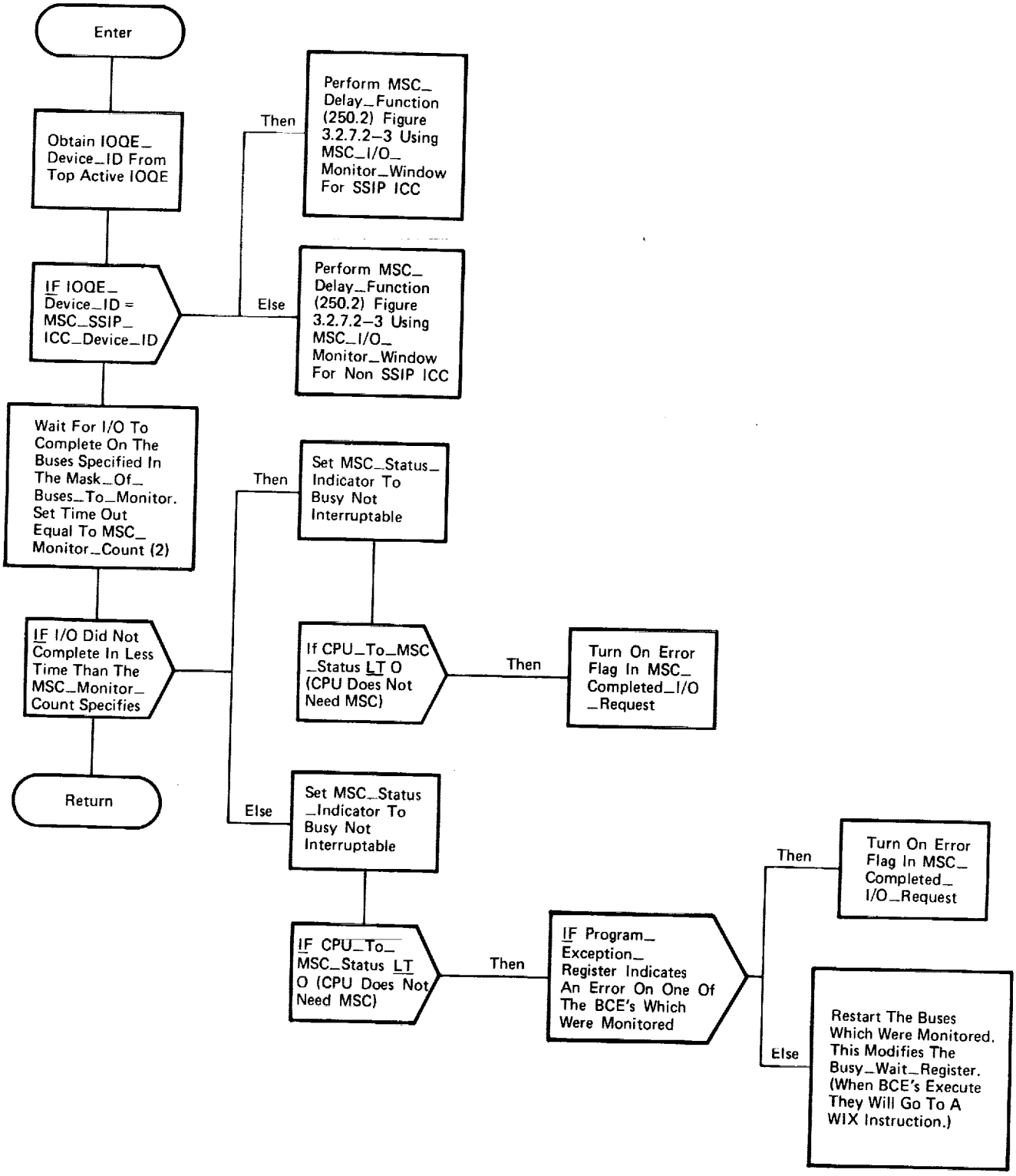


Figure 3.2.7.2-2. MSC_I/O_Monitor MSC_Monitor_Function (250.1)

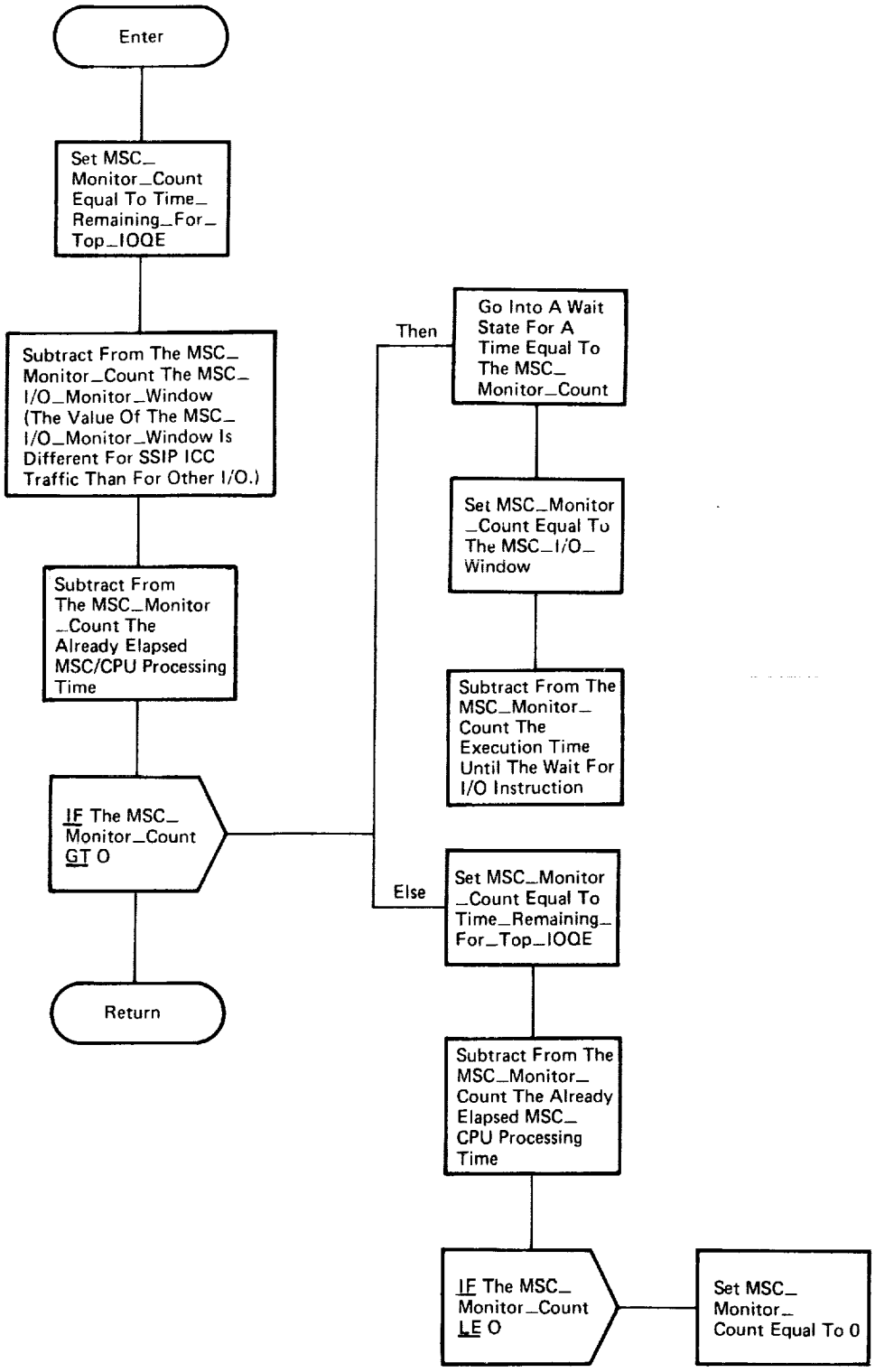


Figure 3.2.7.2-3. MSC_I/O_Monitor MSC_Delay_Function (250.2)

3.2.7.3 MSC_BCE_Reset_Routine (FIOMSETB) (252)

FIOMSETB is an MSC program which causes BCE's to be placed on a WIX instruction.

a. Control Interface - Invoked by (210) Start_MSC_Processor (FIOSTMSC) when the MSC_Program_Counter is loaded.

b. Input - See Table 3.2.7.3-1.

c. Process Description - When the MSC_BCE_Reset_Routine receives control, it starts the buses specified by the Mask_of_Buses_to_Be_Reset. It then passes control to MSC_I/O_Monitor. The buses which were started have been initialized by the Level_C_I/O_Error_Interrupt_Handler to go to a WIX instruction when started. The control flow for this module is presented in Figure 3.2.7.3-1.

d. Output - See Table 3.2.7.3-1.

e. Module References -

1. (251) MSC_I/O_Monitor (FIOMNTR) is CALLed.

2. (268) Common_BCE_Processing (#PFIOECT) is started by an @SIO MSC instruction.

f. Module Attributes - MSC Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.2.7.3-1

NAME MSC_BCE_Reset_Routine (FIOMSETB)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Mask_of_Buses_to_Be_Reset	Y024	I	244	252	FIOMSBMK			
2	Busy_Wait_Register	&006.3	W	250,251, 252,280	251,280, 281				

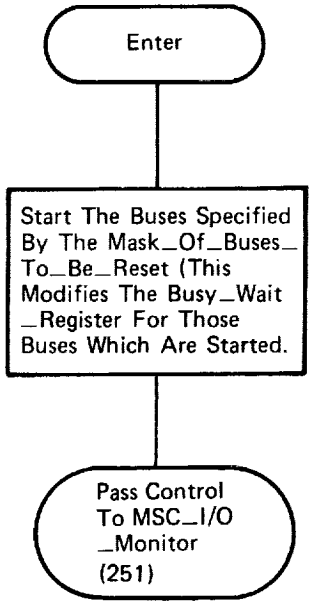


Figure 3.2.7.3-1. MSC_BCE_Reset_Routine (FIOMSETB)

3.2.8 BCE Programs

Most FCOS BCE programs are designed for a specific bus terminal unit to be used with a specific I/O transaction. However, some BCE programs have been made general enough to be used by several I/O transactions to communicate with different bus terminal units. The BCE programs used with TCS I/O requests are examples of generalized BCE programs.

To save CPU setup time, most FCOS BCE programs have been hardcoded. This means that the buffer areas used and the data requested by these programs is never changed.

Some hardcoded BCE programs are structured to facilitate bypassing of commands to retrieve data from LRU's which are failed or powered off.

A BCE program executes in either the command or listen mode. The BCE program may transmit both commands and data and also receive data while in command mode. In listen mode, the BCE program may receive data and listen commands but is not allowed to transmit commands or data. A BCE in listen mode relies on BCE programs in other IOP's to command a subsystem to return data. A listen BCE program executes a Wait-for-Index (WIX) instruction when in the busy state. This WIX instruction monitors for a listen command for another IOP. The listen command contains an index into a table of branch addresses and these addresses are used to start a specific BCE program. The presence of a command to a subsystem on the listen BCE bus is used as a signal to start receiving data when a Receive Data Instruction is being executed by a listen mode BCE. The BCE program in command mode executes a WIX instruction, but the WIX becomes a WAIT for command mode BCE's.

For input operations, a BCE program running in command mode sends a listen command to all listening BCE's. After the listen command has been issued, some delay instructions are executed to allow the listening BCE's to have time to receive the command, interpret it, and get set up to receive the data. After the delays, the commanding BCE executes a #MIN instruction to command the subsystem to return the appropriate data and then prepares itself to handle this data. When the subsystem places its data on the bus, the command and listen mode BCE's are prepared to accept it, and consequently receive exactly the same data at exactly the same time. The data path (annuciation) counters are zeroed by BCE input programs which complete successfully.

For output operations, after a BCE program running in the command mode has sent all of its data to the various subsystems, a listen command will be sent to all listeners to allow them to indicate to their MSC's that the output operation has completed.

The following BCE programs are documented in this section (see Figure 3.2.8-1):

a. (260) DDU_BCE_Processor (FIODDUPG)- These programs output data to the Dedicated Displays.

b. (261) DEU_BCE_Processor (FIODEUPG)- These programs are used to communicate with all DEU's.

c. (262) ICC_BCE_Processor (FIOICCPG)- These programs provide support SSIP ICC data transfers and IPR ICC data transfer between redundant or common set GPC's.

d. (263) LDB_BCE_Processor (FIOLDBPG)- These BCE programs provide GPC and ground communication via the GSE.

e. (264) MDM_BCE_Processor (FIOMDMPG)- These programs include support for TCS I/O request, APU outputs, MTU write nose wheel steering commands, and MDM BITE acquisition.

f. (265) Payload_Discrete_BCE_Processor (FIOPDSPG)- These BCE programs support caution and warning discrete outputs to the payload MDM's.

g. (266) PMU_BCE_Processor (FIOPMUPG)- These BCE programs provide all FCOS I/O support for the PMU.

h. (267) PROM_BCE_Processor (FIOPRMPG)- These BCE programs provide GN&C flight critical inputs and outputs and MTU reads.

i. (268) Common_BCE_Processing (#PFIOECT)- This module contains BCE code for I/O completion common to all BCE programs.

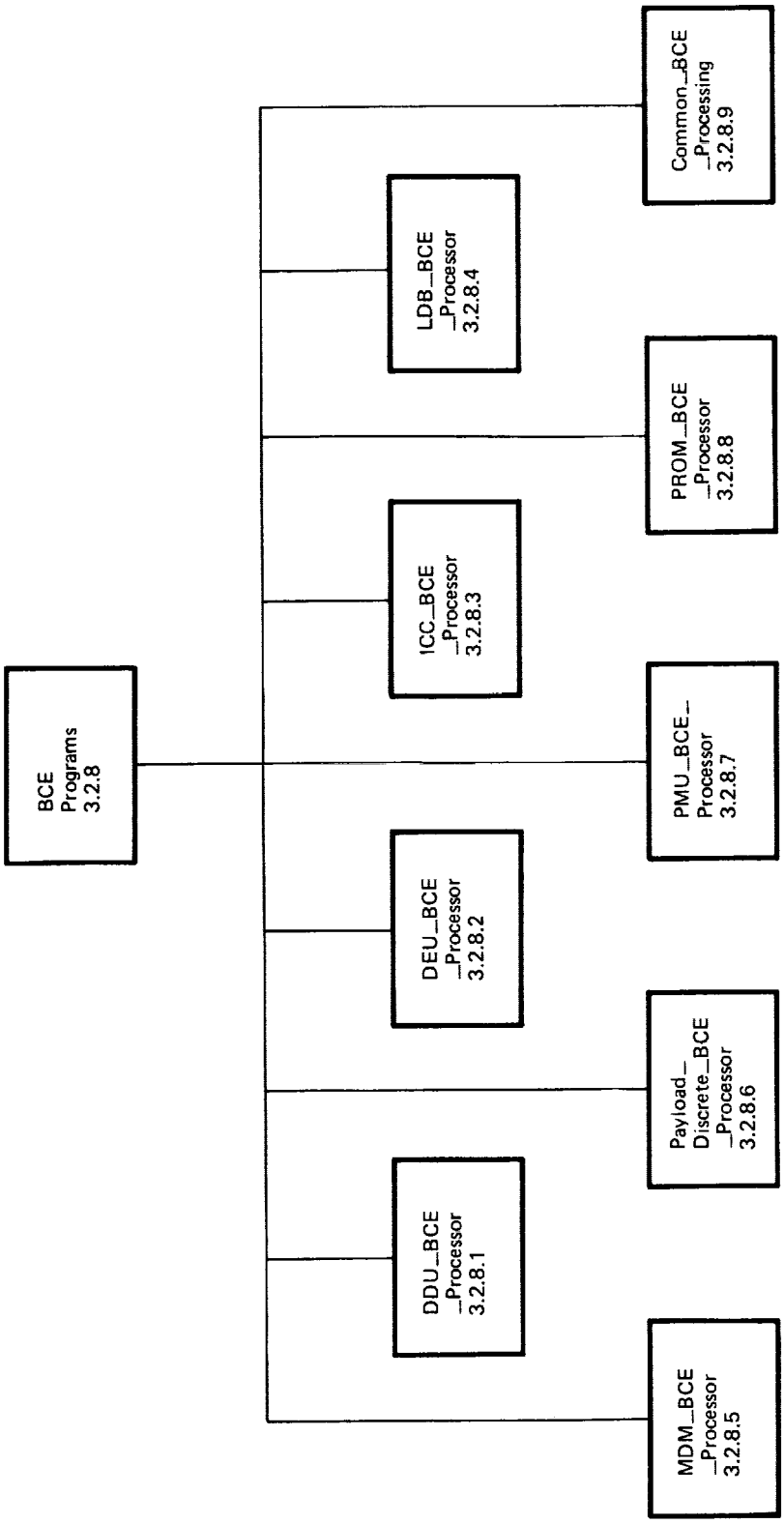


Figure 3.2.8-1. BCE Programs

3.2.8.1 DDU_BCE_Processor (FIODDUPG) (260)

FIODDUPG does output to the Attitude Direction Indicators, Alpha/Mach Indicators, Altitude/Vertical Velocity Indicators, and the Horizontal Situation Indicators attached to DDU1 and DDU2.

a. Control Interface - Initiated by SIO instruction in (250) MSC_Control_Routine (FIOMCNTL).

b. Input - See table 3.2.8.1-1.

c. Process Description - Write 14 words to the ADI, 6 words to the AMI, 6 words to the AVVI, and 10 words to the HSI on DDU1. Repeat the same process for DDU2. Data is from the GN&C I/O Buffers. The control flow for this module is shown in Figure 3.2.8.1-1.

d. Output - See Table 3.2.8.1-1.

e. Module References -
(268) Common_BCE_Processing (#PFIOECT) is invoked.

f. Module Attributes - BCE Program.

g. Template References - None.

h. Error Handling - None.

i. Constraints and Assumptions - None.

j. Detailed Implementation - None.

DATA TABLE 3.2.8.1-1

NAME DDU_BCE_Processor (FIODDUPG)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	BCE_Base_Register	&008.09	W	282,302	260,282				
2	DDU1_Output_Buffer	Y030	I	260	260	FIODDUB1			
3	DDU2_Output_Buffer	Y031	I	260	260	FIODDUB2			

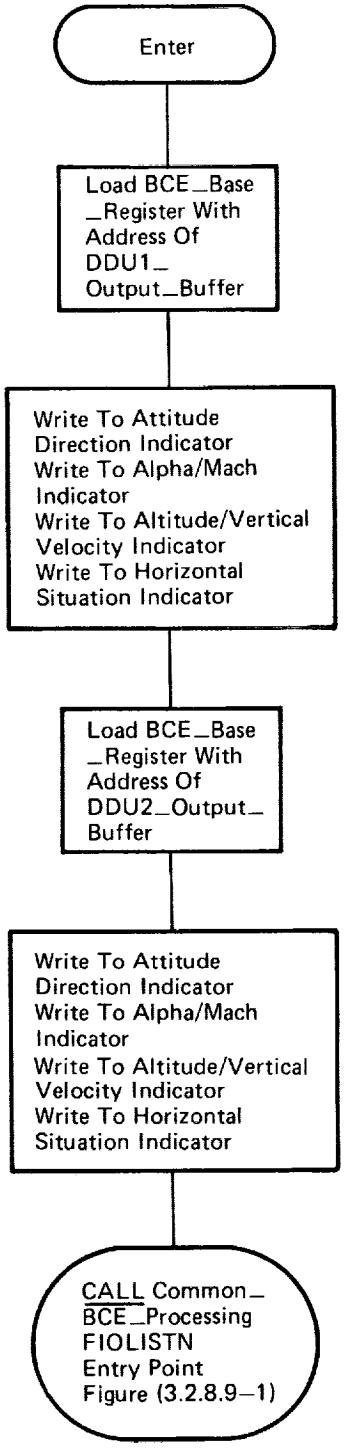


Figure 3.2.8.1-1. DDU_BCE_Processor (FIODDUPG)

3.2.8.2 DEU_BCE_Processor (FIODEUPG) (261)

FIODEUPG performs the DEU operations of poll, dump, memory fill, reset scratch pad line, keyboard request, and bite status request.

a. Control Interface - FIODEUPG is initiated by a SIO instruction in (250) MSC_Control_Routine (FIOMCNTL)

b. Input - See Table 3.2.8.2-1.

c. Process Description - FIODEUPG consists of 6 segments that perform the following DEU operations:

Poll	(261.01)
Dump	(261.02)
Memory Fill	(261.03)
Reset Scratch Pad Line	(261.04)
Keyboard Request	(261.05)
Bite Status Request	(261.06)

The control flow for this module is shown in Figure 3.2.8.2-1.

d. Output - See Table 3.2.8.2-1.

e. Module References -

(268) Common_BCE_Processing (#PFIOECT) is invoked.

f. Module Attributes - BCE Program

g. Template References - N/A

h. Error handling - None

i. Constraints and Assumptions - None

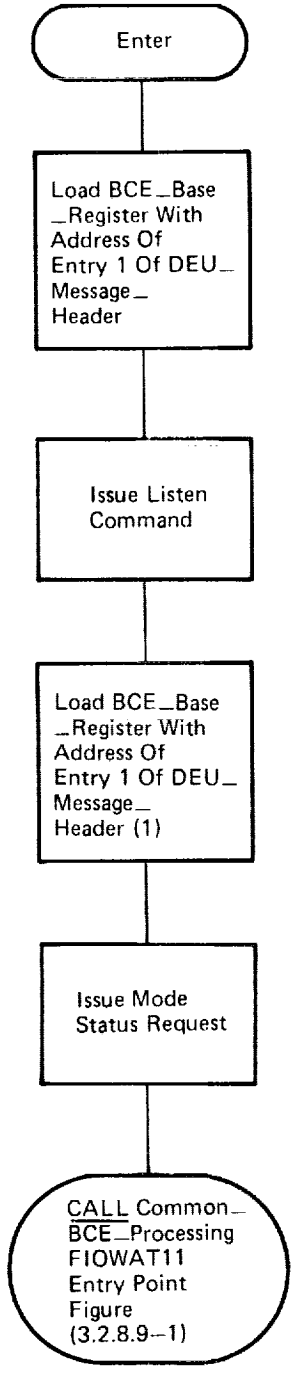
j. Detailed Implementation -

1. Listen Command entry point

DATA TABLE 3.2.8.2-1

NAME DEU_BCE_Processor (FIODEUPG)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	BCE_Base_Register	&008.09	W	SEE App. E	SEE App. E				
2	DEU_Message_Header	J060.02	Z	SEE App. E		FIODBF1P	See App. E		
3	BCE_Base_Register_Table	Y005	I	205,280	SEE App. E	FIOBRE			
4	BCE_Word_Count_Table	Y006	I	205	261,263 264,282	FIOWCE			
5	DEU_BITE_Message_Header	J070.05	Z	261,405, 790	405,790	FIODBF1B			
						FIODBF2B			
						FIODBF3B			



Similar Control Flows For DEU2 And DEU3

Figure 3.2.8.2-1. DEU_BCE_Processor DEU_Poll (261.1)

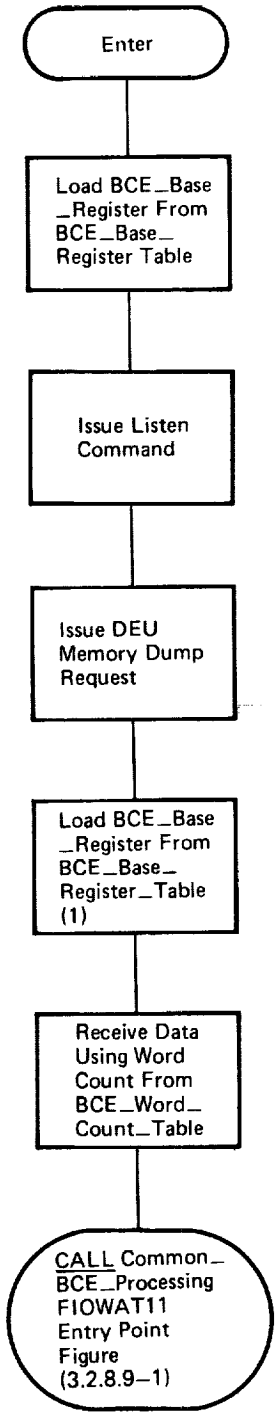


Figure 3.2.8.2–2. DEU_BCE_Processor DEU_Dump (201.2)

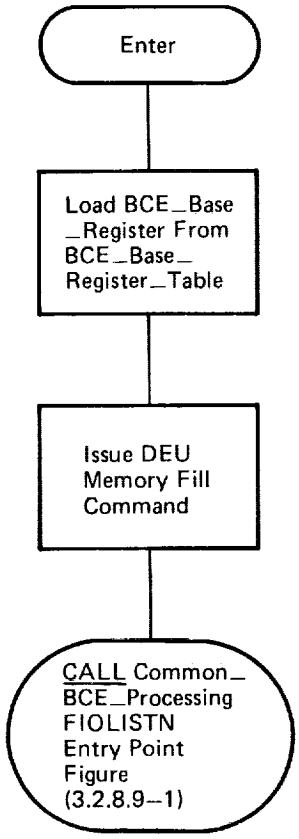


Figure 3.2.8.2–3. DEU_BCE_Processor DEU_Memory_Fill (261.3)

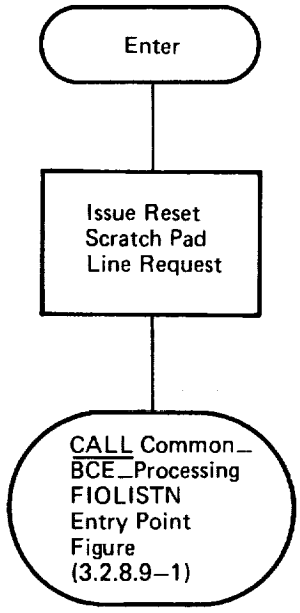
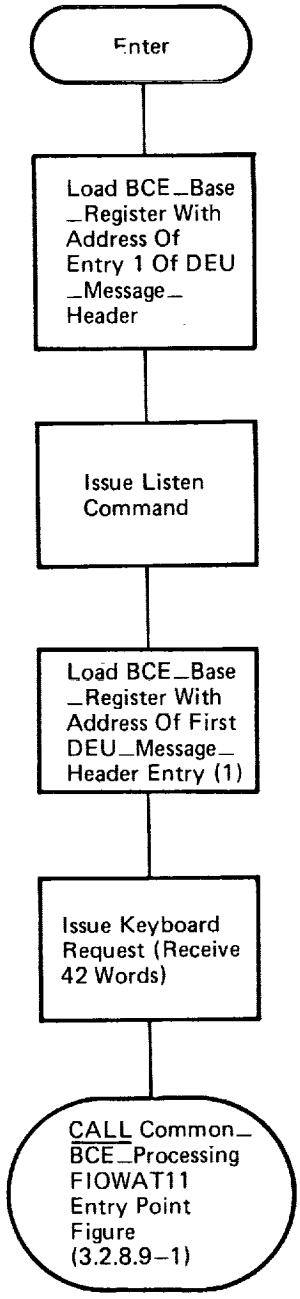
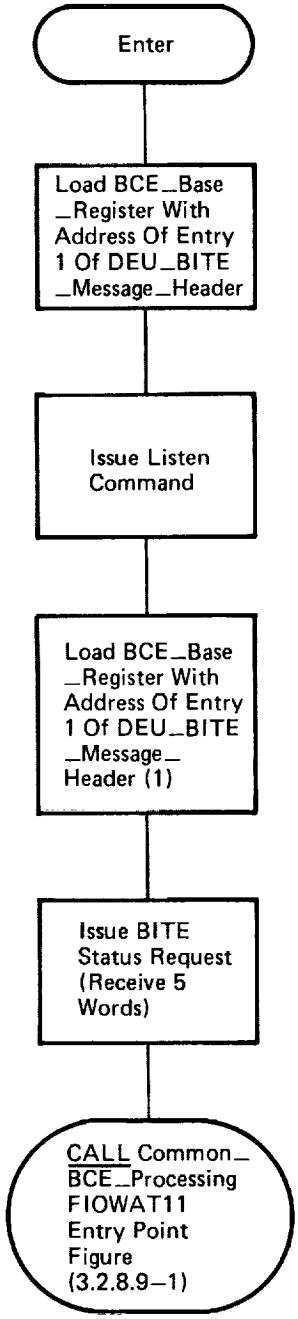


Figure 3.2.8.2–4. DEU_BCE_Processor DEU_Reset_Scratch_Pad_Line (261.4)



Similar Control Flows For DEU2 And DEU3

Figure 3.2.8.2–5. DEU_BCE_Processor DEU_Keyboard_Request (261.5)



Similar Control Flows For DEU2 And DEU 3

Figure 3.2.8.2–6. DEU_BCE_Processing DEU_Bite_Status_Request (261.6)

3.2.8.3 ICC_BCE_Processor (FIOICCPG) (262)

FIOICCPG performs the Inter-Computer Communications for Systems Software Interface Processing and IPR. The SSIP transfers 8, 16, 32, or 64 words per transmission. The IPR common set and redundant set transmissions are 7 words each.

a. Control Interface - FIOICCPG is initiated by a SIO instruction in (250) MSC_Control_Routine.

b. Inputs - See Table 3.2.8.3-1.

c. Process Description - FIOICCPG consists of 6 segments that perform the following ICC operations.

SSIP 8 word transmission and receive	262.01
SSIP 16 word transmission and receive	262.02
SSIP 32 word transmission and receive	262.03
SSIP 64 word transmission and receive	262.04
SSIP 128 word transmission and receive	262.05
IPR common set and redundant set 7 word transmission and receive	262.06

Control flow is presented in Figure 3.2.8.3-1 and 2.

d. Outputs - See Table 3.2.8.3-1.

e. Module References -

(268) Common_BCE_Processing (#PFIOECT) is invoked

f. Module Attributes - BCE Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation -

1. Listen mode entry point

2. Common set entry point

3. Redundant set entry point

4. May be indexed to another WIX instruction or listen mode entry point so that the data can be ignored or received.

5. Similar control flows exist for SSIP_16_Words (262.02), SSIP_32_Words (262.03), SSIP_64_Words (262.04), and SSIP_128_Words (262.05).

DATA TABLE 3.2.8.3-1

NAME ICC_BCE_Processor (FIOICCPG)

NO.	ITEM	ID	ACT	SOURCE	DESTINATION	ASSEMBLER NAME	MML	D	C
1	BCE_Base_Register	&008.09	W	See App. E	See App. E				
2	BCE_Base_Register_Table	Y005	I	205,280	See App. E	FIOBRE			
3	ICC/IPR_Base_Register_Table	Y014	I	205	262	FIOIBRE			

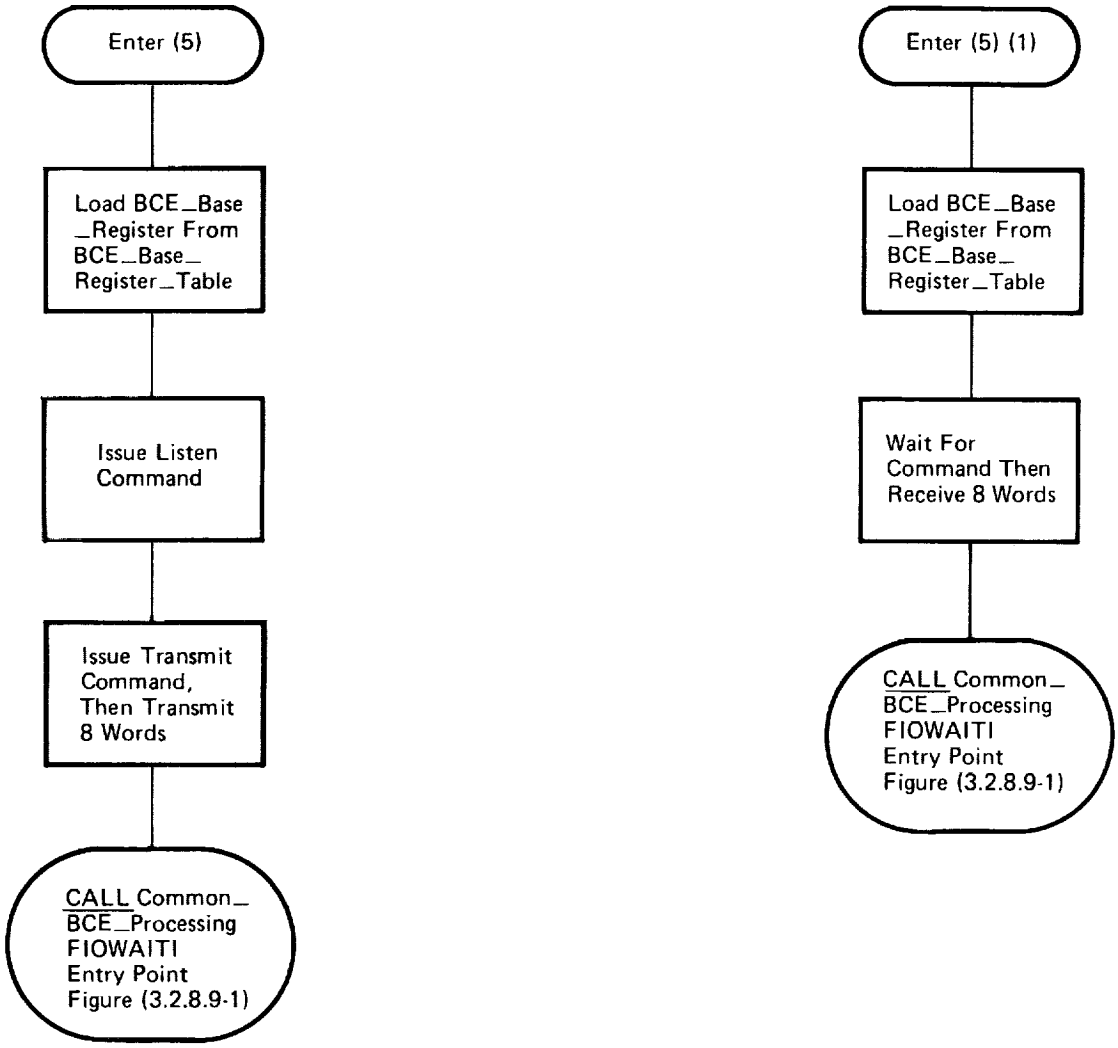


Figure 3.2.8.3-1. ICC_BCE_Processor SSIP_8_Words (262.01)

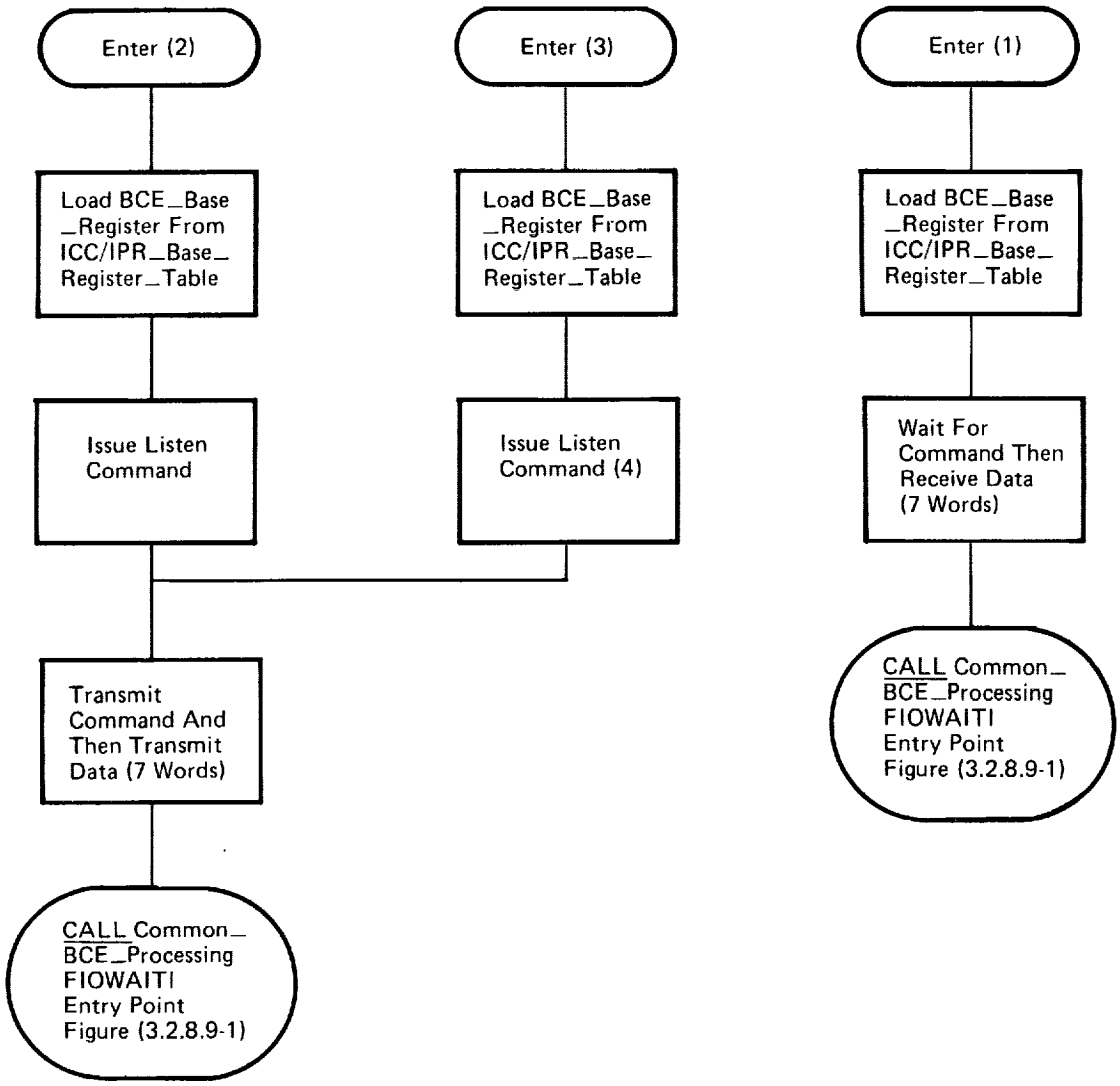


Figure 3.2.8.3-2. ICC_BCE_Processor IPR_7_Words (262.06)

3.2.8.4 LDB_BCE_Processor (FIOLDBPG) (263)

FIOLDBPG performs the LDB operations of status command, interrogate without GPC data, status request, interrogate with GPC data, go ahead command, and transmission enable.

a. Control Interface - FIOLDBPG is initiated by a SIO instruction in (250) MSC_Control_Routine (FIOMCNTL)

b. Input - See Table 3.2.8.4-1.

c. Process Description - FIOLDBPG consists of 5 segments that perform the following LDB operations:

Status Command	(263.01)
Interrogate without GPC Data	(263.02)
Status Request	(263.03)
Interrogate with GPC Data and Go Ahead	(263.04)
Transmission Enable	(263.05)

The control flow for this module is shown in Figure 3.2.8.4-1.

d. Outputs - See Table 3.2.8.4-1.

e. Module References -

(268) Common_BCE_Processing is invoked.

f. Module Attributes - BCE program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation -

1. Listen mode entry point.
2. Command will be either an interrogate with GPC data or a go ahead.
3. Enable command means here comes GPC data.

DATA TABLE 3.2.8.4-1

NAME LDB_BCE_Processor (FIOLDBPG)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	BCE_Base_Register	&008.09	W	See App. E	See App. E				
2	BCE_Base_Register_Table	Y005	I	205,280	See App. E	FIOBRE			
3	BCE_Command_Word_Table	Y007	I	205,280	264,282 283	FIOCWE			
4	Interrogate_Response_Word	V180	O	263,440	440	TFCMLI11,TFCMLSR1			
5	BCE_Word_Count_Table	Y006	I	205	261,263, 264,282	FIOWCE			

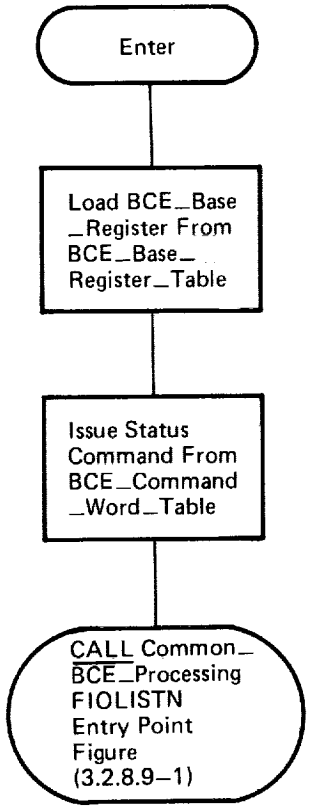


Figure 3.2.8.4–1. LDB_BCE_Processor LDB_Status_Command (264.1)

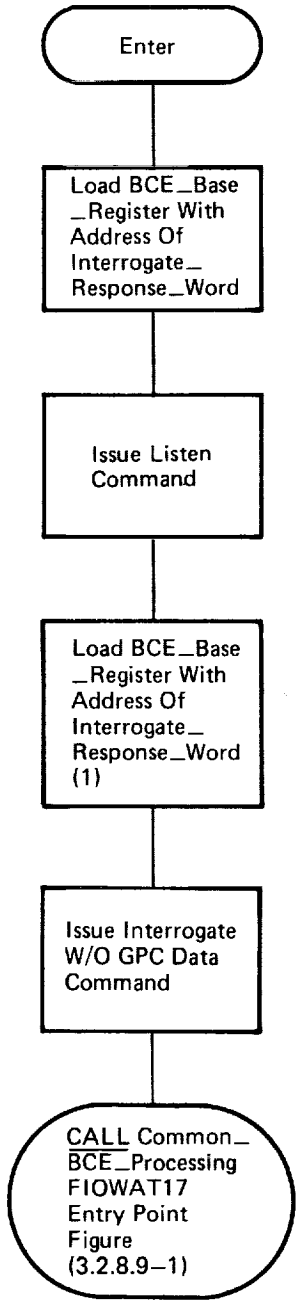


Figure 3.2.8.4-2. LDB_BCE_Processor LDB_Interrogate_W/O_GPC_Data (263.2)

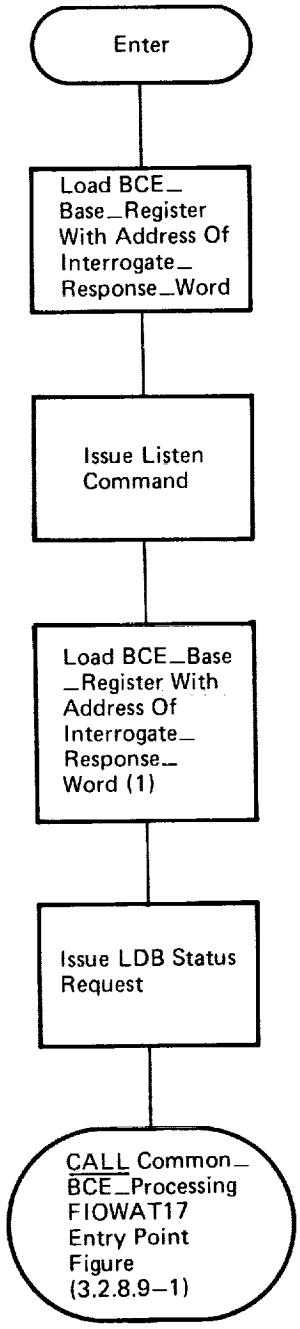


Figure 3.2.8.4-3. LDB_BCE_Processor LDB_Status_Request (263.3)

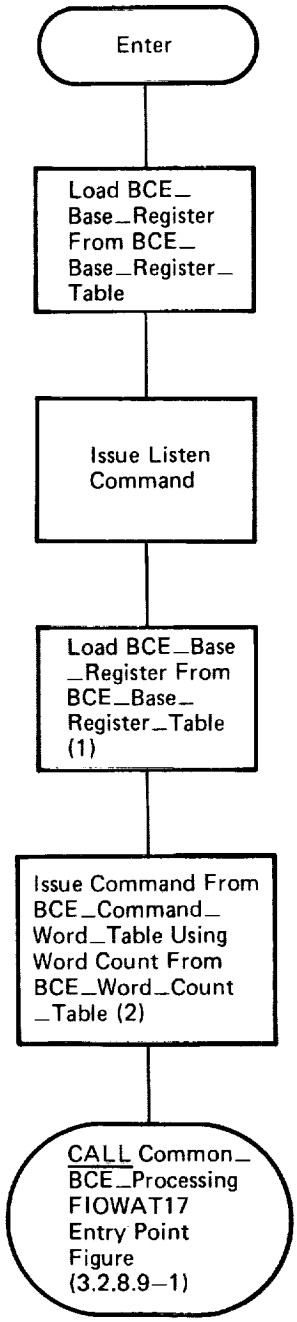


Figure 3.2.8.4–4. LDB_BCE_Processor LDB_Interrogate_W/O_GPC_Data_And_Go_Ahead (263.4)

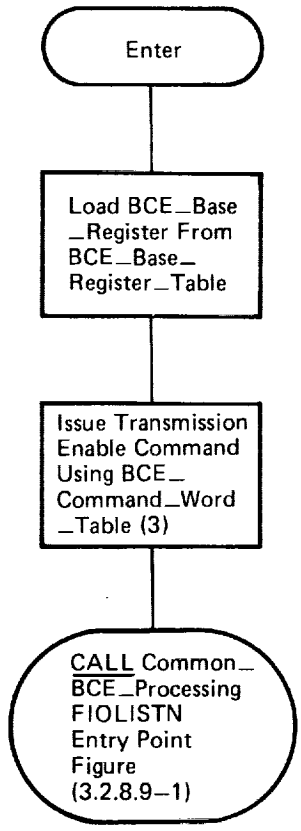


Figure 3.2.8.4–5. LDB_BCE_Processor LDB_Transmission_Enable (263.5)

3.2.8.5 MDM_BCE_Processor (FIOMDMPG) (264)

FIOMDMPG performs the MDM operations of Test Control Supervisor read, Test Control Supervisor write, Test Control Supervisor control, Auxiliary Power Unit fuel quantity write, forward nose wheel steering, master timing unit write, and BITE acquisition.

a. Control Interface - FIOMDMPG is initiated by a SIO instruction in (250) MSC_Control_Routine.

b. Input - See Table 3.2.8.5-1.

c. Process Description - FIOMDMPG consists of 6 segments that perform the following MDM operations:

TCS read	(264.01)	MTU Write	(264.05)
TCS write and control	(264.02)	BITE acquisition	(264.06)
APU write	(264.03)		
Forward nose wheel stering	(264.04)		

The control flow for this module is shown in Figure 3.2.8.5-1.

d. Outputs - See Table 3.2.8.5-1.

e. Module References -

(268) COMMON_BCE_Processing (#PFIOECT) is invoked.

f. Module Attributes - BCE Program.

g. Template References - N/A

h. Error Handling - None

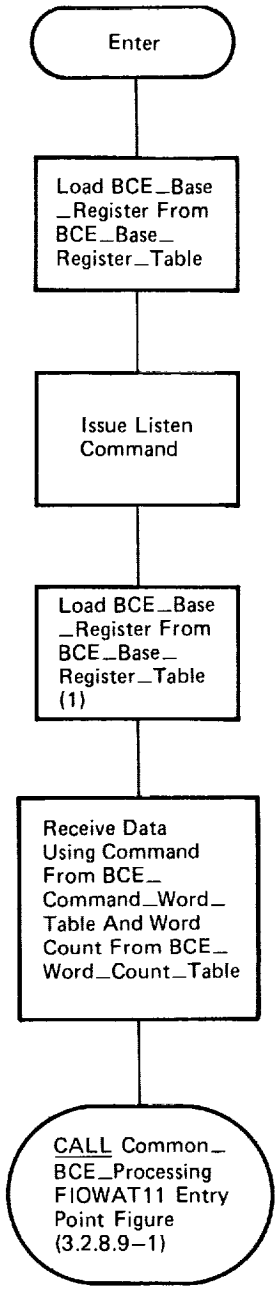
j. Detailed Implementation -

1. Listen command entry point

DATA TABLE 3.2.8.5-1

NAME MDM_BCE_Processor (FIOMDMPG)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	BCE_Base_Register	&008.09	W	See App. E	See App. E				
2	BCE_Base_Register_Table	Y005	I	205,280	See App. E	FIOBRE			
3	BCE_Command_Word_Table	Y007	I	205,280	282,283 264	FIOCWE			
4	BCE_Word_Count_Table	Y006	I	205	263,261, 264,282	FIOWCE			
5	APU_Output_Buffer (IN SM COMPOOL)	Y083	I		264	TFCMAPU1			
6	Nosewheel_Steering (IN GN&C COMPOOL)	Y080	I		264	TFCMFNS1			
7	FF/FA_BITE_Acquisition_Base_Register_Table	Y015	I	205	264	FIOBBRE			
8	Payload_BTU_BITE	I100.10	O	See App. E		CZ2BTPF1 & CZ2BTPF2		X	X



Similar Control Flow For TCS On IUA 12

Figure 3.2.8.5-1. MDM_BCE_Processor MDM_TCS_Read (264.01)

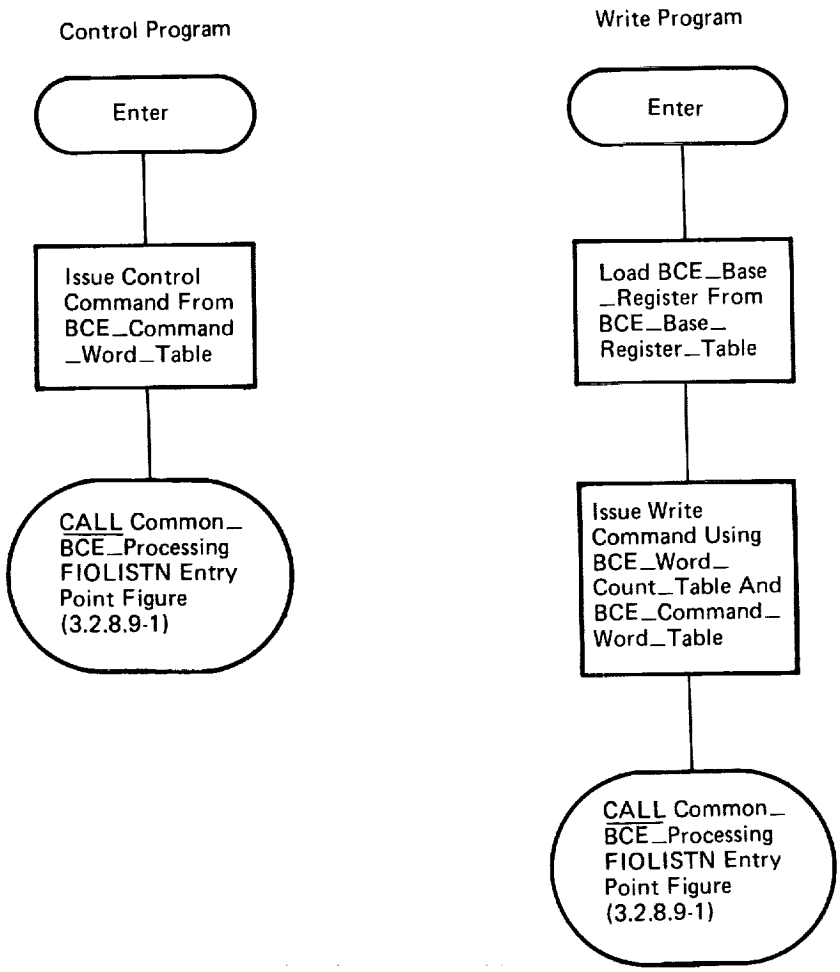


Figure 3.2.8.5-2. MDM_BCE_Processor MDM_TCS_Write_And_Control (264.02)

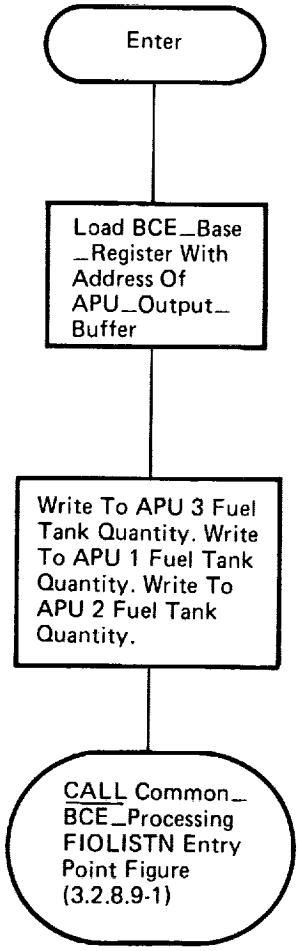


Figure 3.2.8.5-3. MDM_BCE_Processor MDM_APU_Write (264.03)

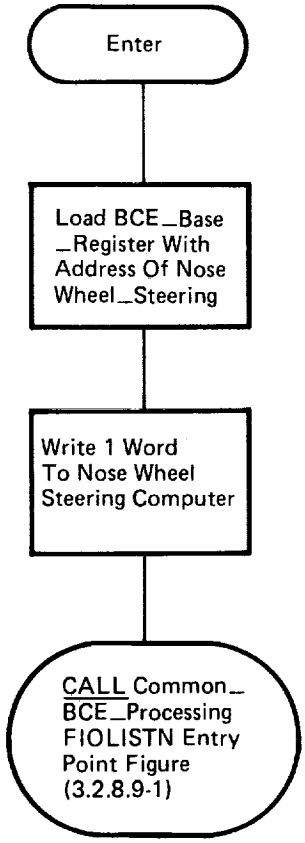


Figure 3.2.8.5-4. MDM_BCE_Processor MDM_Forward_Nose_Wheel_Steering (264.04)

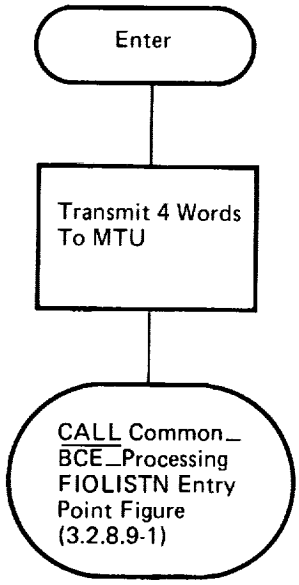


Figure 3.2.8.5-5. MDM_BCE_Processor MDM_MTU_Write (264.05)

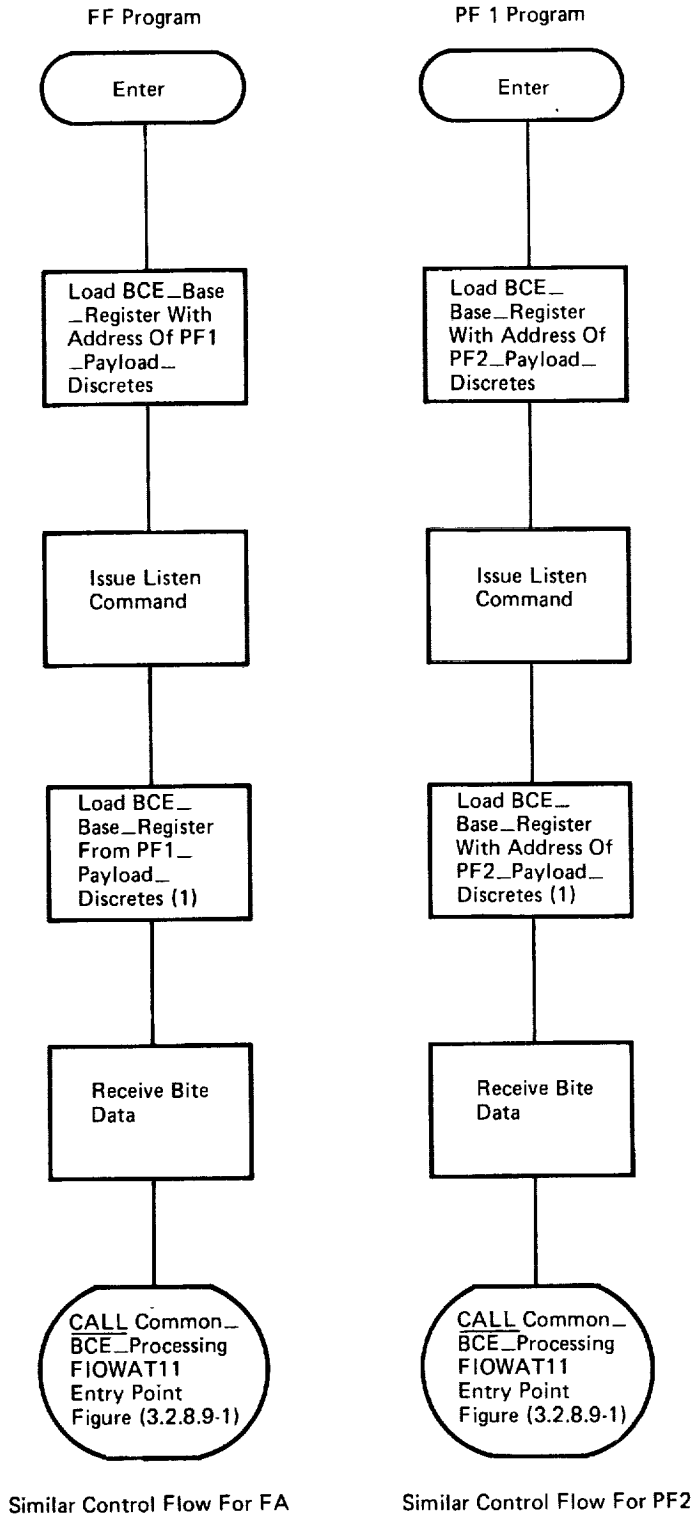


Figure 3.2.8.5-6. MDM_BCE_Processor MDM_Bite_Acquisition (264.06)

3.2.8.6 Payload_Discrete_BCE_Processor (FIOPDSPG)(265)

FIOPDSPG Performs the set and resets of the payload discretes.

a. Control Interface - FIOPDSPG is initiated by a SIO instruction in (250) MSC_Control_Routine.

b. Input - See Table 3.2.8.6 - 1.

c. Process Description - FIOPDSPG will issue a set and reset of the PF1 discretes on card 10 and then on card 2. The same sequence is performed for PF2. The control flow for this module is shown in Figure 3.2.8.6-1.

d. Outputs - See Table 3.2.8.6 - 1.

e. Module References -
(268) Common_BCE_Processing (#PFIOECT) is involved.

f. Module Attributes - BCE Program.

g. Template References - N/A.

h. Error Handling - None.

i. Constraints and Assumption - None.

j. Detailed Implementation - None

DATA TABLE 3.2.8.6-1

NAME Payload_Discrete_BCE_Processor

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	BCE_Base_Register	&008.09	W	See App. E	See App. E				
2	PF1_Payload_Discretes	Y081	O		265	TFCMPFD1			
3	PF2_Payload_Discretes	Y082	O		265	TFCMPFD2			

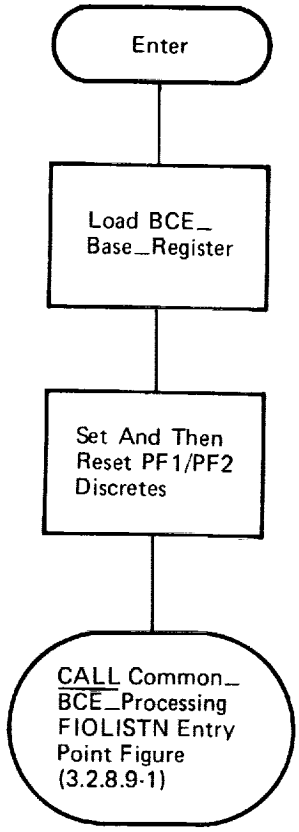


Figure 3.2.8.6-1. Payload_Discrete_BCE_Processor (FIOPDSPG)

3.2.8.7 PMU_BCE_Processor (FIOPMUPG) (266)

FIOPMUPG performs the PMU operations of write to the RAM, 128 KBPS and 64 KBPS read and write, format selection, BITE read, Operational Instrumentation/Payload read, and single data transaction for the Operational Instrumentation/Payload.

a. Control Interface - FIOPMUPG is initiated by a SIO instruction in (250) MSC_Control_Routine.

b. Inputs - See Table 3.2.8.7-1.

c. Process Description - FIOPMUPG consists of 7 segments that perform the following PMU operations:

Write RAM	(266.01)
Write 128/64 KBPS	(266.02)
Read 128/64 KBPS	(266.03)
Read BITE	(266.04)
Format Select	(266.05)
OI/PL Read	(266.06)
OI/PL Single Data Transaction	(266.07)

The control flow for this module is shown in Figure 3.2.8.7-1.

d. Outputs - See Table 3.2.8.7-1.

e. Module References

(268) Common_BCE_Processing (#PFIOECT) is invoked.

f. Module Attributes - BCE Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation

1. Transmission rate (128 or 64 KBPS) determined by command word.

DATA TABLE 3.2.8.7-1

NAME PMU_BCE_Processor

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	BCE_Base_Register	&008.09	W	See App. E	See App. E				
2	PMU_Word_Count_Table	Y009	O	205		FIOPWCE			
3	PMU_Command_Word_Table	Y010	O	205		FIOPCWE			
4	PMU_Base_Register_Table	Y008	O	205		FIOPBRE			
5	BCE_Base_Register_Table	Y005	O	205,280	See App. E	FIOBRE			
6	BCE_Command_Word_Table	Y007	O	205,280	264,282 283	FIOCWE			
7	BCE_Word_Count_Table	Y006	O	205	261,263 264,282	FIOWCE			

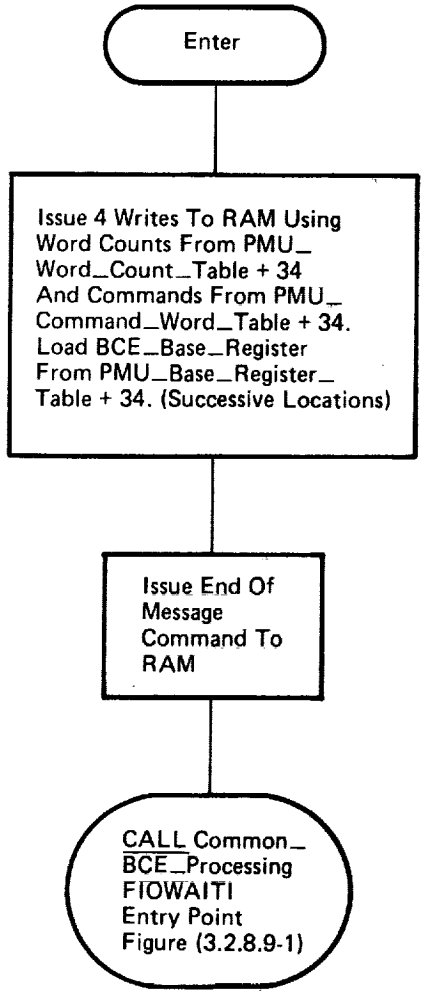
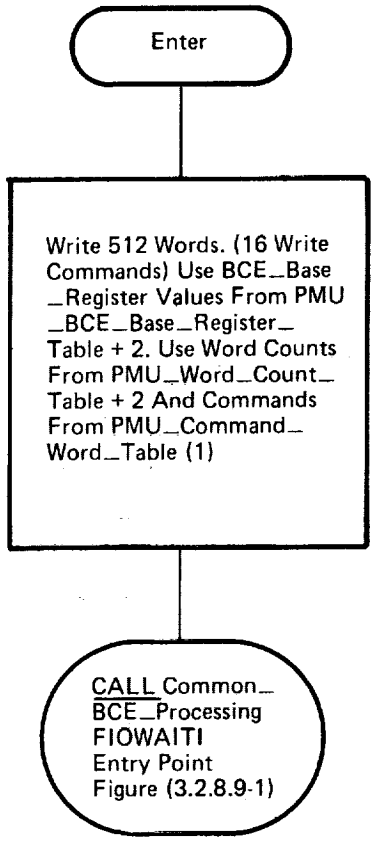


Figure 3.2.8.7-1. PMU_BCE_Processor Write_RAM (266.01)



Similar Control Flow For Read _ 128/64_KBPS (266.03)

Figure 3.2.8.7-2. PMU_BCE_Processor Write_128/64_KBPS (266.02)

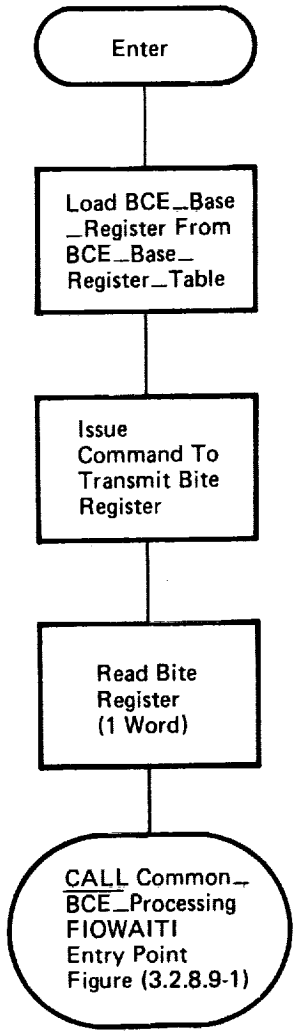


Figure 3.2.8.7-3. PMU_BCE_Processor Bite_Read (266.04)

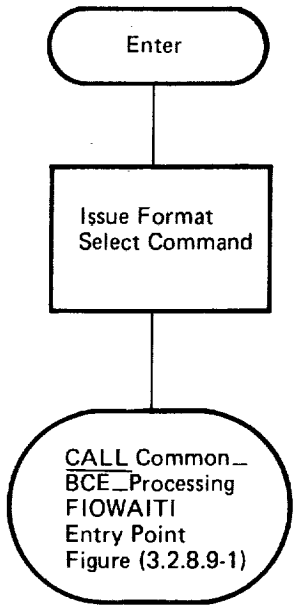


Figure 3.2.8.7-4. PMU_BCE_Processor Format_Select (266.05)

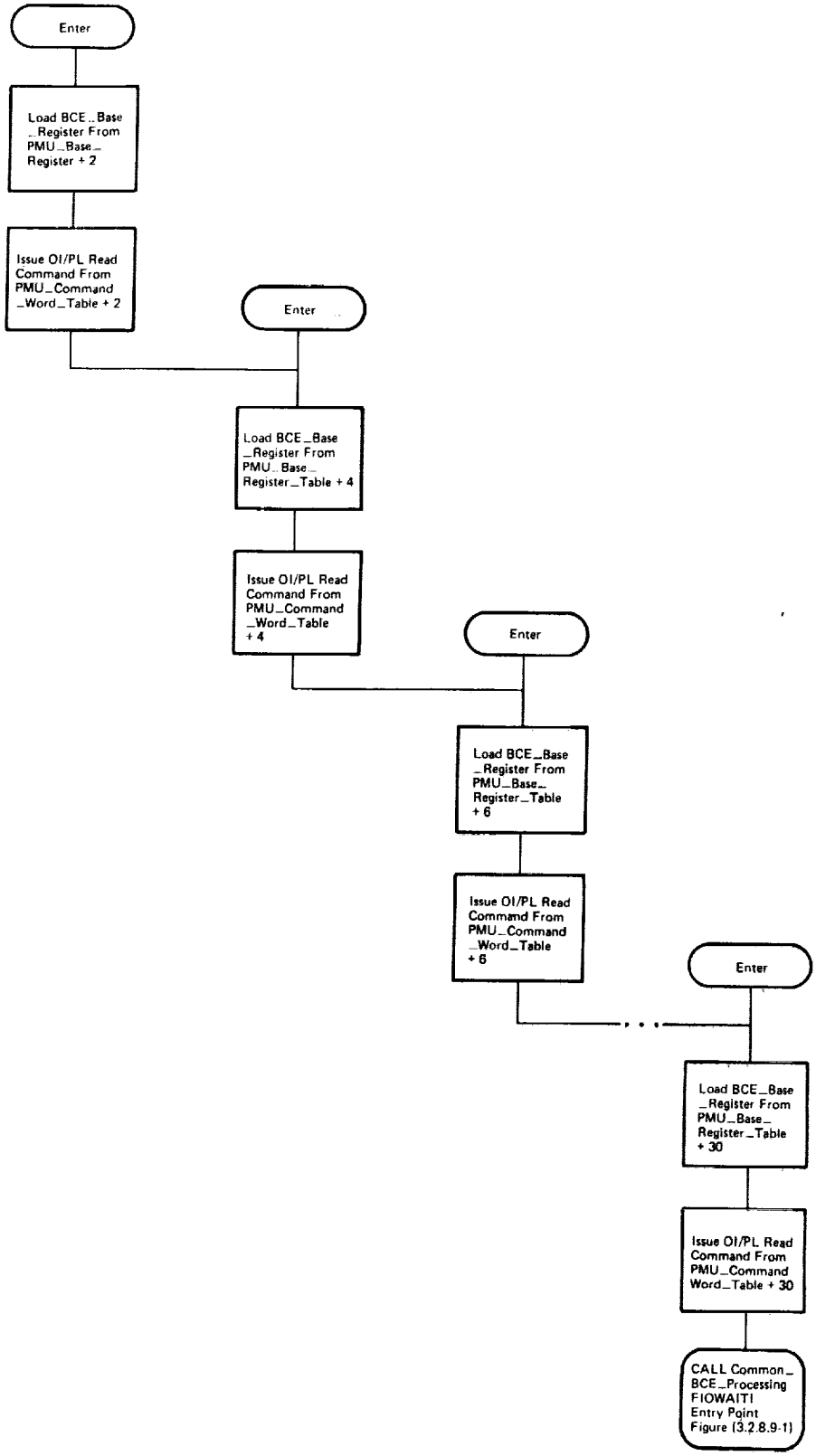


Figure 3.2.8.7-5. PMU_BCE_Processor OI/PL_Read (266.06)

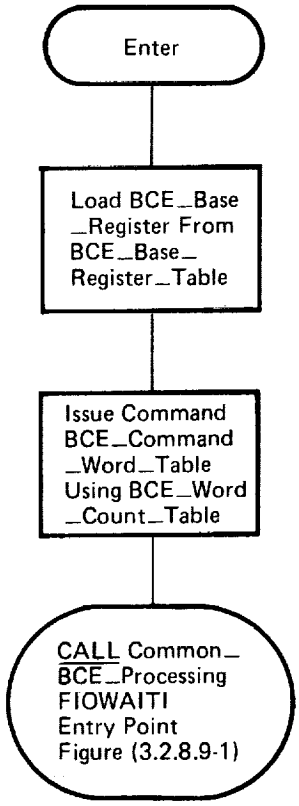


Figure 3.2.8.7-6. PMU_BCE_Processor OI/PL_1_Data_Transaction (266.07)

3.2.8.8 PROM_BCE_Processor (FIOPRMPG) (267)

FIOPRMPG performs the programmable read-only memory sequence I/O and the chained BCE I/O for the FF 1-4 and FA 1-4 MDMs.

a. Control Interface - FIOPRMPG is initiated by a SIO instruction in (250) MSC_Control_Routine.

b. Input - See Table 3.2.8.8-1.

c. Process Description - FIOPRMPG consists of 13 segments that perform the following MDM I/O operations:

FA01-FA03 Input	267.01	FF01 Input 2	267.07
FA04 Input	267.02	FF02 Input 2	267.08
FA Input Attach Point Volts	267.03	FF03 Input 2	267.09
FA01-03 Output	267.04	FF04 Input 2	267.10
FA04 Output	267.05	FF Read MTU	267.11
FF Input 1	267.04	FF01-03 Output	267.12
		FF04 Output	267.13

The control flow for this module is shown in Figure 3.2.8.8-1.

d. Outputs - See Table 3.2.8.8-1.

e. Module References -

(268) Common_BCE_Processing (#PFIOECT) is invoked.

f. Module Attributes - BCE program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation -

1. Listen command entry point.

DATA TABLE 3.2.8.8-1

NAME PROM_BCE_Processor (FIOPRMPG)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	BCE_Base_Register	&J08.04	W	See App. E	See App. E				
2	FA_Input_Buffer	Y070	I	267		TFCMFAI1			
						TFCMFAI2			
						TFCMFAI3			
						TFCMFAI4			
3	FA_Input_Attack_Points_Volts_Buffer	Y071	I	267		TFCMFAI5			
4	FA_Output_Buffer	Y072	O		267	TFCMFAO1			
						TFCMFAO2			
						TFCMFAO3			
						TFCMFAO4			
5	FF_Input_1_Buffer	Y073	I	267		TFCMFI11			
						TFCMFI12			
						TFCMFI13			
						TFCMFI14			
6	FF_Input_2_Discretes	Y074	I	267		TFCMDIS1			
						TFCMDIS2			
						TFCMDIS3			
						TFCMDIS4			
7	FF_Input_2_TACAN	Y075	I	267		TFCMFI21			

DATA TABLE 3.2.8.8-1 (Cont'd)

NAME PROM_BCE_Processor (FIOPRMPG)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
						TFCMFI22			
						TFCMFI23			
						TFCMFI24			
8	FF_Input_2_ADTA	Y076	I	267		TFCMADT1			
						TFCMADT2			
						TFCMADT3			
						TFCMADT4			
9	FF_Input_2_MSBLS	Y077	I	267		TFCMSBL1			
						TFCMSBL2			
						TFCMSBL3			
10	FF_Input_2_IMU	Y078	I	267		TFCMIMU1			
						TFCMIMU2			
						TFCMIMU3			
11	MTU_1_Buffer	#017.2	I	267	955	TFCMMTU1	V91W8850C	X	X
							V91W8852C		
							V91W8854C		
12	MTU_2_Buffer	#018.2	I	267	955	TFCMMTU2	V91W8856C	X	X
							V91W8858C		
							V91W8860C		
13	MTU_3_Buffer	#019.2	I	267	955	TFCMMTU3	V91W8862C	X	X

DATA TABLE 3.2.8.8-1 (Cont'd)

NAME PROM_BCE_Processor (FIOPRMPG)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
							V91W8864C		
							V91W8866C		
14	FF_Output_Buffer	Y079	0		267	TFCMFF01			
						TFCMFF02			
						TFCMFF03			
						TFCMFF04			

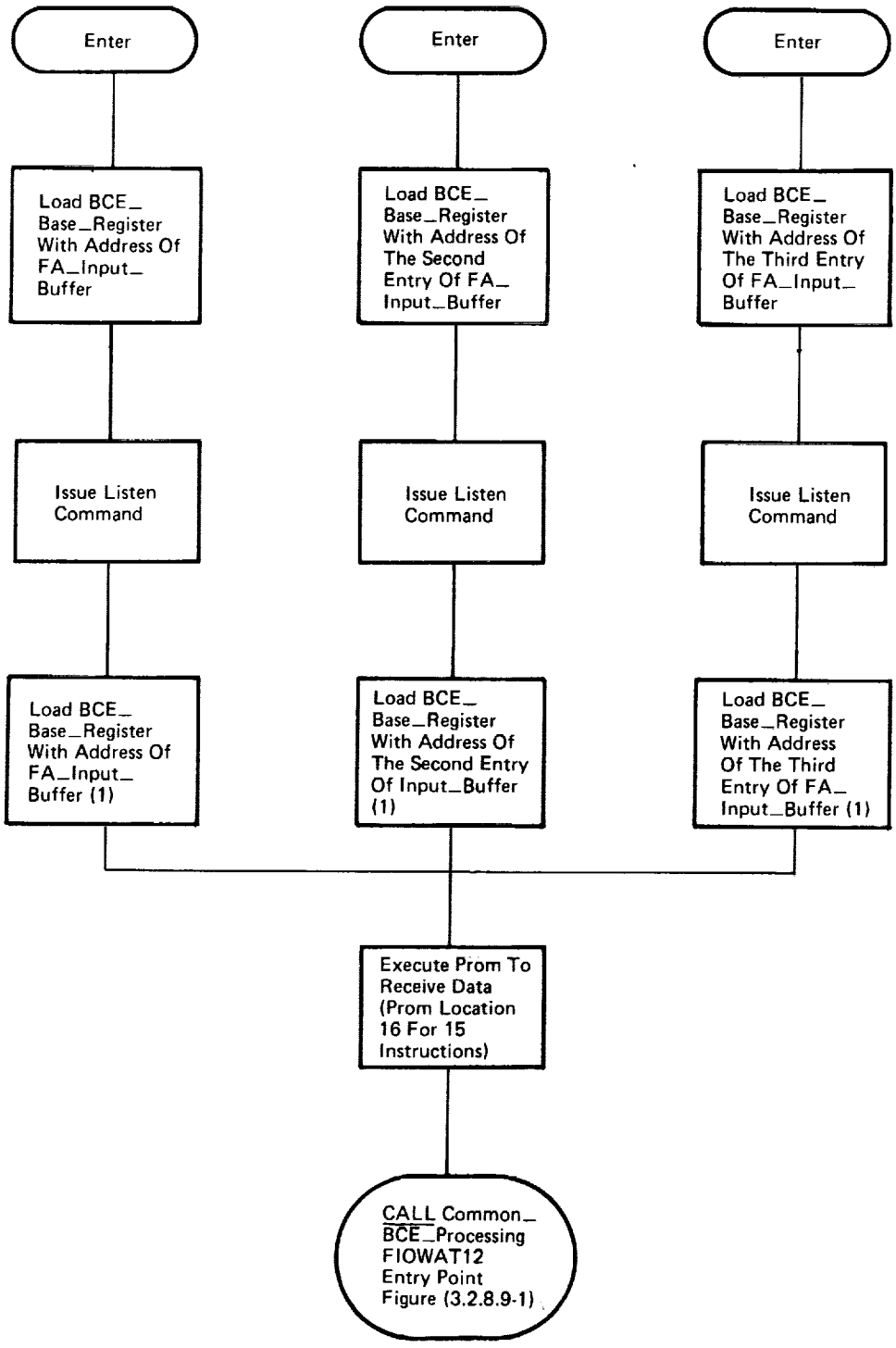


Figure 3.2.8.8-1. Prom_BCE_Processor FA01-03_Input_Prom_Sequence (267.01)

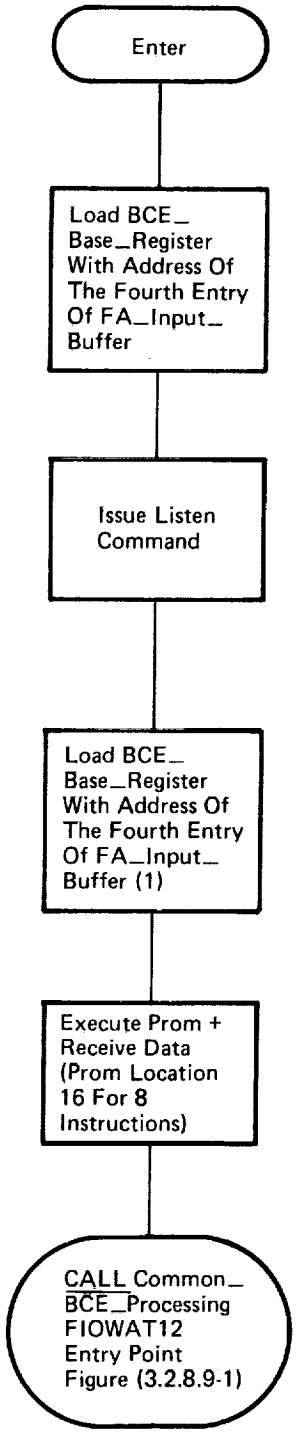


Figure 3.2.8.8-2. Prom_BCE_Processor FA04_Input_Prom_Sequence (267.02)

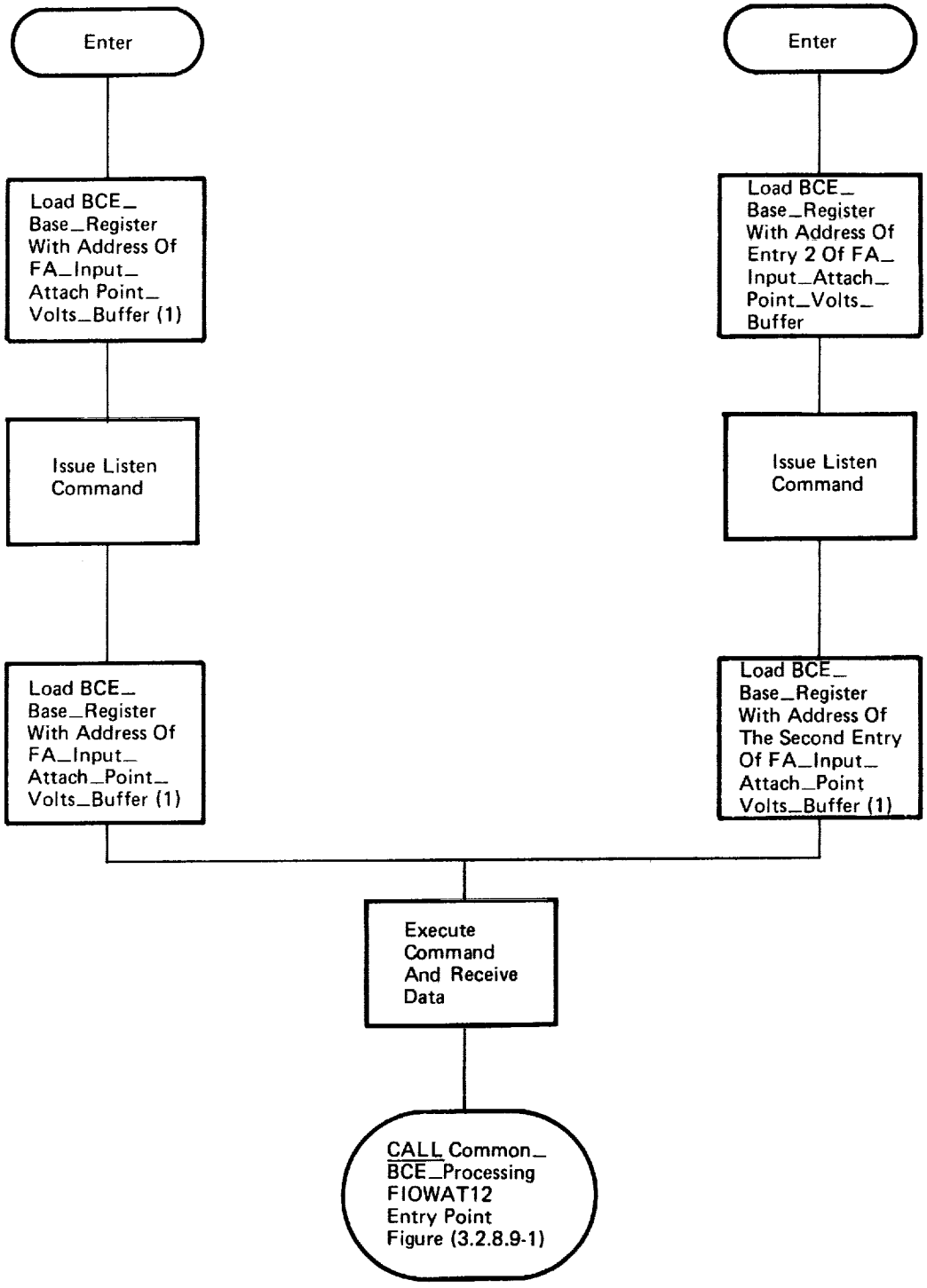


Figure 3.2.8.8-3. Prom_BCE_Processor FA_Input_Attach_Point_Volts (267.03)

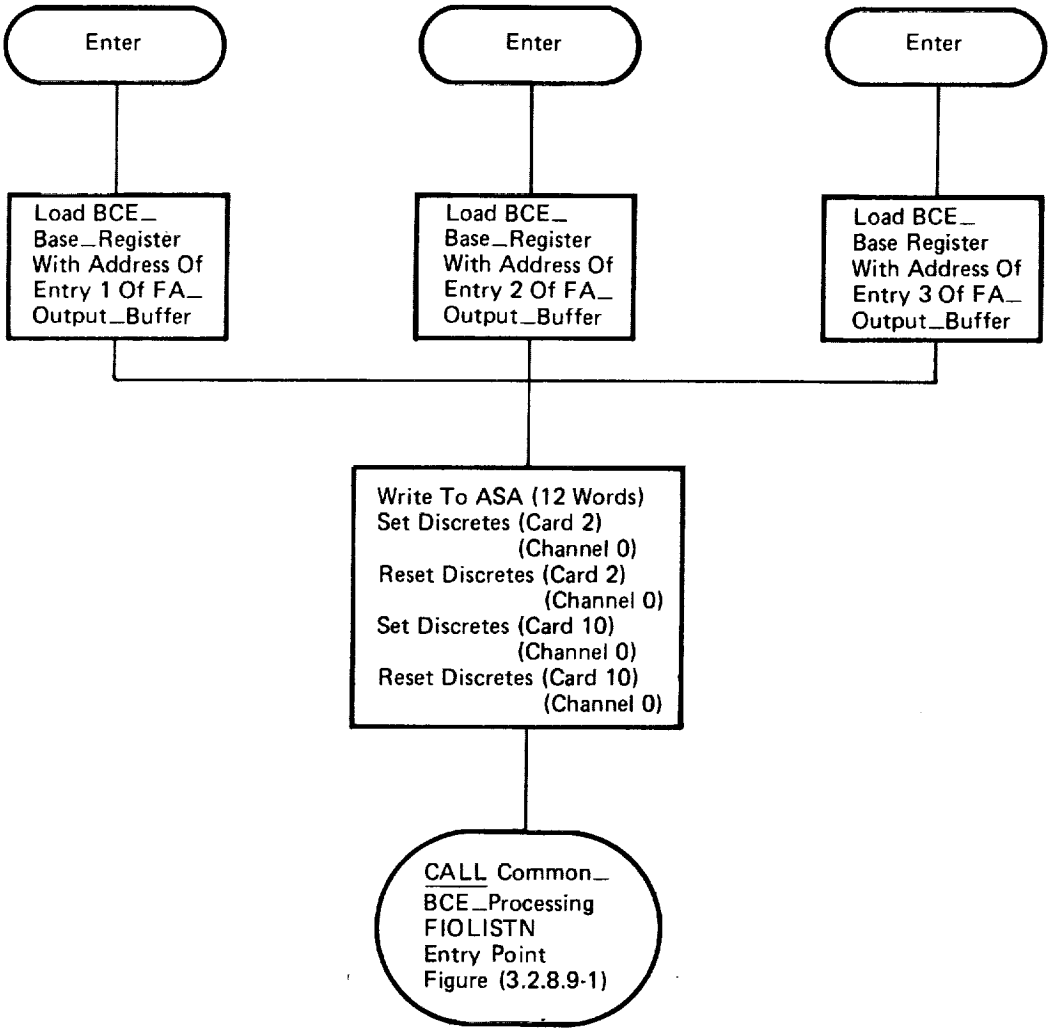


Figure 3.2.8.8-4. Prom_BCE_Processor FA01-03_Output (267.04)

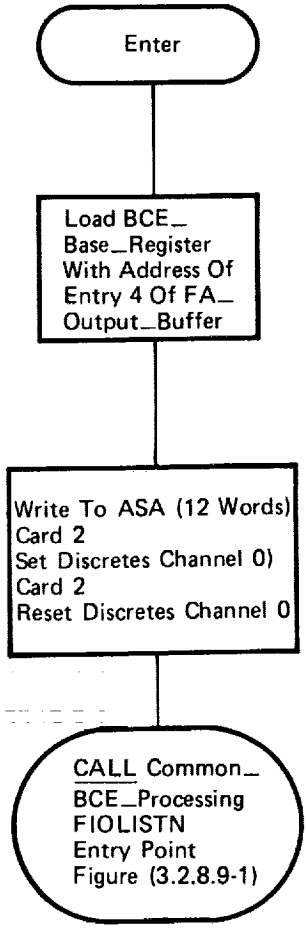


Figure 3.2.8.8-5. Prom_BCE_Processor FA04_Output (267.05)

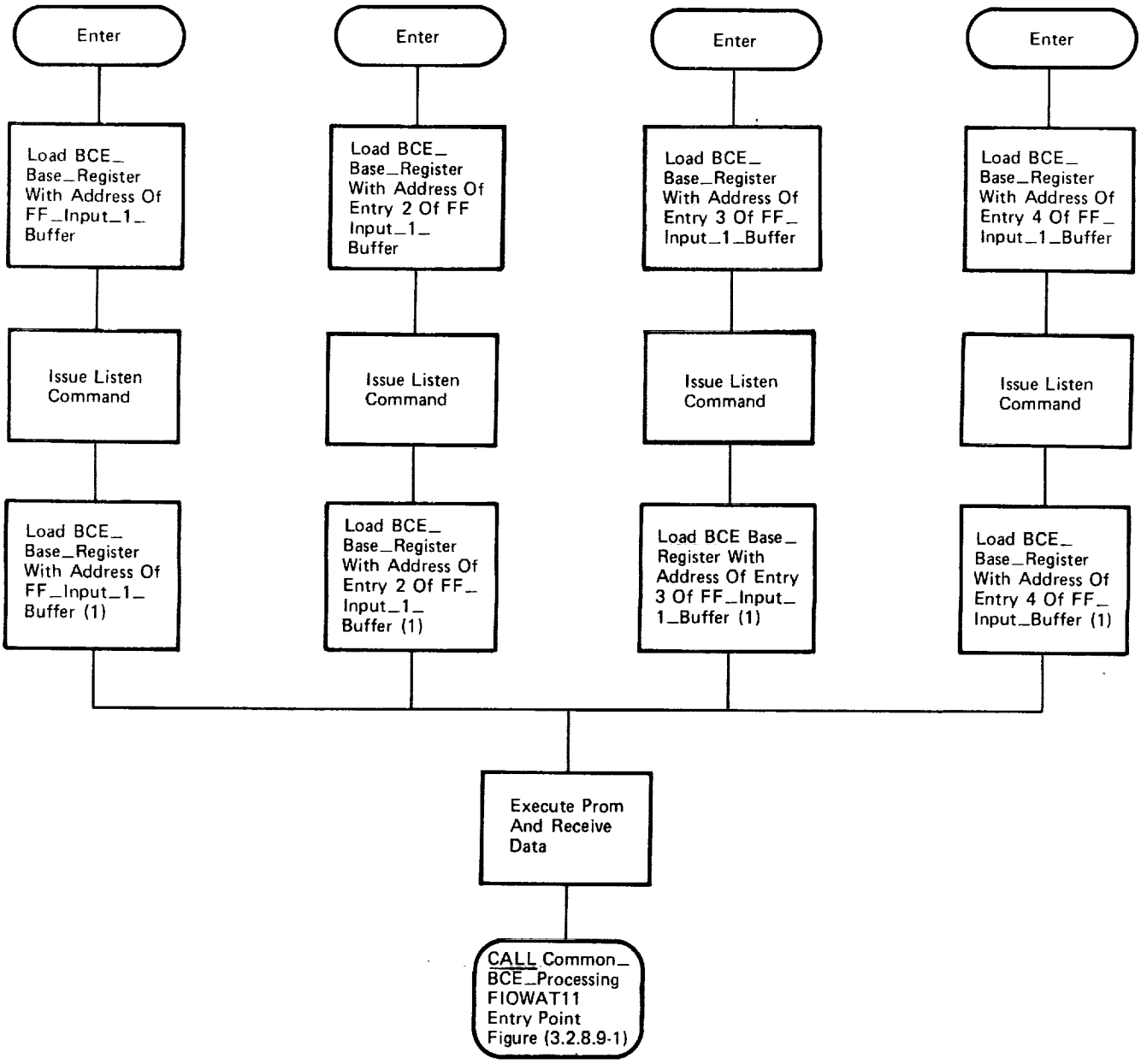
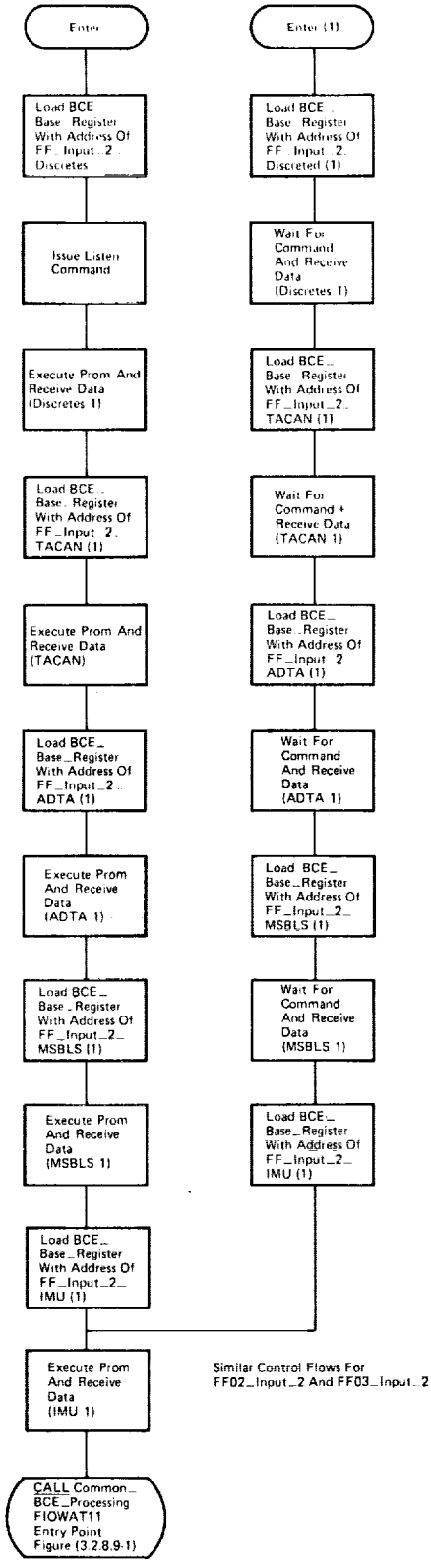


Figure 3.2.8.8-6. Prom_BCE_Processor FF_Input_1 (267.06)



ure 3.2.8.8–7. Prom_BCE_Processor FF01_Input_2 (267.07)

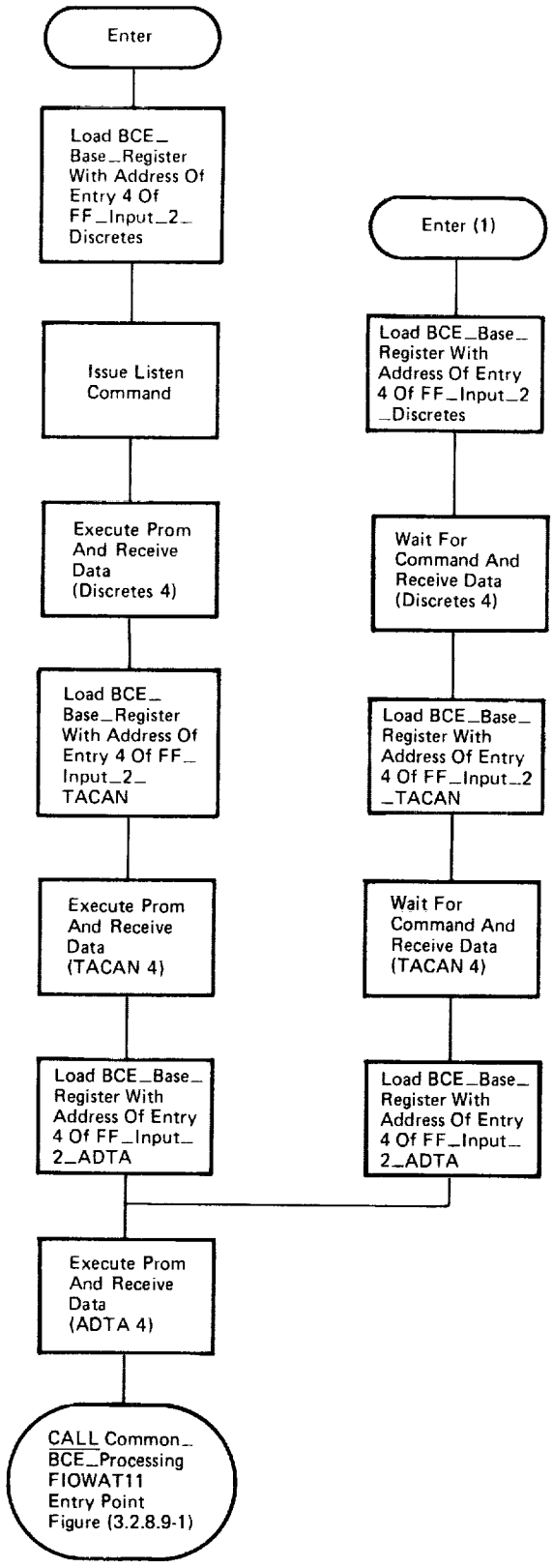


Figure 3.2.8.8-8. Prom_BCE_Processor FF04_Input_2(267.10)

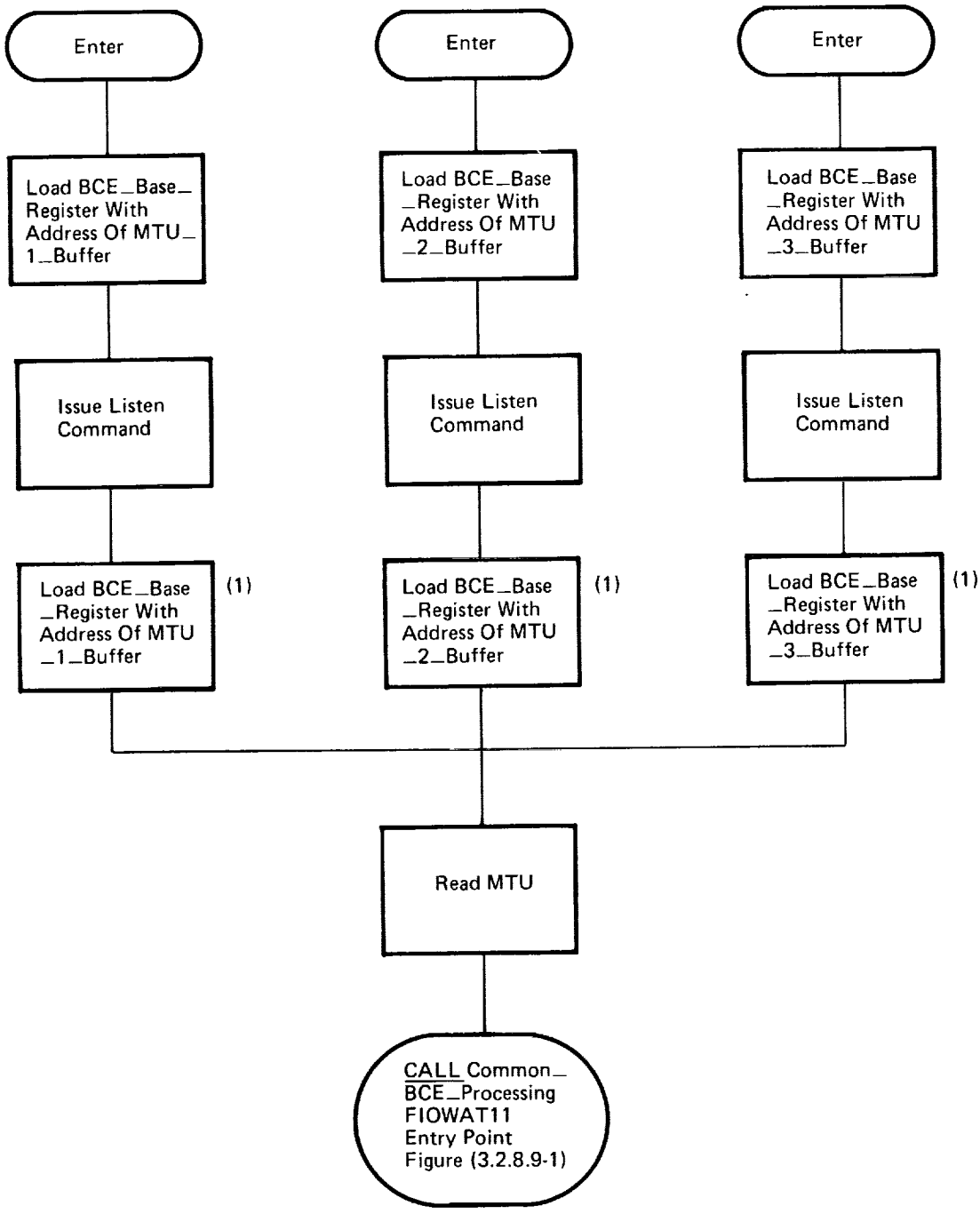


Figure 3.2.8.8-9. Prom_BCE_Processor FF_Read_MTU (267.11)

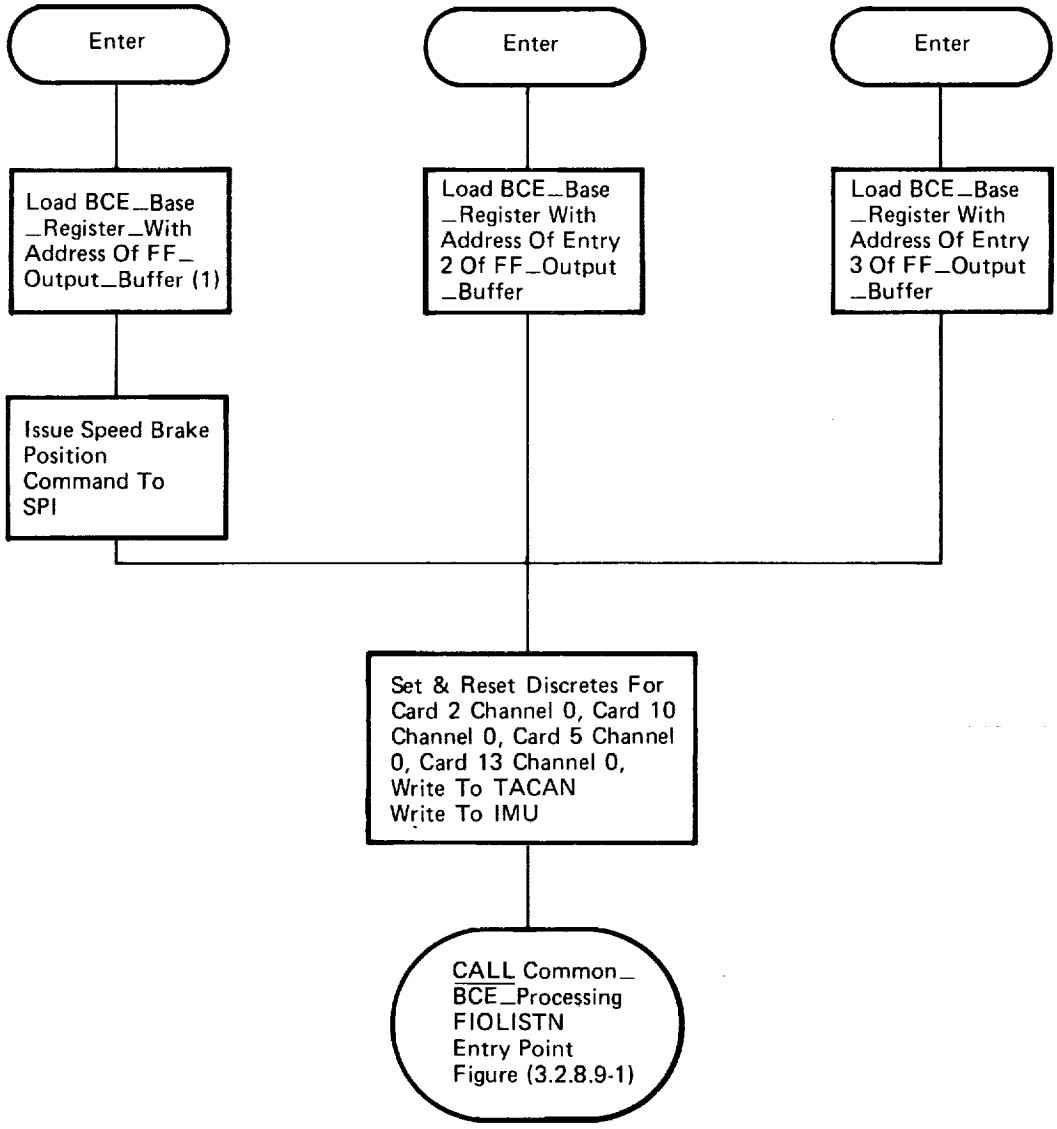


Figure 3.2.8.8-10. Prom_BCE_Processor FF01-03_Output (267.12)

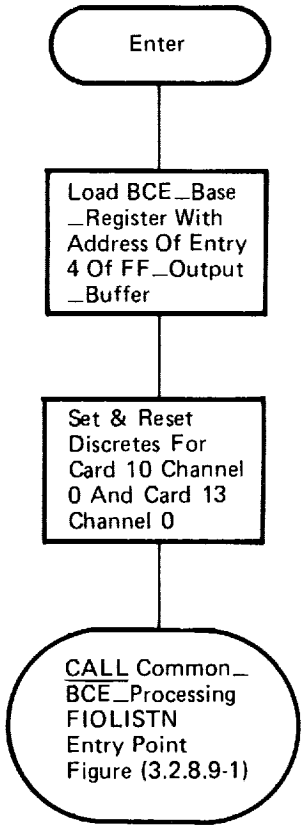


Figure 3.2.8.8-11. Prom_BCE_Processor FF04_Output (267.13)

3.2.8.9 Common_BCE_Processing (#PFIOECT) (268)

#PFIOECT contains BCE code that is entered from other BCE programs via a BCE branch instruction. It is necessary for the BCE code to reside within this compool due to the nature of the STORE STATUS (#SST) and WAIT FOR INDEX (#WIX) BCE instructions. Both instructions require that they be placed within 1024 halfwords of the memory location which they reference. The store status instruction is used to zero Data_Path_Error_Counters while the wait for index instruction references the Wix_Table.

a. Control Interface -

1. @SIO MSC instruction from (251) MSC_I/O_Monitor (FIOMNTR)
2. @SIO MSC instruction from (252) MSC_BCE_Reset_Routine (FIOMSETB)
3. #BU BCE instruction from all BCE programs

b. Input - For BCE's that execute a WAIT FOR INDEX instruction while in the listen mode, an index into the WIX_Table is furnished via a listen command sent by another IOP's BCE operating in the command mode.

See Table 3.2.8.9-1

c. Process Description - When entered from a non-mass memory input transaction BCE program, an IUA_11_N_Counters, IUA_12_N_Counters, or IUA_17_N_Counters field is zeroed and the BCE that is executing leaves the busy state and enters the wait state. An entry from a non-mass memory output transaction BCE program will cause the BCE that is executing to send a listen command and to leave the busy state and enter the wait state.

If no errors were encountered in the I/O transaction, the MSC_I/O_Monitor will issue a @SIO MSC instruction to place the BCE in the busy state executing a WAIT FOR INDEX BCE instruction. If errors were encountered in the I/O transaction, the MSC_BCE_Reset_Routine will issue a @SIO MSC instruction to place the BCE in the busy state executing a WAIT FOR INDEX BCE instruction. If the BCE is in the command mode, the BCE will immediately re-enter the wait state. If, however, the BCE is in the listen mode, the BCE will monitor its bus for a listen command to be sent by another IOP commanding on that bus.

When entered from a mass memory output transaction BCE program, a listen command is sent out, the IUA_12_N_Counters field for this bus is zeroed, and a WAIT FOR INDEX BCE instruction is executed to place commanding BCE's in the wait state and listeners to monitoring their buses for a listen command sent by another IOP commanding on that bus. Entry from a mass memory input transaction BCE program will result in the same processing with the exception of sending out a listen command.

d. Outputs -

Listen commands
See Table 3.2.8.9-1.

e. Module References - None

f. Module Attributes - BCE Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation -

1. For BCE's that are in command mode, the WAIT FOR INDEX instruction is equivalent to a WAIT instruction and the BCE re-enters the wait state until it receives another command from its own MSC telling it to execute another BCE program. The WAIT FOR INDEX instruction places listen mode BCE's back into a loop where they are monitoring the bus for a listen command, which furnishes an index used in picking up the WIX_Table address of the listen mode entry point within the BCE program to be executed.

DATA TABLE 3.2.8.9-1

NAME Common_BCE_Processing (#PFIOECT)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	IUA_12_N_Counters	#200.6	O	244,268	268	FIOB0012			
2	WIX_Table	#201	I	205,220 390	See App. E	FIOWIXTB			
3	IUA_17_N_Counters	#200.9	O	244,268		FIOB0017			
4	IUA_11_N_Counters	#200.5	O			FIOB0611			
5	Data_Path_Error_Counters	#200.4	I	355	268	FIOB0011			

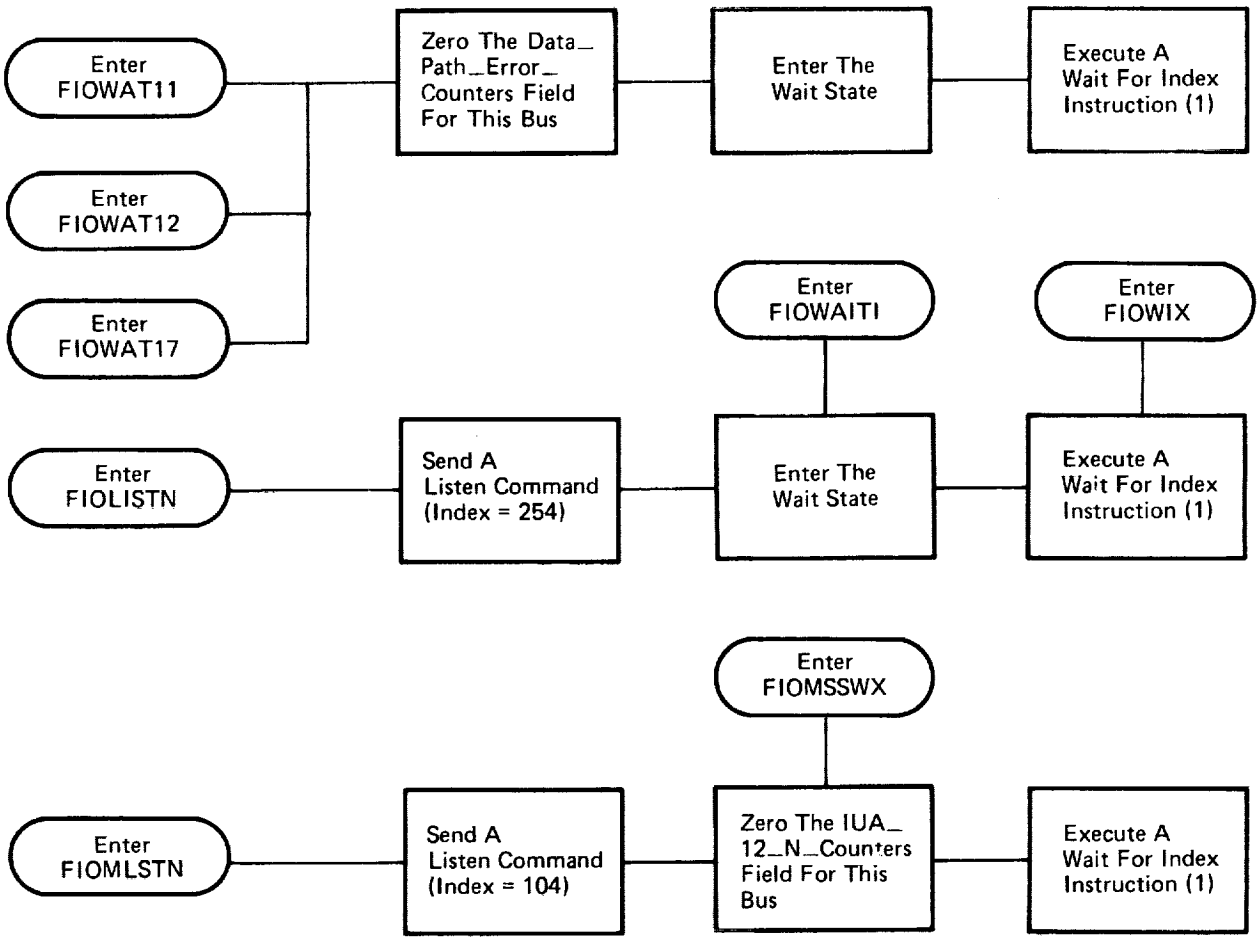


Figure 3.2.8.9-1. Common_BCE_Processing (# PFIOECT)

3.2.9 Mass Memory

Mass Memory Units (MMUs) require I/O Management services due to data verification and hardware characteristics, such as slow access time and fixed length data blocks. These special I/O Management services provide MM service request processing, IOP dispatching, completion processing, error processing, and special IOP programs.

MM service requests are initiated using the generalized I/O SVC services. The MM user requests services by issuing an I/O SVC with the associated parameters initialized. These parameters include the MM operation code used to indicate what function is to be performed, the number of words to be transferred, the requested MM starting block address, and other parameters common to all I/O processing. The I/O SVC service routines when invoked will allocate and initialize an IOQE for the request then relinquish request processing to the special MM I/O Management services. These services will process the MM service request and monitor the completion of the transaction. If the requested MMU is busy, the services will reject the request as in error.

It should be noted, that data is written to MM in 512 16-bit words regardless of the size of the output buffer or word count specified by the requestor. However, the IOP is not required to receive MM data in 512 word blocks. Therefore, reads of less than one block are supported.

MM data verification on read operations is performed by calculating a checksum word from theinput data words and comparing the checksum word against the last input data word. The checksum word is calculated by summing the input data words. The last word of any logical data record is assumed to be a checksum word calculated from the input data words before the data block was originally written to MM.

Data verification on write operations is performed by re-reading the data written to MM and performing a read operation data verification. The data is read into a buffer other than the users output buffer to preserve data integrity.

Mass Memory I/O Management Services consist of the following programs (See Figure 3.2.9-1).

a. Mass_Memory_Manager - Fields and processes all MM service requests and monitors completion of all MM transactions.

b. Mass_Memory_MSC_Processor - Initiates MM BCE programs

c. Mass_Memory_BCE_Processor - MM BCE program used to read, write or position a MMU tape or to read a MM status register.

d. Mass_Memory_Utility_Write_BCE_Processor - MM BCE program used to write 1 to 32 data blocks to MM.

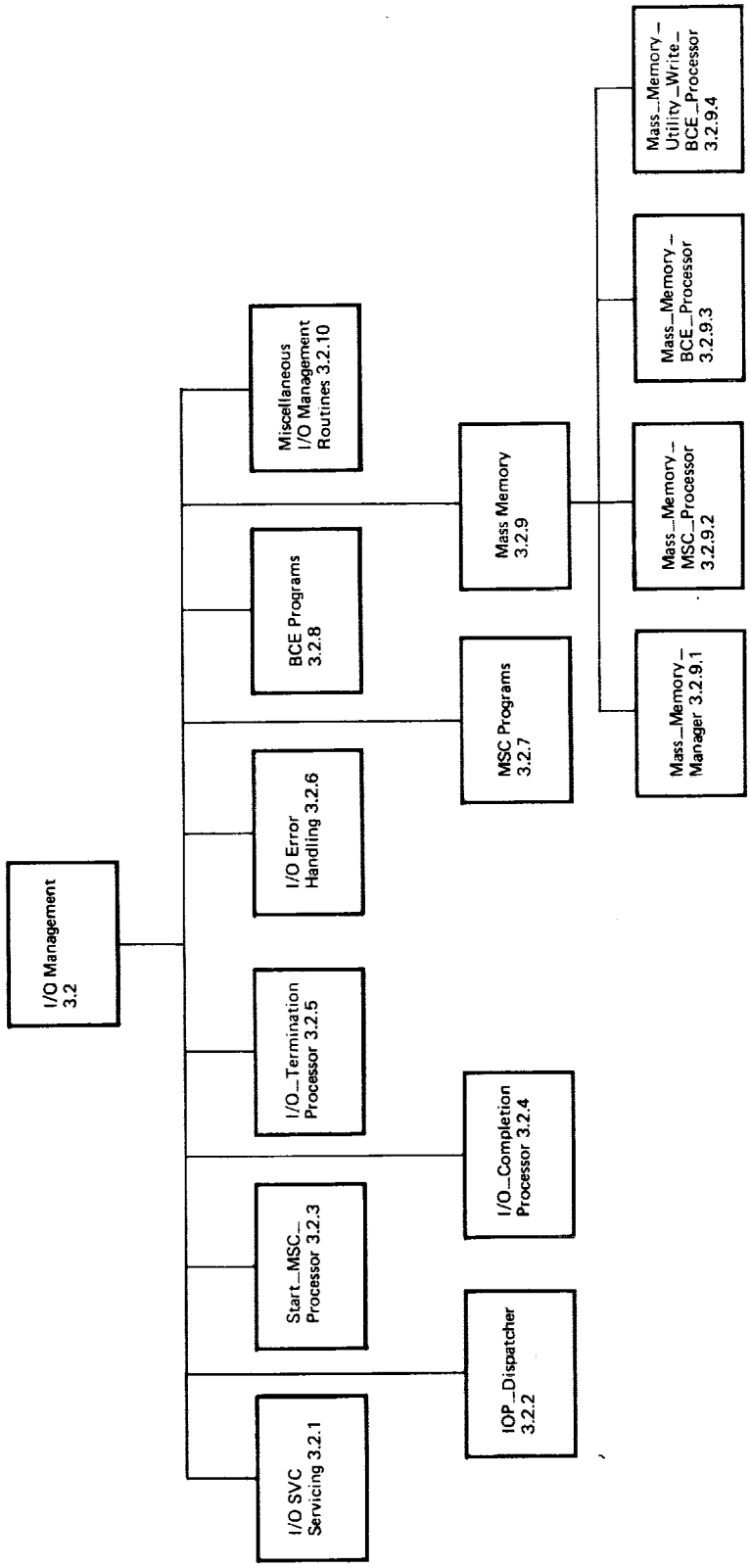

I/O Management 3.2
I/O SVC Servicing 3.2.1
Start_MSC_ Processor 3.2.3
I/O_Termination Processor 3.2.5
I/O Error Handling 3.2.6
BCE Programs 3.2.8
Miscellaneous I/O Management Routines 3.2.10
IOP_Dispatcher 3.2.2
I/O_Completion Processor 3.2.4
MSC Programs 3.2.7
Mass Memory 3.2.9
Mass_Memory_ Manager 3.2.9.1
Mass_Memory_ MSC_Processor 3.2.9.2
Mass_Memory_ BCE_Processor 3.2.9.3
Mass_Memory_ Utility_Write_ BCE_Processor 3.2.9.4

Figure 3.2.9-1.

3.2.9.1 Mass_Memory_Manager(FIOMMMGR)(280)

To Be Provided

3.2.9.2 Mass_Memory_MSC_Processor (FIOMMMSC) (281)

FIOMMMSC is an MSC routine that initiates BCE processing of Mass Memory transactions.

a. Control/Interface - Invoked by (210) Start_MSC_Processor (FIOSTMSC) when the MSC_Program_Counter is loaded.

b. Input - See Table 3.2.9.2-1

c. Process Description - The Mass_Memory_MSC_Processor when invoked loads the MSC_Accumulator_Register with the Mass Memory Start_I/O_BCE_Mask. The processor then starts BCE processing of Mass Memory transactions by ORing the MSC_Accumulator_Register contents with the IOP Busy_Wait_ Register. A 1 in bit position i of the MSC-Accumulator_Register will place BCE (i) in a busy state.

d. Output - None

e. Module Reference -

1. (251) MSC_I/O_Monitor (FIOMNTR) is called.

2. (282) Mass_Memory_BCE_Processor (FIOMMUPG) is started.

3. (283) Mass_Memory_Utility_Write_BCE_Processor (FIOMUWPG) is started.

f. Module Attributes - MSC Program

g. Template References - N/A

h. Error Checks - None

i. Constraints and Assumptions - None

j. Detailed Implementation -

1. FIOMMMGR invokes FIOMMMSC through FIOSTMSC. FIOMMMER loads general purpose register 0 bits 0-15 with the address of FIOMMMSC then calls FIOSTMSC. FIOSTMSC then sets the MSC-Program_Counter with the program address in register 0 and sets the MSC busy.

2. FIOMMMSC unconditionally exits to FIOMNTR.

DATA TABLE 3.2.9.2-1

NAME Mass_Memory_MSC_Processor (FIOMMSC)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Start_I/O_BCE_Mask	Q001.18	I	205,280 305	250	TCVTSIOM			
2	Busy_Wait_Register	&006.3	W	250,251 252,280	251,280 281				
3	MSC_Accumulator_Register	&007.13	W	280	280,281				

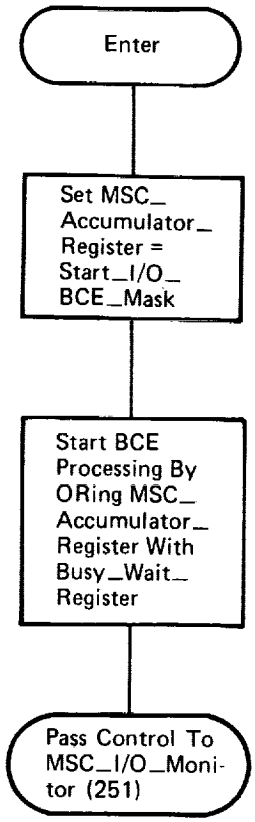


Figure 3.2.9.2-1. Mass_Memory_MSC_Processor (FIOMMMSC)

3.2.9.3 Mass_Memory_BCE_Processor (FIOMMUPG) (282)

The Mass_Memory_BCE_Processor is a CSECT that contains four BCE programs used to read, write, or position a Mass Memory tape and to read a MM status register.

a. Control Interface - Started by SIO instruction in (281) Mass_Memory_MSC_Processor (FIOMMMSC).

b. Input - See Table 3.2.9.3-1

c. Process Description - The read BCE program when invoked will 1) execute a command instruction from the Mass_Memory_Extended_Block_Count_Table to set the MM extended block count, 2) send a LISTEN command to all BCEs in other IOPs on the same bus, 3) delay the BCE for approximately 198 microseconds to allow the listening BCEs to setup, 4) execute a command instruction from the BCE_Command_Word_Table that request s MM data, 5) load the BCE_Base_Register from the BCE_Base_Register_Table, 6) execute a receive data command for the number of words specified in the BCE_Word_Count_Table, and 7) branch to the Common_BCE_Processing program.

The write BCE program when invoked will 1) execute a command instruction from the Mass_Memory_Enable_Write_Table that overrides MM write protection 2) execute a command instruction from the BCE_Command_Word_Table which request s the MM to receive data, 3) load the BCE_Base_Register with the address of the Mass_Memory_Management_Search_Complete_Word_Buffer, 4) execute a receive data command for the search complete word, 5) load the BCE_Base_Register from the BCE_Base_Register_Table, 6) execute a transmit data command to transfer data from the GPC to the MM, and 7) branch to the Common_BCE_Processing program.

The position tape BCE program when invoked will 1) execute a command instruction that sends a LISTEN command to BCEs in other IOPs, 2) delay the BCE 198 microseconds to allow listening buses to setup, 3) load the BCE_Base_Register with the MMU1_BTU_BITE or MMU2_BTU_BITE buffer address, 4) execute a command instruction that requests the MM_Status_Register, 5) executes a read command to receive the MM_Status_Register into a buffer, 6) execute a command instruction from the BCE_Command_Word_Table which positions the MM tape, and 7) branches to the Common_BCE_Processing program.

The read MM_Status_Register BCE program when invoked will 1) execute a command instruction that sends a LISTEN command to BCEs in other IOPs, 2) delays the BCE 198 microseconds to allow listening BCEs to setup, 3) loads the BCE_Base_Register with the MMU1_BTU_BITE or MMU2_BTU_BITE Buffer address, 4) execute a command instruction that requests the MM_Status_Register, 5) executes a read command to receive the MM_Status_Register, and 6) branches to the Common_BCE_Processing program.

d. Output - See Table 3.2.9.3-1.

e. Module References - (298) Common_BCE_Processing(#PFIOECT) is invoked.

f. Module Attributes - BCE Program

g. Template References - None

h. Error Checks - None

i. Constraints and Assumptions - None

j. Detailed Implementation - 1. All BCE programs within this program CSECT are invoked by having their address stored in the MM BCE_Program_Counter by the Mass_Memory_Manager and having the Mass_Memory_MSC_Processor **start** the MM BCE.

DATA TABLE 3.2.9.3-1

NAME Mass_Memory_BCE_Processor (FIOMMUPG)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	MMU1_BTU_BITE	I100.14	R	280,282, 820,880	282	CZ2BTMM1		X	X
2	MMU2_BTU_BITE	I100.15	R	280,820, 880	282	CZ2BTMM2		X	X
3	MM_Status_Register	&012.01	R		282				
4	MM_Block_Count_Register	&012.02	W		282				
5									
6	BCE_Base_Register	&008.09	I	282,302	260,282				
7	BCE_Command_Word_Table	Y007	I	205,280	264,282 283	FIOCWE			
8	BCE_Word_Count_Table	Y006	I	205	261,263 264,282	FIOWCE			
9	BCE_Base_Register_Table	Y005	I,R	205,280	See App. E	FIOBRE			
10	Mass_Memory_Extended_Block_Count_Table	Y012	I,W	280,282	282,283	FIOEBM			
11	Mass_Memory_Enable_Write_Table	Y013	I,W	280,282	282,283	FIOEWM			
12	Mass_Memory_Management_Search_Complete_Word_Buffer	Y022	I,R	282,283	282,283	FIOMSCWB			

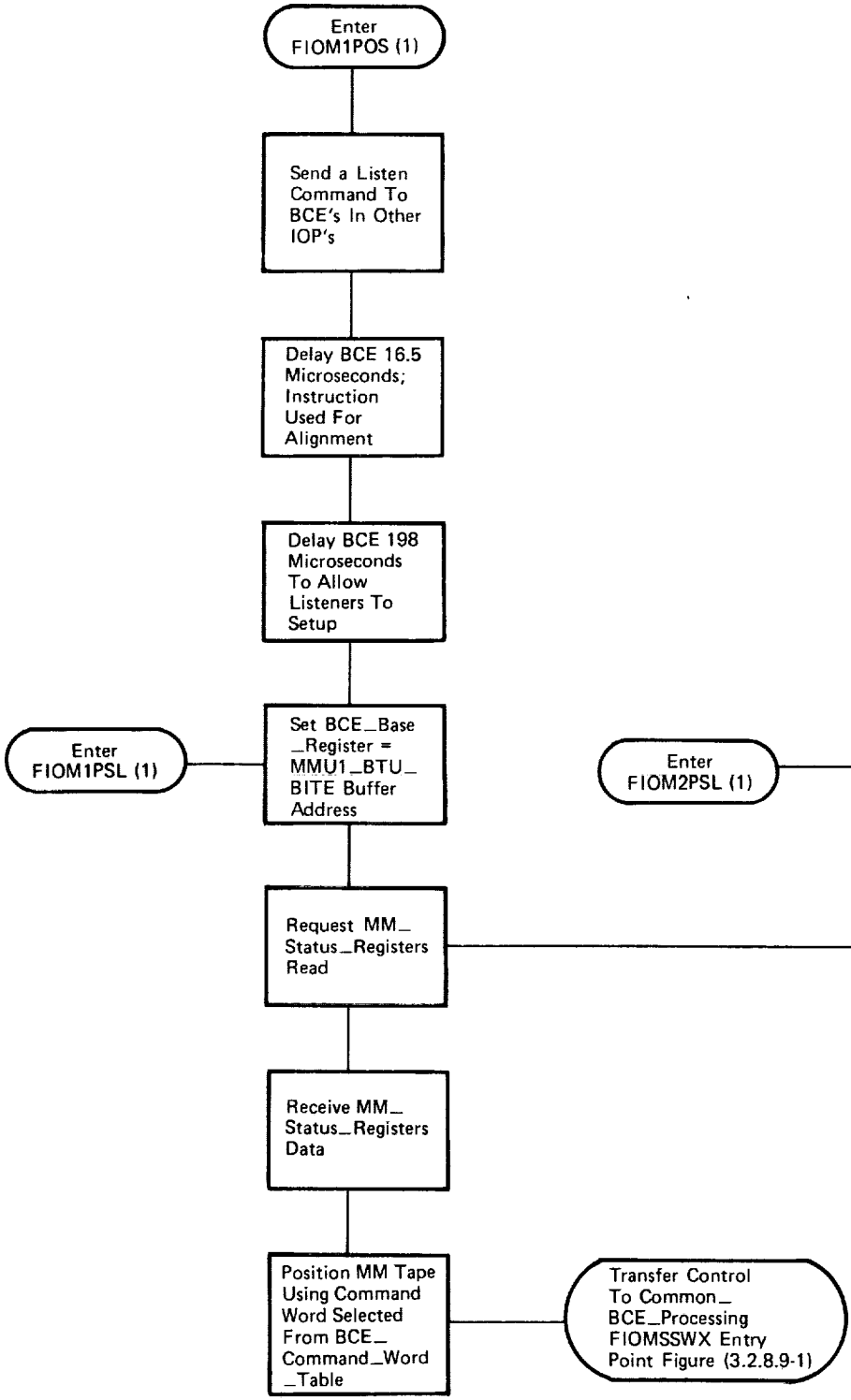


Figure 3.2.9.3-1. Mass_Memory_BCE_Processor Mass Memory Tape Position BCE Program

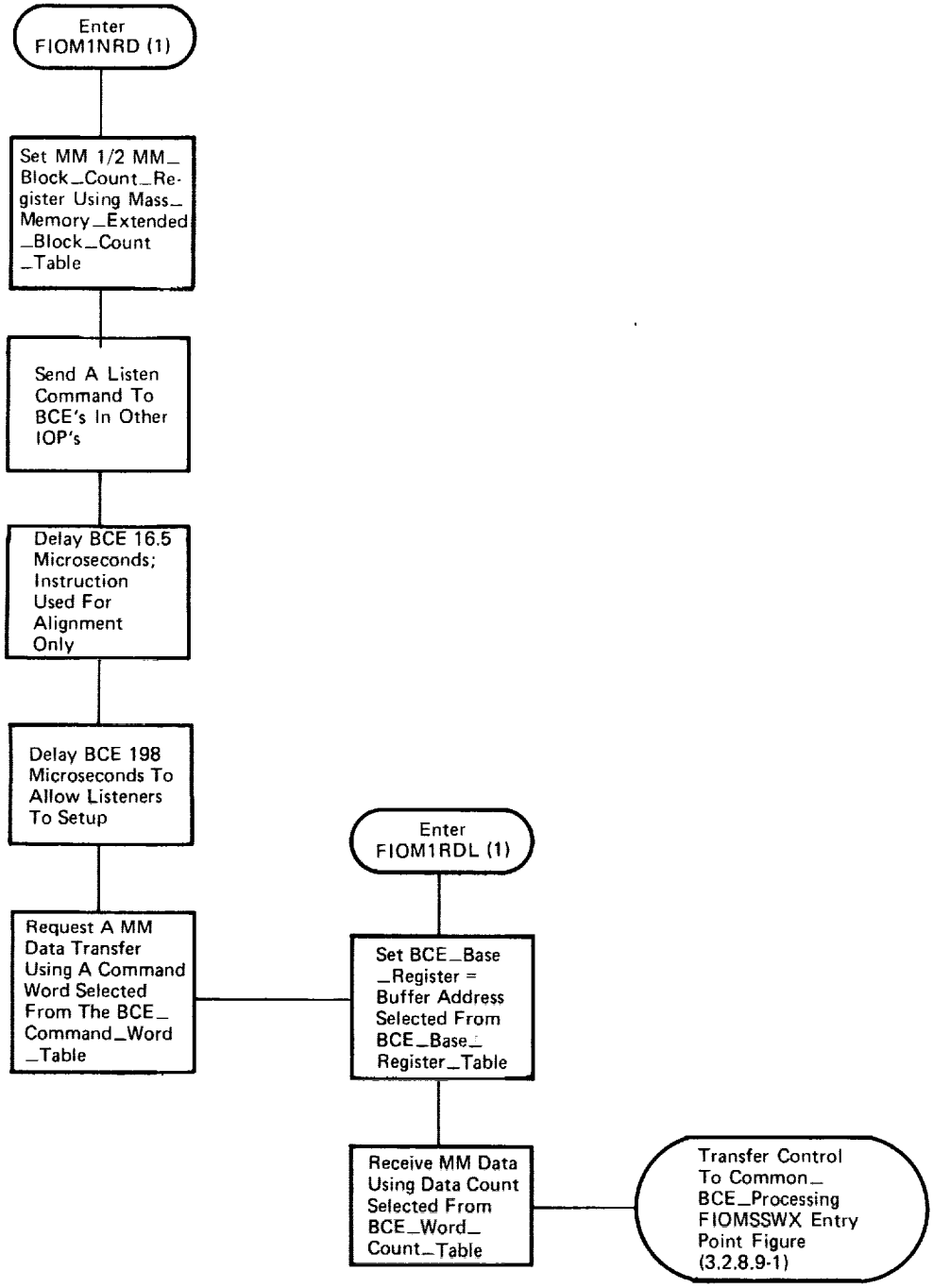


Figure 3.2.9.3-2. Mass_Memory_BCE_Processor Mass Memory Read BCE Program

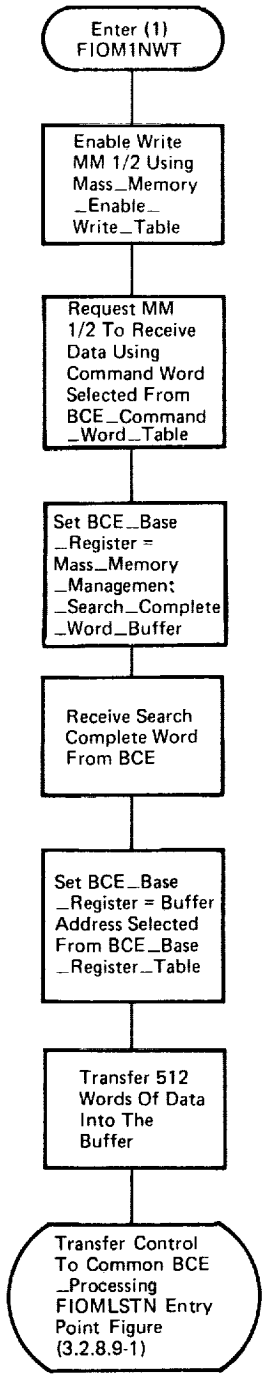


Figure 3.2.9.3-3. Mass_Memory_BCE_Processor Mass Memory Write BCE Program

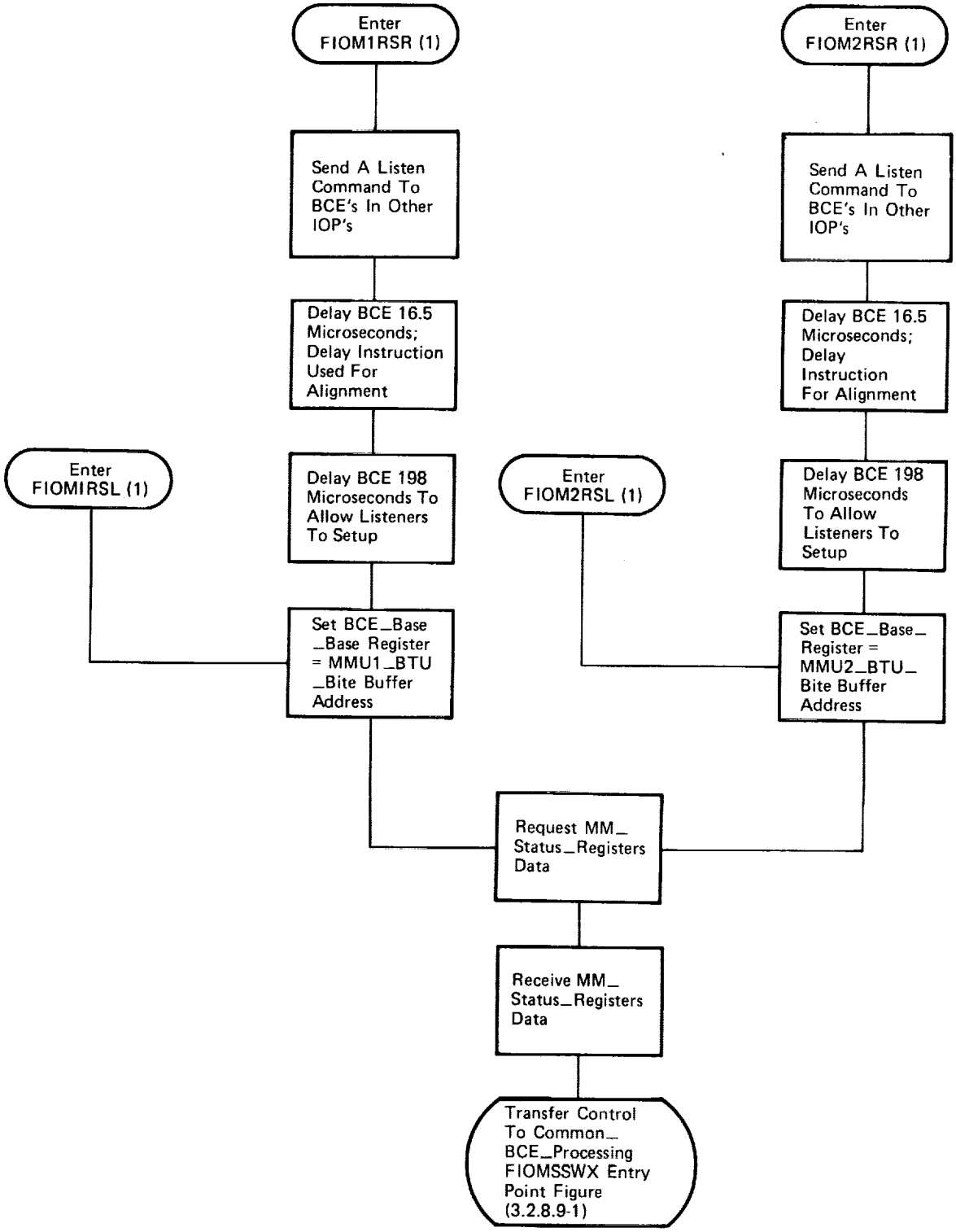


Figure 3.2.9.3-4. Mass_Memory_BCE_Processor Mass Memory Read Status Registers BCE Program

3.2.9.4 Mass_Memory_Utility_Write_BCE_Processor (FIOMUWPG) (283)

FIOMUWPG is a BCE program that writes 1 to 32 blocks of 512 16-bit words to Mass Memory based upon an address stored in the Mass_Memory_BCE_Branch_Table by FIOMMMGR.

a. Control Interface - Started by SIO instruction in (281) Mass_Memory_MSC_Processor (FIOMMMSC).

b. Input - See Table 3.2.9.4-1

c. Process Description - FIOMUWPG when invoked will 1) issue two command instructions that override MM write protection and set the extended blockcount; 2) issue two delay instructions that delay the BCE for approximately 132 microseconds; 3) issue a command instruction requesting MM to receive data; 4) issue 1-32 MM block transmit sequences; and 5) branches to the Common_BCE_Processing program.

   Each MM block transmit sequence with exception of the first will 1) load the BCE_Base_Register_Table with the address of the Mass_Memory_Management_Search_Complete_Word_Buffer, 2) issue a receive instruction to clear the MIA buffer, 3) issue a receive instruction for the search complete word, 4) reload the BCE_Base_Register with the address of a MM Block in the Mass_Memory_Utility_Base_Register_Table, and 5) issue a transmit instruction to output a 512 word block (1 MM block) to MM.

   The first transmit sequence will 1) load the BCE_Base_Register with the address of the Mass_Memory_Management_Search_Complete_Word_Buffer, 2) issue a receive instruction for the search complete word, 3) reload the BCE_Base_Register with the buffer address of the MM BCE entry in the BCE_Base_Register Table, 4) issue a transmit instruction to output 1 MM block, and 5) select from the Mass_Memory_BCE_Branch_Table the address of the next transmit sequence or the Common_BCE_Processing program and branch to it.

d. Outputs - See Table 3.2.9.4-1

e. Module References - (298) Common_BCE_Processing (#PFIOECT) is invoked.

f. Module Attributes - BCE Program

g. Template References - N/A

h. Error Checks - None

i. Constraints and Assumptions - None

j. Detailed Implementation -

1. FIOMMMGR stores in the MM_BCE_Program_Counter the address of FIOMUWPG. FIOMMMGR then calls FIOMMMSC to busy the MM BCE with FIOMUWPG.

There are 32 MM block t ransmit sequences within FIOMUWPG that are used to transmit 1-32 MM blocks to MM during a given execution. The 32 transmit sequences are contiguous and, with exception of the first, identical. Each transmit sequence transfers 1 MM block to MM.

The first transmit sequence is always executed and has as its last instruction an unconditional branch to the next transmit sequence or to the Common_BCE_Processing program. The unconditional branch instruction at t ains the address of the next transmit sequence by indexing the Mass_Memory_BCE_Branch_Table. The Mass_Memory_BCE_Branch_Table is set by FIOMMMGR.

The next transmit sequence is the first transmit sequence in a block of N-1 transmit sequences used to transmit $N(2 \leq N \leq 32)$ MM blocks to MM. An unconditional branch to the Common_BCE_Processing program is executed after the last transmit sequence has been executed.

DATA TABLE 3.2.9.4-1

NAME Mass_Memory_Utility_Write_BCE_Processor (FIOMUWPG)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Mass_Memory_Enable_Write_Table	Y013	I	280, 282	282 283	FIOEWM			
2	Mass_Memory_Extend_Block_Count_Table	Y012	I	280, 282	282 283	FIOEBM			
3	BCE_Command_Word_Table	Y007	I	205 280	264,282, 283	FIOCWE			
4	BCE_Base_Register_Table	Y005	I	205 280	See App. E	FIOBRE			
5	Mass_Memory_Management_Search_Complete_Word_Buffer	Y022	O	282 283	282 283	FIOMSCWB			
6	Mass_Memory_Utility_Base_Register_Table	Y021	I	280	283	FIOBRU			
7	Mass_Memory_BCE_Branch_Table	Y022.1	I	280	283	FIOBBM			

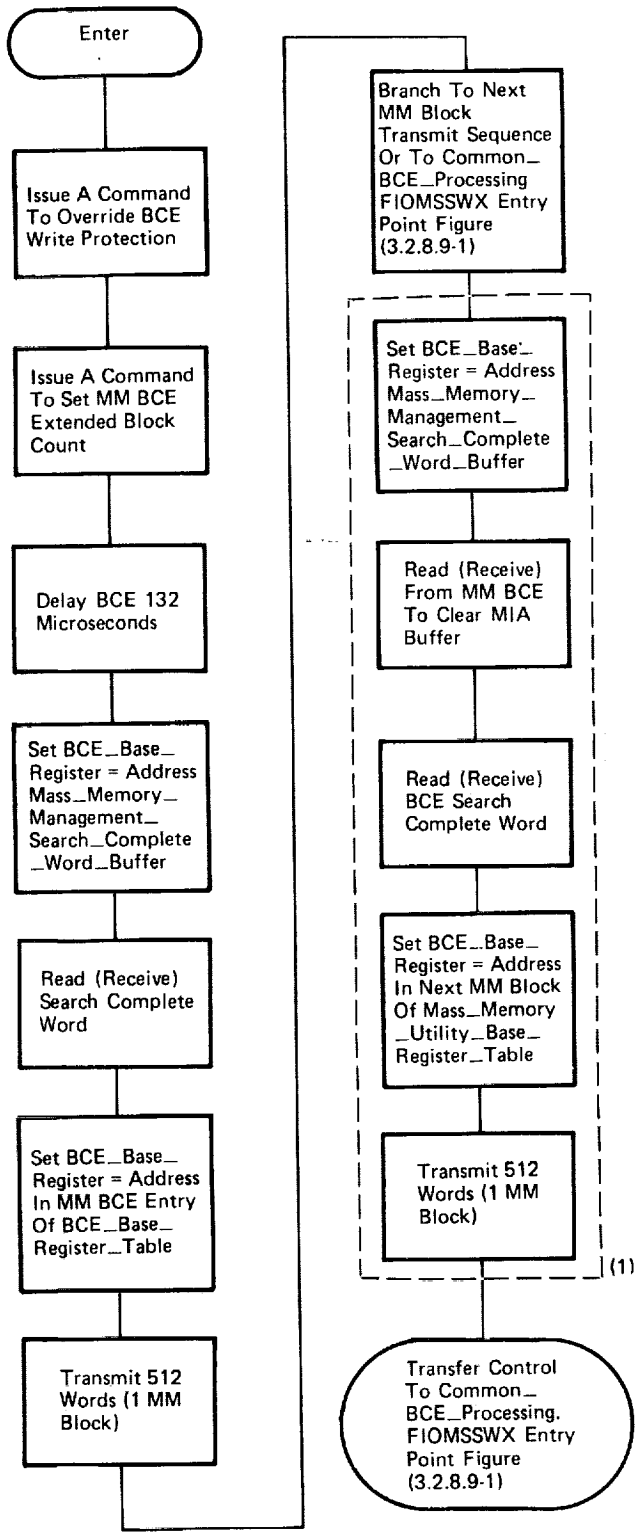


Figure 3.2.9.4-1. Mass_Memory_Utility_Write_BCE_Processor (FIOMUWPG)

3.2.10 Miscellaneous I/O Management Routine

Miscellaneous I/O Management routines are provided for use by application (as well as FCOS) to perform common I/O related processing. Currently, there are 2 such routines. (See Figure 3.2.10-1):

a. BTU_Port_Masking_Routine - Recomputes BTU Port Masks (Composite of data path masks.

b. Checksum_Generator - Computes a checksum for an indicated buffer.

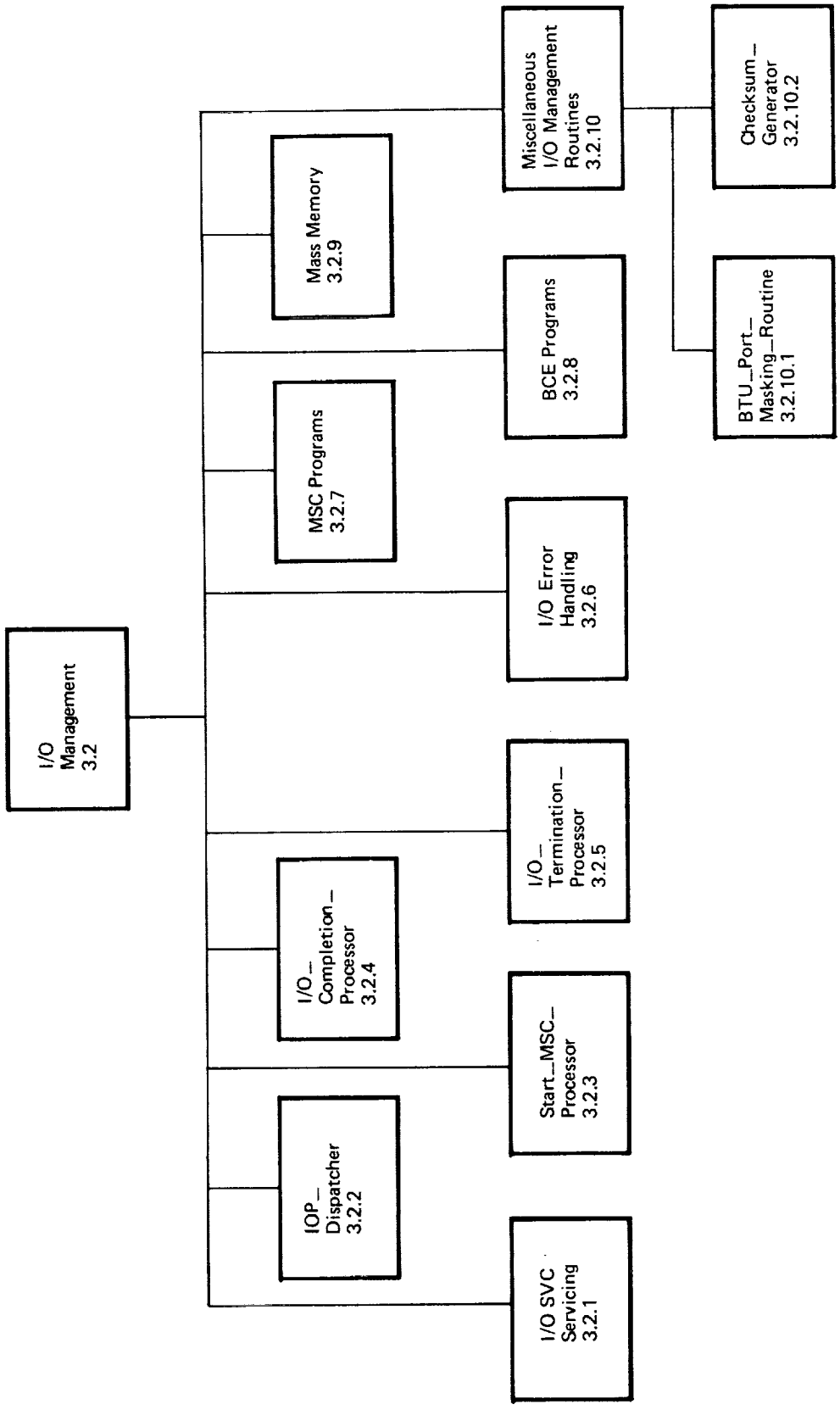


Figure 3.2.10-1.

3.2.10.1 BTU_Port_Masking_Routine (#CFIOBPM) (290)

BTU_Port_Masking_Routine updates the BTU_Port_Mask for each GPC after a data path mask has changed.

a. Control Interface - 1. CALLED by (490) ICC_Message_Router (DME_ICC_ROUT)

b. Input - 1. Register 0 contains the address of the Stack_Frame. See Table 3.2.10.1-1

c. Process Description - BTU_Port_Masking_Routine is CALLed by the HAL routine ICC_Message_Router to update the BTU_Port_Mask(s) after a data path mask has changed.

The address of BTU_Port_Mask and Data_Path_Mask is obtained and placed in registers for later use. For the first of the Interface Unit Adapters (currently two) attached to a GPC the GST_Address_Table is obtained. A temporary composite mask is initialized to zero. For each of the GPC's (currently 4) a LOGICAL SUM of the Data_Path_Mask and the temporary composite mask is computed and stored in BTU_Port_Mask.

The address of the next BTU_Path_Mask and Data_Path_Mask is computed and the process repeated for the last Interface Unit Adapter.

The Control flow for this module is shown in Figure 3.2.10.1-1.

d. Output - See Table 3.2.10.1-1

e. Module References - None

f. Module Attributes - External Procedure

g. Template References - None

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.2.10.1-1

NAME BTU_Port_Masking_Routine (#CFIØBPM)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Stack_Frame	@002	I	101	182 290	TFTSA			
2	GST_Address_Table	@004	I	368	See App. E	FPMGSTAD			
3	BTU_Port_Masks	#024	Ø	290	285,290 355,368	TFCMBPM	See Appendix E	X	X
4	Data_Path_Masks	I010.11	I/Ø	244,290, 355	148,290 665	TGSTDPM	See Appendix E	X	

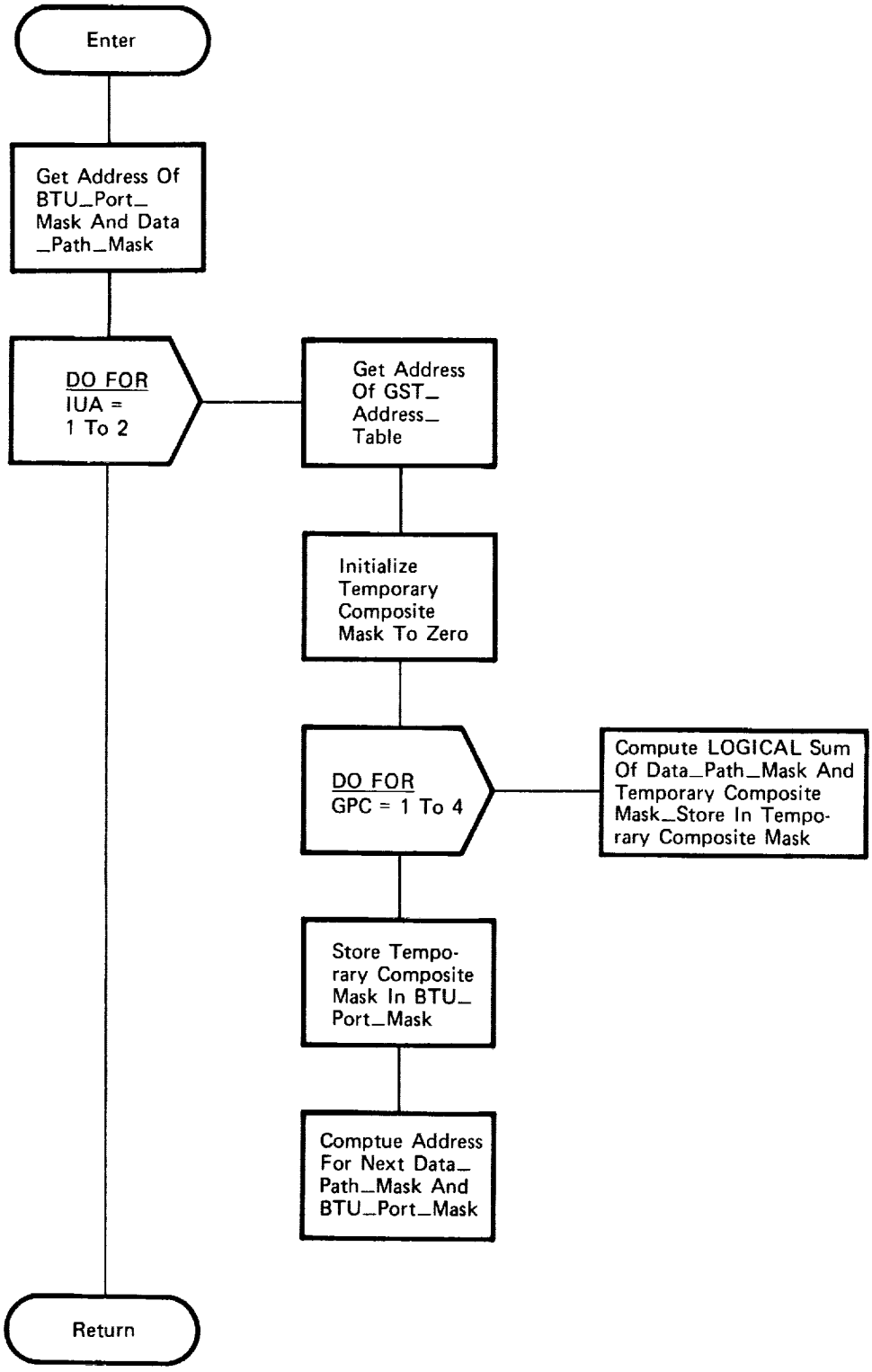


Figure 3.2.10.1-1. BTU_Port_Masking_Routine (# CFIOBPM)

3.2.10.2 Checksum_Generator (#CFIOCGR) (291)

The Checksum_Generator routine, #CFIOCGR, generates a checksum by summing a specified number of contiguous halfwords within a specified data buffer beginning with the first halfword in the data buffer. During the summing, any carry out of a register high order bit position or arithmetic overflow is ignored. After generating the checksum, #CFIOCGR places the checksum in the N or N+1 halfword of the Caller's data buffer.

a. Control Interface - Called by any HAL/S Application requiring a checksum.

b. Input - All input to #CFIOCGR is made via register settings.

Register 0 bits 0-15 contain the address of a stack frame.

Register 5 contains the number of halfwords(N) within the specified data buffer.

Register 6 bits 0-15 contain an indication (0/1) of the number of halfwords to be summed.

0 - Sum N-1 halfwords and place the checksum at data buffer halfword location N.

1 - Sum N halfwords and place the checksum at data buffer halfword location N+1.

Register 7 bits 0-15 contain the data buffer address.

c. Process Description - #CFIOCGR when called will perform in order the following processes.

- Zero the Local_Data_Area_Address
- Calculate the number of data buffer halfwords.
- Calculate an index to the last halfword in the data buffer.
- Calculate an index to the checksum halfword in the data buffer.
- Calculate the checksum by summing the specified number of halfwords in the data buffer beginning with the first halfword.
- Places the checksum in the N or N+1 halfword of the caller's data buffer.

d. Output - The generated checksum is placed in the N or N+1 halfword of the caller's data buffer. See Table 3.2.10.2-1.

e. Module References - None

f. Module Attributes - Program

g. Template References - N/A

h. Error Checks - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.2.10.2-1

NAME Checksum_Generator (#CFIOCGR)

NO.	ITEM	ID	ACT	SOURCE	DESTINATION	ASSEMBLER NAME	MML	D	C
1	Local_Data_Area_Address	@002.9	0	101	108,182, 184,290	TTSALCDA			

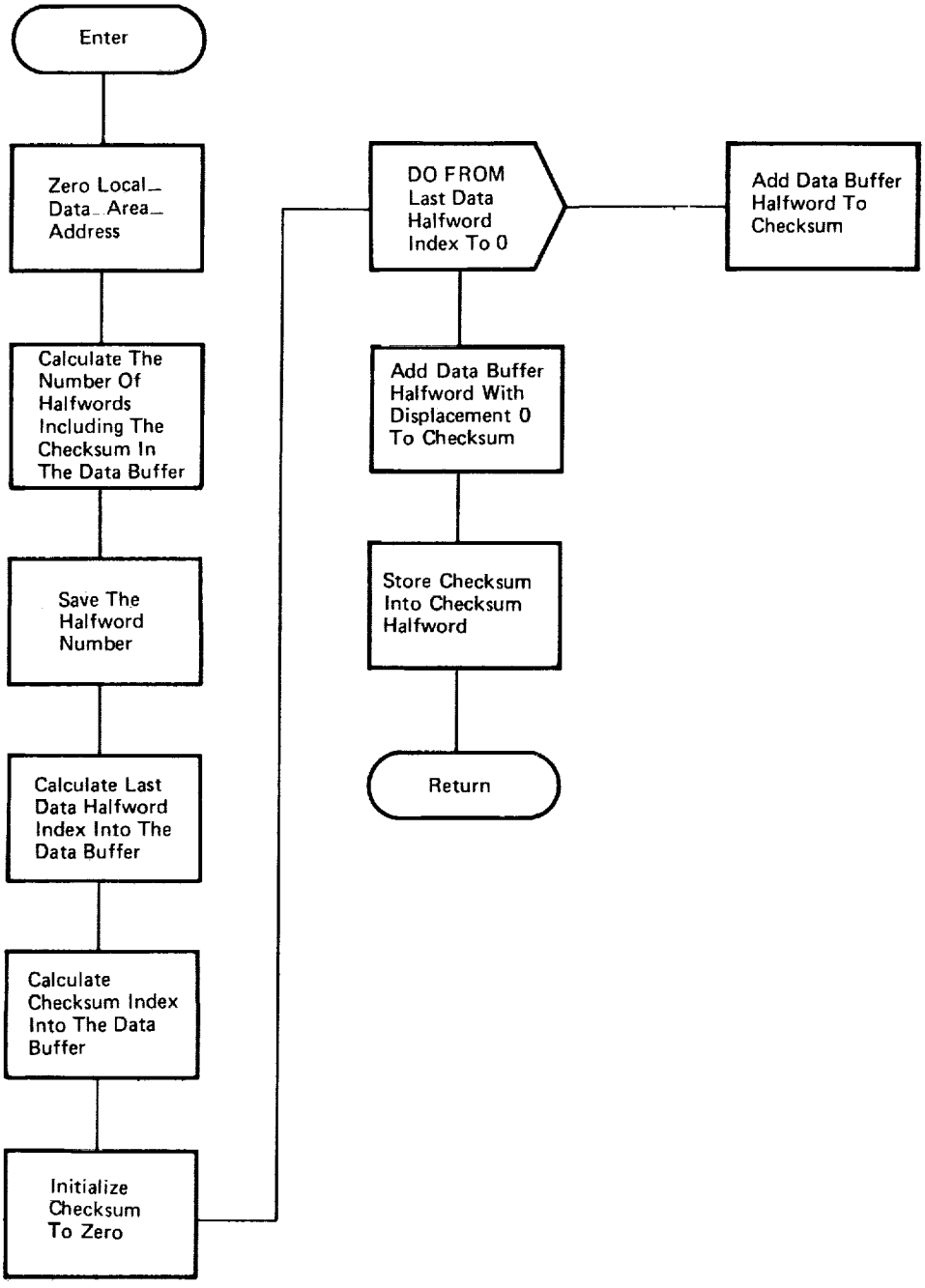


Figure 3.2.10.2-1. Checksum_Generator (# CFIOCGR)

3.3 DPS CONFIGURATION MANAGEMENT

DPS Configuration Management is the FCOS software support required for the control of GPC and IOP Configuration changes. This includes control of both the internal configuration of a single GPC as well as control over the configuration of the DPS.

FCOS Configuration Management provides control and sequencing of GPC initialization. This control includes the loading and initialization of system software into a GPC from mass memory, IOP bus configuration, and verification of the GPC health.

DPS Configuration Management is responsible for GPC synchronization with respect to time, I/O activity, and data exchange.

DPS Configuration Mangement controls configuration changes resulting from internally detected hardware failures, software directive, or user input.

In providing control of individual GPC main memory changes, DPS Configuration Management provides control of memory overlays to predefined memory locations as a result of OPS transitions. The ability to modify main memory or mass memory locations as a result of user input or uplink requests is also provided.

IOP bus configuration changes resulting from hardware failures, sync failures, user requests, or OPS changes are also supported. Figure 3.3-1 illustrates the elements of DPS Configuration Management and shows in which sections detailed descriptions of these elements can be found.

The description of DPS Configuration Management is divided into the following areas:

a. GPC Initialization - The loading of System Software into the GPC from mass memory, initialization of the IOP and initialization of the GPC.

b. Memory Management - FCOS support for overlays, program modification and user modification of BCE chains.

c. Miscellaneous_CM_Request_Processor - FCOS support for SVC requests to read or write discretes, set/reset the CAM lights and set/reset the termination control latches.

d. Bus Configuration - The control of IOP bus configuration changes resulting from hardware failures, sync failures, user requests or OPS changes.

e. GPC Redundancy Management - Provides for GPC synchronization with respect to time, I/O activity and data exchange and also provides for fault detection.

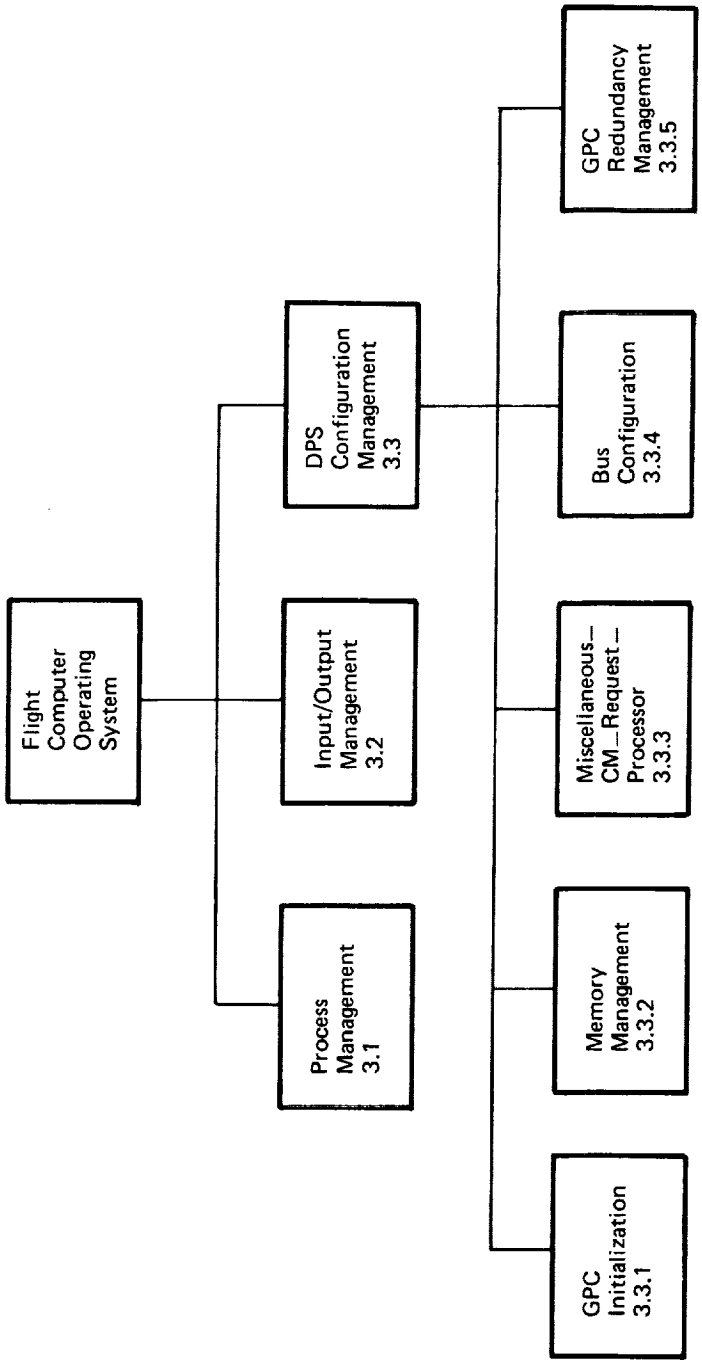


Figure 3.3-1. DPS Configuration Management Hierarchy Diagram

3.3.1 GPC Initialization

FCOS initialization responsibility includes the orderly initialization of a GPC, sequencing it to an operational state, achieving a confidence level for the GPC, initiating the I/O programs, ascertaining the status of other elements of the DPS, and allowing System Control and User Interface to perform its initialization.

There are two types of GPC initialization from an FCOS point of view. The first type is the Initial Program Load, or IPL Initialization that is performed when the user applies power to the GPC in the "HALT" mode, performs hardware IPL, and places the GPC in the "STBY" (standby) mode.

The second type, used when a GPC has been previously IPL'd, is Normal Initialization and is performed when the user applies power to a GPC in the "HALT" mode and then places the GPC in the "STBY" mode. Since FCOS gains control via the System Reset PSW (Program Status Word), which is loaded as a result of the mode switch being switched to "STBY" from "HALT", FCOS must have a way of distinguishing IPL by updating the System Reset PSW with the address of the normal initialization process after the IPL-unique functions are complete.

When power is applied with the mode switch in either the "RUN" or "STBY" position, the computer will go to "WAIT" with interrupts disabled.

Figure 3.3.1-1 illustrates the IPL initialization process. The locations of the software indicated in the figure do not necessarily reflect an actual memory configuration. The Self Test Program (STP) and Software_System_Loader (SSL) are read in by the hardware IPL sequence. The STP gains control to perform validity checks on the GPC. If no errors are found, it passes control to the SSL. The SSL reads FCOS software and permanently resident user interface data from Mass Memory and, at the same time, configures memory protection according to a pre-defined table. After FCOS initialization, System Control Performs its initialization and, through the FCOS program overlay function, brings the applications software into main memory as a result of an operational sequence being selected.

GPC Initialization is divided into the following areas. (See Figure 3.3.1-2).

a. Software_System_Loader - Performs the initialization at Initial Program Load (IPL).

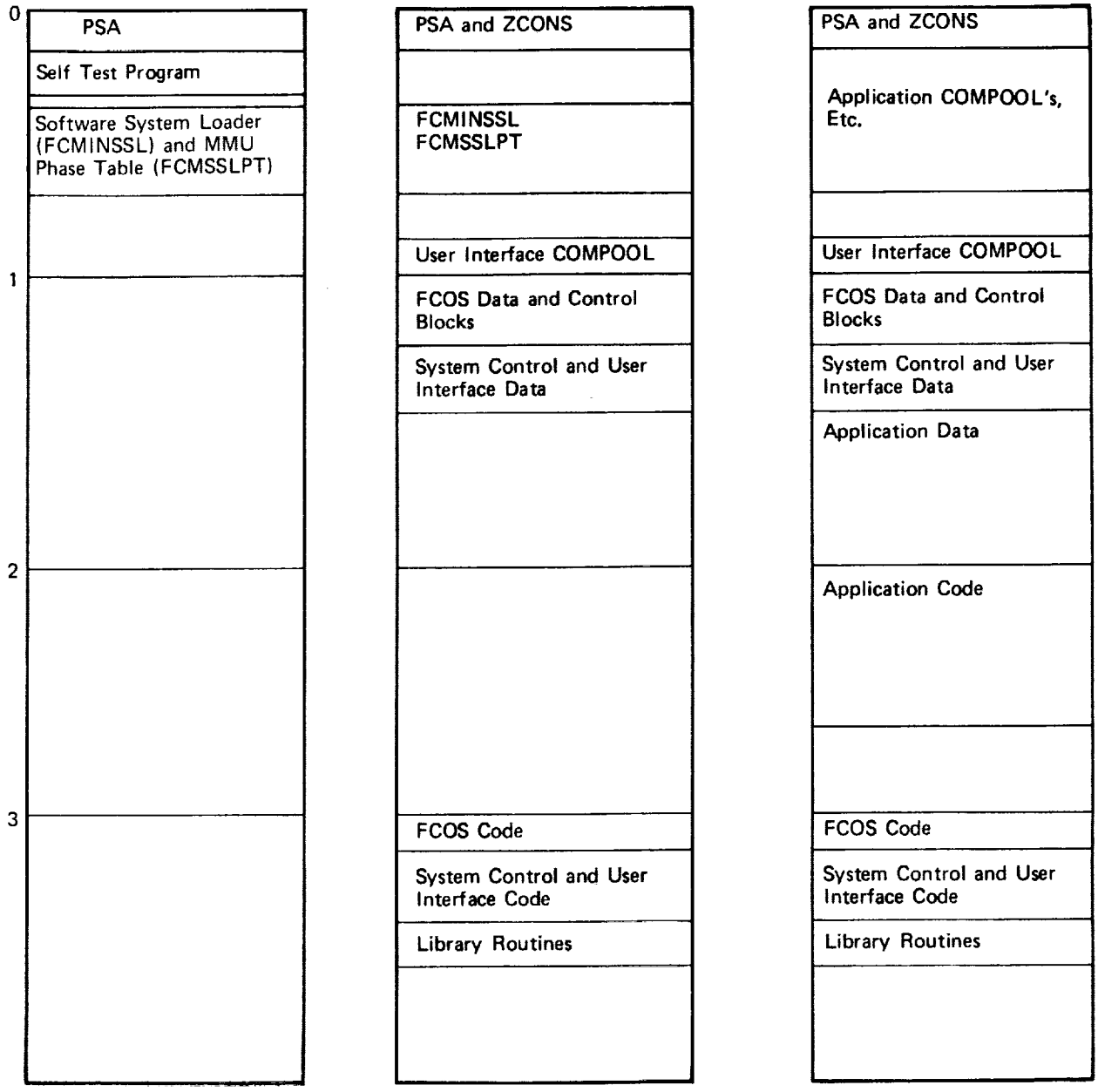
b. SSL_MM_MSC_Processor - The MSC program used by the SSL to read system software from mass memory.

c. SSL_MM_BCE_Processor - The BCE program used by the SSL to read system software from mass memory.

d. System_Load_Selector - The program used to decide which version of system software to read from mass memory.

e. Normal_Initialization_Processor - Performs the initialization when power is applied to a GPC that has been IPL'd.

f. IOP_Initialization_Processor - A service routine used by the SSL and Normal_Initialization_Processor to initialize the IOP.



A. After Hardware IPL

B. After Initial Software Load

C. After OPS Selection

Figure 3.3.1-1. IPL Example

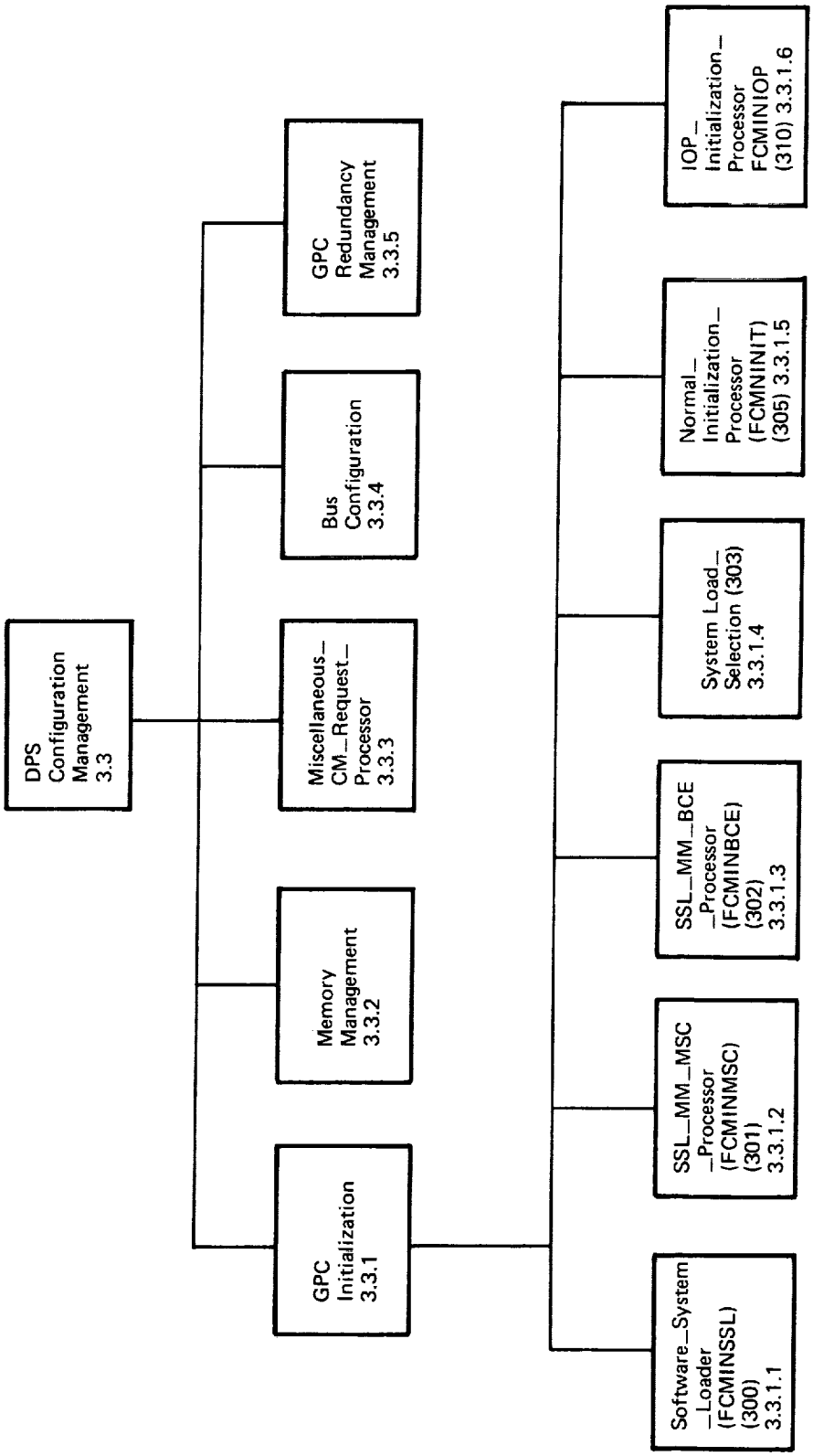


Figure 3.3.1-2. GPC Initialization Hierarchy Diagram

3.3.1.1 Software_System_Loader (FCMINSSL) (300)

FCMINSSL loads permanently resident system software and performs FCOS initialization after hardware IPL.

a. Control Interface - CALLed by the Self Test Program (STP).

b. Input - See Table 3.3.1.1-1.

c. Process Description - First, the FCOS register set, FCOS_BSR_DSR and FCOS_Program_Mask are set in the System_Reset_PSW and External_Interrupt_2_New_PSW. Next, the IOP is initialized to enable mass memory reads. The System_Software_Loader_Phase_Table is used to find a block of data on the mass memory and determine where in main memory the data should be read to. The main memory area is unprotected, the data read and the main memory may or may not be protected depending on the information in the table used to locate the block.

Once the permanently resident code and data have been read the following "one time" initialization functions are performed:

1. The PC1_Software_Portion, PC1_Hardware_Portion, PC2_Software_Portion and PC2_Hardware_Portion are set to their maximum values.

2. The GPC_Id is set to Discrete_Self-Id.

3. Set PSW_Wait_State_Indicator in all PSWS except the System_Reset_PSW and the Power_On_PSW are reset.

4. The Process_Control_Table for GPC_Locator is readied.

5. The System_Reset_PSW is updated to contain the address of the Normal_Initialization_Processor.

6. The IOP is initialized via the IOP_Initialization_Processor.

Once these functions have been performed control is passed to the Processor_Dispatcher. The control flow for this module is presented in Figure 3.3.1.1-1.

d. Output - See Table 3.3.1.1-1.

e. Module References -

1. (103) Processor_Dispatcher (FPMDISP) is CALLed.
2. (301) SSL_MM_MSC_Processor is activated.
3. (310) IOP_Initialization_Processor is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - A checksum is performed after each data block is read from mass memory. If an error is found, the program will try three times to read the data; then, it will enter the wait state.

i. Constraints and Assumptions - None

j. Detailed Implementation -

1. Control is passed via the System_Reset_PSW.
2. Wait State is entered via the Enter_Wait_State_PSW.
3. PSW_External_2_Mask is set to 1 and PSW_Wait_State_Indicator is set to 1.
4. Wait State is entered via the Enter_Wait_State_PSW.
5. Control is passed via the System_Reset_PSW.
6. MSC_Interrupt_Entry_Point is entered by the hardware loading of the External_Interrupt_2_New_PSW.

DATA TABLE 3.3.1.1-1

NAME Software_System_Loader (FCMINSSL)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	FCOS_BSR_DSR	0008.01	L		300	FCMZCPSW			
2	FCOS_Program_Mask	0008.02	L		300	FCMPM			
3	System_Reset_PSW	&001.15	I,O	300	142,190	TPSASRP			
4	External_Interrupt_2_New_PSW	&001.80	I,O	300, 305		TPSAE2NP			
5	System_Software_Loader_Phase_Table	0007	I		300	FCMSSLPT			
6	PC1_Software_Portion	&001.86	O	141,142 300	142	TPSAPC1			
7	PC1_Hardware_Portion	&003	W	300					
8	PC2_Software_Portion	&001.87	O	300	142	TPSAPC2			
9	PC2_Hardware_Portion	&004	W	300					
10	GPC_ID	#001	O	300	See App. E	TFCMID			
11	Discrete_Bits_GPC_ID	&009.01	R		300				
12	PSW_Wait_State_Indicator	&002.24	O	300	320				
13	Power_On_PSW	&001.05	O	300		TPSAPWR			
14	Process_Control_Table	Q003	O	300		TFPCT			
15	Enter_Wait_State_PSW	0008.03	L		300	FCMWTPSW			
16	PSW_External_2_Mask	&002.18	O	300,305					
17	FCMINSSL_Local_Data	0008	L	300	300	FCMDATA			
18	Discrete_BIT_MM1_Ready	&005.7	R		300				
19	Discrete_BIT_MM2_Ready		R		300				
20	BCE_Max_Timeout_Value	0008.04	O	300	300	FCMTMCDE			

DATA TABLE 3.3.1.1-1 (Cont'd)

NAME Software_System_Loader (FCMINSSL)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
21	BCE_Position_Tape_Command_Table	0008.05	L	300		FCMBCEPT			
22	BCE_Base_1	0008.06	L	300	302	FCMBCEB1			
23	BCE_Base_2	0008.07	L	300	302	FCMBCEB2			
24	BCE_Extend_Block_Command_Table	0008.08	L	300		FCMBCEEB			
25	BCE_Word_Count_Partial	0008.09	L	300		FCMBCEC1			
26	BCE_Word_Count_Whole	0008.10	L	300		FCMBCEC2			
27	BCE_Read_Command	0008.11	L	300		FCMBCERD			
28	Checksum_Error_Count	0008.12	L	300	300	FCMECNT			
29	Discrete_Input_Register_A	&005	R	300,340	300,340				
30	MSC_Program_Counter	&007.02	O	210 300					
31	BCE_Program_Counter	&008.02	O	300					

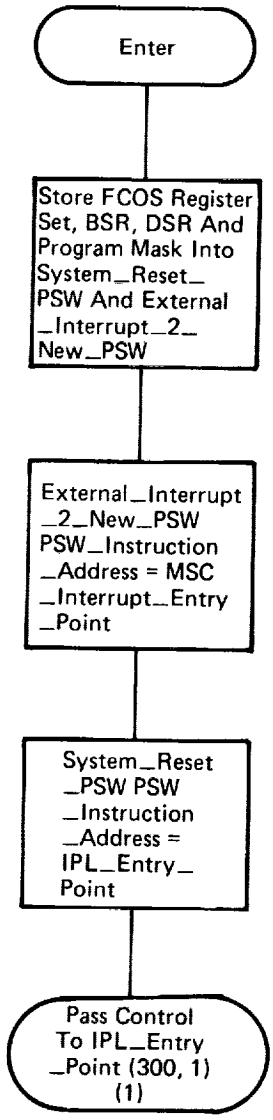


Figure 3.3.1.1-1. Software_System_Loader (FCMINSSL)

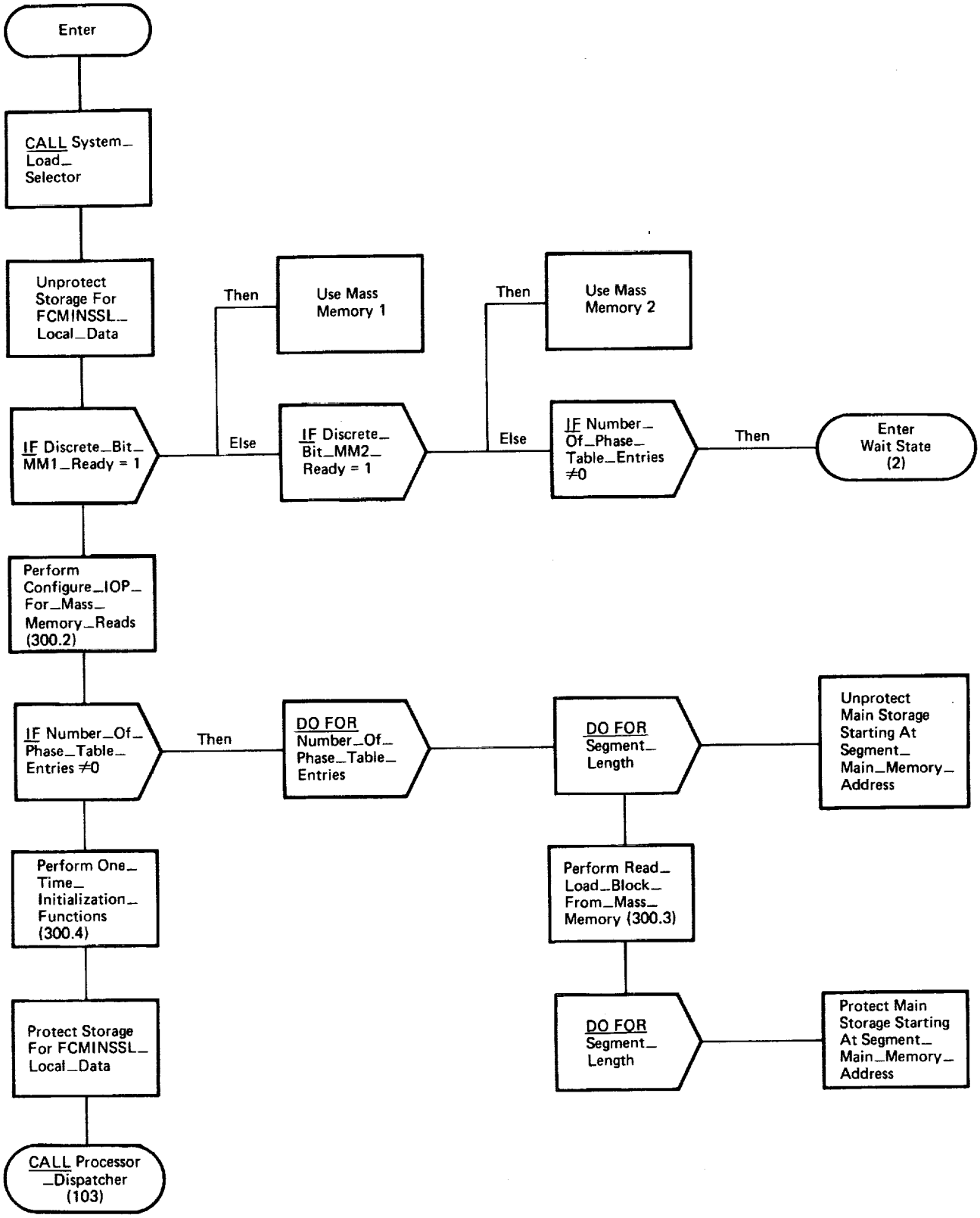


Figure 3.3.1.1-2. Software_System_Loader IPL_Entry_Point (300.1)

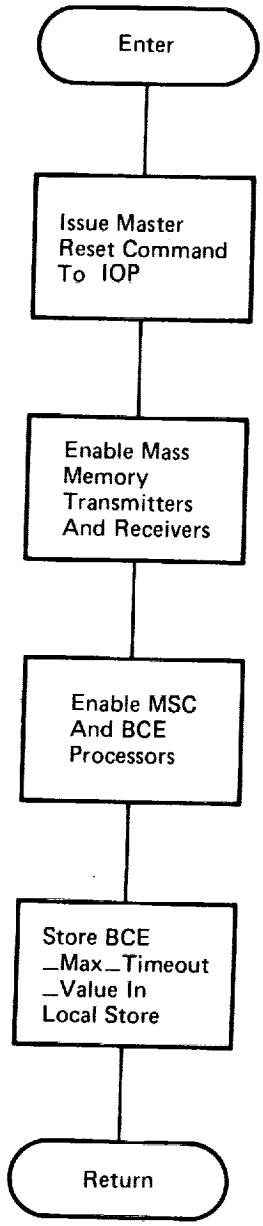


Figure 3.3.1.1-3. Software_System_Loader Configure_IOP_For_Mass_Memory_Reads (300.2)

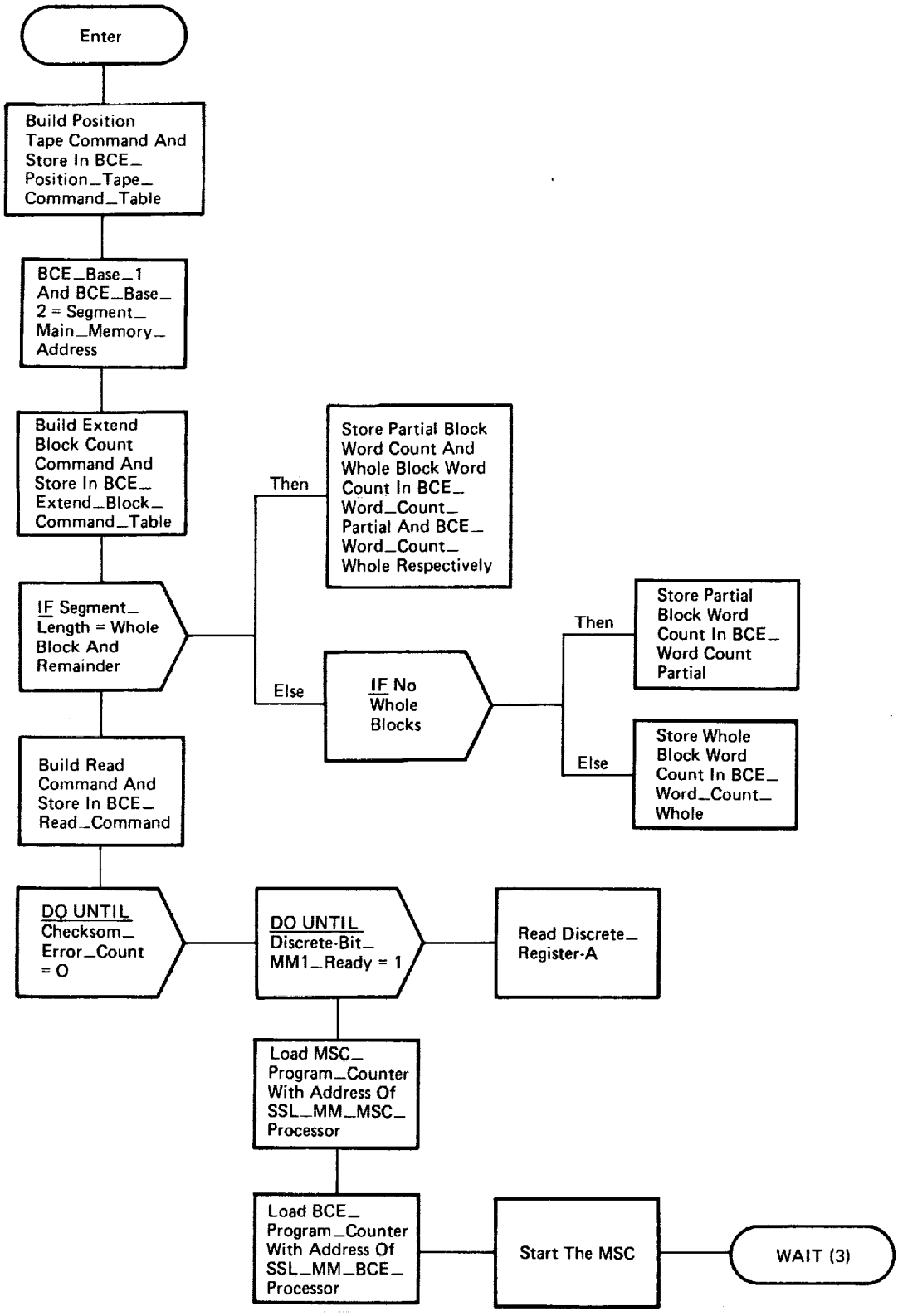


Figure 3.3.1.1-4. Software_System_Loader Read_Load_Block_From_Mass_Memory (300.3)

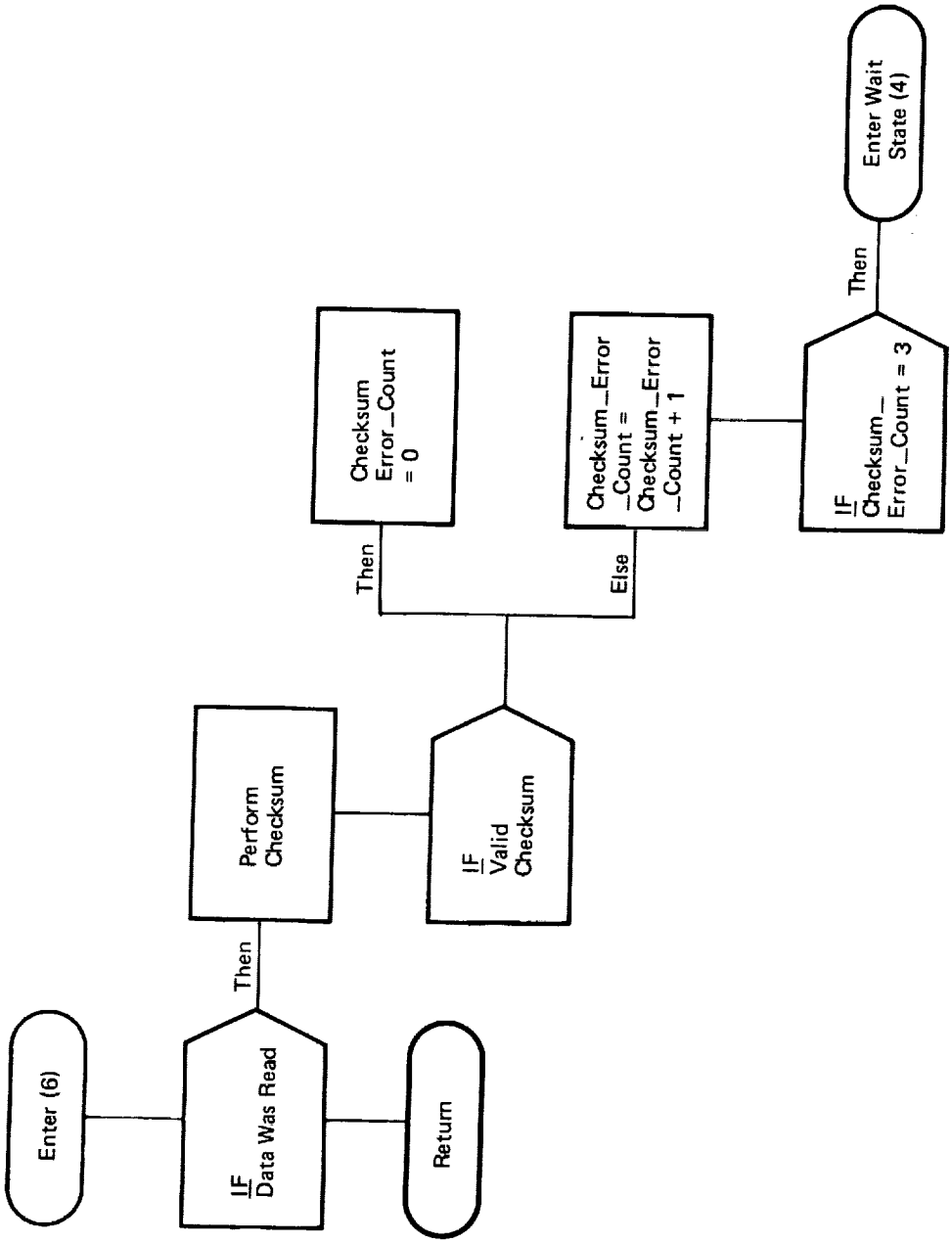


Figure 3.3.1.1-5. Software_System_Loader MSC_Interrupt_Entry_Point (300.4)

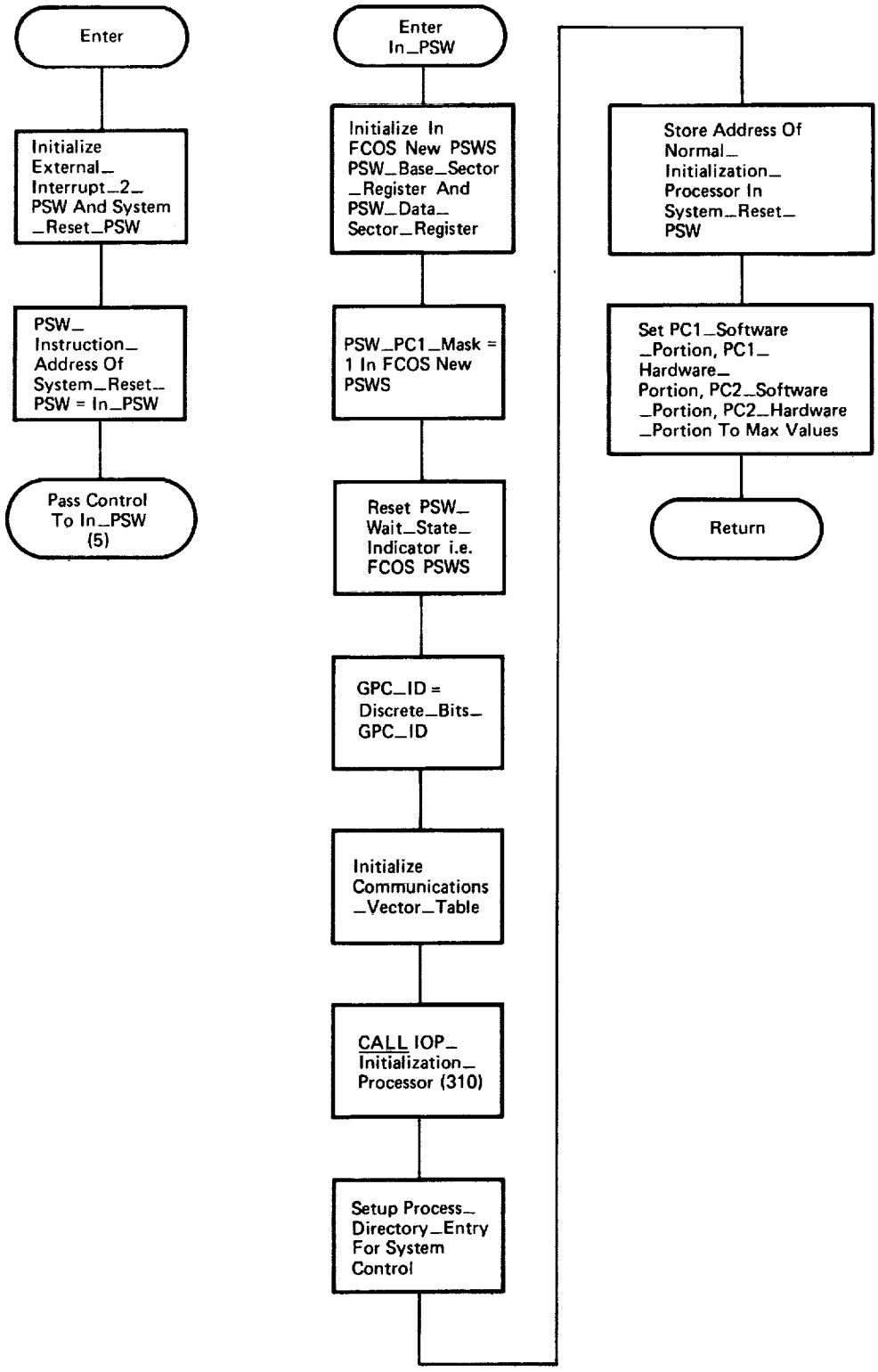


Figure 3.3.1.1-6. Software_System_Loader One_Time_Initialization_Functions (300.5)

3.3.1.2 SSL_MM_MSC_Processor (FCMINMSC) (301)

FCMINMSC is a MSC program used to start the BCE used to read system software from mass memory during IPL.

a. Control Interface - Activated by (300) Software_System_Loader (FCMINSSL). (This activation is accomplished via the PC instruction.)

b. Input - See Table 3.3.1.2-1.

c. Process Description - The SSL_MM_MSC_Processor starts the indicated mass memory BCE, waits for completion, interrupts the CPU upon completion or after maximum time has expired and finally enters a wait state. The control flow for this module is shown in Figure 3.3.1.2-1.

d. Output - See Table 3.3.1.2-1.

e. Module References - None

1. (302) SSL_MM_BCE_Processor (FCMINBCE) is started by an SIO instruction.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.3.1.2-1

NAME SSL_MM_MSC_Processor (FCMINMSC)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Max _Wait_Time	0009	L		301	FCMICNT			
2	MSC_Index_Register	&007.03	R		301				
3	IPL_MM_BCE_Mask	0008.13	I	300	301	FCMIBUSM			
4	MSC_Accumulator_Low	&007.09	R,0	301	301				
5	MSC_Accumulator_High	&007.06	R,0	301	301				

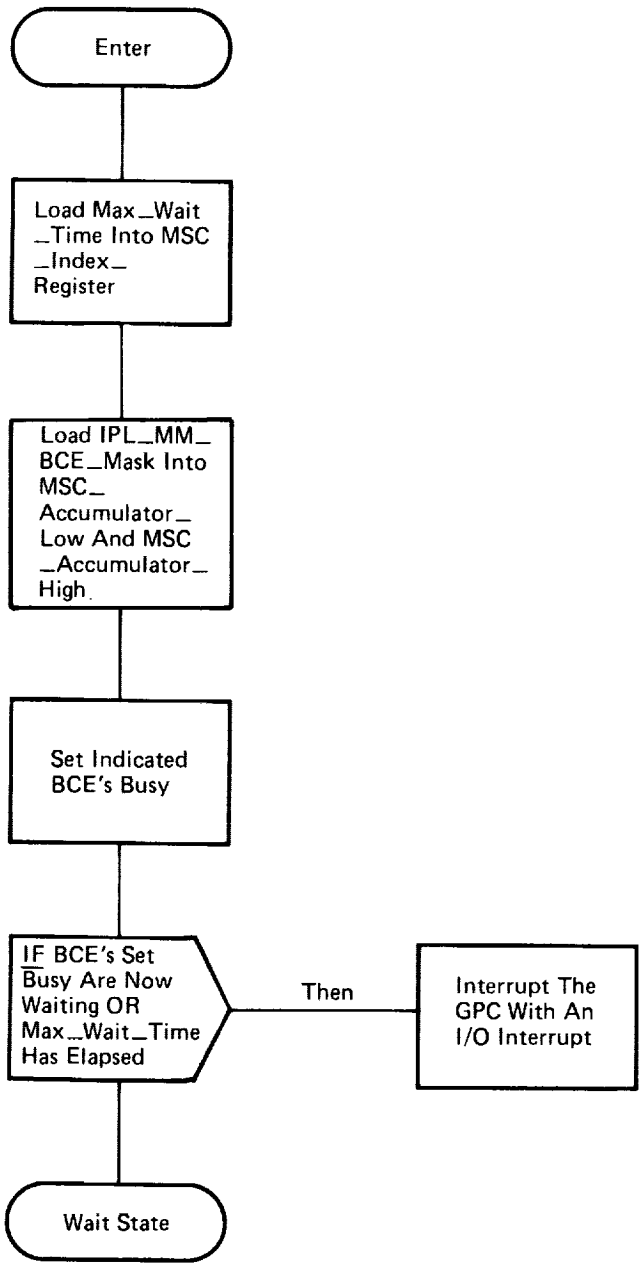


Figure 3.3.1.2-1. SSL_MM_MSC_Processor (FCMINMSC)

3.3.1.3 SSL_MM_BCE_Processor (FCMINBCE) (302)

FCMINBCE consists of two BCE programs used to position and read the mass memory for the (300) Software_System_Loader (FCMINSSL).

a. Control Interface - Activated by the (301) SSL_MM_MSC_Processor (FCMINMSC). (This activation is accomplished via the SIO instruction).

b. Input - See Table 3.3.1.3-1.

c. Process Description - The SSL_MM_BCE_Processor is activated twice every time data is required from mass memory. The first activation is at FCMINMMP - the position tape entry point. The SSL_MM_BCE_Processor positions the mass memory and returns to the wait state. The second activation is at FCMBCMMR - the read tape entry point. The SSL_MM_BCE_Processor reads the mass memory records specified and returns to the wait state. The control flow for this module is shown in Figure 3.3.1.3-1.

d. Output - See Table 3.3.1.3-1.

e. Module References - None

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation

1. Entry point FCMINMMP - position tape.

2. Entry point FCMBCMMR - read tape.

3. This decision is made for FCMINBCE by (300) Software_System_Loader (FCMINSSL). To FCMINBCE it is merely a branch table - Double_Single_Receive_Branch_Table.

DATA TABLE 3.3.1.3-1

NAME SSL_MM_BCE_Processor (FCMINBCE)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Double_Single_Receive_Branch_Table	0008.14	I	300	302	FCMBCEBT			
2	BCE_Base_Register	&008.09	W	282,302	260,282				
3	BCE_Base_1	0008.06	I	300	302	FCMBCEB1			
4	BCE_Base_2	0008.07	I	300	302	FCMBCEB2			

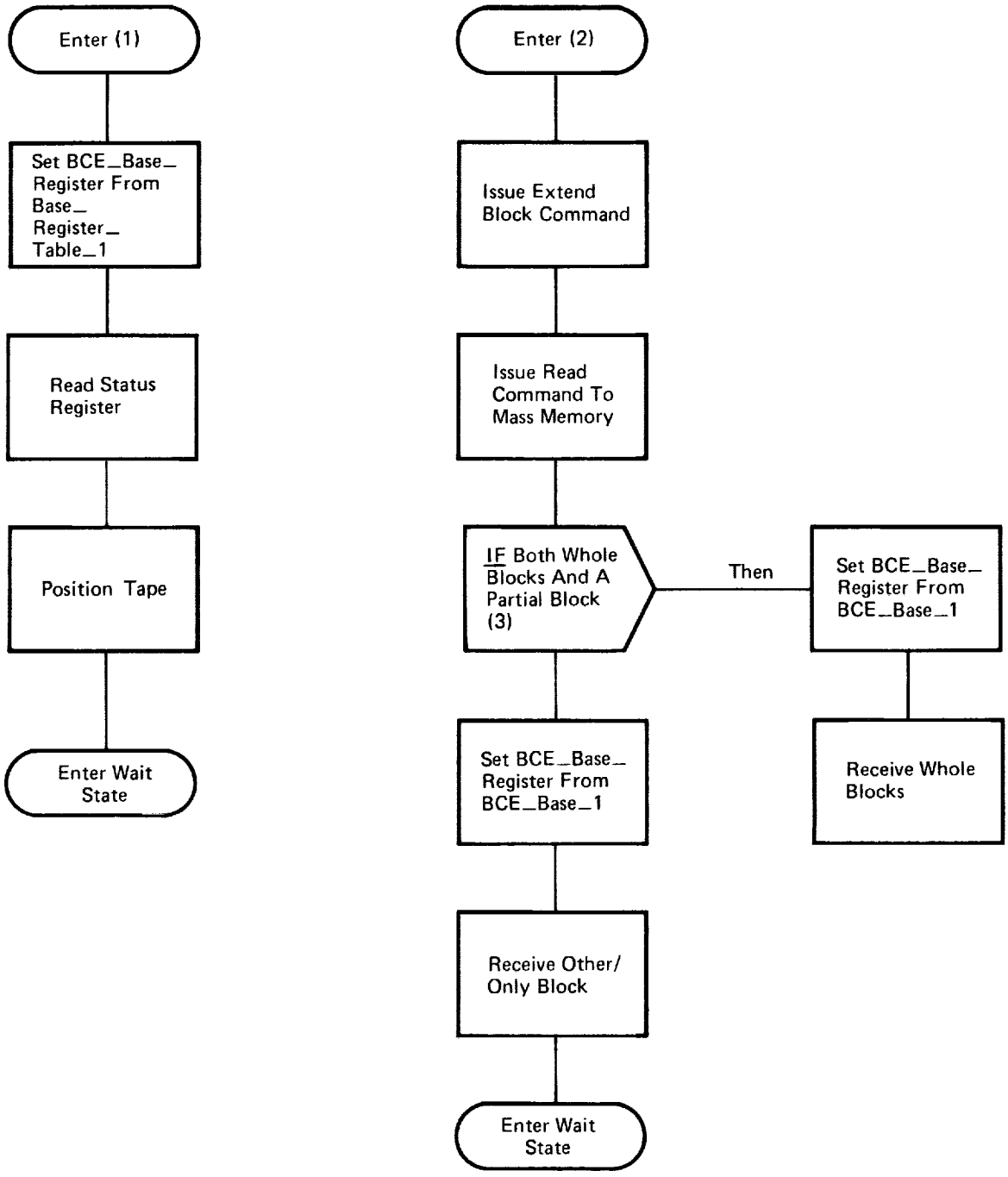


Figure 3.3.1.3-1. SSL_MM_BCE_Processor (FCMINBCE)

3.3.1.4 System_Load_Selector(FCMCH00Z) (303)

To Be Provided

3.3.1.5 Normal _ Initialization_Processor(FCMNINIT)(305)

To Be Provided

3.3.1.6 IOP_Initialization_Processor(FCMINIOP)(310)

To Be Provided

3.3.2 Memory Management

Memory Management maintains the proper main memory data/program configuration and provides for flight updates to the main memory, and provides the capability to disable (and re-enable) elements of BCE chains.

The GPC memory structure is discussed from a functional viewpoint to provide a basis for program overlays and program modifications. Program overlays are used to configure the main memory data/program organization. The source of program overlay is a mass memory unit. Program modifications provide the capability to alter the contents of main memory units.

The GPC memory structure presented in this document does not take into account any power utilization considerations. However, the assumptions presented in this section regarding memory structure do not preclude the consideration of power utilization factors in any final memory structure that may be defined.

The AP-101 computer contains 128K halfwords of main memory. Because of the memory addressing scheme used in the AP-101, main memory is subdivided into four sectors (sectors 0, 1, 2 and 3) of 32K halfwords each.

The FCOS and HAL/S compiler assume that all data and COMPOOL CSECTS (except HAL/S remote data CSECTS) reside in sector 0, which is always addressable, and if necessary one other data sector specified by the data sector register (DSR) field in the PSW. HAL/S remote data CSECTS can be located in any sector and are addressed via a special type address constant. A data or COMPOOL CSECT cannot span sectors. Code CSECTS can reside in any sector; however, only sector 0 and one other sector, specified by the branch sector register (BSR) field in the PSW, can be addressed directly. A code CSECT cannot span sectors. An example of a possible memory layout is presented in Figure 3.3.3-1.

Since the complete flight software package is too large to be permanently resident in main memory, the FCOS provides a program overlay capability to alter the data/program configuration during flight. However, dynamic memory allocation is not supported by the FCOS and all executable code and locations are determined preflight. Multiresidency of data and programs is allowed; however determined preflight. Multiresidency of data and programs is allowed; however, these locations will be determined preflight, not computed, by the FCOS.

All FCOS control tables are allocated at System Generation time using macros provided by FCOS. The use of macros provides the capability to parameterize the control table allocation procedure.

The FCOS control tables are split between two CSECTS. The preferred storage area and communication vector table reside in a CSECT which is located at main memory address 0000. The remaining FCOS control tables reside in a control table CSECT that can be located in any data sector. The control table CSECT contains the following table groups:

- Process Control tables
- Timer/Event queue elements
- I/O queue elements

Each group of tables occupy a contiguous area of memory. When these control tables are allocated, the individual control tables within each group are chained together to form free pool areas. These free pool areas contain all the unused control tables. As the various control tables are required by the FCOS they are unchained from the appropriate free pool area and rechained in the appropriate active control table chain. When an active control table is no longer needed by the FCOS, it is unchained from its active chain and rechained in the approrpiate free pool area for future use. The control table CSECT also contains the temporary stack areas and error log out areas.

Memory Management is divided into the following areas (see Figure 3.3.2-2).

a. Overlay_Processor - Provides the capability to change the configuration of main memory data and code.

b. Program_Modification_Processor - Supports the additions to and modification of programs in main memory.

c. BCE_Element_Bypass_Processor - Processes requests to bypass or restore elements of BCE chains.

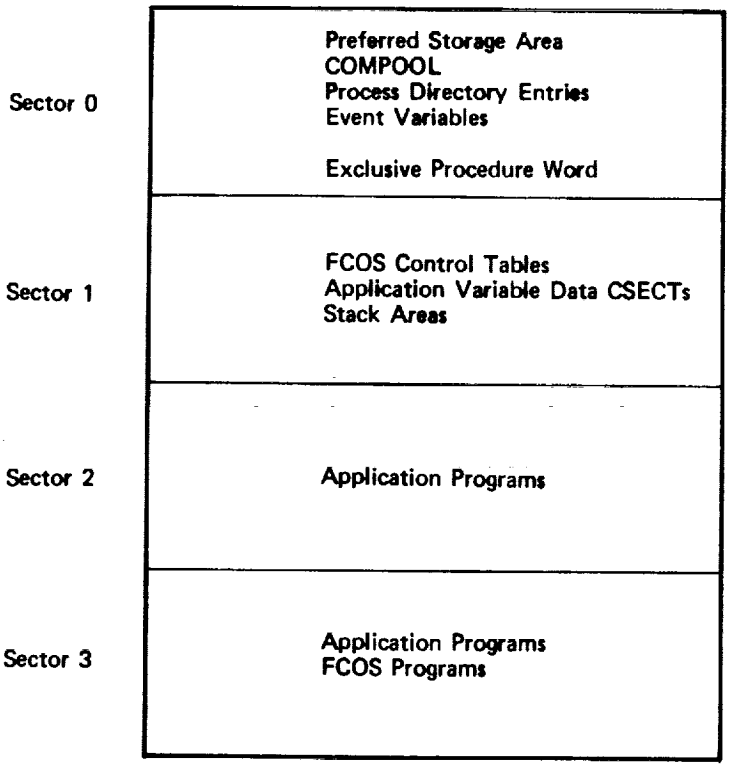


Figure 3.3.2-1. Main Memory Organization Example

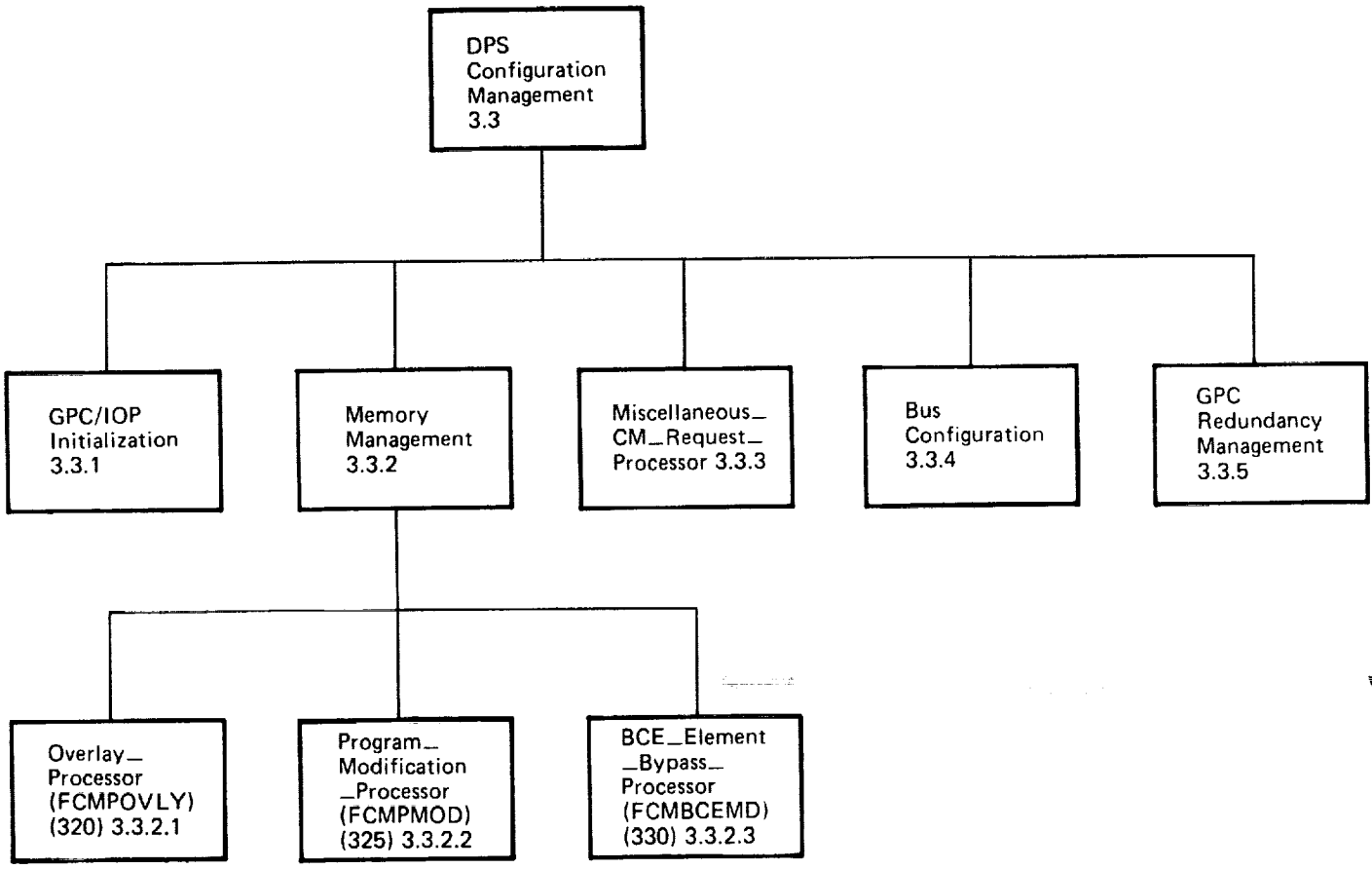


Figure 3.3.2-2. Memory Management Hierarchy Diagram

3.3.2.1 Overlay Processor (FCMPOVLY) (320)

FCMPOVLY provides FCOS support to configure the proper main memory data/ program organization.

a. Control Interface - CALLed by (100) SVC_Handler (FPMSVC)

b. Input - See Table 3.3.2.1-1

c. Process Description - The SVC_Synchronization_Processor is invoked to perform redundant set sync. The FCOS support function associated with program overlays is performed under a PCT with all interrupts enabled. Upon entry to the Overlay_Processor, a PCT is initialized and chained. into the process run queue based on the PCT_Priority of the PCT indicated by the Active_PCT_Address. Information from the Overlay_ Parameter_List is saved in the Process_Interrupt_Register_2, 3, 4, 5 fields. The FCOS_PCT_Indicator is set to communicate to various FCOS modules that this PCT is being used for FCOS processing. The Process_ Dispatcher is then used to dispatch the highest priority ready PCT.

When the Overlay_Processor PCT is dispatched by the Process_Dispatcher, the overlay is initiated.

The Overlay_Source is checked to determine the requested source for the overlay. Currently Mass Memory is the only source supported. The processing consists of the following for the Function_Base_Phase_Number and/or Program_Overlay_Phase_Number specified in the Overlay_Parameter_ List:

1. Reads the Mass_Memory_Phase_Table into core from the mass memory (Issues I/O SVC with wait type I/O). This only done if the Mass_ Memory_Phase_Table is not already in core. If an error occurs, the field indicated in the Overlay_I/O_Status__Address is updated to indicate that all GPC's attempting the transaction have failed and overlay processing is stopped.

2. If the above error does not occur, the following is done for each Segment_Information record associated with the requested phase number:

   - o Unprotects the area to be overlayed
   - o Reads the overlay (Issues wait type I/O SVC)
   - o Protects the overlayed area if necessary (Segment_ Protect_Flag = 1).

The Active_PCT (which is the Overlay_Processor PCT) is then removed from the PCT run queue and the Next_To_Execute_Address is set to the Next_PCT_Address field of the Overlay_Processor PCT. The event specified by the Overlay_Completion_Event_Address is then set to indicate that the overlay request has been serviced. The Event_Evaluator is invoked to evaluate any outstanding EQE's involving this event.

The control flow for this module is shown in Figures 3.3.2.1-1 to 3.3.2.1-3.

d. Output -

1. The event specified in the Overlay_Completion_Event_Address is set.

2. The field specified in Overlay_I/O_Status_Address is updated to reflect errored GPC's if necessary.

3. See Table 3.3.2.1-1.

e. Module References -

1. (103) Process-Dispatcher (FPMDISP) is CALLed.
2. (200) I/O_SVC_Service_Processor (FIOSVC) is invoked via SVC.
3. (174) Event_Evaluator (FPMEVAL) is CALLed.
4. (131) Chain_PCT_Routine (FPMCHPCT) is CALLed.
5. (132) Release_PCT_Routine (FPMRLPCT) is CALLed.
6. (363) SVC_Synchronization_Processor (FCMSSYNC) is called

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - If an I/O error occurs for overlay MM I/O, the field specified by Overlay_I/O_Status_Address is updated by I/O Management to contain a mask of GPC(s) which received the error.

If an I/O error occurs reading the Mass_Memory_Phase_Table, this mask is updated to indicate that all GPC's received an error and the overlay processing is discontinued.

i. Constraints and Assumptions - The Overlay_Processor assumes that the Function_Base_Phase_Number is valid and that the Program_Overlay_Phase_Number is a valid non-zero phase number.

j. Detailed Implementation -

1. There is no return from this CALL to the Process_Dispatcher (See next note).

2. This entry results from the Overlay_Processor PCT becoming the highest priority ready process (entry is from regular Process_Dispatcher processing).

3. This is accomplished via a set system mask (SSM) instruction.

4. Overlay_Processor returns to SVC_Handler.

5. I/O being issued is described by the description in the preceding blocks and the data descriptor table for the Overlay_I/O_SVC_Parameter_List.

6. The protection and unprotection is done using ZCON type addressing to enable protection/unprotection of any sector.

DATA TABLE 3.3.2.1-1

NAME Overlay_Processor (FCMPOVLY)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Overlay_Parameter _List	S019	I						
2	PSW_Interrupt_Code	&002.26	I		180,242,				
					320				
3	SVC_Old_PSW	&001.67	I		See AppendixE	TPSASOP			
4	PCT_Free_Pool_Address	Q001.9	I	131,132	101,280	TCVTPCTP			
				305	320				
5	PCT_Priority	Q003.2	I	101,105 106	See App. E	TPCTPRI			
			O	E					
6	Active_PCT_Address	Q001.2	I	103	See	TCVTOLD			
					Appendix E				
7	Original_Priority	Q003.26	O	101,105	106	TPCTOPRI			
				106,320	131				
8	PCT_Wait_Indicators	Q003.42	O	See Appendix	See Appendix	TPCTWAIT			
				E	E				
9	Interrupt _PSW_Of_Process	Q003.5	O	See Appendix	See App. E	TPCTPSW			
				E					
10	PSW_Instruction_Address	&002.1			183 320				
11	PSW_System_Mask	&002.12			320				
12	PSW_Register_Set	&002.22		182	320				
13	PSW_Machine_Check_Mask	&002.23			320				

DATA TABLE 3.3.2.1-1 (Cont'd)

NAME Overlay_Processor (FCMPOVLY)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
14	PSW_Run_State_Control	&002.25			180,181 320				
15	PSW_Wait_State_Indicator	&002.24		300	320				
16	PSW_Branch_Sector_Register	&002.10			183				
17	PSW_Data_Sector_Register	&002.11							
18	FCOS_PCT_Indicator	Q003.31	O	320		TPCTFLGS			
19	PCT_Flags	Q003.29	O	101,142	107,133,	TPCTFLGS			
				320	140,225				
20	Event_State	@007.2		See Appendix					
				E					
21	Overlay_Completion_Event_Address	S019.5	I		320	TOPLEVNT			
22	Overlay_I/O_Status_Address	S019.6	I		320	TOPLSTAT			
23	Process_Interrupt_Register_1	Q003.7	O	103 182	103	TPCTGPR1			
			I	320	320				
24	Overlay_Source	S019.1	I			TOPLSORC			
25	Process_Interrupt_Register_2	Q003.8	O	103,182,	103,320	TPCTGPR2			
			I	320					
26	Process_Interrupt_Register_3	Q003.9	O	103,182,	103,320	TPCTGPR3			
			I	320					
27	Process_Interrupt_Register_4	Q003.10	O	103,320	103,320	TPCTGPR4			
			I						

DATA TABLE 3.3.2.1-2 (Cont'd)

NAME Overlay_Processor (FCMPOVLY)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
28	Function_Base_Phase_Number	S019.3	I		320	TOPLFNBS			
29	Process_Interrupt_Register_5	Q003.11	O	103	103	TPCTGPR5			
			I	320	320				
30	Program_Overlay_Phase_Number	S019.4	I		320	TOPLPOVL			
31	Overlay_Buffer_Reservation	Q001.39	I	280	280	TCVTODBR			
			O	305 320	320				
32	Next_To_Execute_PCT_Address	Q001.3	O	See App E	See App E	TCVTNEW			
33	Next_PCT_Address	Q003.1	I	101,131 132	See App E	TPCTNXT			
34	Event_Used_Indicator	@007.1			See App E.				
35	Overlay_I/O_SVC_Parameter List	0004	O	320		FCMOVIOS			
36	Overlay_MM_Address	0004.9	O	320					
37	Mass_Memory_Phase_Table_Address	Q001.82	I		320	TCVTMMPT			
38	Overlay_Word_Count	0004.6	O	320					
39	Overlay_Buffer_Address	0004.7	O	320					
40	Mass_Memory_Phase_Table	0002	I		320	FCMMMPT			
			O						
41	Number_Of_Phase_Table_Records	0002.1	I	280	280				
			O	320	320				

DATA TABLE 3.3.2.1-1 (Cont'd)

NAME Overlay_Processor (FCMPOVLY)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
42	Phase_Information	0002.3	I		320				
43	GPC_Self_Status_Table_Address	Q001.58	I	300	See	TCVTGST			
					Appendix E				
44	Redundant_Set_Mask	I010.13	I	See App. E	See App. E		V91M2070P V91M2104P	X	
						TGSTRSM	V91M2138P V91M2172P		
45	GPC _ID	#001	I	300	See App. E	TFCMID			
46	Index_To_Phase_Segments	0002.4	I		320				
47	Segment_Information	0002.6	I		320	TMMPSTRT			
48	Count-Of-Segments	0002.5	I		320				
49	Segment_Mass_Memory_Address	0002.14	I		320	TMMPMMA			
50	Segment_Main_Memory_Address	0002.7	I		320	TMMPADD			
51	Segment_Address_BSR	0002.11	I		320	TMMPPBD			
52	Overlay_Operation_Code	0004.4	O	320	320				
53	Segment_Length	0002.13	I		320	TMMPLNG			
54	Segment_Protrect_Flag	0002.9	I		320	TMMPPBD			

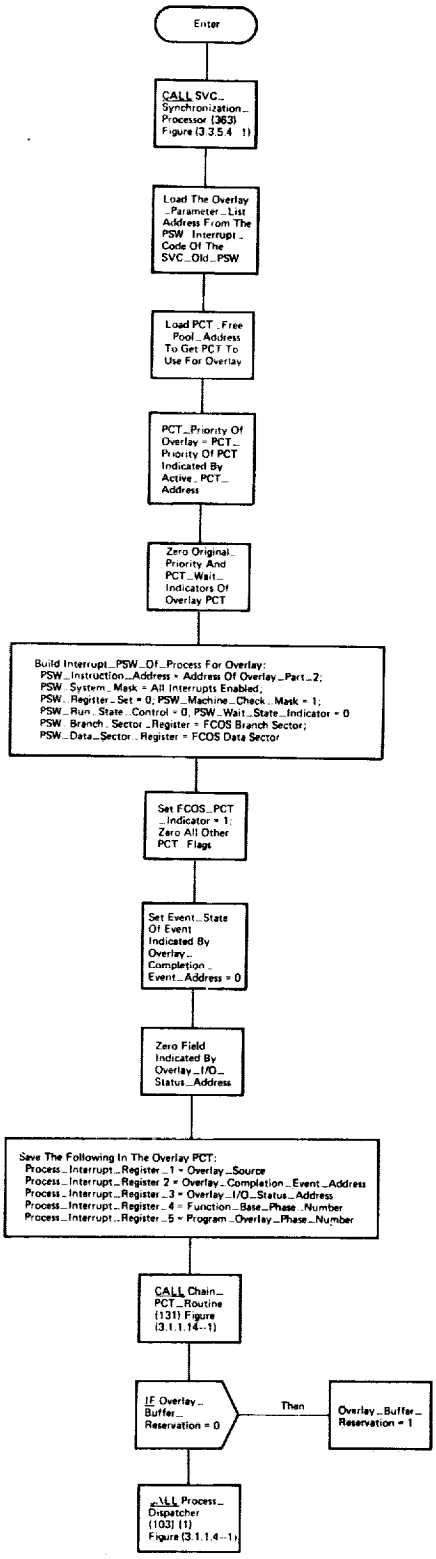


Figure 3.3.2.1-1. Overlay_Processor (FCMPOVLY)

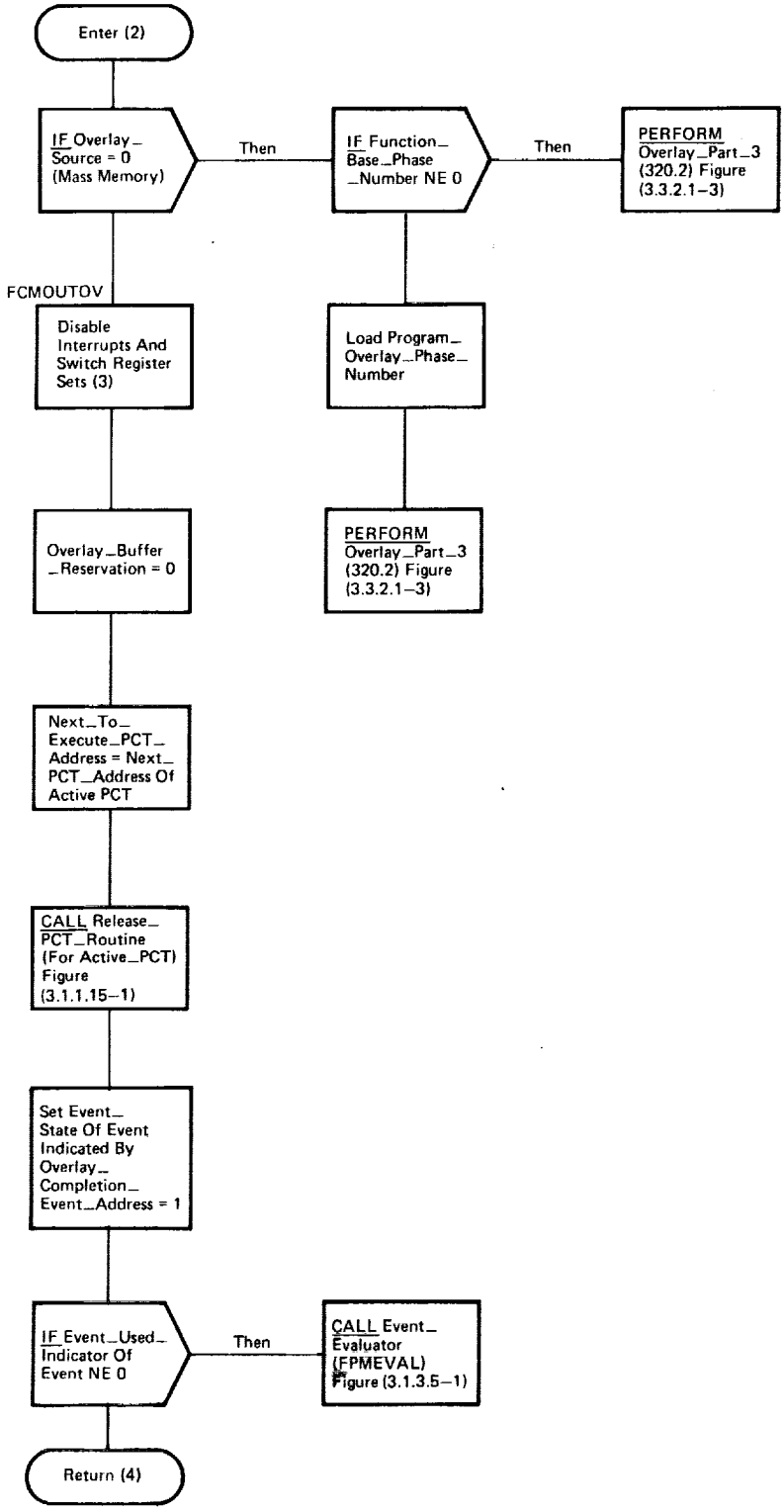


Figure 3.3.2.1-2. Overlay_Processor Overlay_Part_2 (320.1)

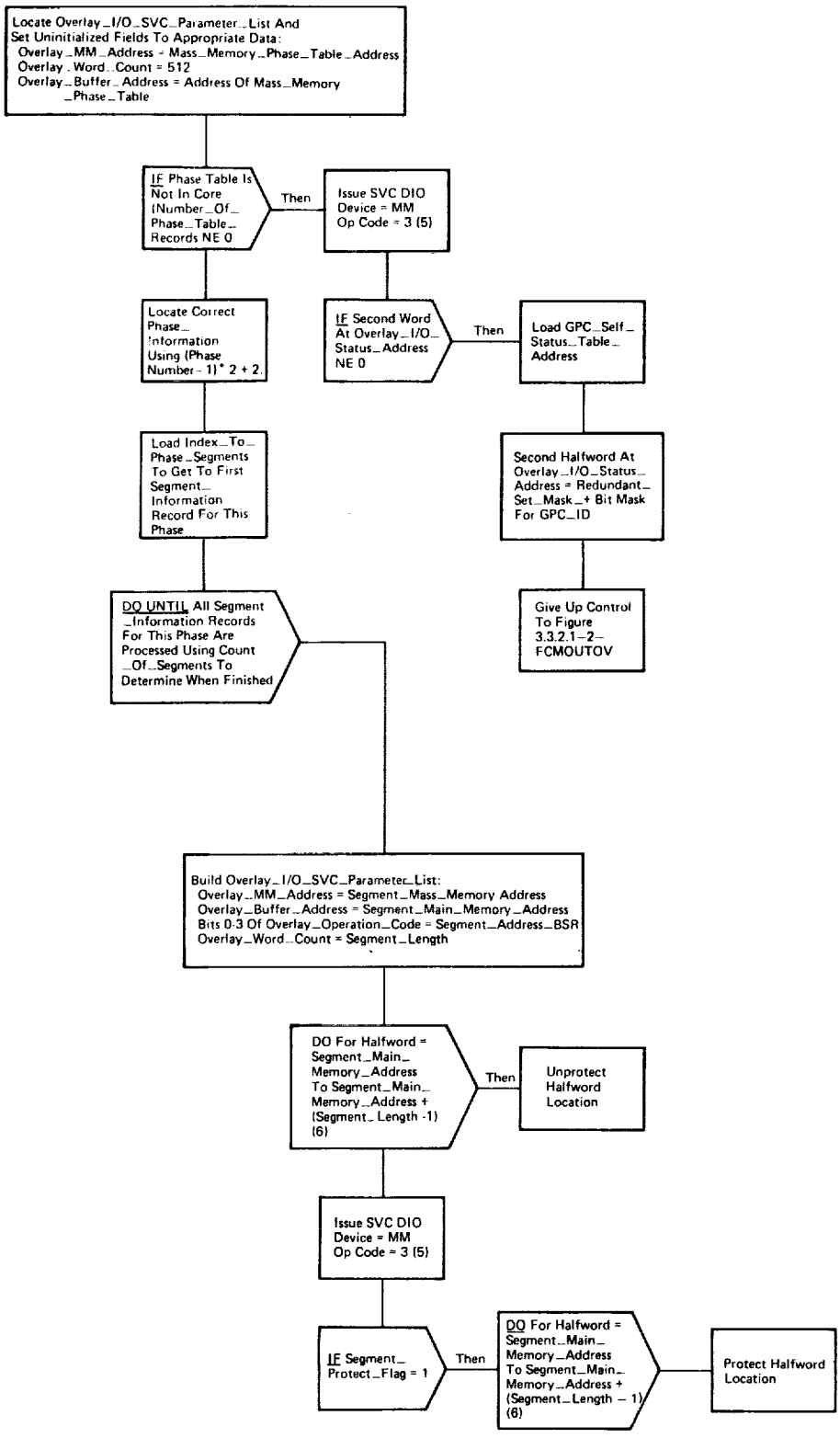


Figure 3.3.2.1-3. Overlay_Processor Overlay_Part_3 (320.2)

3.3.2.2 Program_Modification_Processor (FCMPMOD) (325)

FCMPMOD supports the reading and modification of main memory.

a. Control Interface - CALLed by (100) SVC_Handler (FPMSVC).

b. Input - Register 0 - bits 0-15 contain the address of the Program_Modification_Parameter_List. See Table 3.3.2.2-1.

c. Process Description - Upon entry, the SVC_Synchronization_Processor is CALLed to synchronize all GPC's in the redundant set.

If the Read/Write_Request_Type indicates that this is a read request, the number of halfwords specified by the Program_Modification_Halfword_Count will be moved from the address specified by Main_Memory_Start_Address to a data address specified by Integer_Data_Buffer_Address, Integer_Double_Data_Buffer_Address, Scalar_Data_Buffer_Address, or Scalar_Double_Data_Buffer_Address. The address used is based on one of four indicators being set: Integer_Data, Integer_Double_Data, Scalar_Data, or Scalar_Double_Data. A data address must be specified which corresponds to the indicator that was set.

If the Read/Write_Request_Type indicates that this is a write request, the number of halfwords specified by the Program_Modification_Halfword_Count will be moved one halfword at a time from one of the four addresses mentioned in the preceeding paragraph (depending upon one of the four above mentioned indicators being set) to the address specified by Main_Memory_Start_Address. If the Set_Storage_Protect indicator is set, each halfword will be unprotected prior to the move and then protected after the move.

Lastly, Protect_Interrupt is reset to zero. This is done as an indication that all the data movement has taken place and that no storage protect or other type of error occurred.

d. Outputs - Modified main memory on write requests. See Table 3.3.2.2-1.

e. Module References - (363) SVC_Synchronization_Procesor (FCMSSYNC) is CALLed.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions -

1. The modification or reading of main memory across a sector is prohibited.

2. When modifying main memory, the protect status of the main memory to be modified must be specified. If (and only if) the user specifies the main memory is to be protected after the modification, FCOS will unprotect the main memory prior to the modification and then re-protect the main memory after the modification has taken place. If the user attempts to modify protected main memory and specifies the main memory locations as beirg unprotected, a store protect interrupt will occur. Note that while the protect status of main memory may change from unprotected to protected, protected main memory may not be unprotected using this facility. It is the user's responsibility to verify the validity of the change and to maintain system integrity.

j. Detailed Implementation -

1. The proper data buffer is determined based on one of four indicators being set: Integer_Data, Integer_Double_Data, Scalar_Data, or Scalar_Double_Data. The indicator set will cause data movement to or from a data buffer pointed to by Integer_Data_Buffer_Address, Integer_Double_Data_Buffer_Address, Scalar_Data_Buffer_Address, or Scalar_Double_Data_Buffer_Address. Indexing is used for proper data movement.

DATA TABLE 3.3.2.2-1

NAME Program_Modification_Processor (FCMPMOD)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Main_Memory_Start_Address	S026.12	I		325	TMODADDR			
2	Integer_Data_Buffer_Address	S026.13	I		325	TMODATI			
3	Program_Modification_Flag_Field	S026.1	I		325	TMODFLGS			
4	Program_Modification_Halfword_Count	S026.11	I		325	TMODHWDS			
5	Read/Write_Request_Type	S026.8	I		325	TMODREAD			
6	Set_Storage_Protect	S026.9	I		325	TMODSSP			
7	Status_Flags	I300	I	See App. E	325,620, 830,910	CZ2BFLAG			
8	Protect_Interrupt	I300.	O	325,910	910				
9	Integer_Double_Data_Buffer_Address	S026.14	I		325	TMODATID			
10	Scalar _Data_Buffer_Address	S026.15	I		325	TMODATS			
11	Scalar_Double_Data_Buffer_Address	S026.16	I		325	TMODATSD			
12	Integer_Data	S026.3	I		325	TMODINT			
13	Integer_Double_Data	S026.4	I		325	TMODINTD			
14	Scalar_Data	S026.5	I		325	TMODSLR			
15	Scalar_Double_Data	S026.6	I		325	TMODSLRD			

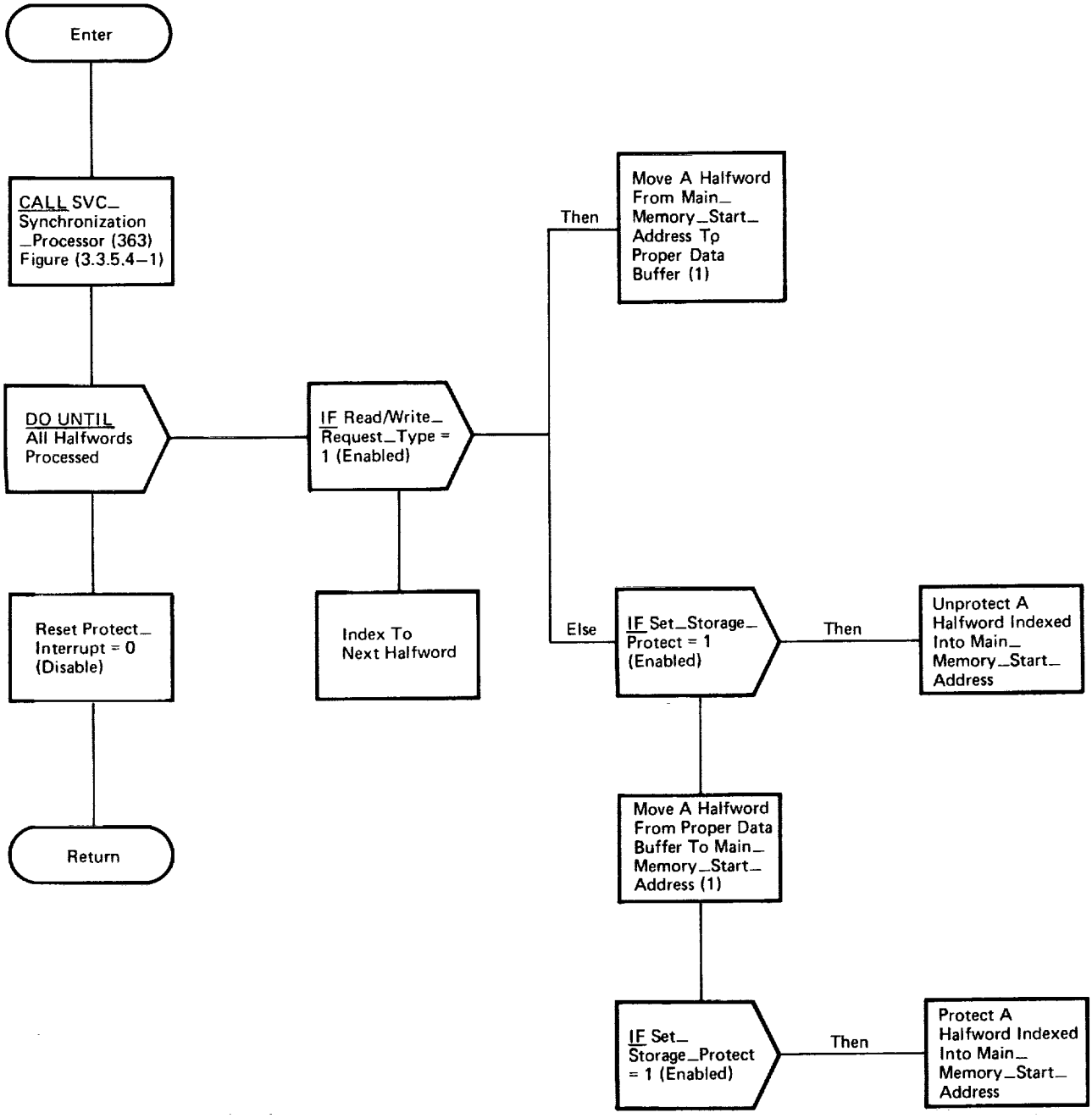


Figure 3.3.2.2-1. Program_Modification_Processor (FCMPMOD)

3.3.2.3 BCE_Element_Bypass_Processor (FCMBCEMD) (330)

FCMBCEMD supports the bypassing of certain BCE elements and the ability to restore them to their original state.

a. Control Interface -

1. CALLed by (100) SVC_Handler (FPMSVC)

2. CALLed by (244) Level_C_I/O_Error_Interrupt_Handler

b. Input - Register 0 - bits 0-15 contain the address of the BCE_Element_Bypass_Parameter_List or a number corresponding to the BCE element to be bypassed. See Table 3.3.2.3-1.

c. Process Description - When CALLed by the SVC_Handler, the SVC_Synchronization_Processor is CALLed to synchronize all GPC's in the redundant set. If BCE_Element_Number is zero, then all BCE elements are restored to their original state. The Transaction_Counter_Table is zeroed as well.

If BCE_Element_Number is non-zero, then an individual BCE chain is restored. A pointer to the first BCE_Element_Modification_Table entry to be used is obtained plus the number of elements to be restored.

The restoration of a BCE element involves overlaying the contents at the address pointed to by BCE_Element_Address with the instruction that was originally a part of the BCE element. This instruction is picked up from Original_BCE_Code. Also, appropriate BCE_Element_Bypass_Indicator_1 or BCE_Element_Bypass_Indicator_2 bits are reset.

When the BCE_Element_Bypass_Processor is entered at entry point FCMBYPAS, a given BCE element is bypassed. This involves overlaying an instruction pointed to by BCE_Element_Address with a set of delay instructions located at BCE_Delay_Instructions as well as setting the appropriate BCE_Element_Bypass_Indicator_1 or BCE_Element_Bypass_Indicator_2 bit.

The control flow for this module is presented in Figure 3.3.2.3-1.

d. Outputs - Modified BCE chains. See Table 3.3.2.3-1.

e. Module References - (363) SVC_Synchronization_Processor (FCMSSYNC) is called.

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation -

1. The FCMBYPAS entry point is used by the Level_C_I/O_Error_Interrupt_ Handler to bypass a BCE element.

DATA TABLE 3.3.2.3-1

NAME BCE_Element_Bypass_Processor (FCMBCEMD)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	BCE_Element_Number	S037.2	I		330				
2	BCE_Element_Bypass_Indicator_1	#030	O	220,244, 330,355	220,244 330	TFCMBCEBI/FIOBCEBI	V91M2252P	X	
3	BCE_Element_Bypass_Indicator_2	#032	O	220,244, 330	220,244 330	TFCMBCBI2/FIOBCBI2	V91M2269P	X	
4	Transaction_Counter_Table	Y029	O	220,244, 330	220,244, 330,355	FIOTCT			
5	BCE_Element_Modification_Table	0006	I		330	FCMBMT			
6	FMPT_Header_Bits	T070	O	See App. E		CDLBERRM			
7	BCE_Element_Address	0006.1	I		330				
8	BCE_Delay_Instructions	0006.2	I		330				
9	Original_BCE_Code	0006.3	I		330				

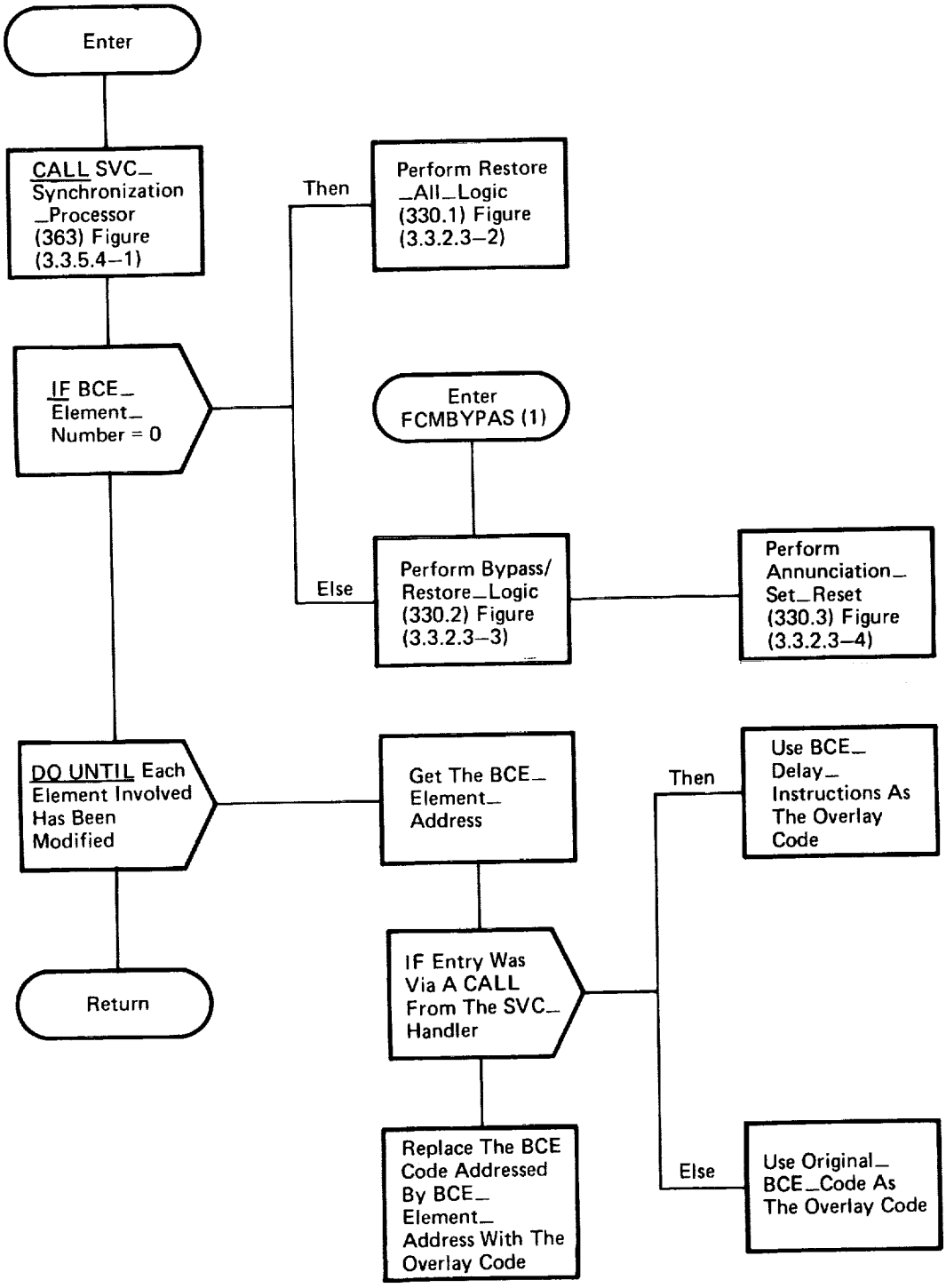


Figure 3.3.2.3-1. BCE_Element_Bypass_Processor (FCMBCEMD)

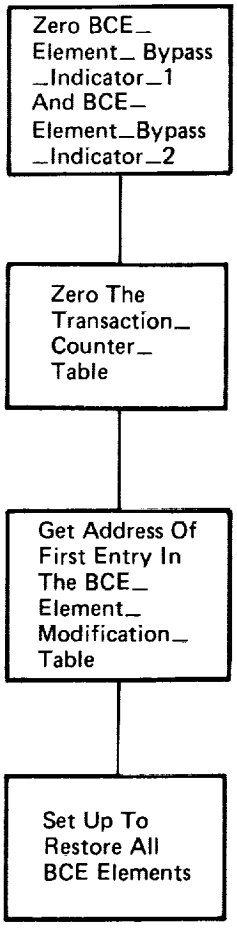


Figure 3.3.2.3-2. BCE_Element_Bypass_Processor Restore_All_Logic (330.1)

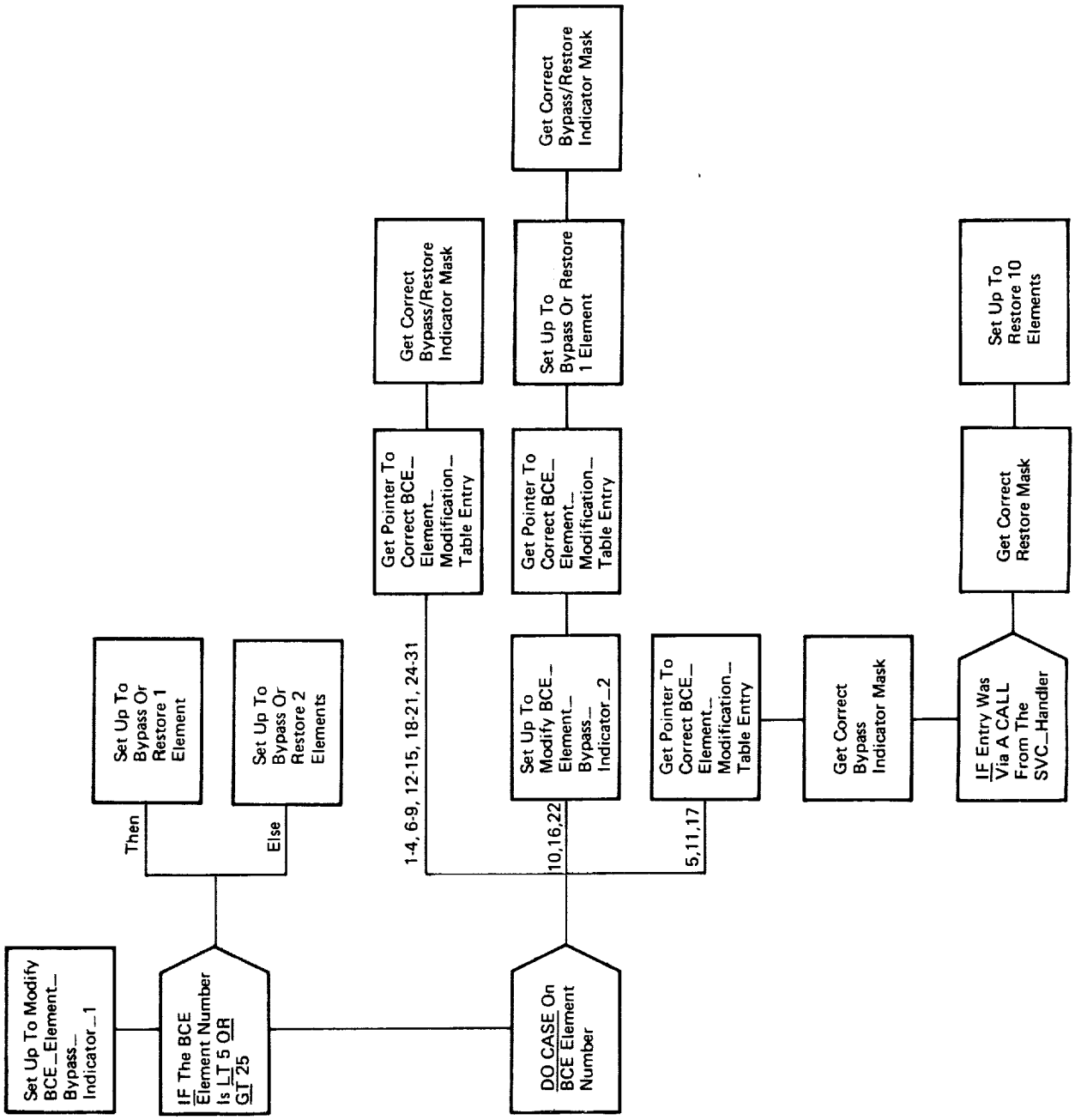


Figure 3.3.2.2-3. Bypass/Restore_Logic

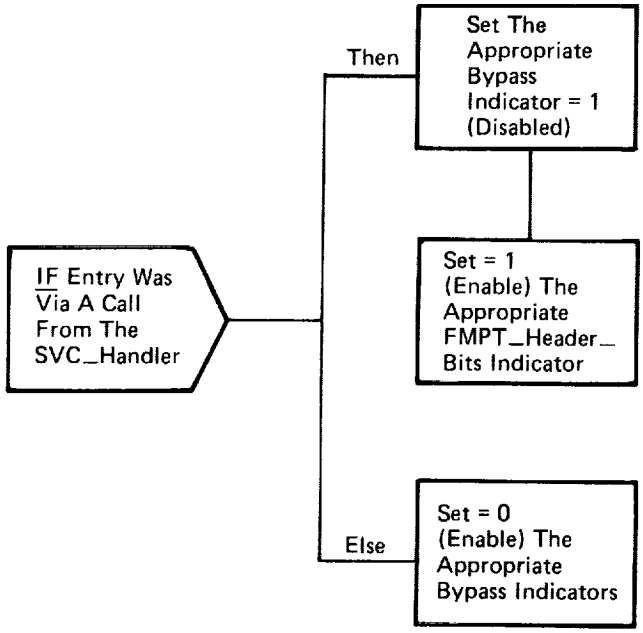


Figure 3.3.2.3-4. BCE_Element_Bypass_Processor Annunciation_Set_Reset (330.3)

3.3.3 Miscellaneous_CM_Request_Processor (FCMSVC) (340)

The Miscellaneous_CM_Request_Processor (FCMSVC) is invoked to service requests for the reading or writing of certain discretes, the setting of the Computer Annunciation Matrix (CAM) lights, or the setting/resetting of the termination control latches.

a. Control Interface - CALLed by (100) SVC_Handling_Routine (FPMSVC).

b. Input - Register 0-bits 0-15 contains the address of the Configuration_Management_SVC_List. See Table 3.3.3-1 for other input to this module.

c. Process Description - When a Configuration Management request SVC is issued, the Miscellaneous_CM_Request_Processor receives control from the SVC_Handling_Routine. It then determines which of the three request types has been made.

1. The following types of requests (CM_SVC_Request=1) to read or write discretes will be accepted.

CM_SVC_OPERATION	OPERATION DESCRIPTION
1	Redundancy management of discrete Register A
2	The RM_Status_Register will be read.
3	Discretes will be set according to the mask input by the program making the request.
4	Discretes will be reset according to the mask input by the program making the request.

2. Two types of requests (CM_SVC_Request=2) for CAM status light changes will be accepted.

CM_SVC_OPERATION	OPERATION DESCRIPTION
1	The fail vote lights can be turned off or on by setting and resetting the fail vote discretes. The mask passed to the program will be used to determine which lights should be turned on. Then the MSC_Set_Fail_Discretes_Program will be started to change the status of the lights.
2	The GPC's fail light can be turned on by loading the test register to similate fail votes by other GPC's. The mask passed to the program will be used to determine which test fail votes to set.

3. The following types of requests (CM_SVC_Request=3) to set or reset termination latches will be accepted.

CM_SVC_OPERATION	OPERATION DESCRIPTION
1	The timeout or voter termination control latches can be set.
2	The timeout or voter termination control latches can be reset.

A software image of the latches, Termination_Control_Patch_Set_Mask, will be kept to allow independent setting/resetting of the latches.

The flow for this module is shown in Figure 3.3.3-1.

d. Output - Outputs from this module are specified in Table 3.3.3-1.

e. Module References - 1. (376) Fail_Discretes_MSC_Processor (FCMSVOTE) is invoked as a result of (3) below.

2. (380) GPC_Discrete_Redundancy_Manager is called. (FCMDSCRM)

3. (210) Start_MSC_Processor (FIOSTMSC) is CALLed.

f. Module Attributes - Program

g. Template References - None

h. Error Checks - None

i. Constraints and Assumptions - None

j. Detailed Implementation

1. Issue the program controlled output command - DISCRETE OUTPUT (SET)

2. Issue the program controlled output command - DISCRETE OUTPUT (RESET)

3. Issue the program controlled output command - LOAD TEST REGISTER

4. Termination_Control_Latch_Set_Mask=LOGICAL SUM of the Termination_Control_Latch_Set_Mask and the Termination_ Control_Latch_Mask.

5. Termination_Control_Latch_Set_Mask=LOGICAL SUM of the Termination_Control_Latch_Set_Mask and the Termination_ Control_Latch_Mask.

   Termination_Control_Latch_Set_Mask=the MODUL02 SUM of the Termination_Control_Latch_Set_Mask and the Termination_ Control_Latch_Mask.

6. Issue the program controlled output command - CONFIGURE TERMINATION CONTROL LATCHES

7. Issue the program controlled input command - READ RM STATUS REGISTER

DATA TABLE 3.3.3-1

NAME Miscellaneous_CM_Request_Processor (FCMSVC)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Configuration_Management_SVC_List	S025	I		340	TFCMS			
2	CM_SVC_Request	S025.2	I		340	TFCMSID			
3	CM_SVC_Operation	S025.3	I		340	TFCMOP			
4	CM_SVC_Buffer	S025.4	I		340	TFCMSBUF			
5	Discrete_Output_Mask	S025.20	I		340	TFCMSBUF			
6	Fail_Discrete_Output_Mask	S025.5	I		340	TFCMSBUF			
7	Test_Voter_Discretes_Input_Mask	S025.11	I		340	TFCMSBUF			
8	Termination_Control_Latch_Mask	S025.17	I		340	TFCMSBUF			
9	Termination_Control_Latch_Set_Mask	#005	O		340	TFCMTRML			
10	RM_Status_Register	&006.07	R/O		340				
11	Set_CAM_Mask	Q001.41	O	340,375	376	TCVTCAMS			
12	Reset_CAM_Mask	Q001.42	O	340,375	376	TCVTCAMR			
13	Discrete_Input_Register_A	&005	R	300,340	300,340				

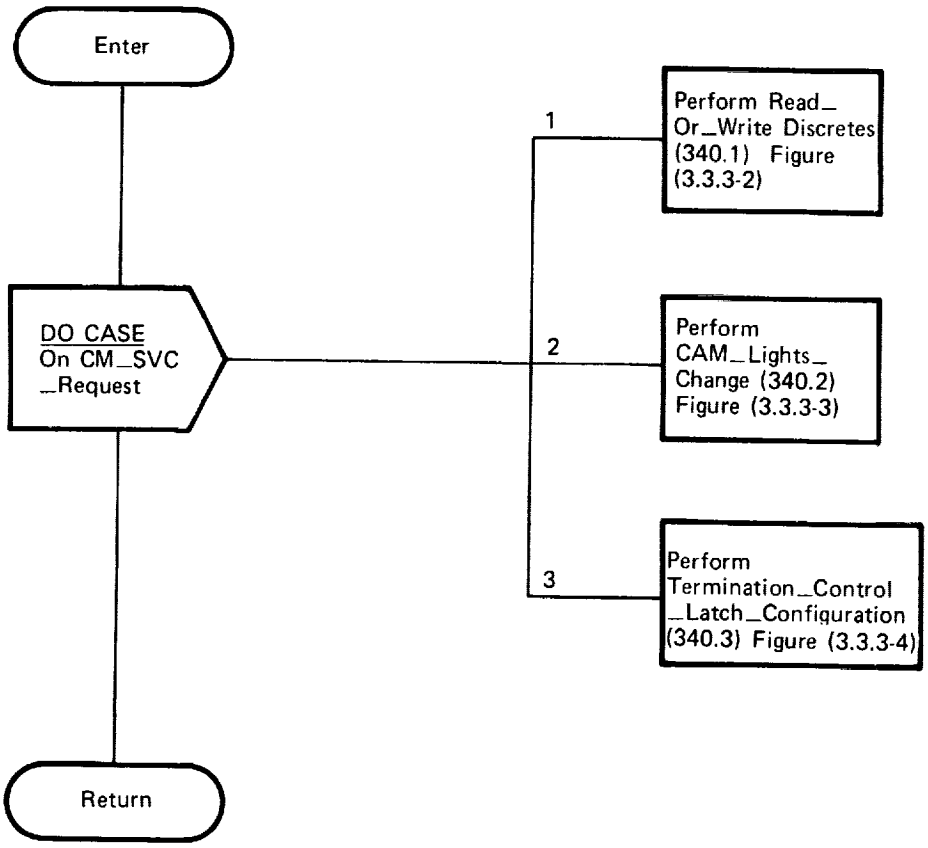


Figure 3.3.3-1. Miscellaneous_CM_Request_Processor (FCMSVC)

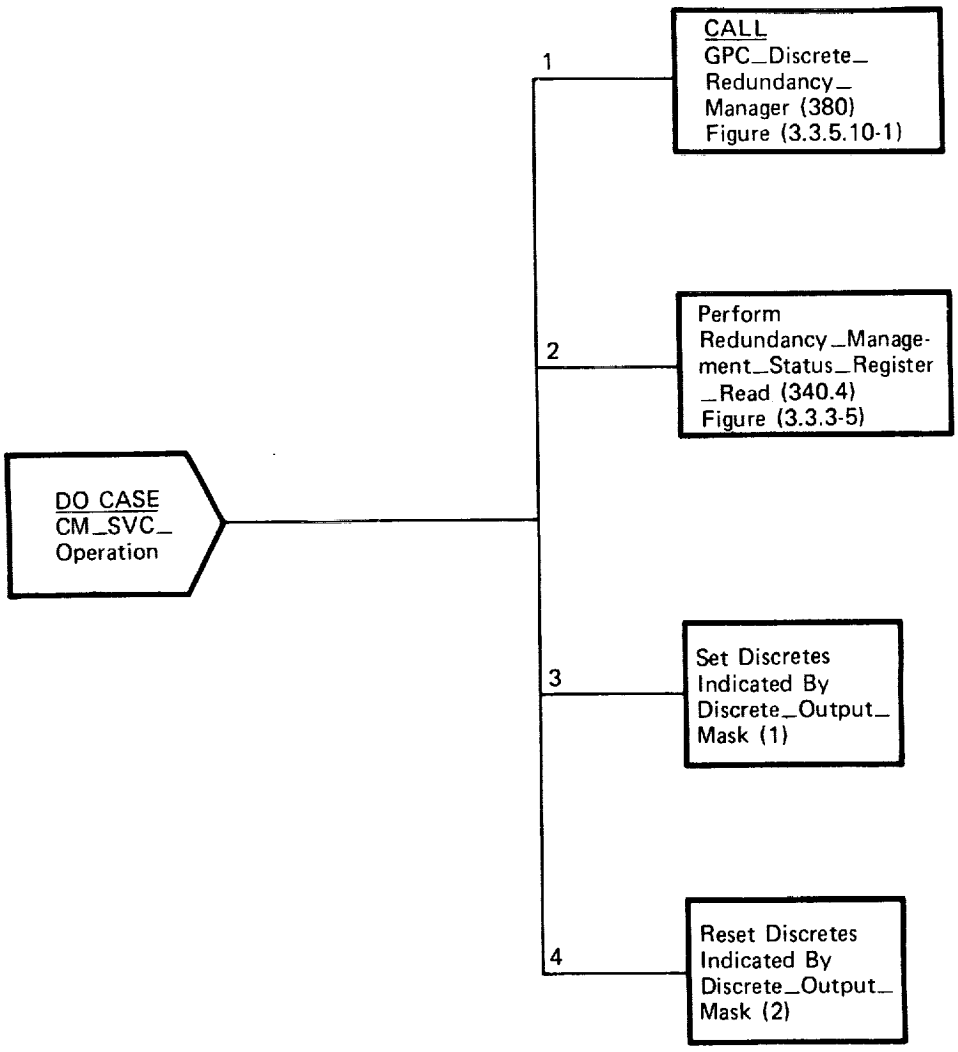


Figure 3.3.3-2. Miscellaneous_CM_Request_Processor Read_Or_Write_Discretes (340.1)

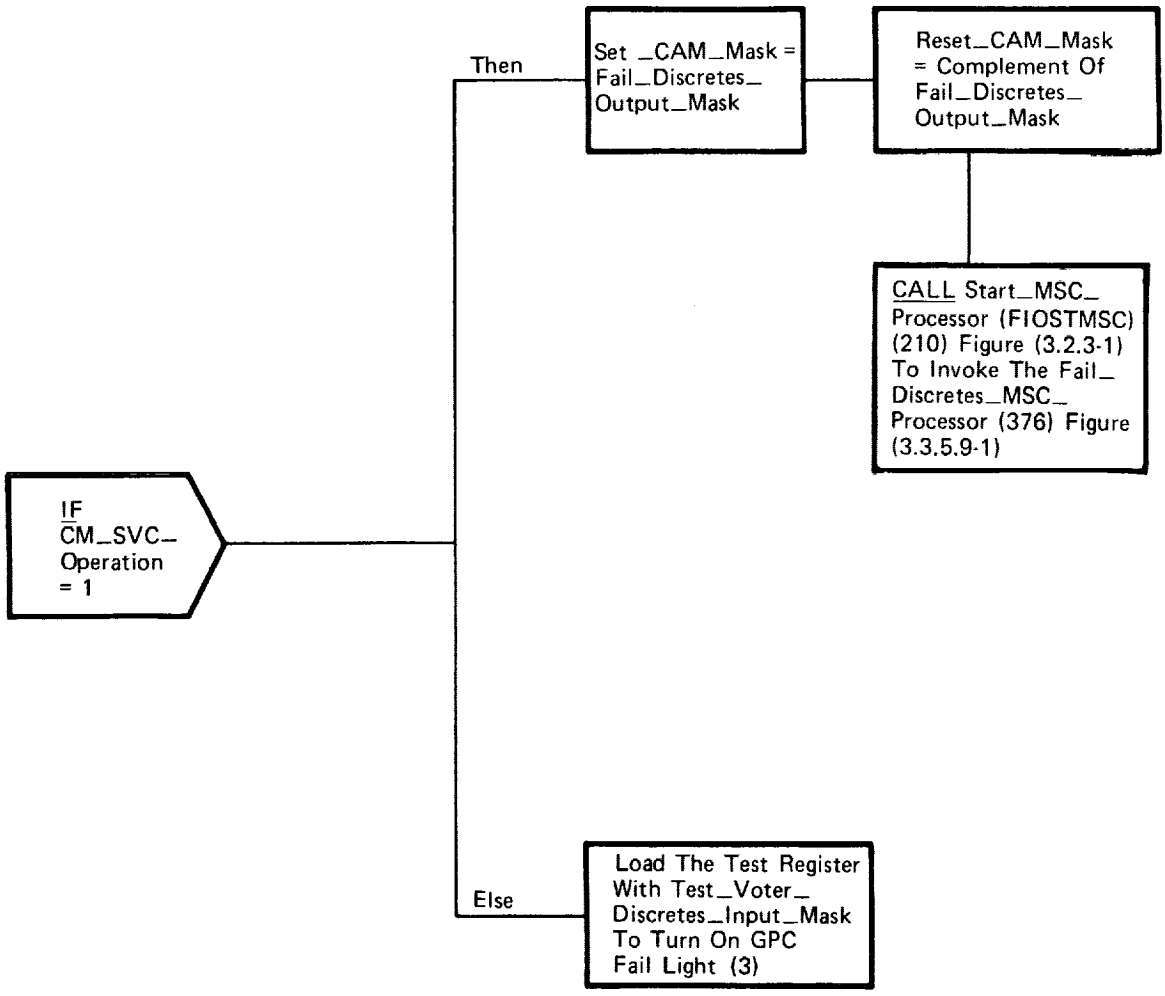


Figure 3.3.3-3. Miscellaneous_CM_Request_Processor CAM_Lights_Change (340.2)

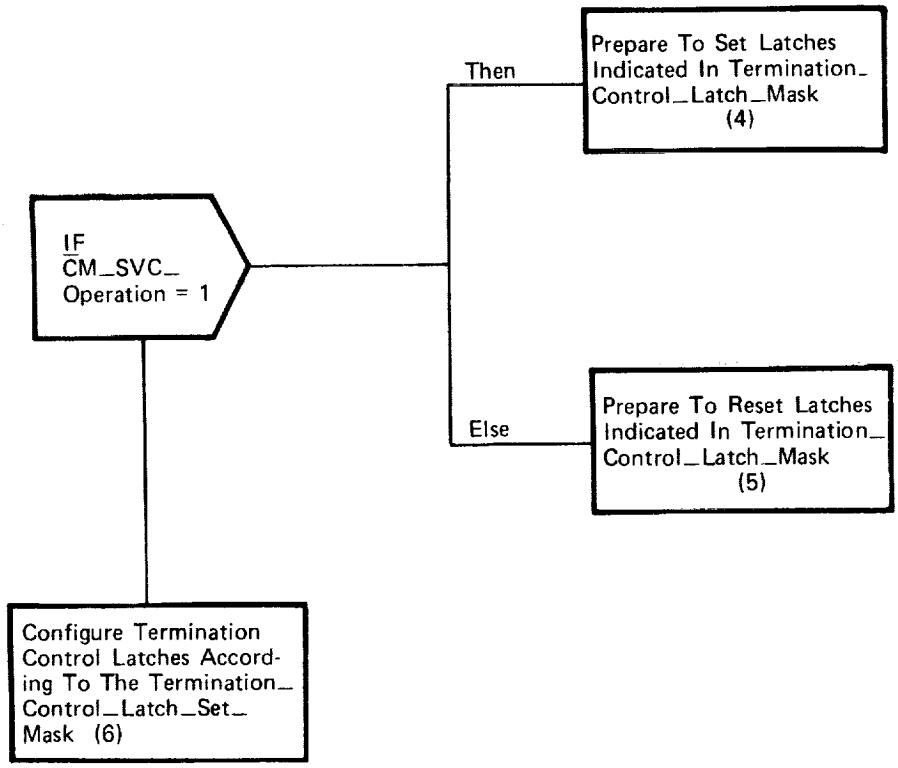


Figure 3.3.3-4. Miscellaneous_CM_Request_Processor Termination_Control_Latch_Configuration (340.3)

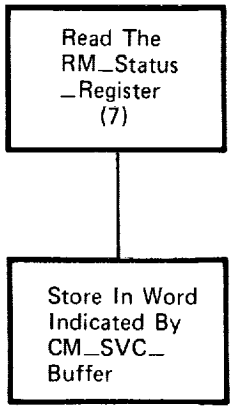


Figure 3.3.3-5. Miscellaneous_CM_Request_Processor Redundancy_Management_Status_Register_Read (340.4)

3.3.4 Bus Configuration

The FCOS provides a set of services for the configuration of the 24 data buses (C-Buses) which interface the Input/Output Processor (IOP) with the Interface Units (IU). The control logic for invoking these services resides in the DPS Reconfiguration portion of System Control.

Configuration of the C-Buses is defined to be controlling the states (enabled or disabled) of the receivers and transmitters associated with each of the Bus Control Elements (BCE) contained within the IOP.

The configurations of the receiver and transmitter associated with each BCE are:

Receiver Enabled	Transmitter Enabled	Mode
Yes	Yes	Command
Yes	No	Listen
No	No	Disabled

Generally, bus configuration is only concerned with enabling and disabling transmitters (changing a BCE status from command to listen mode and vice versa). However, facilities are provided to allow configuring a C-Bus to the disabled mode, or enabling a receiver (configuring a C-Bus from disabled mode to listen mode).

The FCOS also provides a set of services for the configuration of buses associated with logical strings. These services are associated with requests made via the COMPUTER/BUS KEY for string release/assignement/reassignment/moding. These requests are interrogated by the System Control Bus_Configuration_Change (Vol. 2, Part 3, Section 3.2.3.2) which issues the appropriate service request(s) to FCOS. (For ALT, string reassignment/release is not supported by System Control except for nominal string assignment).

For most bus/string reconfiguration requests, System Control coordinates the bus handover at SSIP time. Cooperation is assumed upon successful SSIP sync. For cases of redundant set reconfiguration of buses which are not coordinated by SSIP, sync processing is invoked by FCOS to coordinate the handover.

OPS transition bus configuration is performed in one of two ways. If the active major function is a redundant set, nominal string assignment tables are used to configure the buses upon request from System Control. In cases where nominal string assignment tables are not applicable, System Control determines the assignments to be made and issues the appropriate bus and/or string assignment requests.

FCOS Bus Configuration also provides bus/data path mask management associated with bus/string configurations. These services may result from either direct requests from applications/System Control or from requests implied in other bus configuration requests.

FCOS Bus Configuration also includes processing for bus masking and GPC prime switching (if necessary and enabled) resulting from forced fail to sync and fail to sync. Prime switching resulting from string reconfiguration requests are also included in the processing of those requests.

Prime switching implies switching of the functions allocated to the prime GPC (CPDS Section 4.6.4.5.1). The GPC which will perform the prime function is determined by which GPC is in command of the prime buses (buses needed to perform the prime functions), by the Prime GPC ID field in the GPC Status Table, or by both. As a result, prime switching is accomplished by updating the Prime GPC ID field and by reassigning the prime buses. Since string 5 is maintained in such a way as to always reflect which buses are prime buses, reassignment of string 5 is synonomous with reassignment of prime buses.

To perform these functions FCOS and System Control maintain various tables. System Control maintains the Nominal String Assignments Tables used for redundant set ops transitions. FCOS maintains the Current Configuration Tables which show transmitter and receiver status for each GPC. There are two types of Current Configuration Tables: One shows the current status that should exist based on reconfiguration requests which have been serviced (Requested Current Configuration Table). The other indicates the actual hardware status of the transmitters (True Current Configuration Table). Other bus configuration tables include the Current String Assignment Table, String Definition Tables indicating the buses associated with each string, and the String Mode Indicator showing the current moding of each string. In addition to bus configuration tables, various GPC status information and masks are used.

There also exists a first in-first out Bus Reconfiguration Queue. The Bus_Reconfiguration_Queue_Element (BRQE) contains the information from the parameter list which is needed to service the request. Nominally, this queue consists of zero or one queue element. It exists primarily as a place to store parameter list information when it is necessary to wait for I/O to quiesce on the buses being reconfigured. However, it also serves to queue bus reconfiguration requests when a request is made before a previous request(s) has been serviced.

The functional description of FCOS Bus Configuration presented in the following paragraphs is divided into three areas:

a. Bus_Reconfiguration_Pre-Processor - Performs necessary pre-processing for bus reconfiguration requests.

b. Bus_Reconfiguration_Processor - Services bus reconfiguration requests.

c. Bus/Data Path Mask Management - The setting/resetting of bus and/or data path masks.

Refer to Figure 3.3.4-1 for a hierarchy diagram of FCOS Bus Reconfiguration.

Note that there is no bus configuration initialization processor. The only bus configuration performed automatically by FCOS at initialization time is the enabling of all receivers (except buses 5 and 9 for ALT). All other bus configuration needed at initialization time is performed as a result of bus configuration requests from System Control.

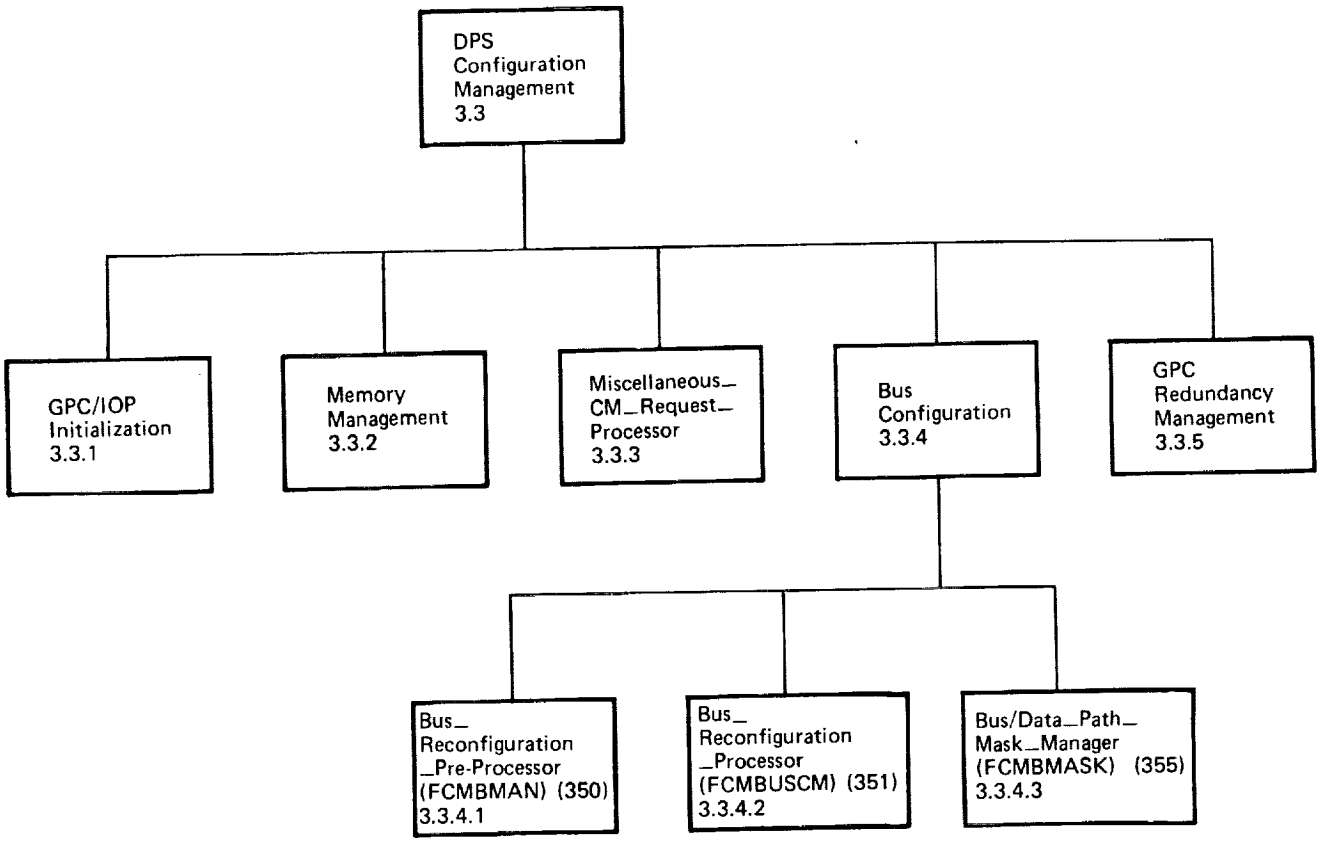


Figure 3.3.4-1. Bus Configuration Hiearchy Diagram

3.3.4.1 Bus_Reconfiguration_Pre-Processor (FCMBMAN)(350)

FCMBMAN performs necessary pre-processing for bus reconfiguration requests.

a. Control Interface -

1. CALLed by (100) SVC_Handler (FPMSVC)
2. CALLed by (220) I/O_Completion_Processor (FIOCMPLT)
3. CALLed by (280) Mass_Memory_Manager (FIOMMMGR)
4. CALLed by (368) Sync_Fail_Processor (FCMSFAIL)

b. Input - Register 0 contains the address of the Bus_Reconfiguration_Parameter_List or zero if invocation is from I/O Completion_Processor or Mass_Memory _Manager. See Table 3.3.4.1-1.

c. Process Description - The Bus_Reconfiguration_Pre-Processor does the following when entered from the SVC_Handler or Sync_Fail_Processor to service a new bus reconfiguration request:

The SVC_Synchronization_Processor is invoked to perform a redundant set sync (except if entry is from the Sync_Fail_Processor). When the LDB_Toggling_Indicator is non-zero, the purpose of this sync is to coordinate the request since an LDB toggling request is a redundant set operation which has not been coordinated by SSIP. In other cases the purpose of the call is to sync redundant GPC's before processing the request. After sync has been performed, the information from the Bus_Reconfiguration_Parameter_List is placed in the BRQE denoted by the BRQE_Free_Pool_Address. If the LDB_Toggling_Indicator is on and a sync failure occurred, bit 0 of the BRQE_Type field is set to indicate that additional processing may be necessary when preparing to service the request (i.e., prime switching may have resulted from the sync failure which would cause a change in the LDB commander).

If there is already an outstanding BRQE (Top_BRQE_Address is not zero), the new BRQE is placed on the bus reconfiguration queue. Otherwise, the request is processed in the following manner.

If sync failure occurred for LDB toggling, the BRQF_Type field is modified if necessary as a result of LDB commander change. The working mask is then built representing the bus(es) which are affected by the request and placed in the Mask_Of Buses_Being_Reconfigured field of the Bus_Management_Work_Area. If all buses indicated in the Mask_Of_Buses_Being_Reconfigured are idle, the Bus_Reconfiguration_Processor is invoked to process the request. Otherwise, Bus_Reconfiguration_Busy/Idle_Indicators are set for the affected buses which are not idle.

The Bus_Reconfiguration_Busy/Idle_Indicators is used by I/O Management to determine when to invoke the Bus_Reconfiguration_Pre-Processor to continue servicing the request (Each time I/O is completed on a bus in this mask, I/O Management removes the indicator for this bus from the Bus_Reconfiguration_Busy/Idle_Indicators instead of marking the bus idle in the BCE_Busy/Idle_Indicators. When all Bus_Reconfiguration_Busy/Idle_Indicators are zero, the Bus_Reconfiguration_Pre-Processor is invoked by I/O Management.). The BCE_Busy/Idle_Indicators are set for all buses in the Mask_Of_Buses_Being_Reconfigured (this is to prevent I/O Management from starting any I/O on the affected buses until after the reconfiguration request has been serviced). When it is necessary to wait for I/O to quiesce before servicing the request, the BRQE is placed on the bus reconfiguration queue. If the request was completely serviced and prime switching is necessary as a result, this processing is repeated for string 5 reassignment.

The Bus_Reconfiguration_Pre-Processor does the following when entered from the I/O_Completion_Processor or the Mass_Memory_Manager:

The Bus_Reconfiguration_Processor is invoked to service the BRQE indicated by the Top_BRQE_Address. The BRQE is then removed from the queue if prime switching is not necessary as a result of processing the request. If prime switching is necessary the BRQE_Type would have been altered to reflect a string 5 reassignment request. If there is another bus reconfiguration request on the queue, the following occurs:

The Mask_Of_Buses_Being_Reconfigured is updated with the correct buses for the request. The special LDB processing is done if necessary. If all buses needed are idle, the Bus_Reconfiguration_Processor is invoked to process the request. Otherwise, the Bus_Reconfiguration_Busy/Idle_Indicators are set for the affected buses which are not idle. If the request has been completely serviced, the BRQE is removed from the queue and the processing is repeated if there is another request. If it is necessary to wait for I/O before servicing the request, the Bus_Reconfiguration_Pre-Processor returns to the calling program.

The control flow for this module is presented in figures 3.3.4.1-1 and 3.3.4.1-2.

d. Output - See table 3.3.4.1-1.

e. Module References -

1. (363) SVC_Synchronization (FCMSSYNC) is CALLed
2. (351) Bus_Reconfiguration_Processor (FCMBUSCM) is CALLed

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - N/A

DATA TABLE 3.3.4.1-1

NAME Bus_Reconfiguration_Pre_Processor (FCMBMAN)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Bus_Reconfiguration_Parameter_List	S036			350,368				
2	LDB_Toggling_Indicator	S036.3	I O		350,368	TBRPSTAT			
3	BRQE_Free_Pool_Address	Q001.46	I O	305 350	350	TCVTBRQF			
4	BRQE_Type	Q008.2	O	350	350	TBRQETYP			
			I	351	351				
5	Bus_Reconfiguration_Request_Type	S036.1	I		350,368	TBRPTYPE			
6	Request_Variable_1	Q008.3	O	350	350	TBRQEV1			
			I	351	351				
7	Reconfiguration_Variable_1_Field	S036.4	I		350,368	TBRPVAR1			
8	Request_Variable_2	Q008.4	O	350	350	TBRQEV 2			
			I	351	351				
9	Reconfiguration_Variable_2_Field	S036.5	I		350,368	TBRPVAR2			
10	Top_BRQE_Address	Q001.45	I	305	350	TCVTBRQ			
			O	350					
11	Next_BRQE_Address	Q008.1	I	350	350	TBRQENXT			
			O						
12	Bus_Reconfiguration_Busy/Idle_Indicators	Q001.44	O	220 280	200	TCVTSBIS			
				305 350	280				
13	Mask_Of_Buses_Being_Reconfigured	Ø003.1	O	350	350	TBMWMBBR			
			I		351				

DATA TABLE 3.3.4.1-1 (Cont'd)

NAME Bus_Reconfiguration_Pre_Processor (FCMBMAN)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
14	BCE_Busy/Idle_Indicators	Q001.17	I	See App. E		TCVTBCEB			
			0						
15	Requested_I/O_Transmitter_Mask	#003	I	351,	205 250	TFCMXMSK			
				355	350 351				
16	GPC_Prime_ID	I050	I	351 700	See App. E	CZ2VGPCP			
				820 880					
17	String_Definition_Table	I070	I	351	350,355 485	CZ2BSDT			
18	Current_String_Assignment_Table	I080	I	351	350,392	CZ2VSAT	V91X1721X		X
				368	830		V91X1722X		
							V91X1723X		
							V91X1724X		
							V91X1725X		

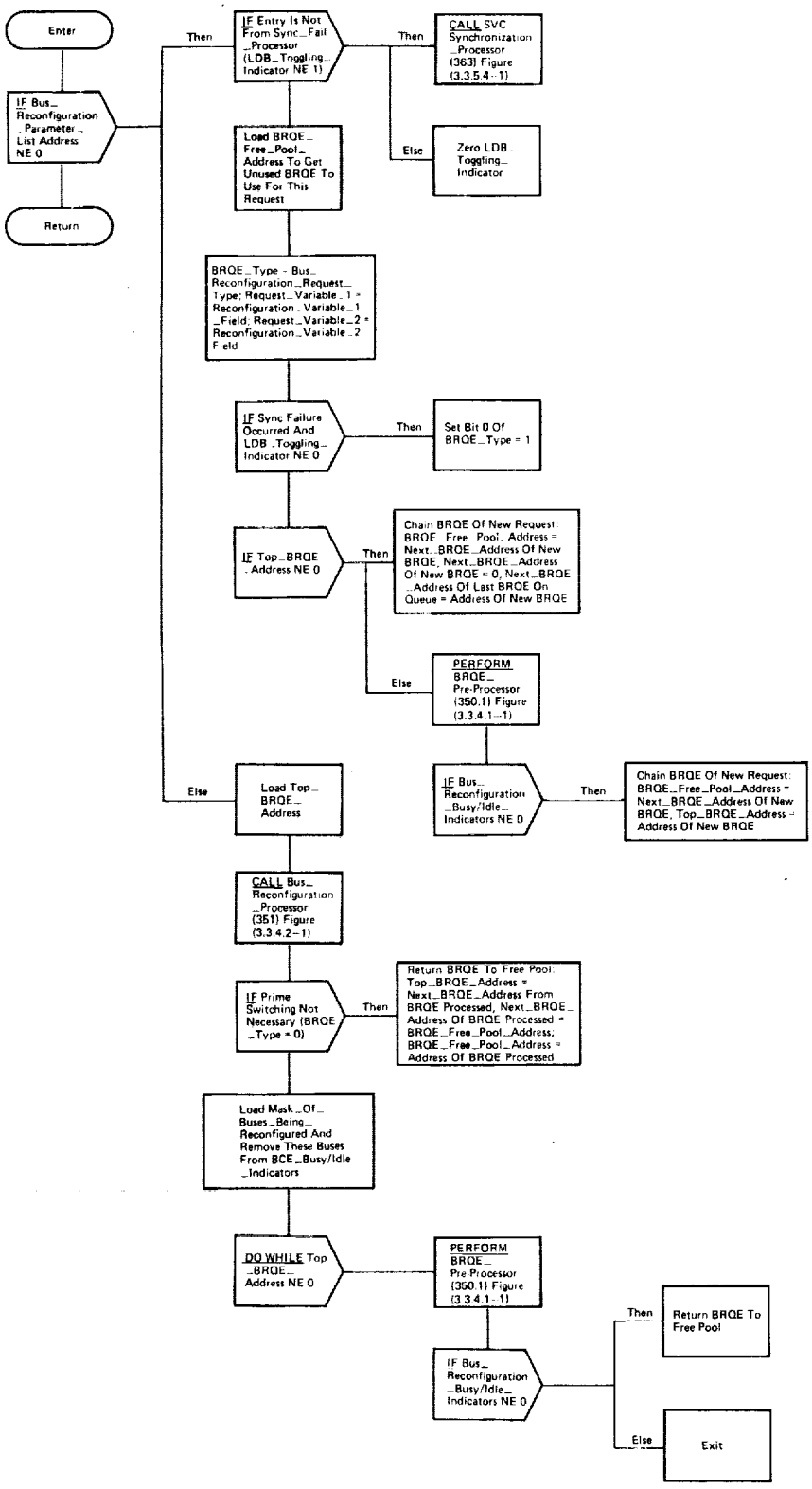


Figure 3.3.4.1-1. Bus_Reconfiguration_Pre_Processor (FCMBMAN)

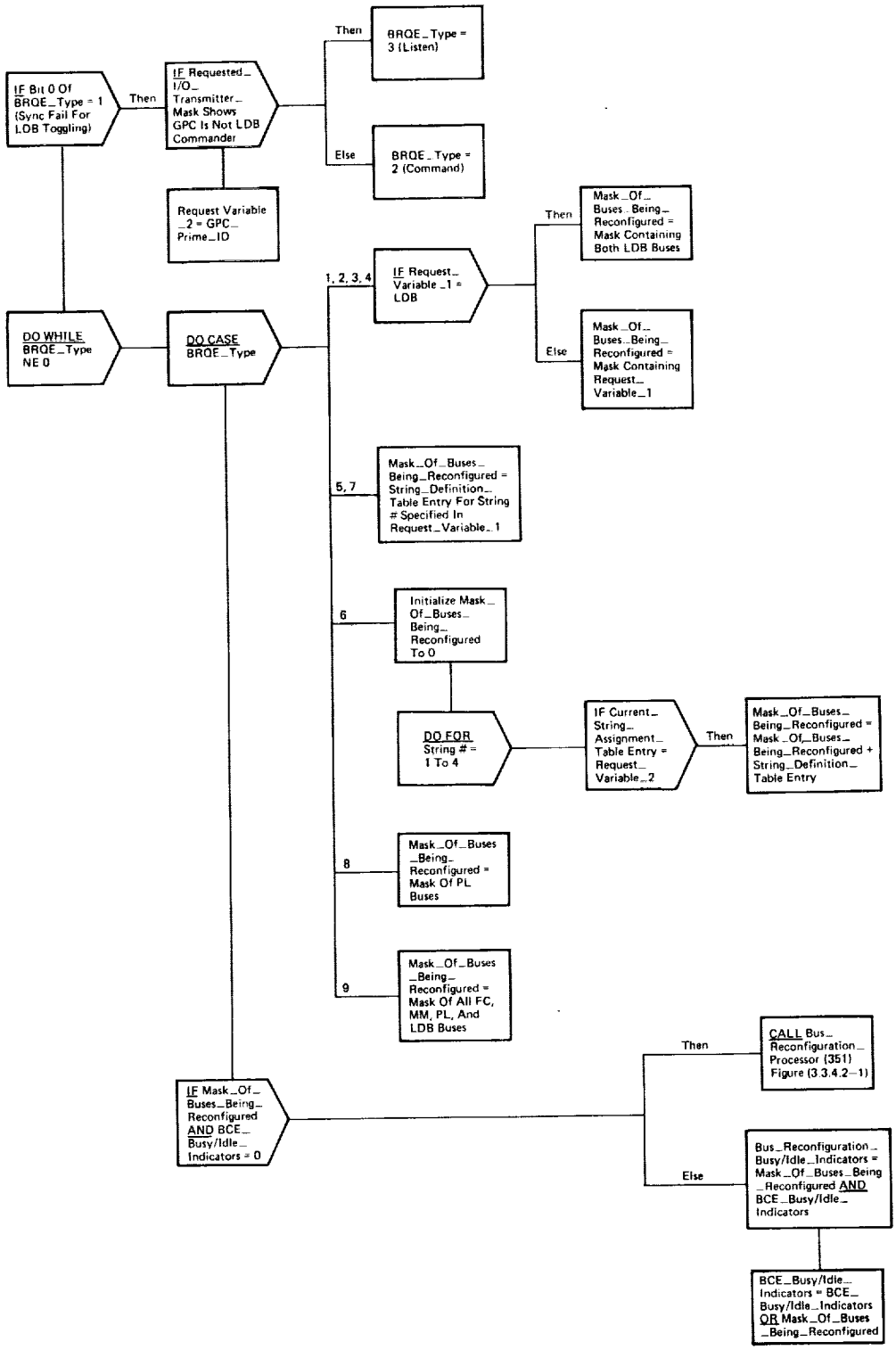


Figure 3.3.4.1-2. Bus_Reconfiguration Pre-Processor BRQE_Pre-Processor (350.1)

3.3.4.2 Bus_Reconfiguration_Processor (FCMBUSCM) (351)

The Bus_Reconfiguration_Processor (FCMBUSCM) effects I/O transmitter and receiver mode changes in support of changes in GPC/Bus assignments.

a. Control Interface CALL'ed by (350) Bus_Reconfiguration_Pre_Processor (FCMBMAN).

b. Input - Register 0 contains the address of a Bus_Reconfiguration_Queue_Element. Also see Table 3.3.4.2-1.

c. Process Description - Initial Bus_Reconfiguration_Processor processing assesses the input BRQE_TYPE and immediately passes control to code unique to that BRQE_TYPE. For all request types, a final CALL to the Bus/Data_Path_Mask_Manager must be made to perform appropriate Mask management implied in the request. The Internal_Parameter_List_Buffer is initialized to indicate a reset Bus_Masks request for the Mask_Of_Buses_Being_Reconfigured. This parameter list may be changed in the subroutines where noted. This call is made at the common return point for all BRQE_TYPE processing. The following paragraphs describe processing for each BRQE_TYPE value.

1. Remove a bus from command mode - The bus transmitter is disabled and the Requested_I/O_Transmitter_Mask is updated. If LDB is being toggled, the BCE_Device_Mask table is updated to reflect the LDB bus change and waiting IOQE's are updated to toggle them to the other LDB.

2. Place a bus in command mode - If the bus is an LDB, the transmitter of the other LDB bus is disabled. It the LDB being put in command mode is not the Current_LDB_Bus reflected in the I/O tables, the Device_Mask_Table and waiting LDB IOQE's are updated to use the new LDB. Next, (for any bus), the transmitter and receiver are enabled and the Requested_I/O_Transmiter_Mask and Requested_I/O_Receiver_Mask are updated. If PL or LDB reassignment, The String_Definition_Table entry for string 5 is updated to reflect whether the bus is or is not in the redundant set. The ICC_String_Mode indicator is set to cause ICC transfer of the new string 5 definition.

3. Place a bus in listen mode - Initial processing is identical to that performed for (2) for LDB operations. Then the bus receiver is enabled and the Requested_I/O_Receiver_Mask is updated. Finally, if the bus is PL or LDB, processing for the String Definition Table is done as for (2).

4. Disable a Bus - The bus transmitter is disabled and the Requested_I/O_Transmitter_Mask is updated. If LDB, the BCE_Device_Mask_Table

and waiting LDB IOQE's are toggled as in (2). If the bus is PL or LDB, processing for the String_Definition_Table is done as for (2). The Internal_Parameter_List_Buffer is modified to indicate to the Bus/Data_Path_Mask_Manager to set bus masks.

5. Remove a string from command Mode - The Current_String_Assignment_Table is set to Request_Variable_2. If GPC self is currently assigned the subject string, all string bus transmitters are disabled. Then, if string 5 (Prime buses) is being reassigned, GPC_Prime_ID is updated to make the new string commander prime. If the string is a FC string being released by prime and the only FC string (1,2, or 3) currently in possesion of its commander and there is a redundant set, the RS_Prime_Switching_Alorigithm is called to attempt to select a new prime GPC. If one is selected the BRQE_TYPE field is updated to reflect string release or assignment, depending on wheither GPC self was or was not prime, respectively, the Request_Variable_1 field is set to 5, and the Request_Variable_2 field is set to the ID of the GPC to be prime.

6. Release all strings - The Current_String_Assignment_Table is updated to indicate that strings which were commanded by the GPC in Request Variable_1 are unassigned. Then, if GPC self is releasing the strings, transmitters for all buses of all strings commanded by self are disabled. Finally, if the releasing GPC is prime (and Redundant_Set_Mask is not 0) prime switching is performed as in (5), above.

   (NOTE: Blocks 5 and 6 change the Internal_Parameter_List_Buffer to indicate a set Bus_Masks request to Bus/Data_Path_Mask_Manager for a string (s) being unassigned or assigned outside of self's redundent set).

7. Assign a string to a GPC - The Current_String_Assignment_Table is updated to reflect new command of the specified string. If the GPC to command is self, transmitters for all buses in the string are enabled. In any case receivers for all buses in the string are enabled. If the subject string is string 5, GPC_Prime_ID is set to the ID of the GPC getting command of the string. If it is not string 5, and if is being removed from prime GPC that will now have none of strings 1,2, or 3 in its command, prime switching is attempted as in (5).

8. Mode a String - The Internal_Parameter_List_Buffer is zeroed to prevent any mask management from being done. The String_Moding_Assignment_Table is then updated to reflect the new mode (Request_Variable_1 field). Only PL moding is supported in ALT. Therefore, no further processing is done if Request_Variable_2 is not 5. If requested mode is not the current mode, the BCE_Device_

Mask_Table and waiting PL IOQE's are updated to do PL I/O to the ports associated with the requested mode.

9. Perform Nominal_String_Assignment_Table configuration - For all FC, LDB, PL, and MM buses, transmitters are disabled and receivers are enabled. Then for each of the 5 strings, the following is done. The Current_String_Assignment_Table is updated to reflect the GPC ID indicated for this string in the Nominal_String_Assignment_Table for the memory configuration specified in Request_Variable_1. If GPC self is the GPC indicated, the transmitters for the buses in this string are enabled and the Requested_I/O_Transmitter_Mask is updated. If the string is unassigned (Nominal_String_Assignment_Table entry for this string =0) or assigned to a GPC that is not in self's Redundant_Set_Mask, the Bus/Data_Path_Mask_Manager is invoked to set Bus_Masks for this string's buses. These buses are removed from the mask of buses which will be passed to the Bus/Data_Path_Mask_Manager for resetting. If the string is the string of prime buses (string 5), GPC_Prime_ID is set to the GPC indicated.

   When all strings have been processed prime switching is done as in (5) if string 5 (prime buses) was unassigned or assigned to a GPC which is not in command of strings 1, 2, or 3.

d. Output - Outputs are listed in Table 3.3.4.2-1.

e. Module References -

   1. (355) Bus/Data_Path_Mask_Manager (FCMBMASK) is CALLed.
   2. (392) RS_Prime_Switching_Algorithm (FCMRSALG) is CALLed.

f. Module attributes - Program

g. Template references - N/A

h. Error Checks - None

i. Constraints and Assumptions - None

j. Detailed Implementation -

   1. A program Controlled (PC) I/O command is issued to directly perform the indicated operation.

DATA TABLE 3.3.4.2-1

NAME Bus_Reconfiguration_Processor (FCMBUSCM)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Bus_Management_Work_Area	0003	I,O			TBMWSTRT			
2	Mask_Of_Buses_Being_Reconfigured	0003.1	I	350	350 351	TBMWMBBR			
3	GPC_Self_Status_Table_Address	Q001.58	I	300	See App. E	TCVTGST			
4	Bus_Reconfiguration_Queue_Element	Q008	I			TFBRQE			
5	Next_BRQE_Address	Q008.1	I	350	350	TBRQENXT			
6	BRQE_Type	Q008.2	I	350 351	350 351	TBRQETYP			
7	Request_Variable_1	Q008.3	I	350 351	350 351	TBRQEV1			
8	Request_Variable_2	Q008.4	I	350 351	350 351	TBRQEV2			
9	ICC_Status_Flags	I320	O	See App. E		CZ2VIF1			
10	ICC_String_Mode	I320.12	O	351		FICCSTRG			
11	Internal_Parameter_List_Buffer	0003.5	O	351 368		TBMWBDPM			
12	Requested_I/O_Transmitter_Mask	#003	O	351 355	205,250, 350,351	TFCMXMSK			
13	Requested_I/O_Receiver_Mask	#004	O	351 355	351	TFCMRMSK			
14	GPC_ID	#001	I	300	See App. E	TFCMID			
15	Current_String_Assignment_Table	I080	O	351 368	350,392, 830	CZ2VSAT			X
16	GPC_Prime_ID	I050	O	351,700, 820,880	See App. E	CZ2VGPCP			
17	GST_Address_Table	@004	I	368	See App. E	FPMGSTAD			
18	IOP_Transmitter_State	I010.06	O	351,355,	See App. E	TGSTTCCT	V91M8706P V91M8740P	X	
				830			V91M8774P V91M8808P		
19	ICC_DST_Status	I320.11	O	351,910 950		FICCGPCP			

DATA TABLE 3.3.4.2-1 (Cont'd)

NAME Bus_Reconfiguration_Processor (FCMBUSCM)

NO.	ITEM	ID	ACT	SOURCE	DESTINATION	ASSEMBLER NAME	MML	D	C
20	String_Moding_Assignment_Table	I090	O	351		CZ2BMODE	V91041110C	X	
21	Current_PL_Bus_Mode	0003.6	I,O	351	351	TBMWCPL			
22	BCE_Device_Mask_Table	Y003	O	351 390	200,201 220,244	F10BCD			
23	Nominal_String_Assignment_Table	I110	I	860,850	351,485, 360	CZ2BNOMS		X	X
24	String_Definition_Table	I070	O	351	350,355, 485	CZ2BSDT			
25	Redundant_Set_Mask	I010.13	I	See App. E		TGSTRSM	V91M2070P V91M2104P	X	
							V91M2138P V91M2172P		
26	Current_LDB_Bus	0003.4	I,O	351	351	TBMWCLDB			
27	I/O_Wait_Queue_Address	Q001.6	I	103	See App. E	TCVTIOW			
28	Address_Of_Next_IOQE_In_Chain	Q007.1	I	200,201 225	205,225	TIOQNXT			
29	Mask_Of_Buses_To_Monitor	Q007.4	O	200,201 351	200,205, 251	TIOQMNTM			
30	IOQE_Device_ID	Q007.43	I	200	205,251, 351	TIOQDVID			
31	Mask_Management_Request_Type	S035.1	I	351	355,368				

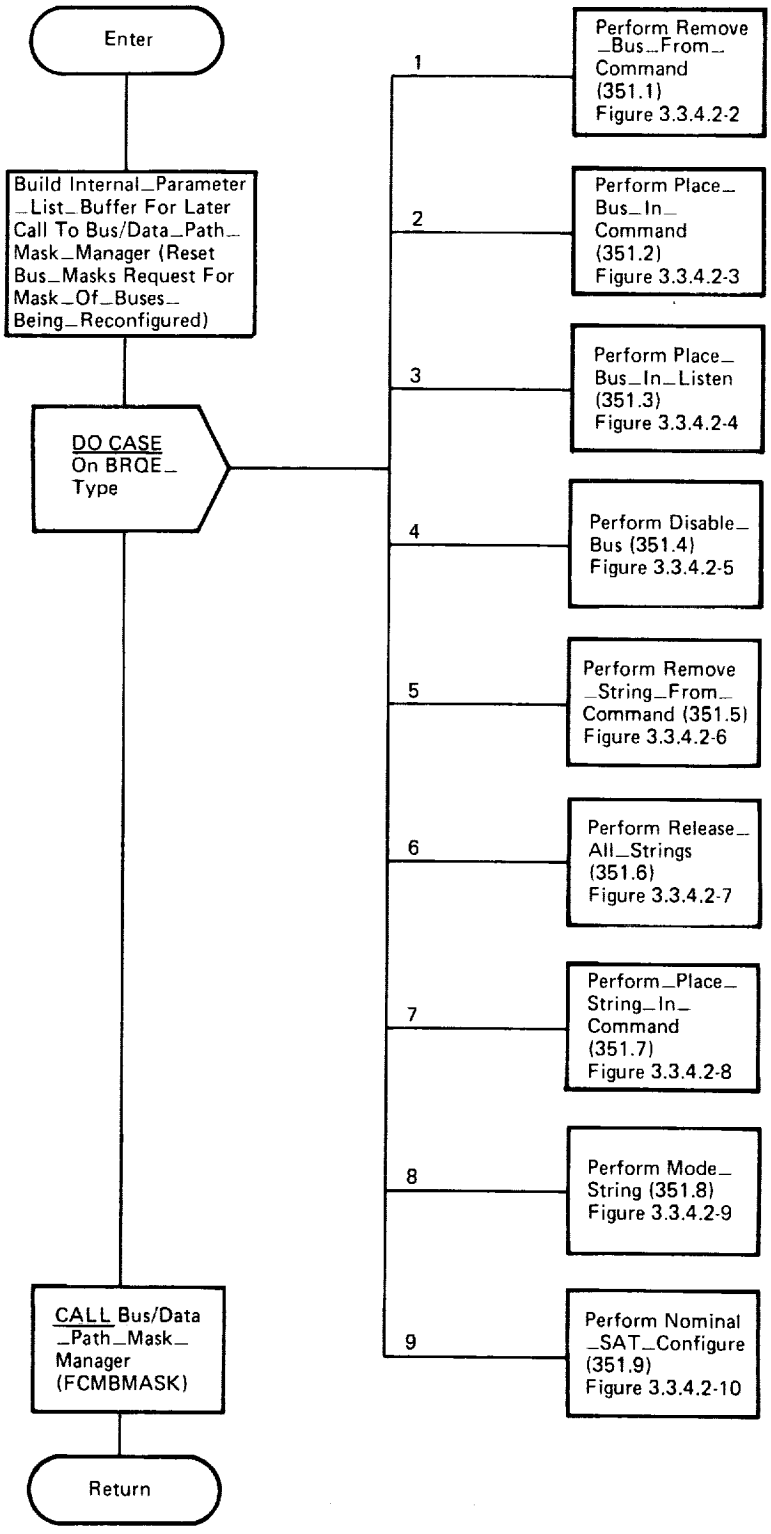


Figure 3.3.4.2-1. Bus_Reconfiguration_Processor (FCMBUSCM)

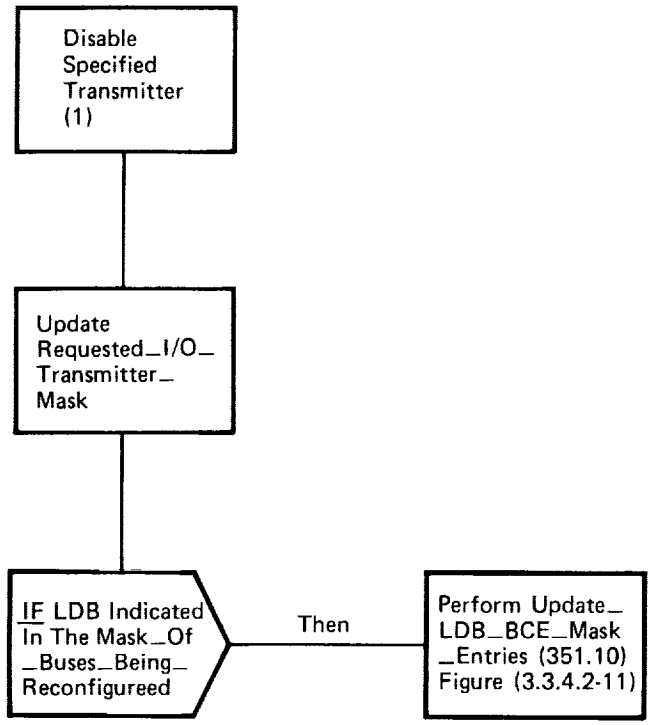


Figure 3.3.4.2-2. Bus_Reconfiguration_Processor Remove_Bus_From_Command (351.1)

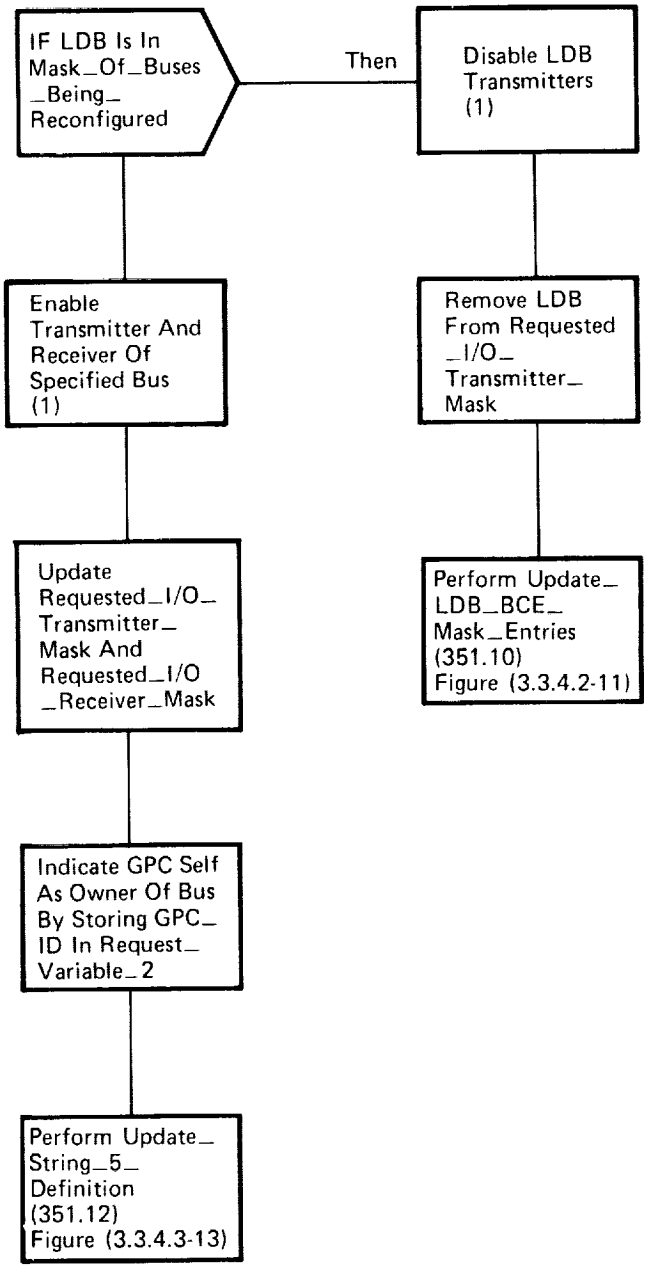


Figure 3.3.4.2-3. Bus_Reconfiguration_Processor Place_Bus_In_Command (351.2)

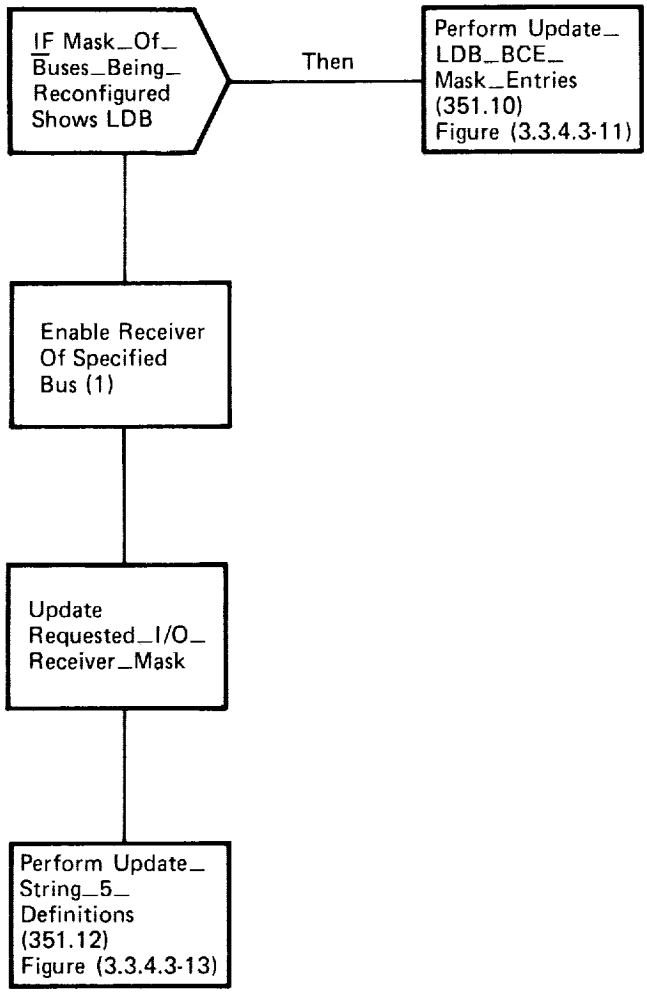


Figure 3.3.4.2-4. Bus_Reconfiguration_Processor Place_Bus_In_Listen (351.3)

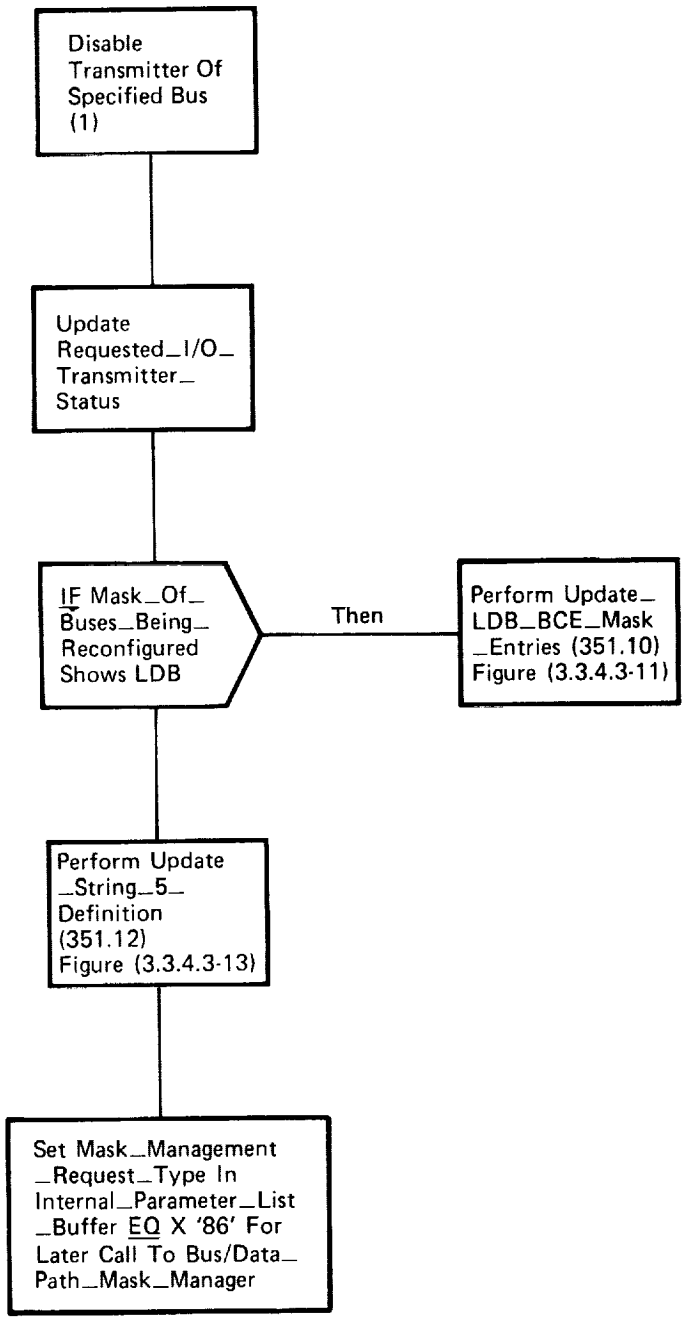


Figure 3.3.4.2-5. Bus_Reconfiguration_Processor Disable_Bus (351.4)

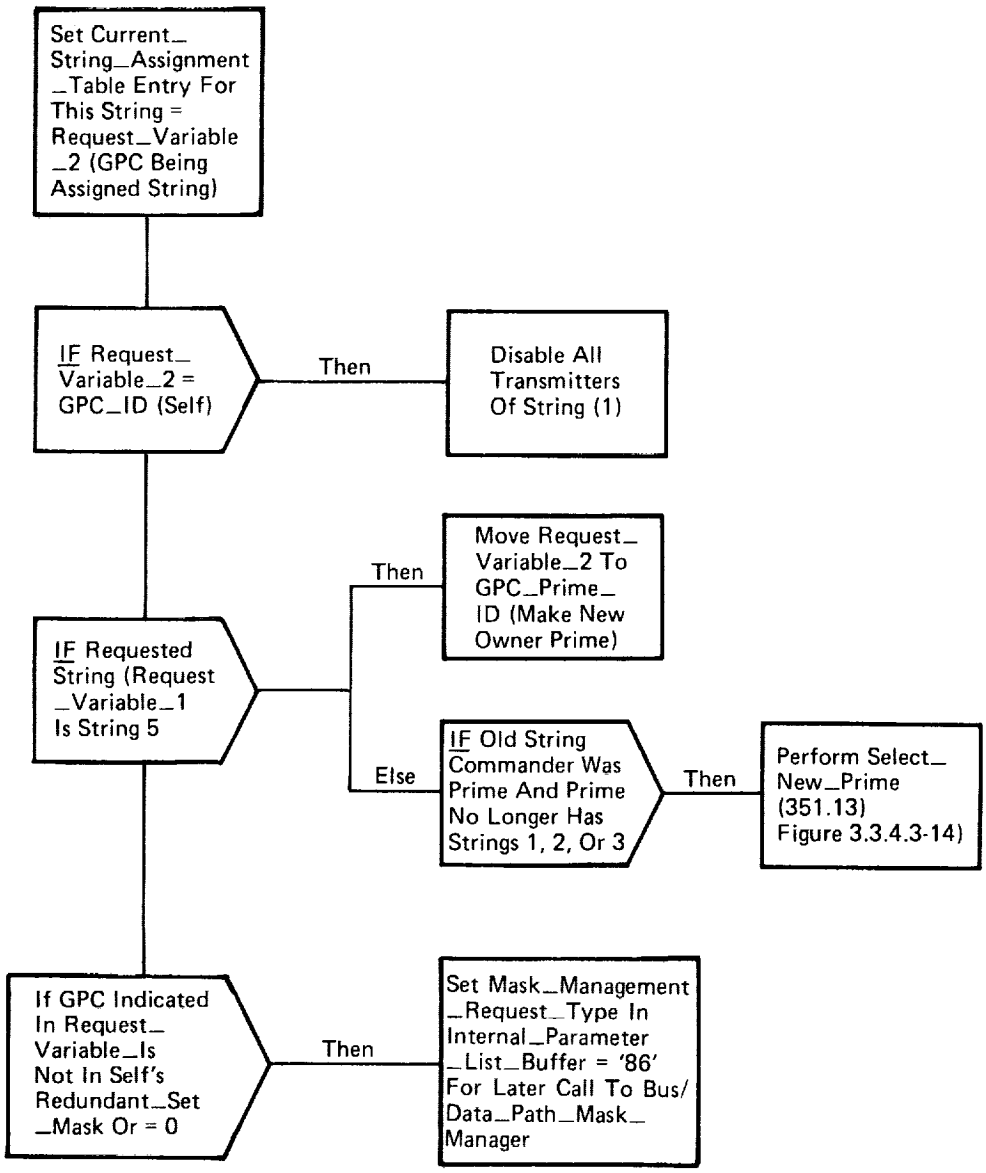


Figure 3.3.4.2-6. Bus_Reconfiguration_Processor Remove_String_From_Command (351.5)

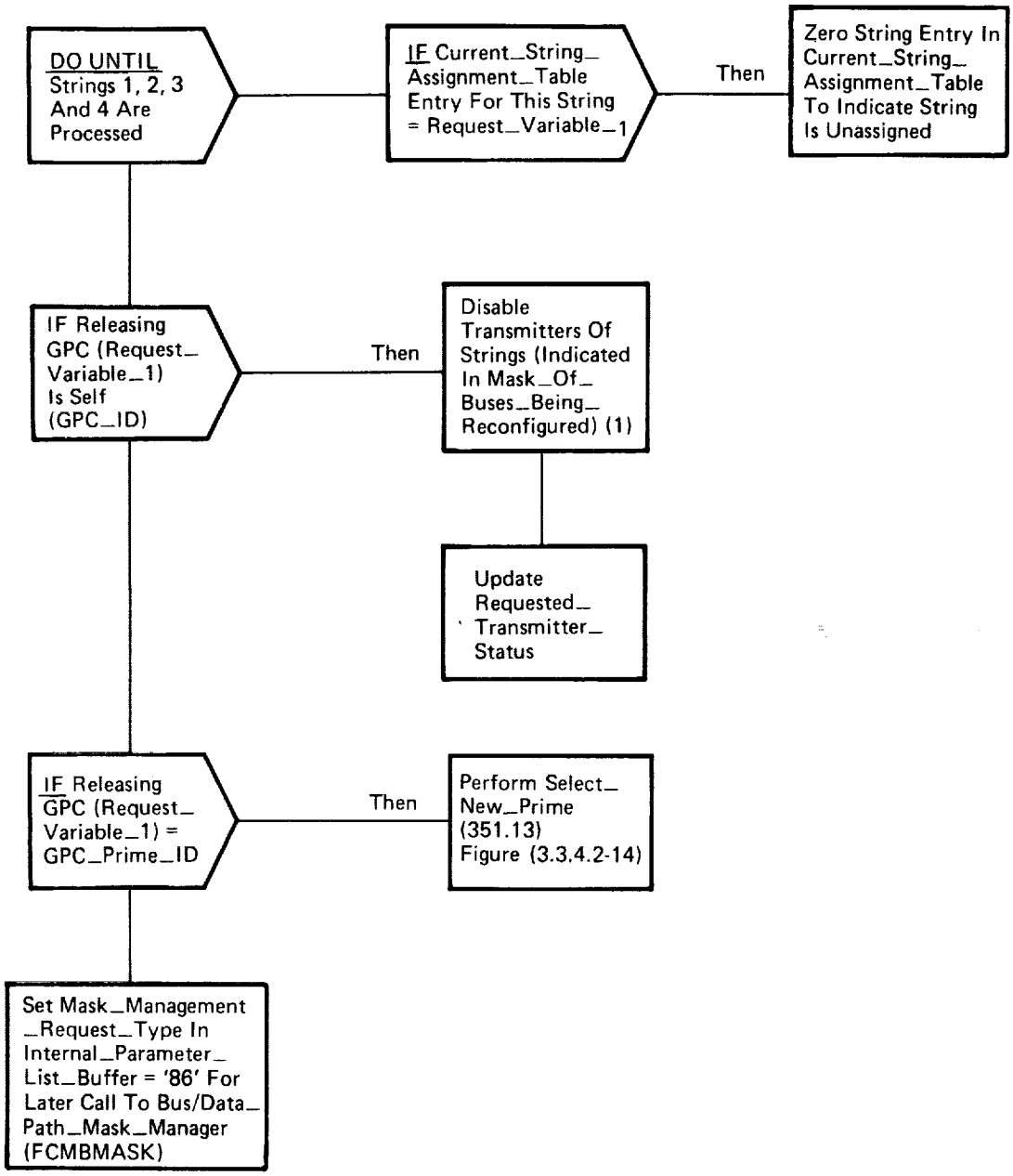


Figure 3.3.4.2-7. Bus_Reconfiguration_Processor Release_All_Strings (351.6)

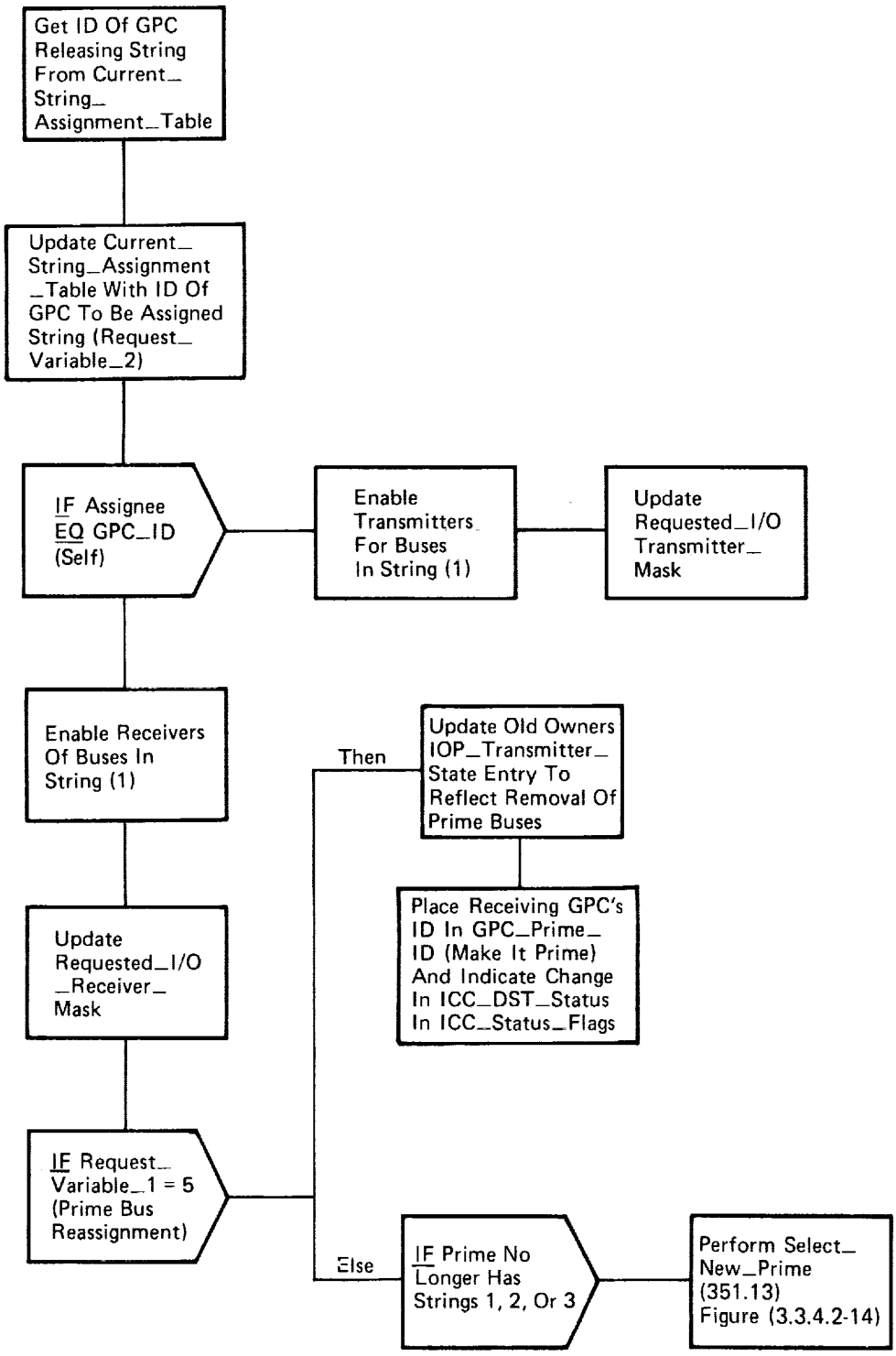


Figure 3.3.4.2-8. Bus_Reconfiguration_Processor Place_String_In_Command (351.7)

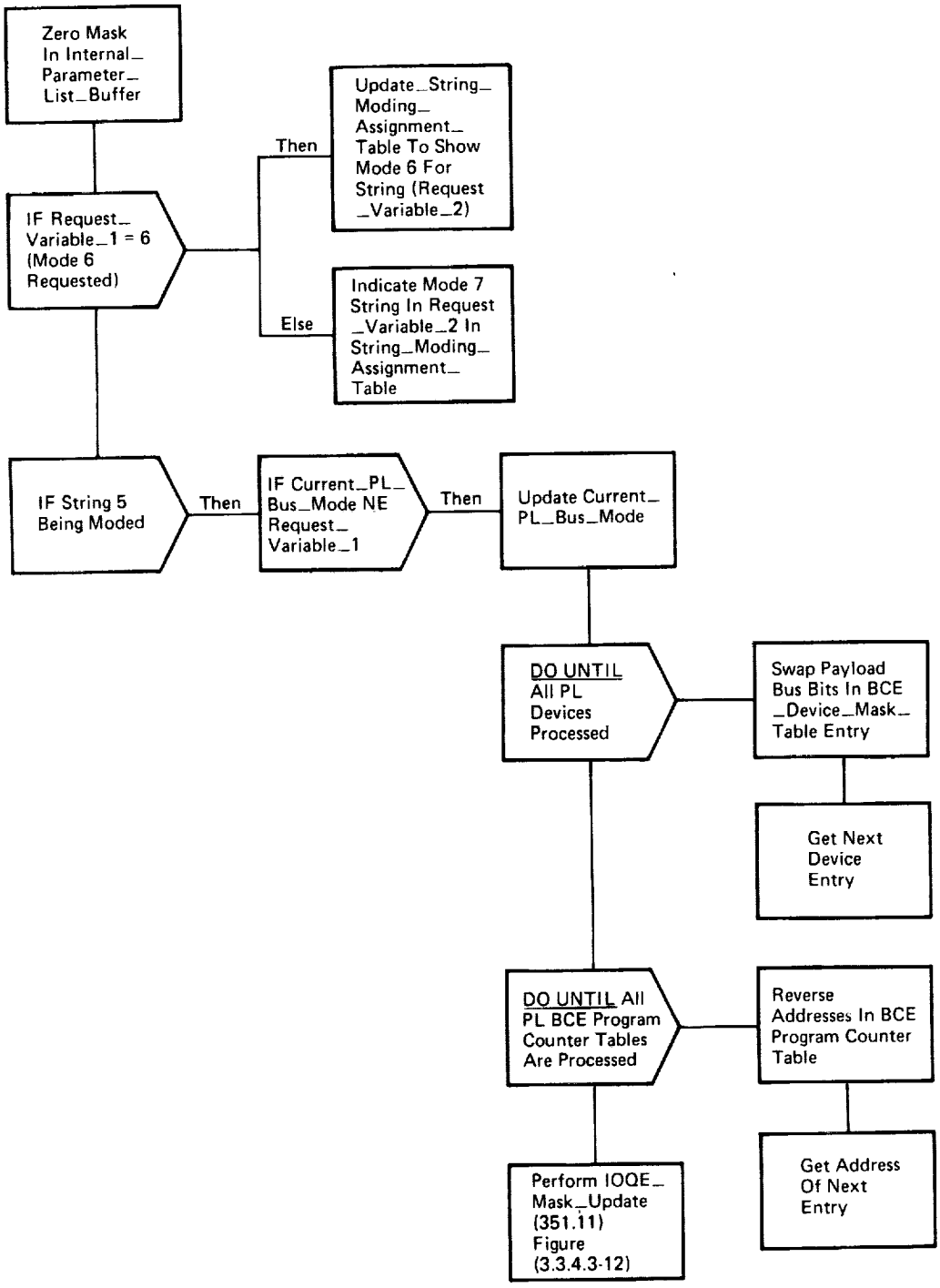


Figure 3.3.4.2-9. Bus_Reconfiguration_Processor Mode_String (351.8)

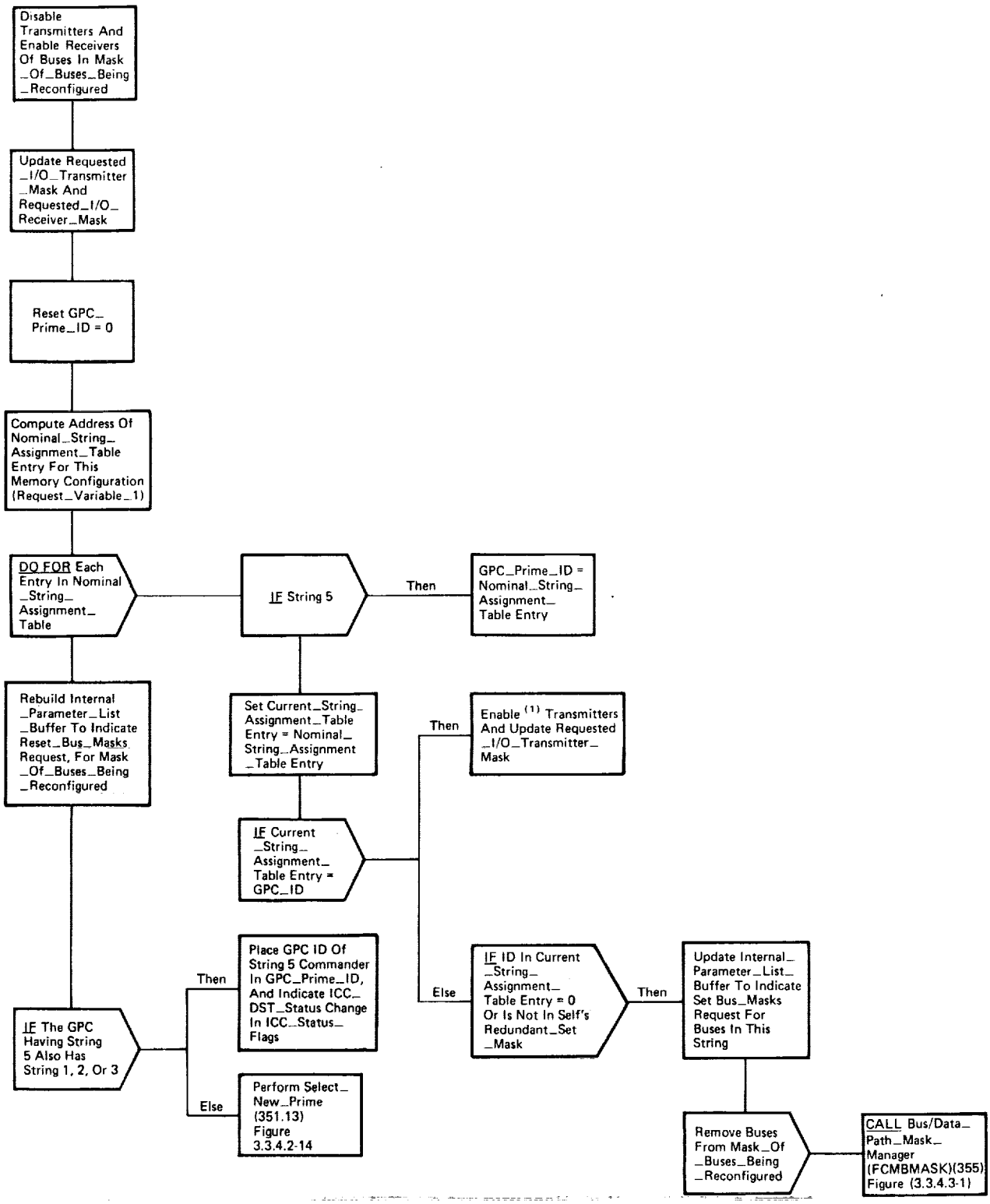


Figure 3.3.4.2–10. Bus_Reconfiguration_Processor Nominal_SAT_Configure (351.9)

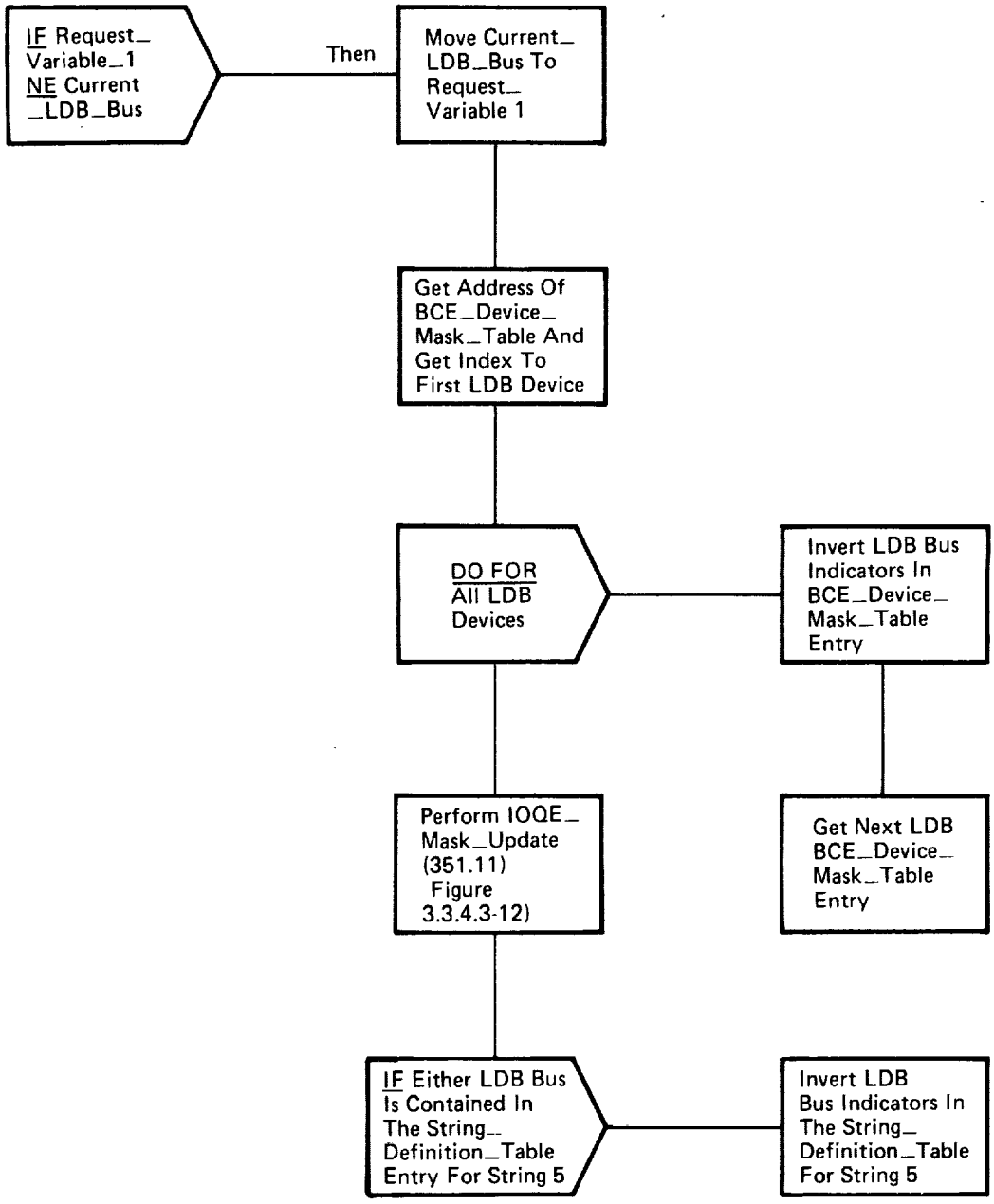


Figure 3.3.4.2-11. Bus_Reconfiguration_Processor Update_LDB_BCE_Mask_Entries (351.10)

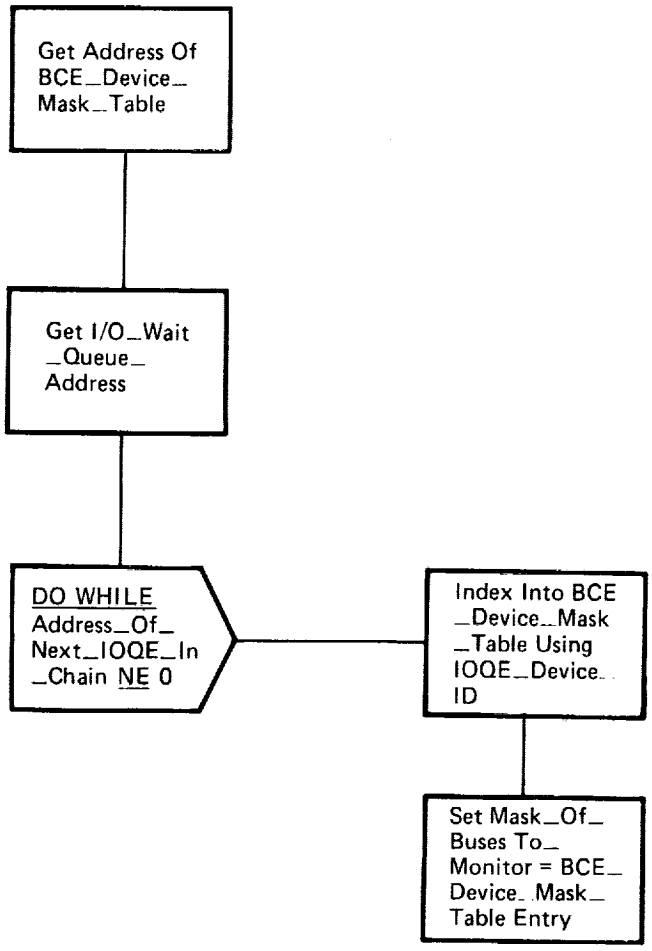


Figure 3.3.4.2-12. Bus_Reconfiguration_Processor IOQE_Mask_Update (351.11)

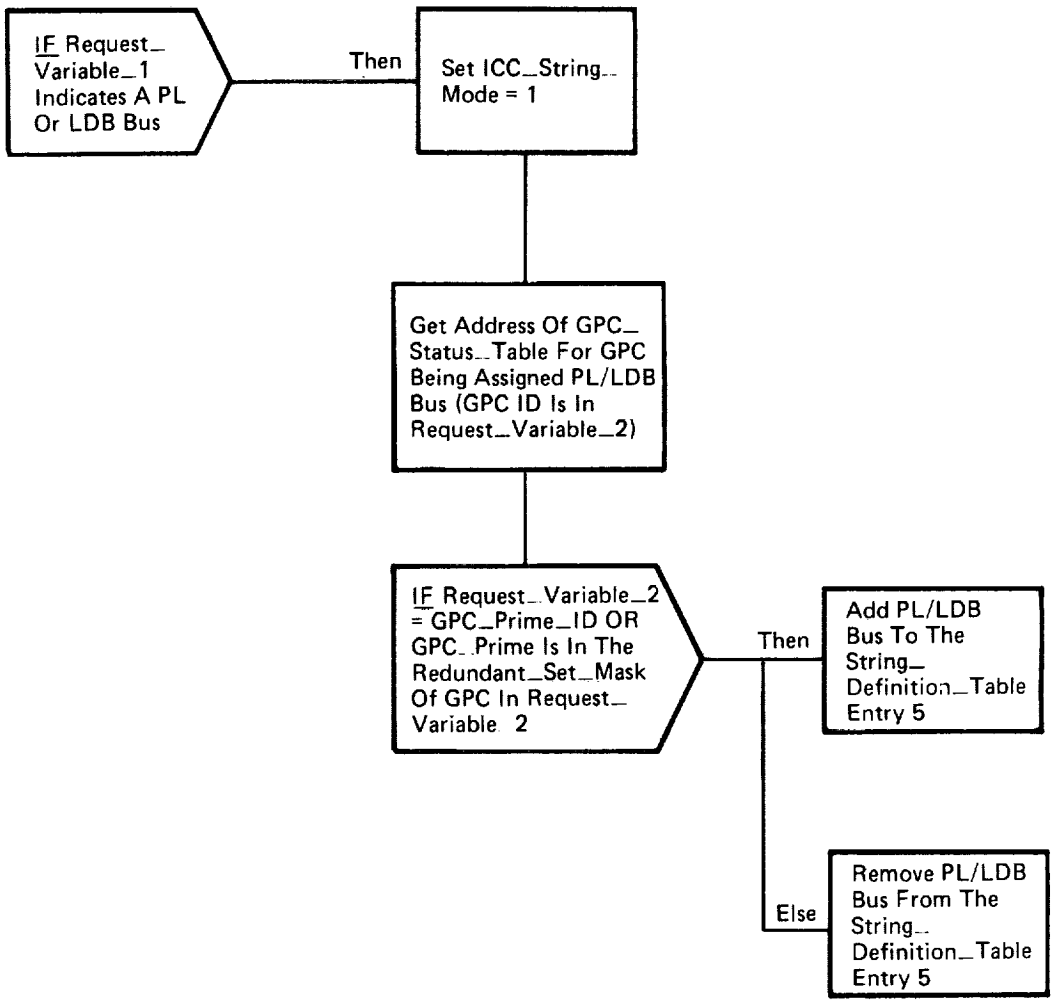


Figure 3.3.4.2-13. Bus_Reconfiguration_Processor Update_String_5_Definition (351.12)

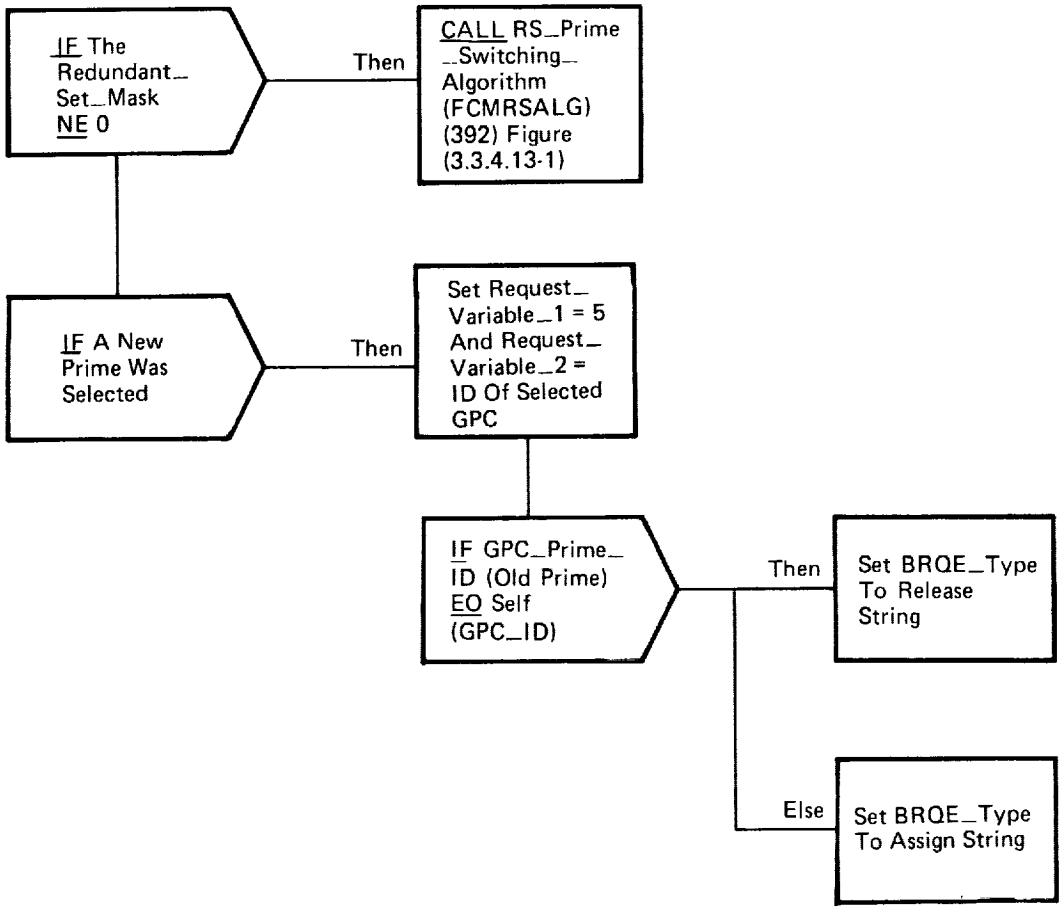


Figure 3.3.4.2-14. Bus_Reconfiguration_Processor Select_New_Prime (351.13)

3.3.4.3 Bus/Data_Path_Mask_Manager (FCMBMASK) (355)

The Bus/Data_Path_Mask_Manager performs the setting/resetting of bus and/or data path masks and error counters.

a. Control Interface -

1. CALLed by (100) SVC_Handler (FPMSVC)
2. CALLed by (351) Bus_Reconfiguration_Processor (FCMBUSCM)
3. CALLed by (368) SYNC_Fail_Processor (FCMSFAIL)
4. CALLed by (244) Level_C_I/O_Error_Interrupt_Handler (FIOERRLC)

b. Input - See Table 3.3.4.3-1.

c. Process Description - The Bus/Data_Path_Mask_Manager first determines if it has been called by the SVC_Handler. If so, the SVC Synchronization_Processor is invoked to obtain redundant set synchronization. The Mask_Management_Request_Type and the Mask_Management_Variable_Field are then used to build a working mask of the buses affected in the request. Non-ICC buses are then processed according to the Mask_Management_Request_Type.

If Mask_Management_Request_Type indicates that Bus_Masks are to be set, the following is done. The bits associated with the buses in the working bus mask are set in Bus_Masks. BCE processors for these buses are halted. If the Bus_Mask_Management_Request_Type indicates that bus masks are being set as a result of a bus reconfiguration request (X'86'), the receivers associated with the buses in the working mask are disabled and the Requested_I/O_Receiver_Mask is updated. BCE_Element_Bypass_Indicator_1 and 2 are updated to indicate all elements associated with these buses.

If Mask_Management_Request_Type indicates that Bus_Masks are to be reset, the following is done. When the OUTPUT switch goes to "TERMINATE" or BFCS is engaged, the hardware disables certain transmitters. Therefore when a request is made to reset Bus_Masks when the OUTPUT switch goes to "NORMAL" (Mask_Management_Request_Type = X'02') or BFCS is disengaged (Mask_Management_Request_Type = X'04'), transmitters which were disabled must be enabled. The Reset_IOP_Processor is then invoked to reset the buses in the working mask (except any buses which are currently busy). Bits associated with buses in the working masks are zeroed in Bus_Masks. BCE_Element_Bypass_Indicator_1 and 2 are updated to not indicate elements associated with buses being reset.

If Mask_Management_Request_Type indicates that Bus_Masks or Data_Path_Masks are to be reset (except for BFCS disengage), the following is done. Bits associated with the buses in the working mask are reset in the Data_Path_Masks and BTU_Port_Masks. Data_Path_Error_Counter_Table entries for these buses are all reset to 0.

For all types of requests, the true current transmitter and receiver status is determined and placed in IOP_Transmitter_State and IOP_Receiver_State.

The control flow for this module is presented in Figures 3.3.4.3-1 and 3.3.4.3-2.

d. Output - See Table 3.3.4.3-1.

e. Module References -

1. (363) SVC_Synchronization_Processor (FCMSSYNC) is called.
2. (244.5) Reset_IOP_Processor (FIORESET) is called.

f. Module Attributes - Program

g. Template References - N/A

h. Error Checks - None

i. Constraints and Assumptions - None

j. Detailed Implementation - 1. A Program Controlled (PC) I/O command is issued directly to perform the indicated operation.

DATA TABLE 3.3.4.3-1

NAME Bus/Data_Path_Mask_Manager (FCMBMASK)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	ICC_Status_Flags	I320	Z	See App. E		CZ2VIF1			
2	ICC_Bus_Masks	I320.07	Z	355					
3	Mask_Management_Parameter_List	S035	I		355,368				
4	Mask_Management_Request_Type	S035.1	I	351	355,368				
5	Mask_Management_Variable_Field	S035.3	I		355,368				
6	String_Definition_Table	I070	I	351	350,355, 485	CZ2BSDT			
7	GPC_ID	#001	I	300	See App. E	TFCMID			
8	Requested_I/O_Transmitter_Mask	#003		351 355	205,250, 350,351	TFCMXMSK			
9	GST_Address_Table	@004	I	368	See App. E	FPMGSTAD			
10	Bus_Masks	I010.09	I,O	355	See App. E	TGSTBMSK	V91M7922P V91M7957P	X	X
							V91M8011P V91M8055P		
11	IOP_Transmitter_State	I010.06	I	351,355, 830	See App. E	TGSTTCCT	V91M8706P V91M8740P	X	
							V91M8774P V91M8808P		
12	GPC_Self_Status_Table_Address	Q001.58	I	300	See App. E	TCVTGST			
13	Requested_IO_Receiver_Mask	#004	I,O	351 355	351	TFCMRMSK			
14	BCE_Busy/Idle_Indicators	Q001.17	I	See App. E		TCVTBCEB			
15	Data_Path_Controller_FC_PL_LDB	I010.22	I,O	244, 355	485, 620	TGSTDPD			X
16	Data_Path_Masks	I010.11	I,O	244,290, 355	148,290 665	TGSTDPM	See App. E	X	
17	BTU_Port_Masks	#024	I,O	290	205,290, 355,368	TFCMBPM	See App. E	X	X
18	ICC_Data_Path_Masks	I320.04	O	355					

DATA TABLE 3.3.4.3-1 (Cont'd)

NAME Bus/Data_Path_Mask_Manager (FCMBMASK)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
19	Data_Path_Error_Counter_Table	#200.3	I		355	FIODPECT			
20	N_Counter_Pointers	#200.1			355	FIOBUS06			
21	Data_Path_Error_Counters	#200.4	0	355	268	FIOB0011			
22	IOP_Receiver_State	I010.08	I,0	335,355	400,440, 815,830	TGSTTCCR			
23	BCE_Element_Bypass_Indicator_1	#030	0	220,244 330,355	220,244, 330	TFCMBCEBI/FIOBCEBI	V91M2252P	X	
24	BCE_Element_Bypass_Indicator_2	#032	0	220,244 330	220,244 330,355	TFCMBCBI2/FIOBCBI2	V91M2269P	X	

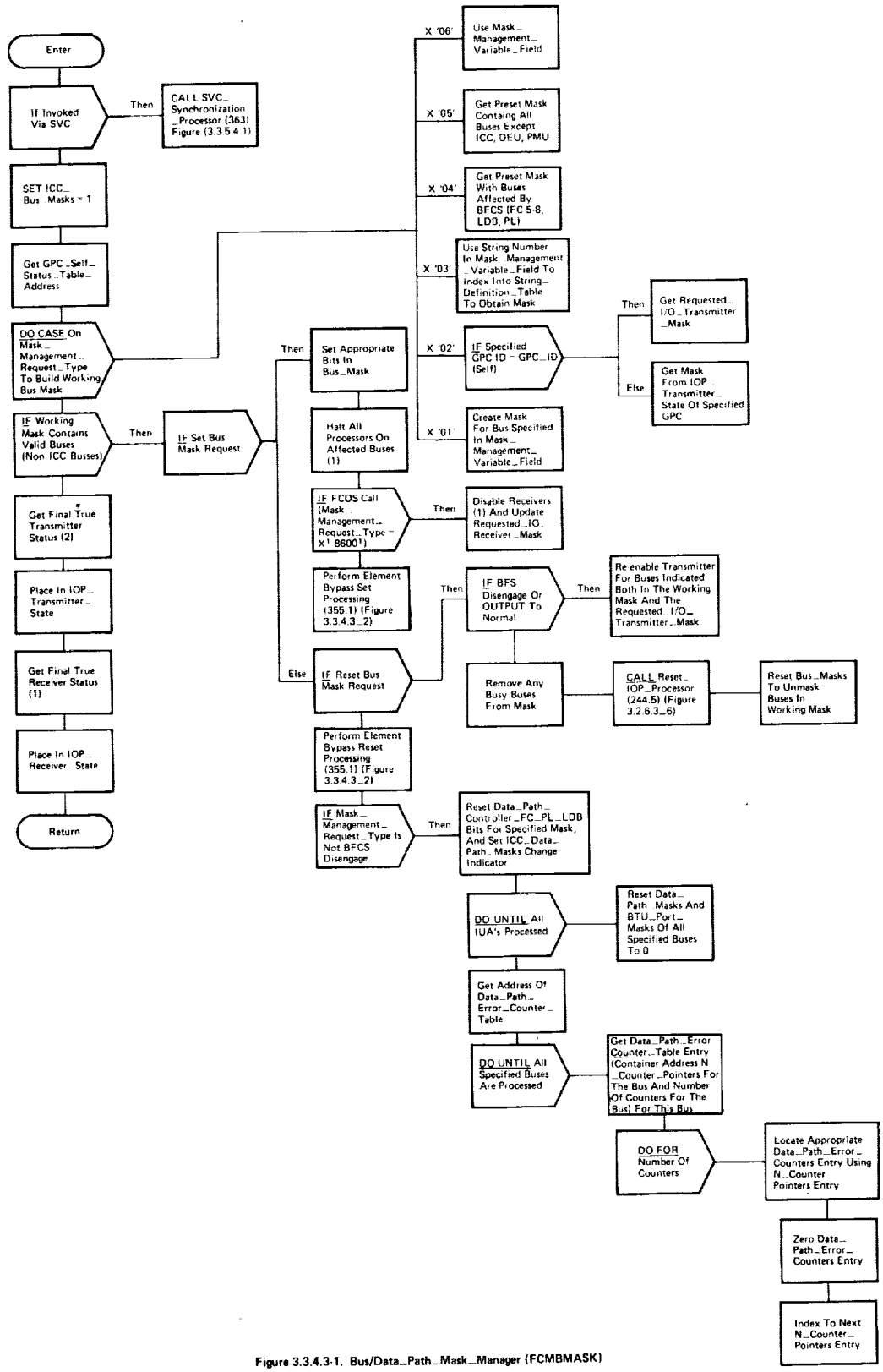


Figure 3.3.4.3-1. Bus/Data_Path_Mask_Manager (FCMBMASK)

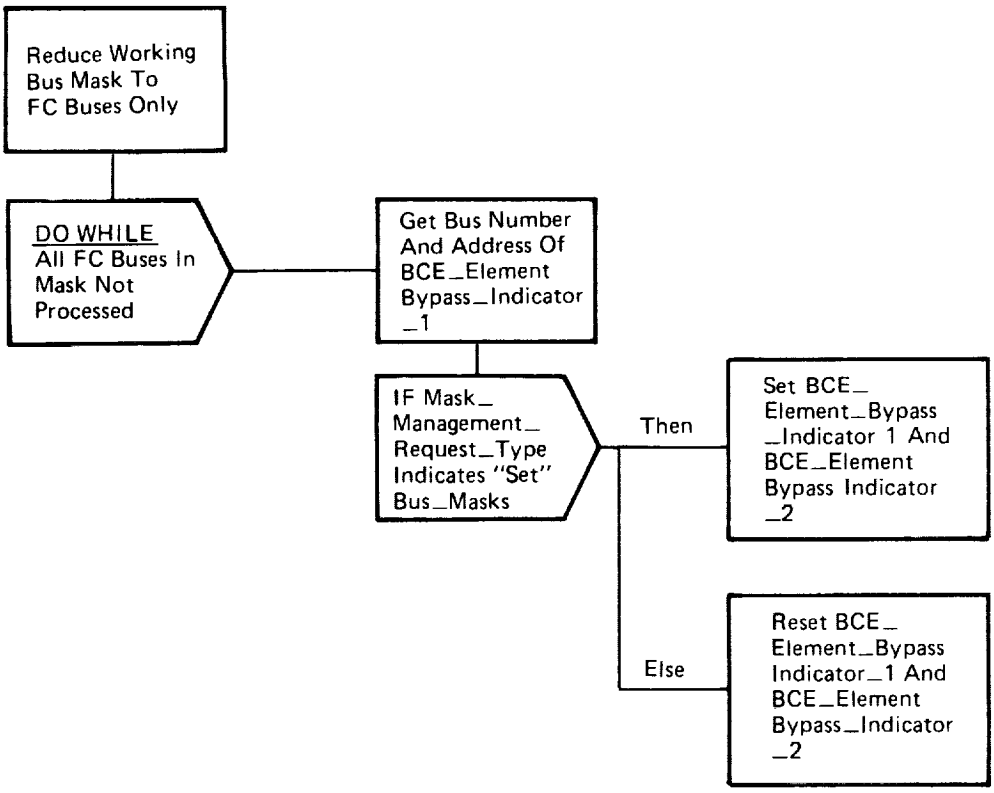


Figure 3.3.4.3-2. Bus/Data_Path_Mask_Manager Element_Bypass_Processing (355.1)

3.3.5 GPC Redundancy Management

GPC Redundancy Management consists of GPC Synchronization for both the common set and redundant set and Fault Detection Identification.

GPC synchronization is designed to keep the GPC's together with respect to time, data gathering, and internal queue manipulation. Because of possible dissimilar processing, the common set is synchronized with respect to time only. This is accomplished by four basic synchronization programs and one special program for SSIP (System Software Interface Processor) sync initialization.

The special sync program, Initial_SSIP_Synchronization_Processor, is used when adding a GPC to the common set as part of initial SSIP processing. It determines if any other GPC's are active and, if so, which ones are members of the common set.

The four basic sync programs are Normal_SSIP_Synchronization_Processor (common Set), I/O_Synchronization_Processor (common and redundant set), SVC_Synchronization_Processor (redundant set), and Timer_Synchronization_Processor (redundant set).

The Normal_SSIP_Synchronization_Processor is used to keep the common set synchronized via the minor cycle SSIP points. Its processing is initiated by the SSIP timer interrupt and includes the time management MTU read every n^{th} minor cycle. It also allows eligible GPC's to join the common set.

The I/O_Synchronization_Processor synchronizes I/O completion interrupts in the redundant set and provides IPR (Input Problem Report) information for protected transaction processing.

SVC Synchronization is done for all SVC's in the redundant set that cause queue modification, data gathering, event modification, changing the state of a process, locked resources, data output, and time processing.

The Timer_Synchronization_Processor is used to synchronize the redundant set GPC's at every interval timer interrupt (PC2).

Synchronization communication between the GPC's is accomplished through the use of GPC sync discrete lines. Each GPC has three lines out (and hence 8 possible codes) and twelve lines in (3 from each of the other 4 GPC's). For code assignments and meanings, see Figure 3.3.2-1.

To prevent a synchronization failure caused by GPC's arriving at different sync points at the same time (e.g., an SVC sync in one GPC and a timer interrupt in another GPC), a synchronization priority scheme is used. The only sync interrupt interference possible is between SVC, I/O complete, and timer interrupts. SVC synchronization is lowest priority and can be interrupted by both I/O complete and timer interrupts. I/O complete synchronization may be interrupted by timer interrupts only, while timer sync is not interruptible. Since normal SSIP sync is

initiated by a timer interrupt, it is of the same priority as timer synchronization. This allows sync processing in each GPC to migrate to the highest priority interrupt currently being processed by any GPC.

Fault Detection Identification is accomplished within the redundant set by transferring sum words over the Inter-Computer Channel and comparing them. GPC fail discretes are set when a GPC determines that it has had a series of consecutive miscompares with another GPC's sum-word.

GPC Redundancy Management is divided into the following areas (See Figure 3.3.5-2).

a. Initial_SSIP_Synchronization_Processor - Provides synchronization when joining a common set.

b. Normal_SSIP_Synchronization_Processor - Synchronizes the common set at every SSIP point.

c. I/O_Synchronization_Processor - Synchronizes GPS in a common or redundant set upon I/O completion interrupt.

d. SVC_Synchronization_Processor - Synchronizes GPC's in a redundant set for certain supervisor calls.

e. Timer_Synchronization_Processor - Synchronizes GPC's in the redundant set at every interval timer interrupt.

f. Sync_Fail_Processor - Performs necessary processing associated with GPC fail to sync or force fail to sync.

g. Sync_Fail_CAM_MSC_Processor - MSC processor to light CAM lights when a fail to sync is received.

h. Fault_Detection_Identification - Compares sum words for all GPC's in the redundant set.

i. Fail_Discretes_MSC_Processor - MSC processor used to light CAM lights when failure is noted by Fault_Detection_Identification.

j. GPC_Discretes_Redundancy_Manager - Compares discretes from all GPC's in the common set and provides one discrete word for all to use.

k. Sync_Mask_Build_Routine - A service routine used by each of the synchronization programs mentioned above.

l. Sync_Fail_Interface_Routine - Builds parameter list need by the Sync_Fail_Processor.

m. RS_Prime_Switching_Algorithm - Selects prime computer for the redundant set.

Number	Binary	Meaning
0	000	Halt/Off/Failed
1	001	Spare
2	010	I/O Complete, with error (IPR)
3	011	I/O Complete, no error (IOC)
4	100	SSIP sync (common set)
5	101	Timer Interrupt
6	110	SVC Interrupt
7	111	Null/Run

Figure 3.3.5-1. Synchronization Codes

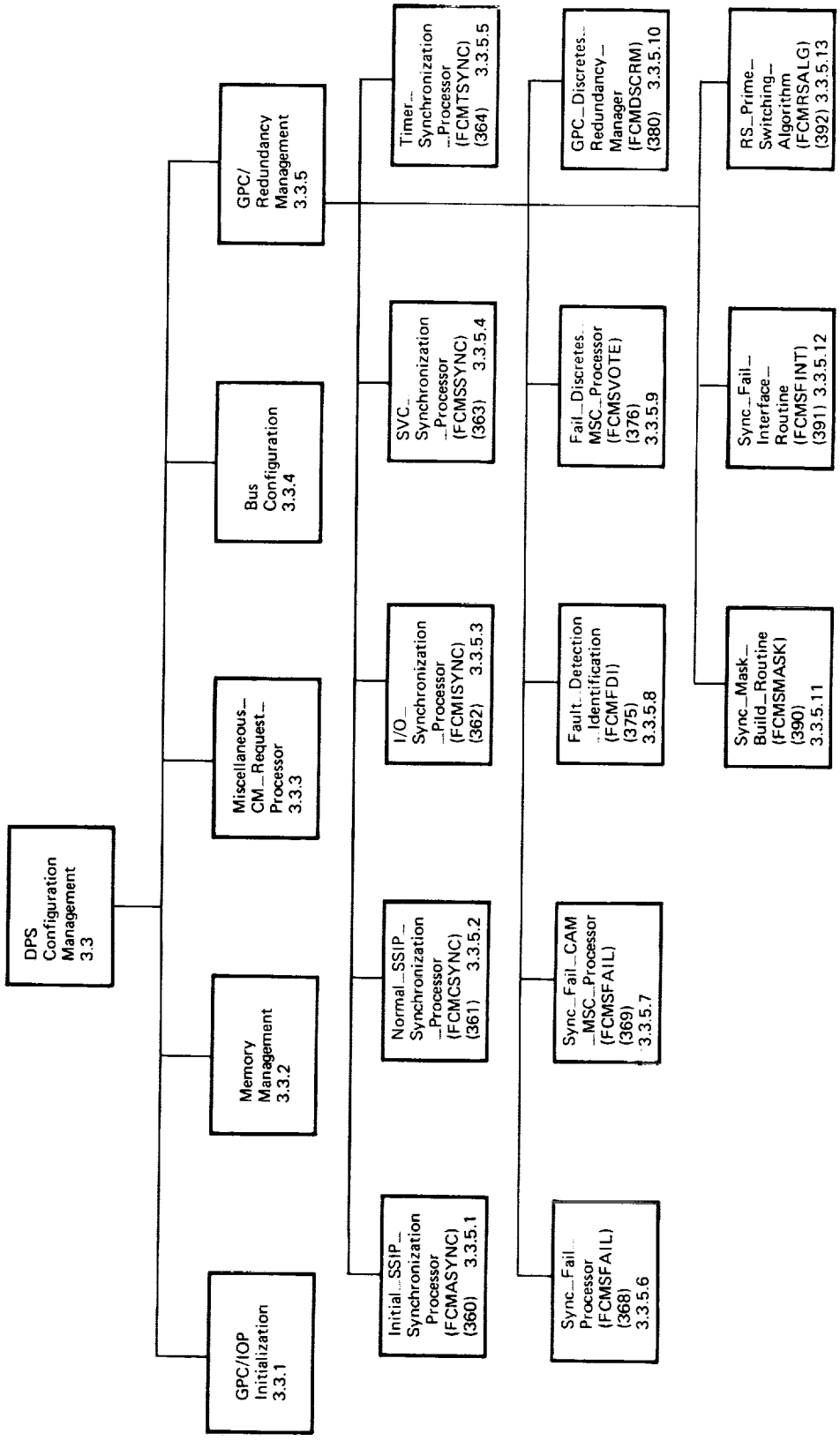


Figure 3.3.5-2. GPC Redundancy Management Hierarchy Diagram

3.3.5.1 Initial_SSIP_Synchronization-Processor (FCMASYNC)(360)

To Be Provided

3.3.5.2 Normal_SSIP_Synchronization_Processor(FCMCSYNC)(361)

To Be Provided

3.3.5.3 I/O_Synchronization_Processor(FCMISYNC)(362)

To Be Provided

3.3.5.4 SVC_Synchronization_Processor (FCMSSYNC) (363)

The SVC_Synchronization_Processor (FCMSSYNC) provides redundant set synchronization prior to the execution of those SVC services that could affect the state or execution path of the software system.

a. Control Interface -

1. CALLed by (330) BCE_Element_Bypass_Processor (FCMBCEMD)
2. CALLed by (350) BUS_Reconfiguration_Preprocessor (FCMBMAN)
3. CALLed by (355) BUS/Data_Path_Mask_Manager (FCMBMASK)
4. CALLed by (325) Program_Modification (FCMPMOD)
5. CALLed by (320) Overlay_Processor (FCMPOVLY)
6. CALLed by (200) I/O_SVC_Service_Processor (FIOSVC)
7. CALLed by (201) Pre_Initialized_I/O_SVC_Processor (FIOSVCP)
8. CALLed by (109) Cancel_Processor (FPMCANCL)
9. CALLed by (107) Close_Processor (FPMCLOSE)
10. CALLed by (149) MTU_Redundancy_Manager (FPMMTURM)
11. CALLed by (105) UPDATE/EXCLUSIVE_Reserve_Processor (FPMRES)
12. CALLed by (171) Reset_Event_Processor (FPMRESET)
13. CALLed by (101) Process_Scheduler (FPMSCHED)
14. CALLed by (170) Set_Event_Processor (FPMSET)
15. CALLed by (172) Signal_Event_Processor (FPMSIGNL)
16. CALLed by (108) Terminate Processor (FPMTERM)
17. CALLed by (144) Time/Date_Application_Request_Processor (FPMTMHAL)
18. CALLed by (104) Wait_Processor (FPMWAIT)
19. CALLed by (148) MTU_Update_Processor (FPMUPMTU)

b. Input - Inputs are shown in Table 3.3.5.4.-1

c. Process Description - The SVC_Synchronization_Processor obtains the Redundant_Set_Sync_Mask to determine if there are other GPC's in the present redundant set. If there are, the SVC_Code_Inverse_Sending_Pattern is placed in the GPC_Sync_Code_Command_Discrete bit positions of the Discrete_Output_Register. Otherwise, no sync code output is performed.

The SVC_Sync_In_Progress_Indicator is then set to direct re-execution of the SVC and the SVC_Synchronization Processor in the event that it is interrupted by I/O or timer expiration during its execution.

Finally, the sync loop is entered and maintained until one of the following occurs:

1. SVC_Sync codes are exchanged with all redundant set members
2. Execution of the loop exceeds a pre-determined number of micro-seconds.
3. A timer or I/O interrupt occurs.

Sync codes are obtained from other GPC's by reading the Discrete_Input_Register_A (see table 3.3.5.4.-1, items 12-23). Stability of the bit values is assured by performing 2 reads and OR'ing the resulting bit values together. Following each pair of reads, GPC's whose bit positions in the Discrete_Input_Register_A contain the SVC Sync pattern are removed from the Leading_Edge_Detection_Mask. Each time a new valid sync pattern is detected, the Sync_Timeout_Counter is reset to its maximum valve. Each time through the loop, if total redundant set agreement has not been reached, timer and I/O interrupts are allowed to occur so that sync code disagreements among the redundant set may be resolved at the highest possible priority level.

Once SVC_Sync_Code discrete patterns have been received from all redundant set GPC's, a delay is performed and the sync codes are re-read from Discrete_Input_Register_A. If any have changed since the prior read, one or more other GPC's has been interrupted by a timer or an I/O completion prior to its detection of this GPC's output SVC_Sync_Code. In this case, the GPC is enabled for timer and I/O interrupts to allow the expected synchronization to occur at the new interrupt level (timer or I/O). Otherwise, a delay is performed to allow sync code detection by other GPC's and the Sync loop is terminated.

After loop termination, the SVC_Sync_Code is removed from the Discrete_Output_Register by replacing it with the null sync code.

In the event of sync loop timeout before valid SVC_Sync_Code discretes have been received from all redundant set members, the Sync_Fail_Interface_Routine and the Sync_Fail_Processor are called to remove the GPC's that failed to sync from the redundant set and perform any required bus reconfiguration.

SVC_Synchronization_Processor flow is shown in Figure 3.3.5.4-1

d. Output - Outputs are shown in Table 3.3.5.4.-1

e. Module References -

1. (391) Sync_Fail_Interface_Routine (FCMSFINT)
2. (368) Sync_Fail_Processor (FCMSFAIL)

f. Module Attributes - Program

g. Template References - N/A.

h. Error Checks - None

i. Constraints and Assumptions - The timing of I/O and Timer interrupt handling and the timing of the sync loop are critical to the logic employed. Changes in these areas must be assessed for impact to this program.

j. Detailed Implementation

1. Issue a Program Controlled Output (PC) command to place sync pattern in the Discrete_Output_Register

2. Issue a Program Controlled Input (PC) command to obtain the current value of the sync bits in the Discrete_Input_Register_A

3. If an interrupt does occur, normal redundant set operation will result in synchronization occuring at the higher priority interrupt level (I/O or Timer) and SVC synchronization will re-occur after higher priority processing has concluded.

DATA TABLE 3.3.5.4-1

NAME SVC_Synchronization_Processor (FCMSSYNC)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Redundant_Set_Sync_Mask	Q001.47	I	305 390	See App. E	TCVTRSSM			
2	SVC_Code_Inverse_Sending_Pattern	Q001.71	I		363	TCVTSVCI			
3	GPC_Sync_Code_Command_Discrete_1	&010.11	W	363	363				
4	GPC_Sync_Code_Command_Discrete_2	&010.13	W	363	363				
5	GPC_Sync_Code_Command_Discrete_3	&010.15	W	363	363				
6	SVC_Sync_In_Progress_Indicator	Q001.49	Ø	220,305, 363	142 220	TCVTSVCS			
7	SVC_Sync_Code	Q001.63	I		363	TCVTSVCC			
8	Leading_Edge_Detection_Mask	Q001.65	I		See App E	TCVTKM			
9	Sync_Timeout_Counter	I580.02	Z	205	363,364	CZ2VSYNC			
10	Null_Sync_Code_Sending_Pattern	Q001.73	I		See App. E	TCVTNULS			
11	Discrete_Bit_GPC_N+1_Sync_Code_1	&005.14	R		363,364, 368				
12	Discrete_Bit_GPC_N+2_Sync_Code_1	&005.15	R		363,364, 368				
13	Discrete_Bit_GPC_N+3_Sync_Code_1	&005.16	R		363,364, 368				
14	Discrete_Bit_GPC_N+4_Sync_Code_1	&005.17	R		363,364, 368				
15	Discrete_Bit_GPC_N+1_Sync_Code_2	&005.18	R		363,364, 368				
16	Discrete_Bit_GPC_N+2_Sync_Code_2	&005.19	R		363,364, 368				
17	Discrete_Bit_GPC_N+3_Sync_Code_2	&005.20	R		363,364, 368				
18	Discrete_Bit_GPC_N+4_Sync_Code_2	&005.21	R		363,364, 368				
19	Discrete_Bit_GPC_N+1_Sync_Code_3	&005.22	R		363,364				

DATA TABLE 3.3.5.4-1 (Cont'd)

NAME SVC_Synchronization_Processor (FCMSSYNC)

NO.	ITEM	ID	ACT	SOURCE	DESTINATION	ASSEMBLER NAME	MML	D	C
20	Discrete_Bit_GPC_N+2_Sync_Code_3	&005.23	R		363,364				
21	Discrete_Bit_GPC_N+3_Sync_Code_3	&005.24	R		363,364				
22	Discrete_Bit_GPC_N+4_Sync_Code_3	&005.25	R		363,364				

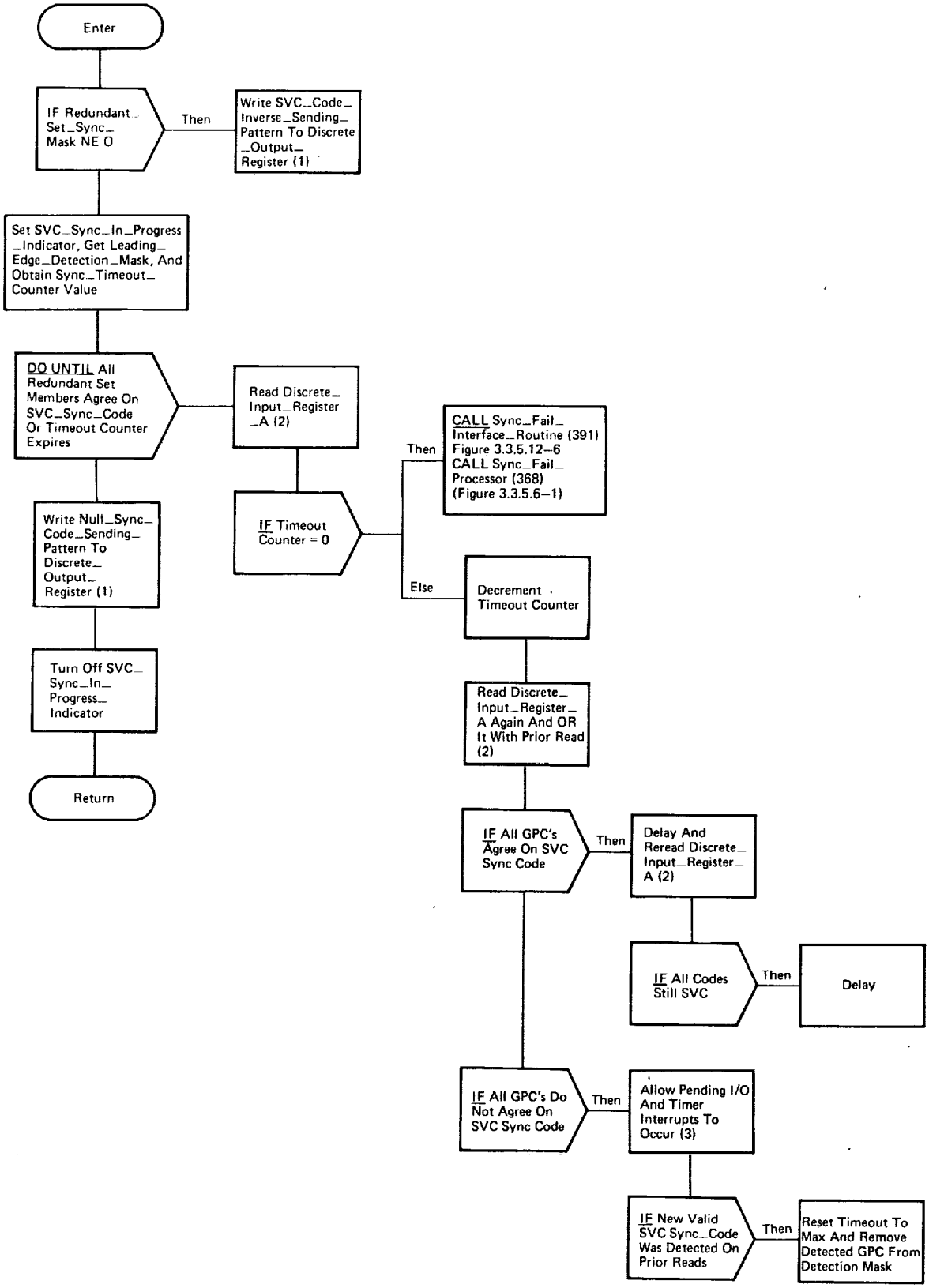


Figure 3.3.5.4-1. SVC_Synchronization_Processor (FCMSSYNC)

3.3.5.5 Timer_Synchronization_Processor (FCMTSYNC) (364)

The Timer_Synchronization_Processor (FCMTSYNC) provides redundant set synchronization prior to the execution of all PC2 driven processing except that provided by the Common_Set_Synchronization_Processor (FCMCSYNC). In addition, certain mass memory I/O operations related to status determination and clearing can cause variations in redundant set processing times. For these operations, this processor provides re-synchronization.

a. Control Interface-

1. CALLed by (280) Mass_Memory_Manager (FIOMMMGR).
2. CALLed by (142) TQE_Expiration_Processor (FPMIHPC2).

b. Input - Inputs are shown in Table 3.3.5.5-1

c. Process Description - The Timer_Synchronization_Processor obtains loop control constants and immediately enters the sync loop. This loop is maintained until the Timer sync codes are exchanged with all redundant set members or until execution of the loop exceeds a pre-determined number of microseconds. Prior to entry to the Timer_Synchronization_Processor, the caller will have placed the Timer_Code_Inverse_Sending_Pattern in the GPC_Sync_Code_Command_Discrete bit positions of the Discrete_Output_Register. This makes the code immediately available to the Timer_Synchronization_Processor executing in other GPC's.

Sync Codes are obtained from other GPC's by reading the Discrete_Input_Register_A (see table 3.3.5.5-1, items 6-16). Stability of the bit values is assured by performing two reads and OR'ing the resulting bit values together. Following each pair of reads, GPC's whose bit positions in the Discrete_Input_Register_A contain the timer sync pattern are removed from the Leading_Edge_Detection_Mask. Each time a new valid sync pattern is detected, the Sync_Timeout_Counter is reset to its maximum value.

Once Timer_Sync_Code discrete patterns have been received from all redundant set GPC's, a delay is performed to allow sync code detection by other GPC's and the sync loop is terminated. After loop termination, the Timer_Sync_Code is removed from the Discrete_Output_Register by replacing it with the null sync code.

In the event of sync loop timeout before valid Timer_Sync_Code discretes have been received from all redundant set members, the Sync_Fail_Interface_Routine and the Sync_Fail_Processor are called to remove the GPC's that failed to exhibit the sync pattern from the redundant set and perform any required bus reconfiguration.

Timer_Synchronization_Processor flow is shown in Figure 3.3.5.5-1.

d. Output - Outputs are shown in Table 3.3.5.5-1.

e. Module References -

1. (391) Sync_Fail_Interface_Routine (FCMSFINT) is called.
2. (368) Sync_Fail_Processor (FCMSFAIL) is called.

f. Module Attributes - Program

g. Template References - N/A

h. Error Checks - None

i. Constraints and Assumptions - The timing of the sync loop is critical to the logic employed. Changes to the loop must be assessed for impact to the loop timeout value.

j. Detailed Implementation -

1. Issue a Program Controlled Input (PC) command to obtain the current value of the sync bits in the Discrete_Input_Register_A.

2. Issue a Program Controlled Output (PC) command to place the Null_Sync_Code_Sending_Pattern in the Discrete_Output_Register.

DATA TABLE 3.3.5.5-1

NAME Timer_Synchronization_Processor (FCMTSYNC)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Redundant_Set_Sync_Mask	Q001.47	I	305 390	See App. E	TCVTRSSM			
2	Timer_Sync_Code	Q001.62	I		364	TCVTTIMC			
3	Sync_Timeout_Counter	I580.02	Z	205	363 364	CZ2VSYNC			
4	Leading_Edge_Detection_Mask	Q001.65	I		See App. F	TCVTKM			
5	Discrete_Bit_GPC_N+1_Sync_Code_1	&005.14	R		363,364, 368				
6	Discrete_Bit_GPC_N+2_Sync_Code_1	&005.15	R		363,364, 368				
7	Discrete_Bit_GPC_N+3_Sync_Code_1	&005.16	R		363,364, 368				
8	Discrete_Bit_GPC_N+4_Sync_Code_1	&005.17	R		363,364, 368				
9	Discrete_Bit_GPC_N+1_Sync_Code_2	&005.18	R		363,364, 368				
10	Discrete_Bit_GPC_N+2_Sync_Code_2	&005.19	R		363,364, 368				
11	Discrete_Bit_GPC_N+3_Sync_Code_2	&005.20	R		363,364, 368				
12	Discrete_Bit_GPC_N+4_Sync_Code_2	&005.21	R		363,364, 368				
13	Discrete_Bit_GPC_N+1_Sync_Code_3	&005.22	R		363,364				
14	Discrete_Bit_GPC_N+2_Sync_Code_3	&005.23	R		363,364				
15	Discrete_Bit_GPC_N+3_Sync_Code_3	&005.24	R		363,364				
16	Discrete_Bit_GPC_N+4_Sync_Code_3	&005.25	R		363,364				
17	Null_Sync_Code_Sending_Pattern	Q001.73	I		See App. E	TCVTNULS			

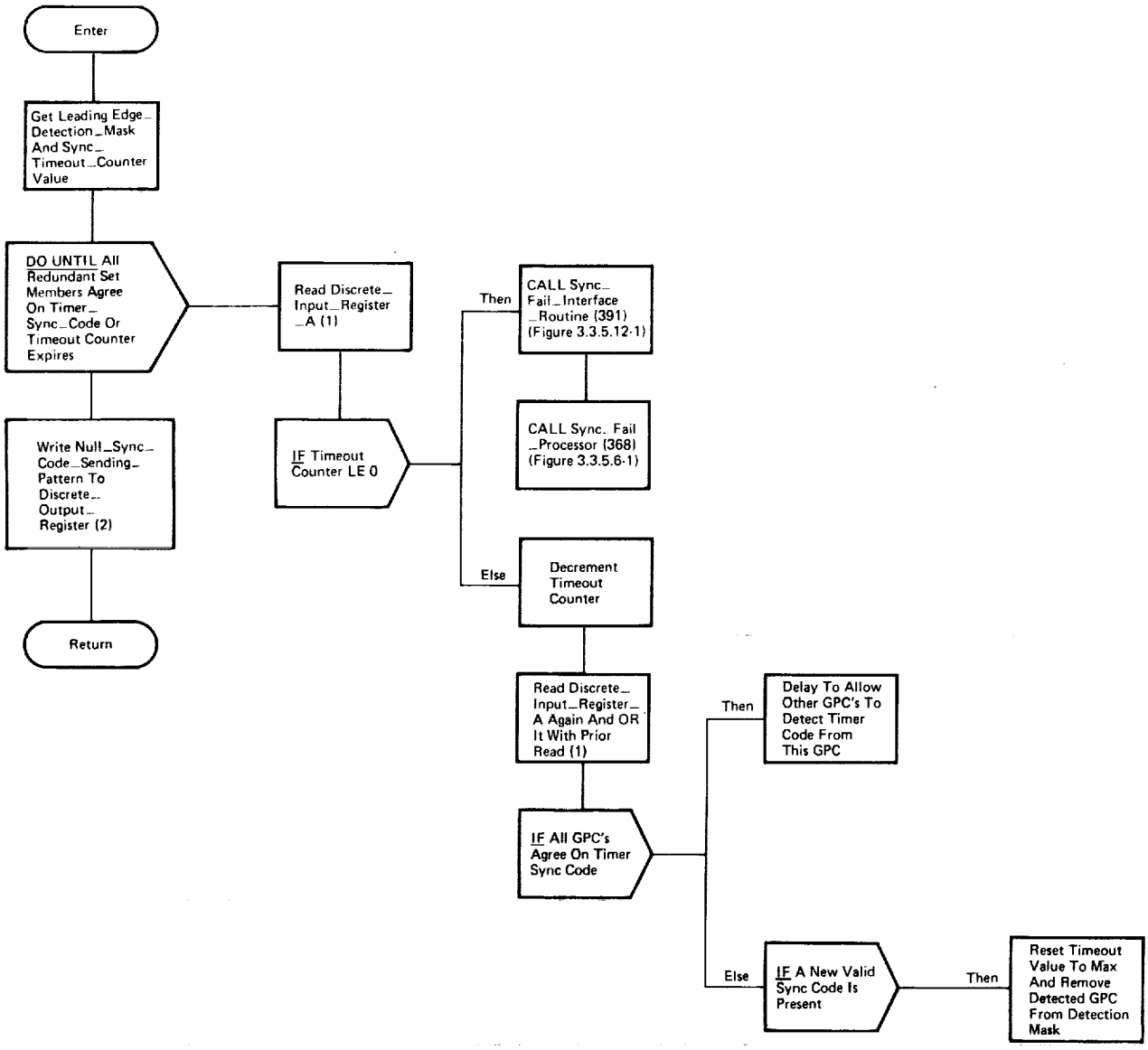


Figure 3.3.5.5-1. Timer_Synchronization_Processor (FCMTSYNC)

3.3.5.6 Sync_Fail_Processor (FCMSFAIL) (368)

The Sync_Fail_Processor (FCMSFAIL) controls the processing required in support of a forced or actual GPC synchronization failure.

a. Control Interface -

1. CALLed by (361) Normal_SSIP_Synchronization_Processor (FCMCSYNC)
2. CALLed by (362) I/O_Synchronization_Processor (FCMISYNC)
3. CALLed by (364) Timer_Synchronization_Processor (FCMTSYNC)
4. CALLed by (363) SVC_Synchronization_Processor (FCMSSYNC)
5. CALLed by (220) I/O_Completion_Processor (FIOCMPLT)
6. CALLed by (244) Level_C_I/O_Error_Interrupt_Handler (FIOERRLC)
7. CALLed by (100) SVC_Handler (FPMSVC)

b. Input - Register-0, bits 16-31 are zero to indicate Force_Fail_to_Sync entry. Other FCOS entries set Register-0 as follows:

Bit 0 - type of sync
    1 = common set
    0 = redundant set
Bit 11-15 represent GPC's 1-5 In each bit position;
    1 = sync failure
    0 = no sync failure
Bit 31- set to 1 to show normal (non force fail) entry.
All other bits must be zero.

c. Process Description - The Sync_Fail_Processor saves the current Redundant_Set_Mask and the GPC_Prime_ID so that they may be restored if pre-maturely modified by early routine logic. Input parameters are then modified to eliminate the difference between FCOS and SVC entries. For fail_self entries the fail mask is obtained from the current Common_Set_Mask (thus fail self is actually accomplished by assuming the failure of all but self).

Modified (as above) inputs are then examined and if a GPC failure is not indicated (can occur on fail self entry with a zero Common_Set_Mask), the IOP transmitter status if brought up to date and the processor returns. If a failed GPC is indicated by input parameters, the Redundant_Set_Mask and Common_Set_Mask are updated to reflect removal of the failing GPC(s). Note that if failure indicaters (input parameters) that self is the only survivor and that a majority of GPC's are being failed, it is assumed that self has failed, and the Fail_Sync_Flag of Initialization_Flags is set before updating the Redundant_Set_Mask and Common_Set_Mask.

After mask update, the Bus/Data_Path_Mask_Manager is called to set bus masks for busses controlled by the failing GPC. Concurrent with this, a mask is constructed for failure annunciation of failing GPC's. This mask is stored following bus mask update.

Next, the Data_Path_Masks for all remaining Redundant set GPC's (as indicated by that GPC's Redundant_Set_Mask) are logically combined to form BTU_Port_Masks for each IUA. The Sync_Mask_Build_Routine is then called to construct a new self sync mask. The Sync Discretes are then read from the Discrete_Input_Register_A for use in determination of a new (if necessary) prime GPC.

If GPC prime failed with 000 discretes (Halted or powered off) or if a failure occurred of prime reducing the redundant set from more than two members, a new prime is selected (if possible). If a new prime is selected, the Bus_Reconfiguration_Pre-Processor is called to assign or release prime buses, (string 5) according to whether or not GPC self was prime in the old redundant set.

Finally, if voter latches are inhibited and if GPC self was an old redundant set member, CAM lights are driven by invoking the Start_MSC_Processor to start up the Sync_Fail_CAM_MSC_Processor. The software CAM_Status_Votes in the GST_Status_Flags is updated. Sync_Fail_Processor flow is shown in Figure 3.3.5.6-1, 3.3.5.6-2, and 3.3.5.6-3.

d. Output - Outputs are shown in Table 3.3.5.6-1.

e. Module References -

1. (355) Bus/Data_Path_Mask_Manager (FCMBMASK) is called.

2. (390) Sync_Mask_Build_Routine (FCMSMASK) is called.

3. (375) Fault_Detection_Identification (FCMFDI) is called at entry label FCMCVTM (Voter_Output_Conversion, 375.1)

4. (210) Start_MSC_Processor (F10STMSC) is called to start the Sync_Fail_CAM_MSC_Processor.

5. (392) RS_Prime_Switching_Algorithm (FCMRSALG) is called.

6. (350) Bus_Reconfiguration_Pre-Processor (FCMBMAN) is called

f. Module Attributes - Program

g. Template References - N/A

h. Error Checks - None

i. Constraints and Assumptions - None

j. Detailed Implementation -

1. The Internal_Parameter_List_Buffer is a shared area used for constructions and passage of Bus and Mask management request parameter lists.

2. A Program Controlled (PC) I/O command is issued directly to perform the indicated operation.

3. This is a closed internal subroutine used to build a mask for voter output. Its code was subroutinized and left in the parent program rather than to remove it and make it a separate program.

4. Fail self is accomplished by assuming failure of all others and thus terminating all discrete and ICC communication with other GPC's. This causes them to unilaterally fail self.

DATA TABLE 3.3.5.6-1

NAME Sync_Fail_Processor (FCMSFAIL)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Bus_Management_Work_Area	0003	O			TBMWSTRT			
2	GPC_Self_Status_Table_Address	Q001.58	I	300	See App. E	TCVTGST			
3	Common_Set_Mask	I010.12	I,O	See App. E.		TGSTCSM	V91M2087P,2121P 2155P, 2200P	X	
4	Redundant_Set_Mask	I010.13	I,O	See App. E		TGSTRSM	V91M2070P,2104P 2138P, 2172P	X	
5	Old_Redundant_Set_Mask	0003.2	O	368	368,392	TBMWRMSK			
6	GPC_Prime_ID	I050	Z	351,700, 820,880	See App. E	CZ2VGPCP			
7	Prime_At_Fail_To_Sync	I340	Z		368	CZ2VPRIM			
8	IOP_Transmitter_State	I010.06	R,O	351,355, 830	See App. E	TGSTTCCT	V91M8706P,8740P 8774P, 8808P	X	
9	Common_Set_GPC_Count	Q001.76	I	390	220 368	TCVTCGCS			
10	Fail_Sync_Flag	Q001.28	I	300,305 368	305,361 362	TCVTIFLG			
11	Internal_Parameter_List_Buffer	0003.5	O	351 368		TBMWBDPM			
12	Fail_To_Sync_GPC	I330	Z	368	485,830, 950	CZ2BFSS			
13	FPMPT_Hdr_Bits	T070	O	368		CDLBERRM			
14	Data_Path_Masks	I010.11	I	244,290, 355	148,290 665	TGSTDPM	See App. E	X	
15	BTU_Port_Masks	#024	I,O	290	205,290, 355,368	TFCMBPM	See App. E	X	X
16	Redundant_Set_GPC_Count	Q001.75	O	390	220 368	TCVTCGRS			
17	Redundant_Set_Sync_Mask	Q001.47	I	305 390	See App. E	TCVTRSSM			
18	Common_Set_Sync_Mask	Q001.48	I	305 390	361,362, 368	TCVTCSSM			
19	Discrete_Bit_GPC_N+1_Sync_Code_1	&005.14	R		363,364 368				

DATA TABLE 3.3.5.6-1 (Cont'd)

NAME Sync_Fail_Processor (FCMSFAIL)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
20	Discrete_Bit_GPC_N+2_Sync_Code_1	&005.15	R		363,364 368				
21	Discrete_Bit_GPC_N+3_Sync_Code_1	&005.16	R		363,364 368				
22	Discrete_Bit_GPC_N+4_Sync_Code_1	&005.17	R		363,364 368				
23	Discrete_Bit_GPC_N+1_Sync_Code_2	&005.18	R		363,364 368				
24	Discrete_Bit_GPC_N+2_Sync_Code_2	&005.19	R		363,364 368				
25	Discrete_Bit_GPC_N+3_Sync_Code_2	&005.20	R		363,364 368				
26	Discrete_Bit_GPC_N+4_Sync_Code_2	&005.21	R		363,364 368				
27	Discrete_Bit_GPC_N+1_Sync_Code_3	&005.22	R		363,364				
28	Discrete_Bit_GPC_N+2_Sync_Code_3	&005.23	R		363,364				
29	Discrete_Bit_GPC_N+3_Sync_Code_3	&005.24	R		363,364				
30	Discrete_Bit_GPC_N+4_Sync_Code_3	&005.25	R		363,364				
31	GPC_ID	#001	I	300	See App. E	TFCMID			
32	GST_Address_Table	@004	I	368	See App. E	FPMGSTAD			
33	Current_Ops	I010.02	I	560,820, 880	See App. E	TGSTPOPS,TGSTGOPS, TGSTSOPS			
34	GST_Status_Flags	I010.14	I	See App. E	See App. E	TGSTIND		X	X
35	Sync_Fail_Set_CAM_Mask	Q001.95	O	368	369	TCVTSETC			
36	Sync_Fail_Reset_CAM_Mask	Q001.94	O	368	369	TCVTRESC			
37	Bus_Reconfiguration_Parameter_List	S036	O		350 368				
38	Bus_Reconfiguration_Request_Type	S036.1	O		350 368	TBRPTYPE			

DATA TABLE 3.3.5.6-1 (cont'd)

NAME Sync_Fail_Processor (FCMSFAIL)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
39	Bus_Reconfiguration_SVC_Number	S036.2	0		100 368				
40	LDB_Toggling_Indicator	S036.3	0		350 368	TBRPSTAT			
41	Reconfiguration_Variable_1_Field	S036.4	0		350 368	TBRPVAR1			
42	Reconfiguration_Variable_2_Field	S036.5	0		350 368	TBRPVAR2			
43	Mask_Management_Parameter_List	S035	0		355 368				
44	Mask_Management_Request_Type	S035.1	0	351	355 368				
45	Mask_Management_SVC_Number	S035.2	0		100 368				
46	Mask_Management_Variable_Field	S035.3	0		355 368				

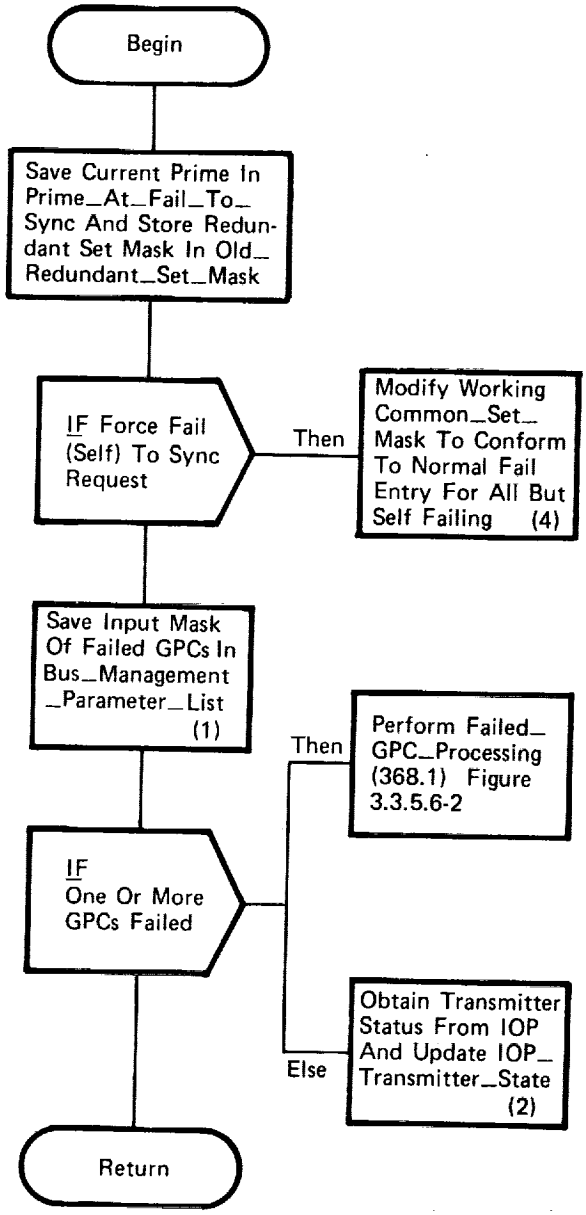


Figure 3.3.5.6-1. Sync_Fail_Processor (FCMSFAIL)

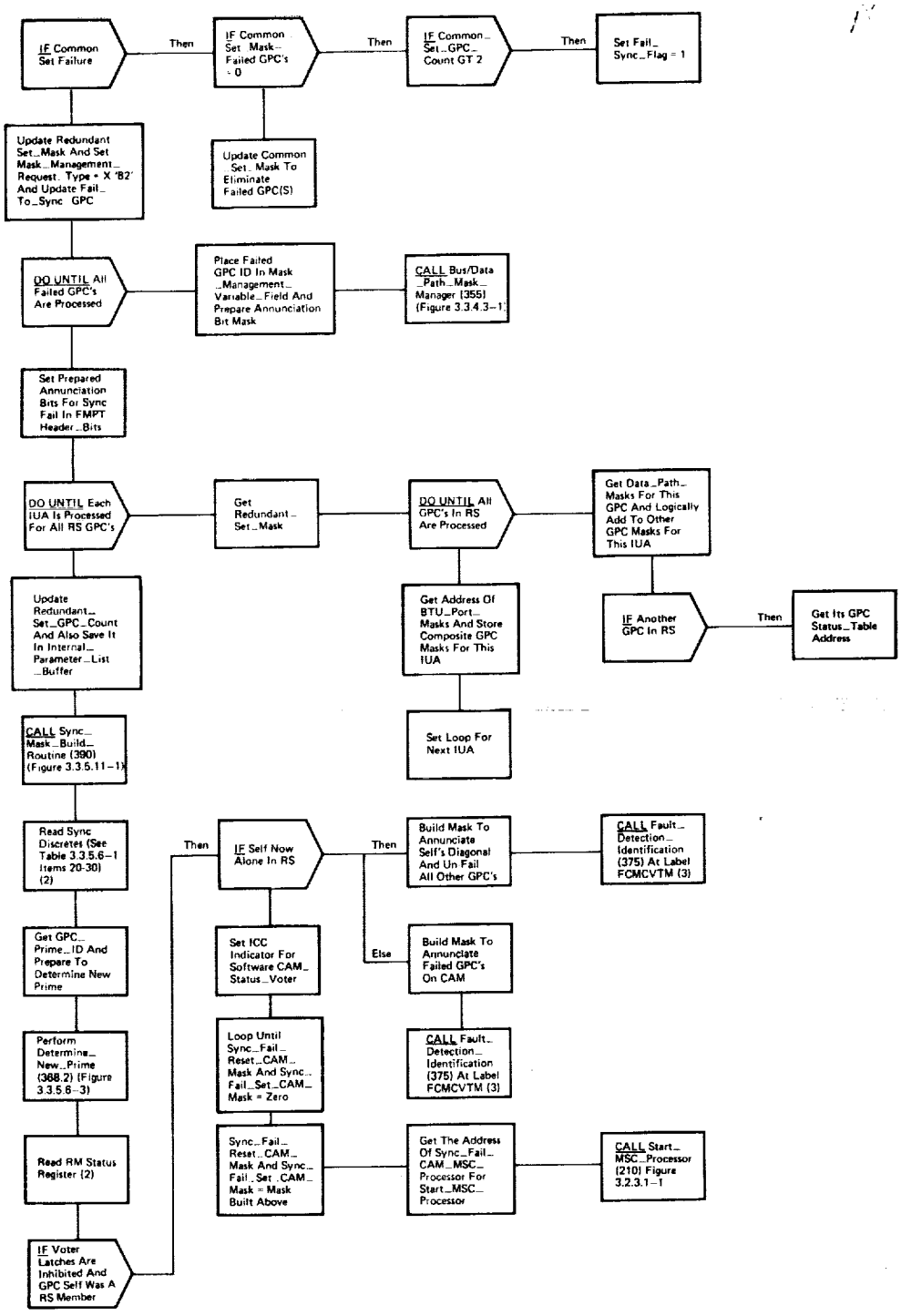


Figure 3.3.5.6-2. Sync_Fail_Processor Failed_GPC_Processing (368.1)

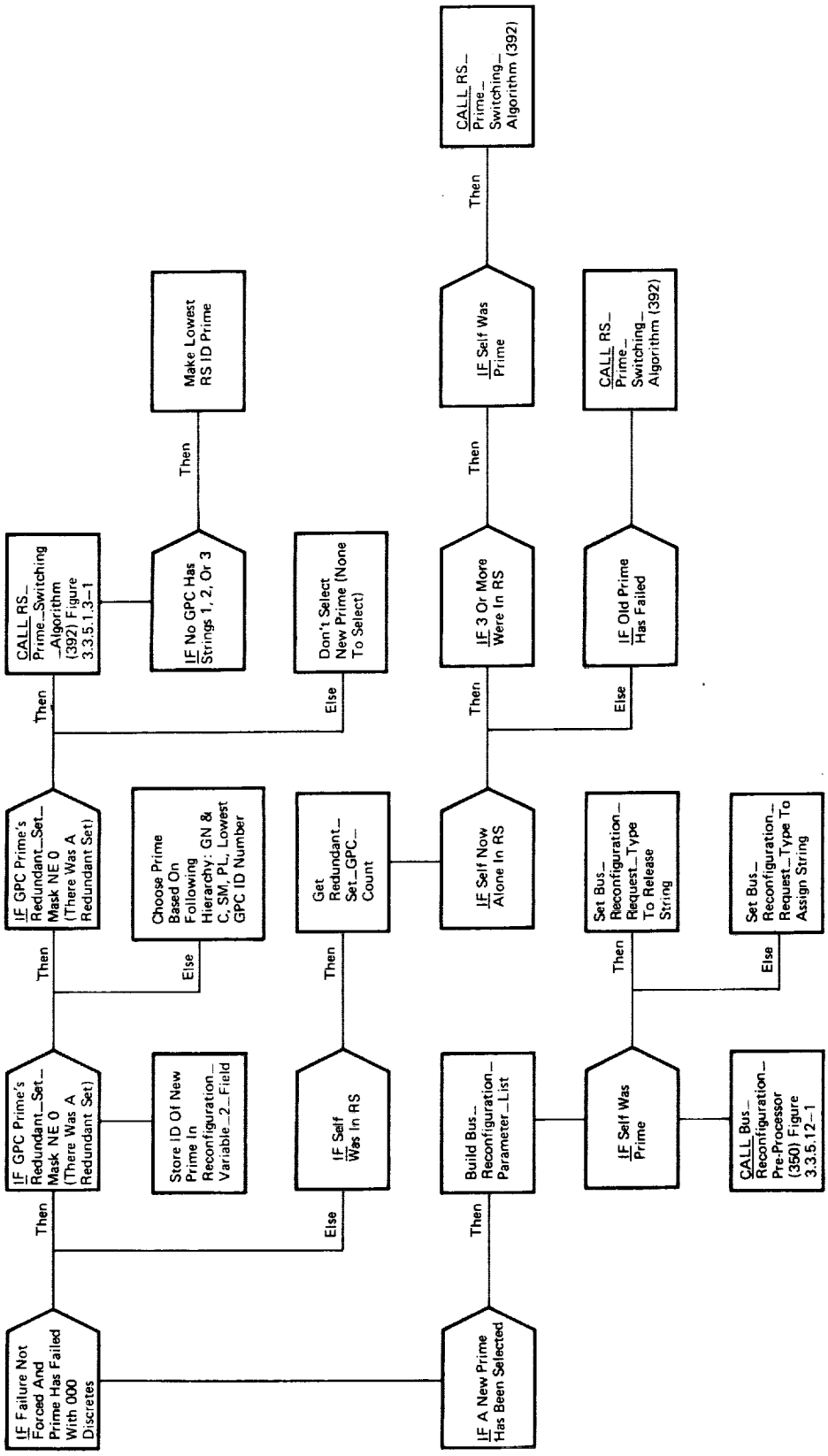


Figure 3.3.5.6-3. Sync_Fail_Processor Determine_New_Prime (368.2)

3.3.5.7 Sync_Fail_CAM_MSC_Processor(FCMSFCAM)(369)

The Sync_Fail_CAM_MSC_Processor performs CAM fail discrete annunciation for the Sync_Fail_Processor.

a. Control Interface - Started by (210) Start_MSC_Processor (FIOSTMSC) on behalf of (368) Sync_Fail_Processor (FCMSFAIL)

b. Inputs - See Table 3.3.5.7-1

c. Processing Description - The GPC fail discretes (CAM) are reset and set as required by the Sync_Fail_Reset_CAM_Mesh and Sync_Fail_Set_CAM_Mask, respectively. Then The Sync_Fail_Set_CAM_Mask and the Sync_Fail_Set_CAM_Mask are set to zero. The control flow for this module is presented in Figure 3.3.5.7-1.

d. Output - The GPC fail discretes may be changed. See Table 3.3.5.7-1.

e. Module References - None

f. Module Attributes - MSC Program

g. Template References - N/A

h. Error checks - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.3.5.7-1

NAME Sync_Fail_CAM_MSC_Processor (FCMSFCAM)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Sync_Fail_Reset_CAM_Mask	Q001.94	I,O	368	369	TCVTRESC			
2	Sync_Fail_Set_CAM_Mask	Q001.95	I,O	368	369	TCVTSETC			

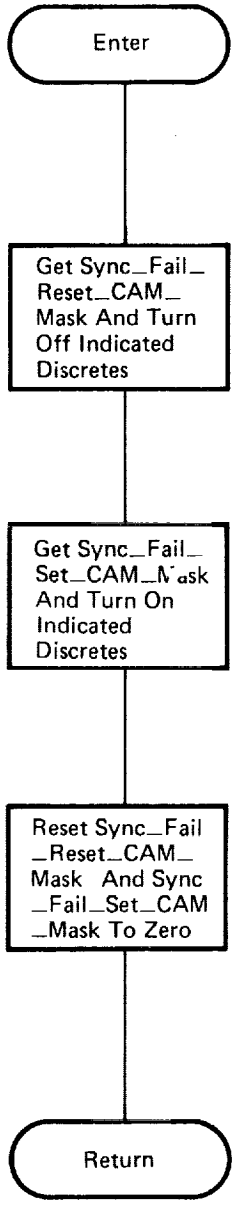


Figure 3.3.5.7-1. Sync_Fail_CAM_MSC_Processor (FCMSFCAM)

3.3.5.8 Fault_Detection_Identification (FCMFDI) (375)

The Fault_Detection_Identification program (FCMFDI) compares the difference of a GPC Sum_Word between GPC self and other GPC's against an allowable Sum_Word_Delta and annunciates compare failures via the CAM lights.

a. Control Interface

1. CALLed by (220) I/O_Completion_Processor (FIOCMPLT)
2. CALLed by (368) Sync_Fail_Processor (FCMSFAIL) at entry FCMCVTM (Voter_Output_Conversion)

b. Input - See Table 3.3.5.8-1

c. Process Description - If the Redundant_Set_Mask is null, no processing is performed. Otherwise, the Set_CAM_Mask and Reset_CAM_Mask words are set to zero and a working mask is prepared with indications for any GPC whose Sum_Word differs from GPC self's Sum_Word by more than Sum_Word_Delta. If all other GPC's are failed, it is assumed that GPC self is at fault, and the working mask is modified to reflect a GPC self failure. Then, for each RS GPC, the following is performed.

The CAM_Counters word is incremented if the GPC has failed, otherwise, it is set to zero. If the error counter reaches the threshold value of CAM_NOGO_Counter, the GST_Status_Flags and FMPT_HDR_Bits are set to reflect the failure. If GPC self has failed, all CAM lights are set to reflect a self failure (other GPC bits in self's column are reset). Otherwise the failing GPC is reflected in the Set_CAM_Mask word unless its failure has been previously annunciated.

Finally, if any Set_CAM_Mask bits are set, the failing GPC bits are reformatted for CAM output and the Start_MSC_Processor is CALLed to perform actual annunciation.

d. Output - CAM lights may be turned on. See Table 3.3.5.8-1.

e. Module References -

1. (376) Fail_Discretes_MSC_Processor (FCMSVOTE) is referenced to prepare it for execution by (2) below.

2. (210) Start_MSC_Processor (FIOSTMSC) is CALLed to cause execution of (1) above.

f. Module Attributes - Program

g. Template References - none

h. Error Checks - none

i. Constraints and Assumptions - none

j. Detailed Implementation -

1. A program controlled (PC) I/O instruction is issued.

2. FIOSTMSC is used to accomplish the immediate execution of FCMSVOTE by the MSC.

DATA TABLE 3.3.5.8-1

NAME Fault_Detection_Identification (FCMFDI)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Set_CAM_Mask	Q001.41	O	340,375	376	TCVTCAMS			
2	Reset_CAM_Mask	Q001.42	O	340,375	376	TCVTCAMR			
3	ICC_Buffer_Address_Table	0005	I		142,149	FCMICCAD			
4	Sum_Word_Delta	I580.01	Z		375	CZ2VSUMW			
5	Sum_Word	I610.01	Z	750	375	TICCRMSW			
6	GST_Status_Flags	I010.14	Z	See Appendix E		TGSTIND TGSTIND		X	X
7	ICC_Status_Flags	I320	Z	See Appendix E		CZ3VIF1,CZ2VIF2			
8	Redundant_Set_Mask	I010.13	I	See Appendix E		TGSTRSM		X	
9	GPC_ID	#001	I	300	See App. E	TFCMID			
10	CAM_Counter_Table_Address	Q001.52	I	300	375	TCVTCNTA			
11	CAM_NOGO_Counter	I580.03	I,O	375	375	CZ2VCAMN			
12	FMPT_HDR_Bits	T070	I,O	See Appendix E		CDLBERRM			
13	CAM_Counter_ICC_Indicator_Mask	Q001.55	I	300	375	TCVTCNTM			
14	ICC_GST_Status	I320.09	O	See App. E	142				
15	CAM_Counters	I240	I/O	375,820, 880	485,660, 665	CZ2CNTG1			
16	CAM_Status_Votes	I010.18	I/O	375,820, 880	620	TGSTIND			X

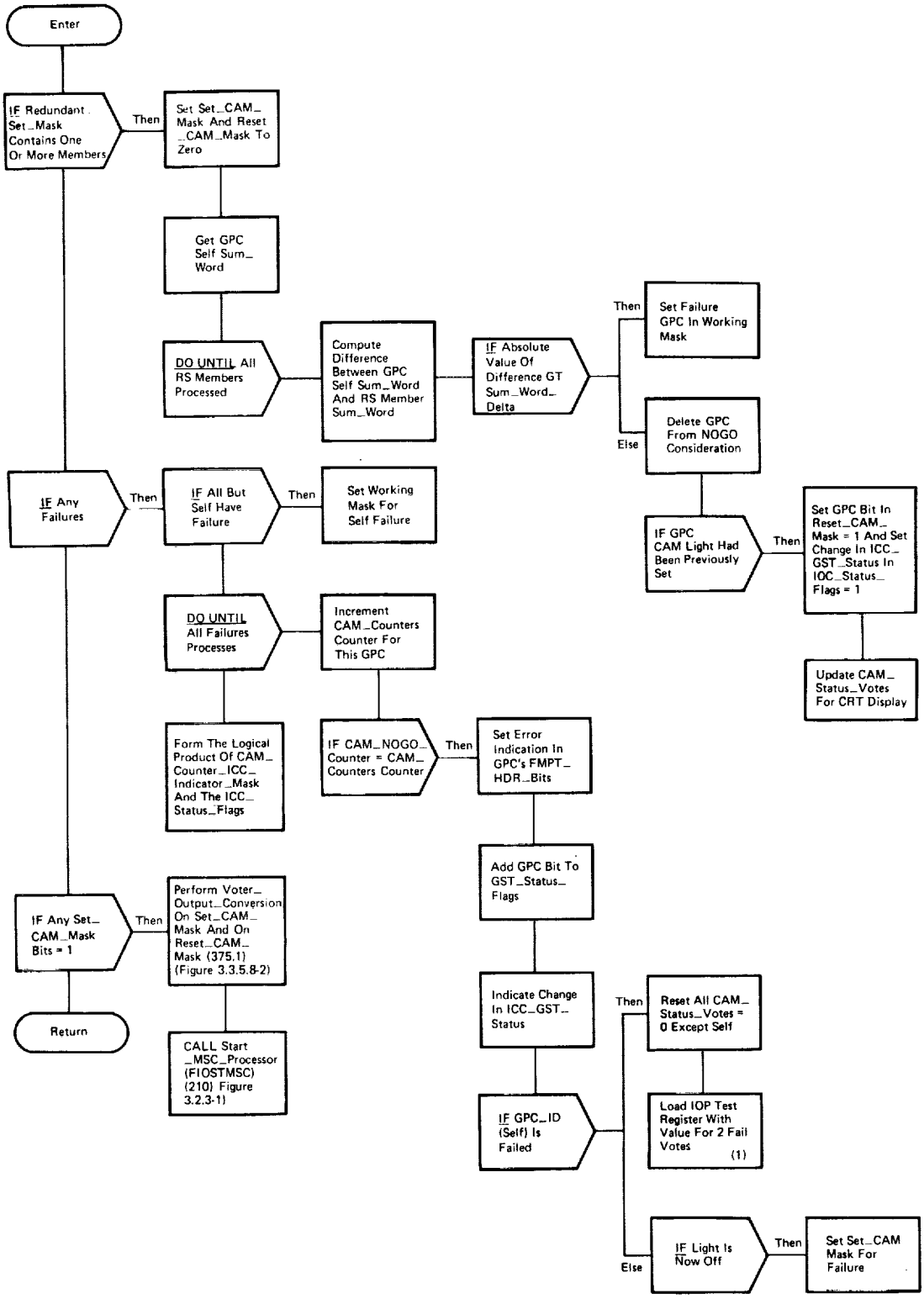


Figure 3.3.5.8-1. Fault_Detection_Identification (FCMFDI)

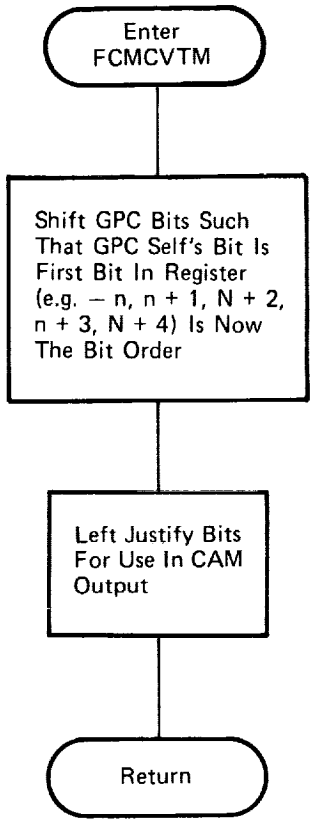


Figure 3.3.5.8-2. Fault_Detection_Identification Voter_Output_Conversion (375.1)

3.3.5.9 Fail_Discretes_MSC_Processor (FCMSVOTE) (376)

The Fail_Discretes_MSC_Processor performs Fail Discretes annunciation.

a. Control Interface -

Started by (210) Start_MSC_Processor (FIOSTMSC) on behalf of (340) Miscellaneous_CM_Request_Processor (FCMSVC) and (375) Fault_Detection_Identifcation (FCMFDI).

b. Inputs - See table 3.3.5.9-1.

c. Processing Description - The GPC fail discretes (CAM) are reset and set as required by the Reset_CAM_Mask and Set_CAM_Mask, respectively. Then the Reset_CAM_Mask and the Set_CAM_Mask are set to zero. The control flow for this module is presented in Figure 3.3.5.9 - 1.

d. Output - The GPC fail discretes may be changed, also see Table 3.3.5.9-1.

e. Module References - None

f. Module Attributes - MSC Program

g. Template References - N/A

h. Error Checks - None

i. Constraints and Arsumptions - None

j. Detailed Implementation - None

DATA TABLE 3.3.5.9-1

NAME Fail_Discretes_MSC_Processor (FCMSVOTE)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Set_CAM_Mask	Q001.41	I,O	340,375	376	TCVTCAMS			
2	Reset_CAM_Mask	Q001.42	I,O	340,375	376	TCVTCAMR			

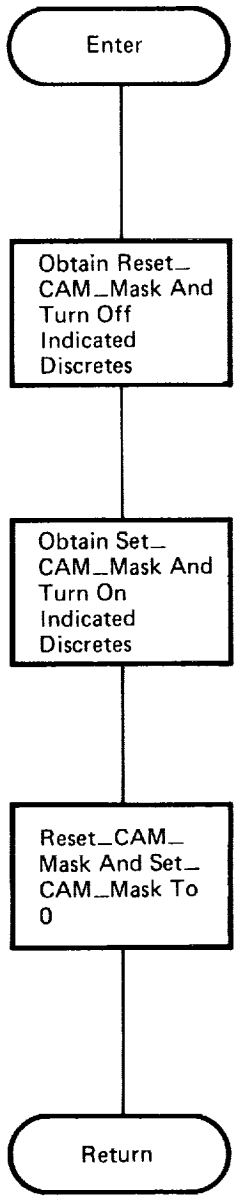


Figure 3.3.5.9-1. Fail_Discrete_MSC_Processor (FCMSVOTE)

3.3.5.10 GPC_Discrete_Redundancy_Manager (FCMDSCRM)(380)

To Be Provided

3.3.5.11 Sync_Mask_Build_Routine(FCMSMASK)(390)

To Be Provided

3.3.5.12 Sync_Fail_Interface_Routine(FCMSFINT)(391)

To Be Provided

3.3.5.13 RS_Prime_Switching_Algorithm(FCMRSALG)(392)

RS_Prime_Switching_Algorithm is used to choose a new prime GPC when prime switch is necessary within the redundant set.

a. Control Interface

1. CALLed by (351) Bus_Reconfiguration_Processor (FCMBUSCM)
2. CALLed by (368) Sync_Fail_Processor (FCMSFAIL)

b. Input - Register 4 contains the GPC ID of GPC not to be prime when CALLed from Sync_Fail_Processor ((FCMSFAIL).

Register 1 Bits 0-15 contains the address of the Bus_Management_Work_Area. See table 3.3.5.13-1.

c. Process Description - RS_Prime_Switching_Algorithm (FCMRSALG) will select a new GPC to be prime from among those GPC's identified in the redundant set.

The ID of the GPC that has failed is obtained from input Register 4. The address of GPC__Self_Status_Table_Address is obtained. The ID of the GPC that failed is compared to self's ID. If equal the Old_Redundant_Set_Mask is used. If not equal an updated Redundant_Set_Mask is formed using GPC_ID and Redundant_Set_Mask.

For each GPC starting with the lowest ID number in the Redundant_Set_Mask check if the GPC is in command of strings 1, 2 or 3 (as indicated in the Current_String_Assignment_Table). If it is, indicate this GPC to be new prime (place in register 0 for return to caller) and stop search.

The control flow for this module is shown in Figure 3.3.5.13-1.

d. Outputs -

1. Register 0 contains the GPC ID of GPC to be prime in integer form, value of 1, 2, 3 or 4.

e. Module References - None

f. Module Attributes - Program

g. Template References - N/A

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - N/A

DATA TABLE 3.3.5.13-1

NAME RS_Prime_Switching_Algorithm (FCMRSALG)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	GPC_ID	#001	I	300	See App. E	TFCMID			
2	Current_String_Assignment_Table	I080	I	351,368	350,392, 830	CZ2VSAT	V91X1721X		X
							V91X1722X		
							V91X1723X		
							V91X1724X		
							V91X1725X		
3	GPC_Self_Status_Table_Address	Q001.58	I	300	See App. E	TCVTGST			
4	Old_Redundant_Set_Mask	0003.2	I	368	368,392	TBMWRMSK			
5	Bus_Management_Work_Area	0003	I			TBMWSTRT			
6	Redundant_Set_Mask	I010.13	I	See Appendix E		TGSTRSM	V91M2070P	X	
							V91M2104P		
							V91M2138P		
							V91M2172P		

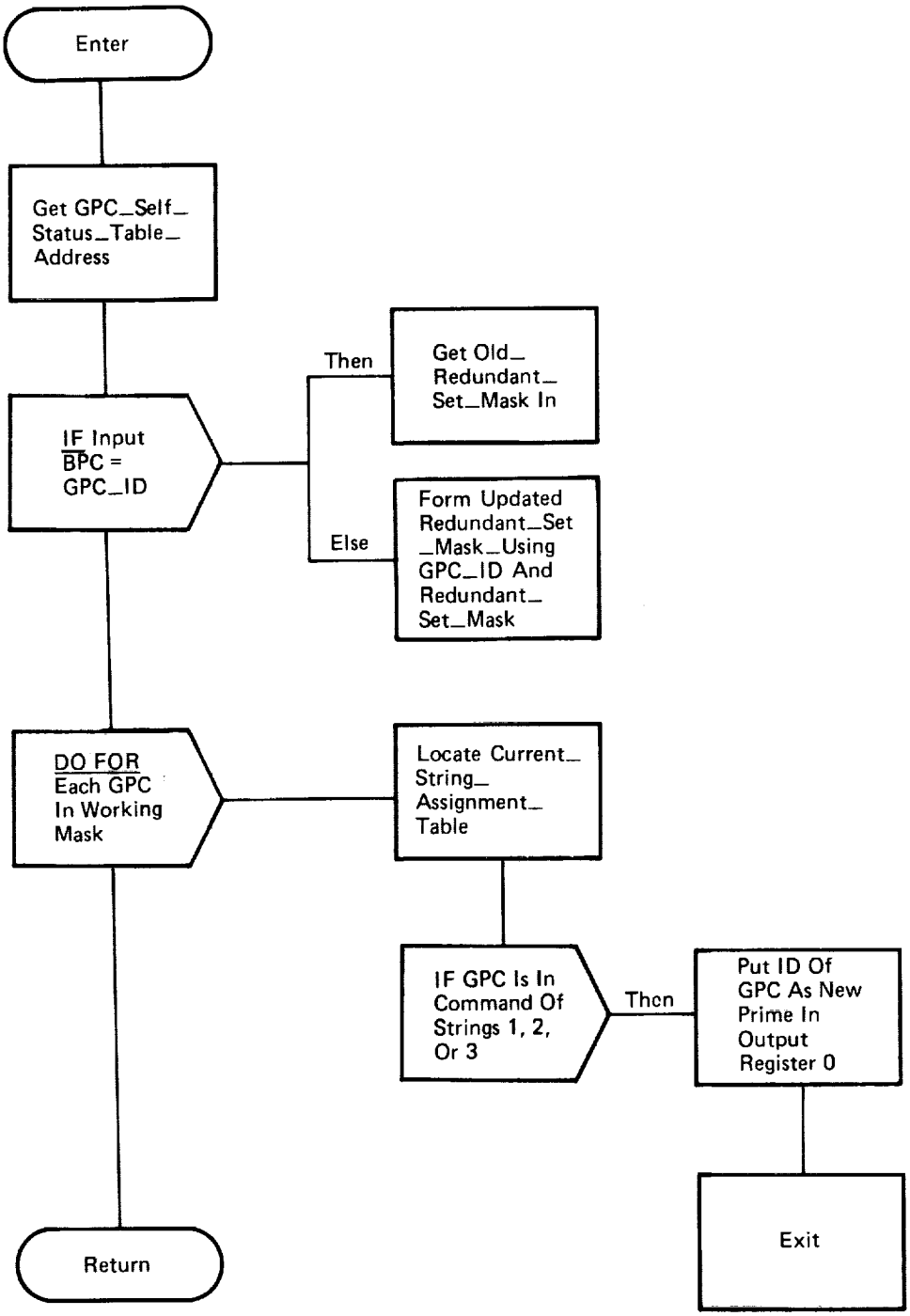


Figure 3.3.5.13-1. RS_Prime_Switching_Algorithm (FCMRSALG)

CPDS - Detailed Design Cross Reference

APPENDIX A

To Be Provided

Resource Allocation Summary

Appendix B

(To Be Provided)

Appendix C

FCOS Module List

FCOS MODULE LIST

Module ID	Module Name	Assembler Name	Section
100	SVC_Handler	FPMSVC	3.1.1.1
101	Process_Scheduler	FPMSCHED	3.1.1.2
102	Process_Switcher	FPMSWTCH	3.1.1.3
103	Process_Dispatcher	FPMDISP	3.1.1.4
104	Wait_Processor	FPMWAIT	3.1.1.5
105	UPDATE/EXCLUSIVE_Reserve_Processor	FPMRES	3.1.1.6
106	UPDATE/EXCLUSIVE_Release_Processor	FPMREL	3.1.1.7
107	Close_Processor	FPMCLOSE	3.1.1.8
108	Terminate_Processor	FPMTERM	3.1.1.9
109	Cancel_Processor	FPMCANCL	3.1.1.10
110	OPS_Cancel_Processor	FPMOPSCN	3.1.1.11
120	Select_I/O_Processor	FPMSIO	3.1.1.12
130	Hybrid_Dispatching_Routine	#CFPMHYD	3.1.1.13
131	Chain_PCT_Routine	FPMCHPCT	3.1.1.14
132	Release_PCT_Routine	FPMRLPCT	3.1.1.15
133	Free_PCT_Routine	FPMFRPCT	3.1.1.16
140	Timer_Queue_Generator	FPMTMENQ	3.1.2.1
141	GPC_Clock_Expiration_Processor	FPMIHPC1	3.1.2.2
142	TQE_Expiration_Processor	FPMIHPC2	3.1.2.3
143	TQE_Dequeue_Processor	FPMTMDEQ	3.1.2.4
144	Time/Date_Application_Requests_Processor	FPMTMHAL	3.1.2.5
148	MTU_Update_Processor	FPMUPMTU	3.1.2.9

FCOS MODULE LIST

Module ID	Module Name	Assembler Name	Section
149	MTU_Redundancy_Manager	FPMMTURM	3.1.2.7
150	Time_Conversion_SVC_Servicer	FPMTMCVT	3.1.2.8
160	Convert_To_Fixed_Point_Routine	FPMCVTFX	3.1.2.9
161	Convert_To_Floating_Point_Routine	FPMCVTFL	3.1.2.10
162	Current_GMT_Routine	FPMGMTIM	3.1.2.11
163	Program_Counter_2_Update_Routine	FPMITUPD	3.1.2.12
164	Fixed_To_MTU_Format_Conversion_Routine	FPMFXMTU	3.1.2.13
165	MTU_To_Fixed_Format_Conversion_Routine	FPMMTUFX	3.1.2.14
166	Chain_TQE_Routine	FPMCHTQE	3.1.2.15
167	Expiration_Time_Update_Routine	FPMUPTOX	3.1.2.16
170	Set_Event_Processor	FPMSET	3.1.3.1
171	Reset_Event_Processor	FPMRESET	3.1.3.2
172	Signal_Event_Processor	FPMSIGNL	3.1.3.3
173	Event_Queue_Generator	FPMEVENQ	3.1.3.4
174	Event_Evaluator	FPMEVAL	3.1.3.5
175	EQE_Dequeue_Processor	FPMEVDEQ	3.1.3.6
180	Program_Interrupt_Handler	FPMIHPGM	3.1.4.1
181	Instruction_Monitor_Interrupt_Handler	FPMIHIM	3.1.4.2
182	Process_Error_Recovery_Processor	FPMSDERR	3.1.4.3
183	Process_Error_Logger	FPMERLOG	3.1.4.4
184	Forced_Close_Processor	FPMFCLOS	3.1.4.5
185	Application_Error_Number_Request_Processor	FPMSTAT	3.1.4.6
191	Idle_Time_Processor	FPMIDLE	3.1.5

FCOS MODULE LIST

Module ID	Module Name	Assembler Name	Section
200	I/O_SVC_Service_Processor	FIOSVC	3.2.1.1
201	Pre-Initialized_I/O_SVC_Processor	FIOSVCP	3.2.1.2
205	IOP_Dispatcher	FIOPDISP	3.2.2
210	Start_MSC_Processor	FIOSTMSC	3.2.3
220	I/O_Completion_Processor	FIOCMPLT	3.2.4
225	I/O_Termination_Processor	FIOPURGE	3.2.5
240	Level_A_I/O_Error_Interrupt_Handler	FIOERRLA	3.2.6.1
242	Level_B_I/O_Error_Interrupt_Handler	FIOERRLB	3.2.6.2
244	Level_C_I/O_Error_Interrupt_Handler	FIOERRLC	3.2.6.3
246	I/O_Error_Log_Routine	FIOLGERR	3.2.6.4
250	MSC_Control_Routine	FIOMCNTL	3.2.7.1
251	MSC_I/O_Monitor	FIOMNTR	3.2.7.2
252	MSC_BCE_Reset_Routine	FIOMSETB	3.2.7.3
260	DDU_BCE_Processor	FIODDUPG	3.2.8.1
261	DEU_BCE_Processor	FIODEUPG	3.2.8.2
262	ICC_BCE_Processor	FIOICCPG	3.2.8.3
263	LDB_BCE_Processor	FIOLDBPG	3.2.8.4
264	MDM_BCE_Processor	FIOMDMPG	3.2.8.5
265	Payload_Discretes_BCE_Processor	FIOPDSPG	3.2.8.6
266	PMU_BCE_Processor	FIOPMUPG	3.2.8.7
267	PROM_BCE_Processor	FIOPRMPG	3.2.8.8
268	Common_BCE_Processing	#PFIOECT	3.2.8.9
280	Mass_Memory_Manager	FIOMMMGR	3.2.9.1

FCOS MODULE LIST

Module ID	Module Name	Assembler Name	Section
281	Mass_Memory_MSC_Processor	FIOMMMSC	3.2.9.2
282	Mass_Memory_BCE_Processor	FIOMMUPG	3.2.9.3
283	Mass_Memory_Utility_Write_BCE_Processor	FIOMUWPG	3.2.9.4
290	BTU_Port_Masking_Routine	#CFIOBPM	3.2.10.1
291	Checksum_Generator	#CFIOCGR	3.2.10.2
300	Software_System_Loader	FCMINSSL	3.3.1.1
301	SSL_MM_MSC_Processor	FCMINMSC	3.3.1.2
302	SSL_MM_BCE_Processor	FCMINBCE	3.3.1.3
303	System_Load_Selection	FCMCHOOZ	3.3.1.4
305	Normal_Initialization_Processor	FCMNINIT	3.3.1.5
310	IOP_Initialization_Processor	FCMINIOP	3.3.1.6
320	Overlay_Processor	FCMPOVLY	3.3.2.1
325	Program_Modification_Processor	FCMPMOD	3.3.2.2
330	BCE_Element_Bypass_Processor	FCMBCEMD	3.3.2.3
340	Miscellaneous_CM_Request_Processor	FCMSVC	3.3.3
350	Bus_Reconfiguration_Pre-Processor	FCMBMAN	3.3.4.1
351	Bus_Reconfiguration_Processor	FCMBUSCM	3.3.4.2
355	Bus /Data_Path_Mask_Manager	FCMBMASK	3.3.4.3
360	Initial_SSIP_Synchronization_Processor	FCMASYNC	3.3.5.1
361	Normal_SSIP_Synchronization_Processor	FCMCSYNC	3.3.5.2
362	I/O_Synchronization_Processor	FCMISYNC	3.3.5.3
363	SVC_Synchronization_Processor	FCMSSYNC	3.3.5.4
364	Timer_Synchronization_Processor	FCMTSYNC	3.3.5.5

FCOS MODULE LIST

Module ID	Module Name	Assembler Name	Section
368	Sync_Fail_Processor	FCMSFAIL	3.3.5.6
369	Sync_Fail_CAM_MSC_Processor	FCMSFCAM	3.3.5.7
375	Fault_Detection_Identification	FCMFDI	3.3.5.8
376	Fail Discretes_MSC_Processor	FCMSVOTE	3.3.5.9
380	GPC_Discrete_Redundancy_Manager	FCMDSCRM	3.3.5.10
390	Sync_Mask_Build_Routine	FCMSMASK	3.3.5.11
391	Sync_Fail_Interface_Routine	FCMSFINT	3.3.5.12
392	RS_Prime_Switching_Algorithm	FCMRSALG	3.3.5.13

APPENDIX D

Error Conditions

ALT SYSTEM SOFTWARE ERRORS

MSG #	ERROR CONDITION	SOFTWARE RESPONSE	FMPT ID	TEXT MAJOR	TEXT MIN	CLASS
MM001	CHECKSUM ERROR OR I/O ERROR ON MMU TRANSACTION	REQ REJ	R4	I/O TRANSIENT	MMU	3
MM002	REQUESTED MMU TRANSACTION WHEN MMU IS OFF/BUSY	REQ REJ	M1	MMU OFF/BUSY		3
TM001	AUTOMATIC TIME SOURCE CHANGE	NEXT SOURCE SELECTED	T1	MTU	RM	3
OS001	GPC RM FAIL	TURN ON CAM LIGHTS	G2	GPC	RM	2
OS002	CPU DUTY CYCLE HIGH	NONE	G3	GPC	CPU	3
OS003	GPC1 LOSS OF SYNC	IGNORE SYNC W/FAILED GPC	S1	LOSS OF SYNC	1	3
OS004	GPC2 LOSS OF SYNC	IGNORE SYNC W/FAILED GPC	S2	LOSS OF SYNC	2	3
OS005	GPC3 LOSS OF SYNC	IGNORE SYNC W/FAILED GPC	S3	LOSS OF SYNC	3	3
OS006	GPC4 LOSS OF SYNC	IGNORE SYNC W/FAILED GPC	S4	LOSS OF SYNC	4	3
IO001	I/O ERROR ON FF1/P1	DATA PATH INH	I1	I/O ERROR	FF1	3
IO002	I/O ERROR ON FF1/P2	DATA PATH INH	I1	I/O ERROR	FF1	3

MSG #	ERROR CONDITION	SOFTWARE RESPONSE	FMPT	TEXT MAJOR	TEXT MIN	CLASS
I0003	I/O ERROR ON FF2/P1	DATA PATH INH	I2	I/O ERROR	FF2	3
I0004	I/O ERROR ON FF2/P2	DATA PATH INH	I2	I/O ERROR	FF2	3
I0005	I/O ERROR ON FF3/P1	DATA PATH INH	I3	I/O ERROR	FF3	3
I0006	I/O ERROR ON FF3/P2	DATA PATH INH	I3	I/O ERROR	FF3	3
I0007	I/O ERROR ON FF4/P1	DATA PATH INH	I4	I/O ERROR	FF4	3
I0008	I/O ERROR ON FF4/P2	DATA PATH INH	I4	I/O ERROR	FF4	3
I0009	I/O ERROR ON FA1/P1	DATA PATH INH	I5	I/O ERROR	FA1	3
I0010	I/O ERROR ON FA2/P1	DATA PATH INH	I6	I/O ERROR	FA2	3
I0011	I/O ERROR ON FA3/P1	DATA PATH INH	I7	I/O ERROR	FA3	3
I0012	I/O ERROR ON FA4/P1	DATA PATH INH	I8	I/O ERROR	FA4	3
I0013	I/O ERROR ON PF1/P1	DATA PATH INH	I9	I/O ERROR	PF1	3
I0014	I/O ERROR ON PF1/P2	DATA PATH INH	I9	I/O ERROR	PF1	3
I0015	I/O ERROR ON PF2/P1	DATA PATH INH	I10	I/O ERROR	PF2	3
I0016	I/O ERROR ON PF2/P2	DATA PATH INH	I10	I/O ERROR	PF2	3
I0017	I/O ERROR ON LF1/P1	SWITCH BUSES	I11	I/O ERROR	LF1	3

MSG #	ERROR CONDITION	SOFTWARE RESPONSE	FMPT	TEXT MID	MAJOR	MIN	CLASS
I0018	I/O ERROR ON LF1/P2	SWITCH BUSES	I11		I/O ERROR	LF1	3
I0019	I/O ERROR ON LA1/P1	SWITCH BUSES	I12		I/O ERROR	LA1	3
I0020	I/O ERROR ON LA1/P2	SWITCH BUSES	I12		I/O ERROR	LA1	3
I0021	I/O ERROR ON DEU1	NO DATA PATH INH	I13		I/O ERROR	DEU1	3
I0022	I/O ERROR ON DEU2	NO DATA PATH INH	I14		I/O ERROR	DEU2	3
I0023	I/O ERROR ON DEU3	NO DATA PATH INH	I15		I/O ERROR	DEU3	3
I0024	I/O ERROR ON PCMMU	NO DATA PATH INH	I16		I/O ERROR	PCM	3
OS014	GPC1 ICC ERROR	FORCE FAIL TO SYNC	I17		I/O ERROR	ICC	3
OS015	GPC2 ICC ERROR	FORCE FAIL TO SYNC	I17		I/O ERROR	ICC	3
OS016	GPC3 ICC ERROR	FORCE FAIL TO SYNC	I17		I/O ERROR	ICC	3
OS017	GPC4 ICC ERROR	FORCE FAIL TO SYNC	I17		I/O ERROR	ICC	3
OS038	QUEUE OVERFLOW (IBM DESIGN)	REQ IGNORED	G6				
OS039	CYCLIC PROCESS WRAPAROUND (IBM DESIGN)	CYCLE SKIPPED	G7				

MSG #	ERROR CONDITION	SOFTWARE RESPONSE	FMPT	IDT	TEXT MAJOR	TEXT MIN	CLASS
OS040	CUMULATIVE ERROR INTERRUPTS (IBM DESIGN)	VARYING RESPONSE ON EACH INTERRUPT	G4		GPC	BITE	3
OS041	BCE ELEMENT BYPASS	BYPASS CHAIN ELE	E1		BCE ELE BYPASS	1	3
OS042	BCE ELEMENT BYPASS	BYPASS CHAIN ELE	E2		BCE ELE BYPASS	2	3
OS043	BCE ELEMENT BYPASS	BYPASS CHAIN ELE	E3		BCE ELE BYPASS	3	3
OS044	BCE ELEMENT BYPASS	BYPASS CHAIN ELE	E4		BCE ELE BYPASS	4	3
OS045	BCE ELEMENT BYPASS	BYPASS CHAIN ELE	E5		BCE ELE BYPASS		3

ERROR			SYSTEM DEFAULT ACTION	ERROR RESPONSE
Group	Code	ERROR	(Application Errors)	For FCOS ERROR
02	0D	Attempt to start cycle of cyclic process still in previous cycle.	Skip Cycle(ignore timer interrupt) and continue.	N/A
03	00	Illegal Operation Code	FORCE CLOSE process and continue	Dispatch application process
	01	Priviliged Instruction	FORCE CLOSE process and continue	N/A
	03	CPU Address Specification	FORCE CLOSE process	Dispatch application
	04	Fixed-Point Overflow	See action for 0309-030C	Ignore and continue
	05	Significance	See action for 0309-030C	Ignore and continue
	07	CPU Protection Violation	FORCE CLOSE process	Dispatch application
	09	Exponent Underflow(Floating Point)	Bump count of program interrupts for this process	Ignore and continue
	0A	Overflow(convert)	and	Ignore and continue
	0B	Exponent Overflow (Floating point)	Ignore if count max allowed	Ignore and continue
			or	
	0C	Divide (Floating-Point)	FORCE CLOSE process if count = max allowed	Ignore and continue
	14	Instruction Monitor	FORCE CLOSE process and continue	Dispatch application process
06	User	User Errors	Ignore and continue	N/A

NOTE: All errors are logged.

APPENDIX E

DATA DESCRIPTION TABLE

The Data Descriptors Appendix provides detailed information about each parameter. Parameters in the Data Descriptors can be cross referenced to parameters in the modular Data Tables via a parameter identification tag (ID). The definition of each field in the table follows.

1. ID - A parameter ID has been assigned to each data item in the format

   ZXXX [. XX]

   where:

   Z - Single character denoting the function of the parameter, chosen from one of the following:

   - A - Moding of Control Segments (See Vol II, P2)
   - D - Display Formats (See Vol II, P3)
   - B - Mass Memory (See Vol II, P2)
   - T - Annunciation (See Vol II, P2)
   - V - LDB (See Vol II, P2)
   - F - Downlist (See Vol II, P2)
   - W - Read/Write of Main Memory (See Vol II, P3)
   - H - Time Management (See Vol II, P3)
   - I - FCOS/CI Common Compool (See Vol II, P3)
   - J - Common (Miscellaneous) (See Vol II, P2)
   - L - User Interface Local Variable (See Vol II, P2)
   - X - System Control Local Variable (See Vol II, P3)
   - Y - FCOS I/O Management (See Vol II, P1)
   - @ - FCOS Process Management (See Vol II, P1)
   - 0 - FCOS Configuration Management (See Vol II, P1)
   - # - FCOS COMPOOL (See Vol II, P1)
   - Q - FCOS Control Blocks (See Vol II, P1)
   - S - FCOS SVC Parameter List (See Vol II, P1)
   - & - Hardware Parameters (See Vol II, P1)

   XXX - A unique three digit number assigned to each major division. (A major division is a structure, table, array, or a single parameter not contained in one of the previously mentioned items.

   [.XX] - Second level qualifier for elements of arrays, parts of structures or tables or the bits in a flag or discrete word.

2. ITEM - A unique meaningful English_Equivalent_Name for the data item.

3. HAL/Assembler Name - The HAL/S or Assembler Name used in coding. The column is blank if no name is assigned as is the case with bits in a flag word. (User Interface and System Control column name is "HAL NAME" FCOS column is "ASSEMBLER NAME").

4. Description - A concise statement of the nature of the parameter and its purpose.

5. Source - A listing of the three digit ID's of all modules updating the parameter and/or one of the following special codes:

   ILD - The parameter is of the Initial Load (I-LOAD) type (see Level A CPDS, Table 6-4).

   MRW - The parameter is available to memory read write (see Level A CPDS, Table 6-4).

6. Destination - A list of the three digit ID's of all modules referencing the parameter and/or one of the following special codes:

   DL - The parameter is available for downlink.
   CRT - The parameter is available for display.

7. Attributes - The attributes of the parameter are as follows:

   a. Data Type - The HAL/S declaration type or assembler language type

ST(n)	-	Structure with n copies
A(m,n),k	-	Array of m,n dimensions of HAL parameter type k
BS(n)	-	Bit string of length n
BT	-	Bit
C(n)	-	Character string of length n
SC	-	Scalar/Assembler Language Floating Point
I	-	Integer/Assembler Language Fixed Point Halfword
BO	-	Boolean
M(n,m)	-	Matrix of n,m dimensions
V(n)	-	Vector of n elements
E	-	Event
Y	-	Halfword Address Constant
AC	-	Fullword Address Constant
Z	-	Fullword Indirect Address Constant
X	-	Hexadecimal
F(n)	-	Fullword Fixed Point n Copies

NOTE: Double length of a parameter type is denoted by prefixing a 'D' to the above characters.

b. Initial Value - The initialization value if applicable;

INIT(value)

c. Units of Measure - The units of measure of the parameter if applicable:

UM(unit)

d. Other - Any of the attributes such as REPLACE, LOCU(h), etc.

8. MML - The MML number associated with the parameter. The column is blank if none is defined.

In part 1 (FCOS) bits in a word will be numbered starting with 0. In parts 2 and 3 (UI and SC) bits in a word will be numbered starting with 1.

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
&001	PREFERRED_STORAGE_AREA	TFPSA	THE PSA IS THE ZONE OF MAIN MEMORY RESERVED FOR HARDWARD/SOFTWARE INTERFACE AND COMMUNICATION (RESIDES IN ECMPSA)			ST	
&001.01	RESERVED_AREA_1	TPSARES1	RESERVED AREA			A(4),I,INIT(0)	
&001.05	POWER_ON_PSW	TPSAPWR	POWER ON PSW	300		Y,INIT(0) X, INIT('0800000F0000')	
&001.06	RESERVED_AREA_2	TPSARS2	RESERVED AREA			A(4),I,INIT(0)	
&001.10	RESERVED_AREA_3	TPSARS3	RESERVED AREA			A(4),I,INIT(0)	
&001.14	POWER_OFF_PSW	TPSAPWRF	POWER OFF PSW	300		Y,INIT(0) X, INIT('0800000F0000')	
&001.15	SYSTEM_RESET_PSW	TPSASRP	SYSTEM RESET PSW	300	142 190	Y,INIT(ADDR(ECMINSSL)) X, INIT('0800000F0000')	
&001.16	RESERVED_AREA_4	TPSARS4	RESERVED AREA			A(2),I,INIT(0)	
&001.17	CVT_ADDRESS	TPSACVTA	ADDRESS OF THE CVT			Y,INIT(ADDR(ECMCVT))	
&001.18	UNUSED_AREA	TPSASPR1	UNUSED			I,INIT(0)	
&001.19	RESERVED_AREA_5	TPSARS5	RESERVED AREA			A(4),I,INIT(0)	
&001.23	RESERVED_AREA_6	TPSARS6	RESERVED AREA			A(4),I,INIT(0)	
&001.27	RESERVED_AREA_7	TPSARS7	RESERVED AREA			A(4),I,INIT(0)	
&001.31	RESERVED_AREA_8	TPSARS8	RESERVED AREA			A(4),I,INIT(0)	
&001.35	RESERVED_AREA_9	TPSARS9	RESERVED			A(4),I,INIT(0)	
&001.39	RESERVED_AREA_10	TPSARS10	RESERVED AREA			A(4),I,INIT(0)	
&001.43	RESERVED_AREA_11	TPSARS11	RESERVED AREA			A(4),I,INIT(0)	
&001.47	RESERVED_AREA_12	TPSARS12	RESERVED AREA			A(4),I,INIT(0)	
&001.51	RESERVED_AREA_13	TPSARS13	RESERVED AREA			A(4),I,INIT(0)	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
&001.55	MACHINE_CHECK_OLD_PSW	TPSAMCOP	MACHINE CHECK OLD PSW			A(2),F,INIT(0)	
&001.56	MACHINE_CHECK_NEW_PSW	TPSAMCNP	MACHINE CHECK NEW PSW	300		Y,INIT(0) X, INIT('0800000A0000')	
&001.57	PROGRAM_INTERRUPT_OLD_PSW	TPSAPIOP	PROGRAM CHECK OLD PSW		180	A(2),F,INIT(0)	
&001.58	PROGRAM_INTERRUPT_NEW_PSW	TPSAPINP	PROGRAM CHECK NEW PSW	300		Y,INIT (ADDR (EPMIHPGM)) X, INIT('0800000F0000')	
&001.59	RESERVED_AREA_14	TPSARES3	RESERVED AREA			A(4),I,INIT(0)	
&001.63	RESERVED_AREA_15	TPSARES4	RESEVED AREA			A(4),I,INIT(0)	
&001.67	SVC_OLD_PSW	TPSASOP	SUPERVISOR CALL OLD PSW		100 101 104 105 108 109 170 171 172 200 201 320	A(2),F,INIT(0)	
&001.68	SVC_NEW_PSW	TPSASNP	SUPERVISOR CALL NEW PSW	300	101 140	Y,INIT(ADDR (EPMSVC) X, INIT('0800200F0000')	
&001.69	PROGRAM_COUNTER_1_OLD_PSW	TPSAC1OP	PROGRAM COUNTER 1 OLD PSW		141	A(2),F,INIT(0)	
&001.70	PROGRAM_COUNTER_1_NEW_PSW	TPSAC1NP	PROGRAM COUNTER 1 NEW PSW	300		Y,INIT (ADDR (EPMIHPC1)) X, INIT('0800200F0000')	
&001.71	PROGRAM_COUNTER_2_OLD_PSW	TPSAC2OP	PROGRAM COUNTER 2 OLD PSW		142	A(2),F,INIT(0)	
&001.72	PROGRAM_COUNTER_2_NEW_PSW	TPSAC2NP	PROGRAM COUNTER 2 NEW PSW	300		Y,INIT (ADDR (EPMIHPC2)) X, INIT('0800200F0000')	

PREFERRED STORAGE AREA

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
&001.73	INSTRUCTION_MONITOR_OLD_PSW	TPSAIMOP	INSTRUCTION MONITOR OLD PSW		181	A(2),F,INIT(0)	
&001.74	INSTRUCTION_MONITOR_NEW_PSW	TPSAIMNP	INSTRUCTION MONITOR NEW PSW	300		Y,INIT (ADDP (EPM(HIM)) X, INIT('0800000F0000')	
&001.75	EXTERNAL_INTERRUPT_0_OLD_PSW	TPSAE0P	EXTERNAL INTERRUPT 0 OLD PSW		240	A(2),F,INIT(0)	
&001.76	EXTERNAL_INTERRUPT_0_NEW_PSW	TPSAE0NP	EXTERNAL INTERRUPT 0 NEW PSW	300		Y,INIT (ADDP (EIOERPLA)) X, INIT('0800200F0000')	
&001.77	EXTERNAL_INTERRUPT_1_OLD_PSW	TPSAE1OP	EXTERNAL INTERRUPT 1 OLD PSW		242	A(2),F,INIT (0)	
&001.78	EXTERNAL_INTERRUPT_1_NEW_PSW	TPSAE1NP	EXTERNAL INTERRUPT 1 NEW PSW	300		Y,INIT (ADDP (EIOERPLB)) X, INIT('0800200F0000')	
&001.79	EXTERNAL_INTERRUPT_2_OLD_PSW	TPSAE2OP	EXTERNAL INTERRUPT 2 OLD PSW	300	220 244	A(2),F,INIT (0)	
&001.80	EXTERNAL_INTERRUPT_2_NEW_PSW	TPSAE2NP	EXTERNAL INTERRUPT 2 NEW PSW	300 305		Y,INIT (ADDP (ECMSSLEX)) X,INIT('0800200C0000')	
&001.81	EXTERNAL_INTERRUPT_3_OLD_PSW	TPSAE3OP	EXTERNAL INTERRUPT 3 OLD PSW			A(2),F, INIT(0)	
&001.83	SPECIAL_INTERRUPT_OLD_PSW	TPSASIOP	SPECIAL INTERRUPT OLD PSW			A(2),F,INIT (0)	
&001.84	SPECIAL_INTERRUPT_NEW_PSW	TPSASINP	SPECIAL INTERRUPT NEW PSW	300		Y,INIT(0) X, INIT('0800200F0000')	
&001.85	RESERVED_AREA_16	TPSARES5	RESERVED AREA			A(16),I, INIT(0)	
&001.86	PC1_SOFTWARE_PORTION	TPSAPC1	SOFTWARE PORTION OF PROGRAM COUNTER 1	141 142 300	142	X,INIT (FFFF)	
&001.87	PC2_SOFTWARE_PORTION	TPSAPC2	SOFTWARE PORTION OF PROGRAM COUNTER 2	300	142	X,INIT (FFFF)	
&001.88	FCOS_WORK_AREA	TPSAWORK	FCOS WORK AREA	103		A(14),I,	

PREFERRED STORAGE AREA

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
				182 185 300 305		INIT(0)	
£001.89	GPR_SET_1_PUT_AWAY	TPSAPAR1	GENERAL PURPOSE REGISTER SET 1 PUT AWAY AREA			A(16),I,INIT(0)	
£001.90	GPR_SET_2_PUT_AWAY	TPSAPAR2	GENERAL PURPOSE REGISTER SET 2 PUT AWAY AREA			A(16),I,INIT (0)	
£001.91	FLOATING_POINT_ REGISTER_PUT_AWAY	TPSAPAFP	FLOATING POINT REGISTER PUT AWAY			A(16),I INIT (0)	
£001.92	MICROCODE_PUT_AWAY	TPSAPAMR	MICROCODE PUT AWAY AREA			A(16)I,INIT(0)	
£001.93	RESERVED_PUT_AWAY	TPSAPARS	RESERVED PUT AWAY AREA			I,INIT (0)	
£001.94	PC1_PUT_AWAY	TPSAPAC1	PROGRAM COUNTER 1 PUT AWAY AREA			I,INIT (0)	
£001.95	PC2_PUT_AWAY	TPSAPAC2	PROGRAM COUNTER 2 PUT AWAY AREA			I,INIT (0)	
£001.96	OP_REGISTER_PUT_ AWAY	TPSAPAUR	OP REGISTER PUT AWAY AREA			I,INIT (0)	
£001.97			RESERVED			A(60),I, INIT(0)	

PREFERRED STORAGE AREA

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
&002	PROGRAM_STATUS_WORD		THE PSW CONTAINS THE BASIC INFORMATION REQUIRED FOR PROPER PROGRAM EXECUTION. THE FOLLOWING DESCRIBES THE VARIOUS FIELDS OF THE PSW WHICH IS APPLICABLE TO EVERY PSW.				
&002.1	PSW_INSTRUCTION_ADDRESS		HALFWORD ADDRESS OF THE NEXT INSTRUCTION TO BE EXECUTED. ACTUAL ADDRESS IS DETERMINED BY APPENDING PSW_BRANCH_SECTOR REGISTER TO BITS 1-15 OF THIS FIELD	183 320		Y	
&002.2	PSW_PROGRAM_MASK		PROGRAM MASK FIELD WHICH CONTAINS CPU STATUS AND INDICATOR BITS DESCRIBED IN THE NEXT FEW FIELDS.	192		BS(8)	
&002.3	PSW_CONDITION_CODE		PSW_PROGRAM_MASK BITS 0 AND 1 CONTAIN THE CURRENT CONDITION CODE RESULTING FROM CERTAIN ARITHMETIC LOGICAL OR I/O INSTRUCTIONS.			BS(2)	
&002.4	PSW_CARRY_INDICATOR		PSW_PROGRAM_MASK BIT 2: 1=CARRY OUT OF HIGH ORDER BIT POSITION OF REGISTER HAS OCCURRED 0= CARRY OUT HAS NOT OCCURED.			BT	
&002.5	PSW_OVERFLOW_INDICATOR		PSW_PROGRAM_MASK BIT 3: 1= OVERFLOW HAS OCCURED 0= OVERFLOW HAS NOT OCCURED			BT	
&002.6	PSW_OVERFLOW_MASK		PSW_PROGRAM_MASK BIT 4: 1= FIXED POINT ARITHMETIC OVERFLOW INTERRUPTS ARE ENABLED. 0= OVERFLOW INTERRUPTS ARE DISABLED.	192		BT	
&002.7	PSW_RESERVED_FIELD_1		PSW_PROGRAM_MASK BIT 5 IS RESERVED (ALWAYS=0)			BT	
&002.8	PSW_EXPONENT_UNDERFLOW_MASK		PSW_PROGRAM_MASK BIT 6: 1= FLOATING POINT EXPONENT UNDERFLOW INTERRUPTS ARE ENABLED. 0=FLOATING POINT EXPONENT UNDERFLOW INTERRUPTS ARE DISABLED			BT	
&002.9	PSW_SIGNIFICANCE_MASK		PSW_PROGRAM_MASK BIT 7: 1= SIGNIFICANCE INTERRUPTS ARE ENABLED.			BT	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	OST	ATTRIBUTES	MML
			0= SIGNIFICANCE INTERRUPTS ARE DISABLED.				
&002.10	PSW_BRANCH_SECTOR_REGISTER		REPLACE THE HIGH ORDER BIT OF A BRANCH ADDRESS WHEN THAT HIGH ORDER BIT IS A 1		193	BS(4)	
&002.11	PSW_DATA_SECTOR_REGISTER		REPLACES THE HIGH ORDER BIT OF A DATA ADDRESS WHEN THAT HIGH ORDER BIT IS A 1			BS(4)	
&002.12	PSW_SYSTEM_MASK		INTERRUPT MASK BITS WHICH INDICATE IF A TYPE OF INTERRUPT WILL BE ENABLED OR DISABLED		320	BS(8)	
&002.13	PSW_PC1_MASK		PSW_SYSTEM_MASK BIT 0: 1= PROGRAM COUNTER 1 INTERRUPTS ENABLED. 0= PROGRAM COUNTER 1 INTERRUPTS DISABLED			BT	
&002.14	PSW_PC2_MASK		PSW_SYSTEM_MASK BIT 1: 1= PROGRAM COUNTER 2 INTERRUPTS ENABLED 0= PROGRAM COUNTER 2 INTERRUPTS DISABLED			BT	
&002.15	PSW_INSTRUCTION_MONITOR_MASK		PSW_SYSTEM_MASK BIT 2: 1= INSTRUCTION MONITOR INTERRUPTS ENABLED. 0= INSTRUCTION MONITOR INTERRUPTS DISABLED			BT	
&002.16	PSW_EXTERNAL_0_MASK		PSW_SYSTEM_MASK BIT 3: 1= EXTERNAL 0 INTERRUPTS ENABLED 0= EXTERNAL 0 INTERRUPTS DISABLED			BT	
&002.17	PSW_EXTERNAL_1_MASK		PSW_SYSTEM_MASK BIT 4: 1= EXTERNAL 1 INTERRUPTS ENABLED. 0= EXTERNAL 1 INTERRUPTS DISABLED.			BT	
&002.18	PSW_EXTERNAL_2_MASK		PSW_SYSTEM_MASK BIT 5: 1=EXTERNAL 2 INTERRUPTS ENABLED. 0=EXTERNAL 2 INTERRUPTS DISABLED.	300 305		BT	
&002.19	PSW_EXTERNAL_3_MASK		PSW_SYSTEM_MASK BIT 6: 1=EXTERNAL 3 INTERRUPTS ENABLED 0=EXTERNAL 3 INTERRUPTS DISABLED			BT	
&002.20	PSW_EXTERNAL_4_MASK		PSW_SYSTEM_MASK BIT 7: 1= EXTERNAL 4 INTERRUPTS ENABLED 0= EXTERNAL 4 INTERRUPTS DISABLED			BT	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
			RESERVED. ALWAYS= 0			BS (4)	
&002.21	PSW_RESERVED_FIELD_2						
&002.22	PSW_REGISTER_SET		GENERAL PURPOSE REGISTER SET SELECTION: 0= SET 1 (USED BY APPLICATION) 1= SET 2 (USED BY FCOS)	182	320	BT	
&002.23	PSW_MACHINE_CHECK_MASK		MACHINE CHECK INTERRUPT MASK: 0= MACHINE CHECKS DISABLED. 1= MACHINE CHECKS ENABLED.		320	BT	
&002.24	PSW_WAIT_STATE_INDICATOR		WAIT STATE FIELD: 0=NOT WAIT STATE 1=WAIT STATE	300	320	BT	
&002.25	PSW_RUN_STATE_CONTROL		RUN STATE CONTROL BIT: 0= SUPERVISOR STATE (FCOS) 1= PROBLEM STATE		180 181 320	BT	
&002.26	PSW_INTERRUPT_CODE		INTERRUPT CODE FOR PROGRAM OR MACHINE CHECKS OR SVC PARAMETER LIST ADDRESS FOR SVC INTERRUPT.		180 242 320	I OR V	

PROGRAM STATUS WORD(PSW)

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
&003	PC1_HARDWARE_PORTION		HARDWARE PORTION OF PROGRAM COUNTS DOWN BY ONE EVERY MICROSECOND	300		H	
&004	PC2_HARDWARE_PORTION		HARDWARE PORTION OF PROGRAM COUNTS DOWN 1 EVERY MICROSECOND.	300		H	
&005	DISCRETE_INPUT_REGISTER_A			300 340	300 340	BS(32)	
&005.01	DISCRETE_BIT_HALT		DISCRETE_INPUT_REGISTER_A BIT 0: 1= HALT COMMAND MODE IS ON. IOP AND CPU EXECUTION IS INHIBITED. 0= HALT COMMAND MODE IS OFF.			BT	
&005.02	DISCRETE_BIT_STANDBY		DISCRETE_INPUT_REGISTER_A BIT 1: 1= STANDBY MODE COMMAND IS ON. 0= STANDBY MODE COMMAND IS OFF.			BT	
&005.03	DISCRETE_BIT_RUN_MODE		DISCRETE_INPUT_REGISTER_A_BIT 2: 1= RUN MODE IS ON. 0= RUN MODE IS OFF BT				
&005.04	DISCRETE_BIT_IPL_ACTIVATE		DISCRETE_INPUT_REGISTER_A BIT 3: 1= GPC IPL ACTIVATE COMMAND IS ON. 0= SPC IPL ACTIVATE COMMAND IS OFF.			BT	
&005.05	DISCRETE_BIT_MM1_SELECT		DISCRETE_INPUT_REGISTER_A BIT 4: 1= SELECT MM1 TO PERFORM GPC IPL. 0= DO NOT SELECT MM1.			BT	
&005.06	DISCRETE_BIT_MM2_SELECT		DISCRETE_INPUT_REGISTER_A BIT 5: 1= SELECT MM2 TO PERFORM GPC IPL. 0= DO NOT SELECT MM2			BT	
&005.07	DISCRETE_BIT_MM1_READY		DISCRETE_INPUT_REGISTER_A BIT 6: 1= MM1 IS IN STANDBY AND CAPABLE OF ACCEPTING COMMANDS 0= MM2 IS PERFORMING AN OPERATION, IS FAILED OR POWER IS OFF.	300		BT	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
&005.08	DISCRETE_BIT_MM2_READY		DISCRETE_INPUT_REGISTER_A_BIT 7: 1= MM2 IS IN STANDBY AND IS CAPABLE OF ACCEPTING COMMANDS. 0= MM2 IS PERFORMING AN OPERATION, IS FAILED OR POWER IS OFF.		300	BT	
&005.09	DISCRETE_BITS_SPARE_FIELD_A1		DISCRETE_INPUT_REGISTER_A_BITS 8-10.			BS(3)	
&005.10	DISCRETE_BIT_BFCS_ENGAGE		DISCRETE_INPUT_REGISTER_A_BIT 11: 1= BACKUP FLIGHT CONTROL SYSTEM (BFCS) IS ENGAGED. 0= BFCS IS NOT ENGAGED.			BT	
&005.11	DISCRETE_BIT_IOP_TERMINATE_A		DISCRETE_INPUT_REGISTER_A_BIT 12: 1= THE COMPUTER OUTPUT SWITCH IS IN THE TERMINATE POSITION. (THE IOP IS TO TERMINATE I/O OVER DATA BUSES FC1-FC4). 0= THE COMPUTER OUTPUT SWITCH IS IN THE NORMAL POSITION. (I/O MAY CONTINUE.)			BT	
&005.12	DISCRETE_BIT_IOP_TERMINATE_B		DISCRETE_INPUT_REGISTER_A_BIT 13: 1= THE COMPUTER OUTPUT SWITCH IS IN THE TERMINATE POSITION OR THE BFCS ENGAGE SWITCH IS ON. (THE IOP IS TO TERMINATE I/O OVER DATA BUSES FC5-FC8, PL1-PL2, L81-L82) 0= THE COMPUTER OUTPUT SWITCH IS IN THE NORMAL POSITION AND THE BFCS ENGAGE SWITCH IS OFF. (I/O MAY CONTINUE).			BT	
&005.13	DISCRETE_BITS_SPARE_FIELD_A2		DISCRETE_INPUT_REGISTER_A BITS 14-19			BS(6)	
&005.14	DISCRETE_BIT_GPC_N+1_SYNC_CODE_1		DISCRETE_INPUT_REGISTER_A_BIT 20		363 364 368	BT	
&005.15	DISCRETE_BIT_GPC_N+2_SYNC_CODE_1		DISCRETE_INPUT_REGISTER_A_BIT 21		363 364 368	BT	
&005.16	DISCRETE_BIT_GPC_N+3_SYNC_CODE_1		DISCRETE_INPUT_REGISTER_A_BIT 22		363 364 368	BT	
&005.17	DISCRETE_BIT_GPC_		DISCRETE_INPUT_REGISTER_A_BIT 23		363	BT	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
	N+4_SYNC_CODE_1				364 368		
&005.18	DISCRETE_BIT_GPC_ N+1_SYNC_CODE_2		DISCRETE INPUT REGISTER_A_BIT 24		363 364 368	BT	
&005.19	DISCRETE_BIT_GPC_ N+2_SYNC_CODE_2		DISCRETE INPUT REGISTER A_BIT 25		363 364 368	BT	
&005.20	DISCRETE_BIT_GPC_ N+3_SYNC_CODE_2		DISCRETE_INPUT_REGISTER_A_BIT 26		363 364 368	BT	
&005.21	DISCRETE_BIT_GPC_ N+4_SYNC_CODE_2		DISCRETE_INPUT_REGISTER_A_BIT 27		363 364 368	BT	
&005.22	DISCRETE_BIT_GPC_ N+1_SYNC_CODE_3		DISCRETE_INPUT_REGISTER_A_BIT 28		363 364	BT	
&005.23	DISCRETE_BIT_GPC_ N+2_SYNC_CODE_3		DISCRETE_INPUT_REGISTER_A_BIT 29		363 364	BT	
&005.24	DISCRETE_BIT_GPC_ N+3_SYNC_CODE_3		DISCRETE_INPUT REGISTER_A_BIT 30		363 364	BT	
&005.25	DISCRETE_BIT_GPC_ N+4_SYNC_CODE_3		DISCRETE_INPUT_REGISTER_A_BIT 31		363 364	BT	
&006	IOP_STATUS_ REGISTERS		ICP PROCESSING STATE IS MAINTAINED IN THE IOP_STATUS_REGISTERS				
&006.1	HALT REGISTER		A 25 BIT REGISTER WHICH PROVIDES INDIVIDUAL CONTROL OF THE PROCESSORS TO STOP PROGRAM EXECUTION POSITION 0= MSC. POSITIONS 1 THROUGH 24= BCE'S 1 THROUGH 24.			BS(25)	
&006.2	PROGRAM_EXCEPTION_ REGISTER		A 25 BIT REGISTER WHICH INDICATES THAT A PROCESSOR HAS ENCOUNTERED SOME PROBLEM IN THE EXECUTION OF A PROGRAM POSITION 0= MSC. POSITION 1 THROUGH 24= BCE'S 1 THROUGH 24.		251	BS(25)	
&006.3	BUSY_WAIT_REGISTER		A 25 BIT REGISTER WHICH INDICATES WHETHER OR NOT A PROCESSOR IS EXECUTING A PROGRAM	250 251 252	251 280 281	BS(25)	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
			POSITION 0= MSC. POSITIONS 1 THROUGH 24= BCE'S 1 THROUGH 24.	280			
E006.4	INDICATOR_REGISTER		A 24 BIT REGISTER WHICH IS USED FOR COMMUNICATION BETWEEN PROCESSORS. THERE IS ONE BIT ASSIGNED TO EACH BCE. BS(24)				
E006.05	MIA_TRANSMIT_ENABLE_REGISTER		A 24 BIT REGISTER WHICH DESIGNATES WHETHER OR NOT A MIA TRASMITTER IS ENABLED. 0= DISABLED 1= ENABLED			BS(24)	
E006.06	MIA_RECEIVE_ENABLE_REGISTER		A 24 BIT REGISTER WHICH DESIGNATES WHETHER OR NOT A MIA RECEIVER IS ENABLED. 0= DISABLED 1= ENABLED			BS(24)	
E006.07	RM_STATUS_REGISTER		A 32 BIT REGISTER WHICH CONTAINS THE STATUS OF VARIOUS FUNCTIONS ASSOCIATED WITH THE RM HARDWARE LOGIC.		340	BS(32)	
E006.10	INTERRUPT_REGISTER_A		A 32 BIT REGISTER ASSOCIATED WITH ONE OF THE INTERRUPT LINES FROM THE IOP TO THE GPC. IT IS USED TO DENOTE CERTAIN IOP ERRORS. UNDEFINED BITS ARE NOT USED.		240	BS(32)	
E006.11	GO/NO-GO-TIMER_INTERRUPT		BIT 0-GO/NO-GO TIMEOUT INDICATOR 0= NO INTERRUPT 1= INTERRUPT		240	BT	
E006.12	IOP_FAIL_LATCH_INTERRUPT		BIT 1- COMPUTER FAILURE 0= NO INTERRUPT 1= INTERRUPT		240	BT	
E006.13	C/M_IDLE_INTERRUPT		BIT 2- IOP CONTROL/MONITOR IDLE MODE INDICATOR 0= NO INTERRUPT 1= INTERRUPT		240	BT	
E006.14	ROS_PARITY_ERROR_INTERRUPT		BIT 3- IOP ROS PARITY ERROR INDICATOR 0= NO INTERRUPT 1= INTERRUPT		240	BT	
E006.15	IOP_FAULT_INTERRUPT		BIT 4- IOP OSCILLATOR FAILURE INDICATOR		240	BT	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	OST	ATTRIBUTES	MML
			0= NO INTERRUPT 1= INTERRUPT				
&006.20	INTERRUPT_REGISTER_B		A 32 BIT REGISTER ASSOCIATED WITH ONE OF THE INTERRUPT LINES FROM THE CPU. IT IS USED TO DESIGNATE PARITY AND DMA ERRORS. UNDEFINED BITS ARE NOT USED.		242	BS(32)	
&006.21	PCI/PCO_PARITY_ERROR_INTERRUPT		BIT 0- PCI/PCO PARITY ERROR INDICATOR 0= NO INTERRUPT 1= INTERRUPT			BT	
&006.22	DMA_INSTRUCTION_PARITY_ERROR_INTERRUPT		BIT 1- DMA INSTRUCTION PARITY ERROR INDICATOR 0= NO INTERRUPT 1= INTERRUPT			BT	
&006.23	DMA_PARITY_ERROR_INTERRUPT		BIT 2- DMA PARITY ERROR INDICATOR 0= NO INTERRUPT 1= INTERRUPT			BT	
&006.24	INTERRUPT_REGISTER_B_SPARE		BIT 3- NOT USED			BT	
&006.25	DMA_QUEUE_OVERFLOW_INTERRUPT		BIT 4- DMA QUEUE OVERFLOW INDICATOR 0= NO INTERRUPT 1= INTERRUPT			BT	
&006.26	DMA_TIME_OUT INTERRUPT		BIT 5- DMA TIMEOUT INDICATOR 0= NO INTERRUPT 1= INTERRUPT			BT	
&006.30	INTERRUPT_REGISTER_C		A 32 BIT REGISTER ASSOCIATED WITH ONE OF THE INTERRUPT LINES FROM THE TOP TO THE CPU. THIS REGISTER CONTAINS THE PROGRAMMABLE INTERRUPTS GENERATED BY THE MSC PROCESSOR.		251	BS(32)	
&007	MSC_LOCAL_STORE					ST	
&007.1	MSC_BANK_A_UNWED		MSC_LOCAL_STORE BANK A LOCATIONS 0-1				
&007.02	MSC_PROGRAM_		MSC_LOCAL_STORE BANK A LOCATION 2		210	BS(18)	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
	COUNTER		ADDRESS OF INSTRUCTION BEING EXECUTED.	300			
&007.03	MSC_INDEX_REGISTER		MSC_LOCAL_STORE BANK A LOCATION 3 USED FOR STORAGE OR GENERATING A MEMORY ADDRESS.		301		
&007.04	MSC_BANK_B_UNUSED		MSC_LOCAL STORE BANK B LOCATIONS 0-1				
&007.05	MSC_INSTRUCTION_HIGH		MSC_LOCAL_STORE BANK B LOCATION 2 UPPER 16 BITS OF PRESENT MSC INSTRUCTION.			I	
&007.06	MSC_ACCUMULATOR_HIGH		MSC_LOCAL_STORE BANK B LOCATION 3 BITS 0-15 OF MSC ACCUMULATOR.	301	301	I	
&007.07	MSC_BANK_C_UNUSUAL 1		MSC_LOCAL_STORE BANK C LOCATIONS 0-1				
&007.08	MSC_INSTRUCTION_LOW		MSC_LOCAL_STORE BANK C LOCATION 2 LOWER 16 BITS OF LAST INSTRUCTION WORD READ FROM MEMORY.	301	301	I	
&007.09	MSC_ACCUMULATOR_LOW		MSC_LOCAL_STORE BANK C LOCATION 3 BITS 16-31 OF MSC ACCUMULATOR	301	301	I	
&007.10	MSC_BANK_C_UNUSED_2		MSC_LOCAL STORE BANK C LOCATIONS 4-5			BS(18)	
&007.11	MSC_EXTERNAL_CALL_REGISTER		MSC_LOCAL_STORE BANK C LOCATION 6 SET BY THE CPU TO POINT TO A PROGRAM TO BE EXECUTED BY THE MSC				
&007.12	MSC_STATUS_REGISTER		MSC_LOCAL_STORE BANK C LOCATION 7 CONTAINS COPIES OF MSC BUSY/WAIT BIT AND MSC PROGRAM EXCEPTION BIT. ALSO INDICATES THE CAUSE OF THE EXCEPTION IF THE LATTER IS SET.			BS(19)	
&007.13	MSC_ACCUMULATOR_REGISTER		COMBINATION OF MSC_ACCUMULATOR -HIGH (&007.06) AND MSC_ACCUMULATOR -LOW (&007.09).	280	280 281	BS(32)	
&008	BCE_LOCAL_STORE					ST	
&008.01	BCE_BANK_A_UNUSED		BCE_LOCAL_STORE BANK A LOCATIONS 0-1				

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
&008.02	BCE_PROGRAM_ COUNTER		BCE_LOCAL_STORE BANK A LOCATION 2 CONTAINS THE MAIN MEMORY ADDRESS OF THE BCE INSTRUCTION BEING EXECUTED.	300		BS(18)	
&008.03	BCE_IDENTIFY_ REGISTER		BCE_LOCAL_STORE BANK A LOCATION 3 CONTAINS A VALUE THAT IS TWICE THE BCE NUMBER.			BS(6)	
&008.04	BCE_BANK_B_UNUSED		BCE_LOCAL_STORE BANK B LOCATIONS 0-1	264 267	262		
&008.05	BCE_INSTRUCTION_ REGISTER_HIGH		BCE_LOCAL_STORE BANK B LOCATION 2 CONTAINS THE HIGH HALF OF THE CURRENT INSTRUCTION				
&008.06	BCE_MAX_TIME_OUT_ REGISTER		BCE_LOCAL_STORE BANK B LOCATION 3 INDICATES THE TIME A BCE SHOULD WAIT FOR A SUBSYSTEM RESPONSE.	300	302	BS(18)	
&008.07	BCE_BANK_C_UNUSED_ 1		BCE_LOCAL_STORE BANK C LOCATIONS 0-1				
&008.08	BCE_INSTRUCTION_ REGISTER_LOW		BCE_LOCAL_STORE BANK C LOCATION 2 CONTAINS LOW HALF OF CURRENT INSTRUCTION.				
&008.09	BCE_BASE_REGISTER		BCE_LOCAL_STORE BANK C LOCATION 3 CONTAINS THE MAIN MEMORY ADDRESS OF AN I/O BUFFER.	261 263 264 265 267 282 302	260 261 262 263 266 282	BS(18)	
&008.10	BCE_BANK_C_UNUSED_ 2		BCE_LOCAL_STORE BANK C LOCATION 4				
&008.11	BCE_INTERFACE_UNIT_ ADDRESS_REGISTER		BCE_LOCAL_STORE BANK C LOCATION 5 CONTAINS THE SUBSYSTEM ADDRESS OF THE SUBSYSTEM PRESENTLY IN COMMUNICATION WITH THIS BCE.			BS(5)	
&008.12	BCE_STATUS_HIGH		BCE_LOCAL_STORE BANK C LOCATION 6 HIGH HALF OF BCE STATUS REGISTER.		282	BS(16)	
&008.13	BCE_STATUS_LOW		BCE_LOCAL_STORE BANK C LOCATION 7 LOW HALF OF BCE STATUS REGISTER		282	BS(32)	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
&008.14	BCE_STATUS_REGISTERS		BCE_LOCAL_STORE_BANK_C_LOCATIONS 6 AND 7 OF BCE STATUS REGISTERS		292	BS(32)	
&009	DISCRETE_INPUT_REGISTER_B					BS(32)	
&009.01	DISCRETE_BITS_GPC_ID		DISCRETE_INPUT_REGISTER_B_BITS 0-2 BIT 0 1 2 0 0 1 GPC 1 0 1 0 GPC 2 0 1 1 GPC 3 1 0 0 GPC 4 1 0 1 GPC 5		300	BS(3)	
&009.02	DISCRETE_BITS_SPARE_FIELD_BI		DISCRETE_INPUT REGISTER_B_BITS 3-7			BS(24)	
&010	DISCRETE_OUTPUT_REGISTER					BS(32)	
&010.01	GPC_IDENTIFICATION_SOURCE		DISCRETE_OUTPUT_REGISTER_BIT 0 1= VOLTAGE SOURCE FOR GPC IDENTIFICATION DISCRETE INPUTS DI0,DI1,AND DI2 IS APPLIED. 0= VOLTAGE SOURCE FOR GPC IDENTIFICATION IS NOT APPLIED.			BT	
&010.02	DISCRETE_OUTPUT_SPARE_BITS_FIELD_1		DISCRETE_OUTPUT_REGISTER_BITS 1-6			BS(6)	
&010.03	I/O_ACTIVE_INDICATOR		DISCRETE_OUTPUT_REGISTER_BIT 7 1= THE GPC IS READY TO PERFORM NORMAL DATA BUS TRANSMISSION AFTER COMPLETION OF GPC INITIALIZATION. 2= THE GPC IS POWERED OFF IS POWERED ON BUT IN AN IOP GENERAL RESET STATE, OR IS IN A GPC INITIALIZATION STATE PRIOR TO PERFORMING NORMAL DATA BUS TRANSMISSION.			BT	
&010.04	DISCRETE_OUTPUT_SPARE_BIT_FIELD_2		DISCRETE_OUTPUT_REGISTER BIT 8			BT	
&010.05	GPC_READY_INDICATOR		DISCRETE_OUTPUT_REGISTER BIT 9 1= GPC(1-4)IS CONDITIONAL FOR MCDS			BT	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
			KEYBOARD COMMUNICATION (STBY OP RUN MODE), OR GPC(5) IS CONDITIONED FOR RUN MODE BACKUP FUNCTIONS. 0= GPC(1-5)IS NOT CONDITIONED FOR FUNCTIONAL USE.				
&010.06	DISCRETE_OUTPUT_ SPARE BIT_FIELD_3		DISCRETE_OUTPUT_REGISTER_BIT 10			BT	
&010.07	GPC_IPL_INDICATOR		DISCRETE_OUTPUT_REGISTER BIT 11 1= GPC IS PERFORMING IPL. 0= GPC IS NOT PERFORMING IPL.			BT	
&010.08	MM1_RESET		DISCRETE_OUTPUT_REGISTER BIT 12 1= MM1 IS TO TERMINATE CURRENT OPERATIONS AND ENTER THE STANDBY STATE. THE CURRENT OPERATION (IF ANY) WILL CONTINUE UNTIL THE END OF THE CURRENT 512 WORD BLOCK. 0= MM1 IS TO CONTINUE CURRENT OPERATION.			BT	
&010.09	MM2_RESET		DISCRETE_OUTPUT_REGISTER BIT 13 1= MM2 IS TO TERMINATE CURRENT OPERATION AND ENTER THE STANDBY STATE. THE CURRENT OPERATION (IF ANY) WILL CONTINUE UNTIL THE END OF THE CURRENT 512 WORD BLOCK. 0= MM2 IS TO CONTINUE CURRENT OPERATION.			BT	
&010.10	DISCRETE_OUTPUT_ SPARE_BITS_FIELD_3		DISCRETE_OUTPUT_REGISTER BITS 14-19			BS(6)	
&010.11	GPC_SYNC_CODE_ COMMAND_DISCRETE_1		DISCRETE_OUTPUT_REGISTER BIT 20	363	363	BT	
&010.12	DISCRETE_OUTPUT_ SPARE BITS_FIELD_4		DISCRETE_OUTPUT_REGISTER BITS 21-23			BS(3)	
&010.13	GPC_SYNC_CODE_ COMMAND_DISCRETE_2		DISCRETE_OUTPUT_REGISTER BIT 24	363	363	BT	
&010.14	DISCRETE_OUTPUT_ SPARE_BITS_FIELD_5		DISCRETE_OUTPUT_REGISTER BITS 25-27			BS(3)	
&010.15	GPC_SYNC_CODE_ COMMAND_DISCRETE_3		DISCRETE_OUTPUT_REGISTER BIT 28	363	363	BT	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
E010.16	DISCRETE_OUTPUT_SPARE_BITS_FIELD_6		DISCRETE OUTPUT REGISTER BITS 29-31			BS(3)	
E011.1	VOTER_LATCH		PROVIDES A VOTER FAILED SIGNAL.	495		BT	
E011.2	TIMER_LATCH		PROVIDES A TIMEOUT SIGNAL.	495		BT	
E012	MM_LOCAL_STORE						
E012.01	MM_STATUS_REGISTER		TWO 16-BIT STATUS REGISTERS (A AND B) CONTAINED WITHIN THE MMU		282	BS(32)	
E012.02	MM_BLOCK_COUNT_REGISTER		MM BLOCK COUNT TO BE READ		282	BS(16)	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
#001	GPC_ID	TFCMID	THE ID OF GPC SELF (VALID NUMBERS ARE 1,2,3,4,5)	300 183 300 320 351 355 392 400 490 495 505 630 750 810 820 830 880	142	I,INIT(1)	
#002	INITIALIZATION_ FLAGS	TFCMFLAG	INITIALIZATION FLAGS	101 720	101 750	BS(15)I,INIT(0)	
#002.1	IPL_IN_PROGRESS_ INDICATOR		BIT_IPL INDICATOR 1=IN PROGRESS 0=NOT IN PROGRESS			BT	
#002.2	NORMAL INITIALIZATION_ INDICATOR		BIT 1 NORMAL INITIALIZATION INDICATOR 1=IN PROGRESS 0=NOT IN PROGRESS			BT	
#002.3	POWER_TRANSIENT_ INDICATOR		BIT 2 POWER TRANSIENT 1=IN PROGRESS 0=NOT IN PROGRESS			BT	
#002.4			BIT 3 IS NOT USED			BT	
#002.5			BIT 4 IS NOT USED			BT	
#002.6			BIT 5 IS NOT USED			BT	
#002.7			BIT 6 IS NOT USED			BT	
#002.8			BIT 7 IS NOT USED			BT	
#002.9	TMP_ENABLE_ INDICATOR		BIT 8 TMP INDICATOR 1=ENABLED 0=DISABLED			BT	
#002.10	SSIP_SCHEDULE_ INDICATOR		BIT 9 SSIP SCHEDULE INDICATOR 1=IN PROGRESS 0=NOT IN PROGRESS	101	101	BT	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
#002.11			BIT 10 IS NOT USED			BT	
#002.12			BIT 11 IS NOT USED			BT	
#002.13			BIT 12 IS NOT USED			BT	
#002.14			BIT 13 IS NOT USED			BT	
#002.15			BIT 14 IS NOT USED			BT	
#002.16			BIT 15 IS NOT USED			BT	
#003	REQUESTED_I/O_ TRANSMITTER_MASK	TFCMXMSK	THE CURRENT TRANSMITTER CONFIGURATION WHERE BITS 1-24 REPRESENT BUSES 1-24	351 355	205 250 350 351	BS(32),INIT(0)	
#004	REQUESTED_I/O_ RECEIVER_MASK	TFCMRMSK	THE CURRENT RECEIVER CONFIGURATION WHERE BITS 1-24 REPRESENT BUSES 1-24	351 355	351	BS(32),INIT(HFX 78BEEF80)	
#005	TERMINATION_ CONTROL_LATCH_SET_ MASK	TFCMTRML	MASK FOR SETTING TERMINATION CONTROL LATCHES.BITS 0-29 ARE NOT USED		340	BS(32),INIT(0)	
#005.1	TIMER_TERMINATION_ CONTROL_LATCH		BIT 30-TIMER TERMINATION_CONTROL LATCH INDICATOR 1=SET 0=NOT SET		240	BT	
#005.2	VOTER_TERMINATION_ CONTROL_LATCH		BIT 31-THE VOTER TERMINATION CONTROL LATCH INDICATOR 1=SET 0=NOT SET		240	BT	
#006	GO/NO_GO_CLOCK_ VALUE	TFCMGNGV	GO/NO CLOCK VALUE		142	DI,INIT(-235) UMI,768 MICROSECONDS)	
#007	TMP_PROCESSING_ FREQUENCY	TFCMTMP	FREQUENCY OF TIME MANAGEMENT PROCESSING (TMP) IN MINOR CYCLES			I,INIT(18)	
#008	TMP_PROCESSING_ PHASE_COUNT	TFCMTMPC	CURRENT NUMBER OF CYCLES REMAINING BEFORE TMP			I,INIT(0)	
#009	GO/NO_GO_ PROCESSING_ FREQUENCY	TFCMGNGF	GC/NO_GC PROCESSING FREQUENCY IN SSTP MINOR CYCLES		142	I,INIT(4)	
#010	DUTY_CYCLE_ COMPUTATION_ FREQUENCY	TFCMDCYF	DUTY CYCLE COMPUTATION FREQUENCY IN SSIP MINOR CYCLES		142	I,INIT(25)	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
#011	GO/NO_GO_PHASE_COUNT	TFCMGNGP	GO/NO_GO PHASE COUNT SSIP CYCLES UNTIL GO/NO_GO PROCESSING	142		I,INIT(4)	
#012	DUTY_CYCLE_COMPUTATION_PHASE_COUNT	TFCMDCYP	DUTY CYCLE PHASE COUNT-SSIP CYCLES UNTIL DUTY CYCLE COMPUTATION	142		I,INIT(25)	
#013	MTU_ACCUMULATOR_GO/NO_GO_STATUS	TFCMMASB	GO/NO_GO STATUS OF MTU ACCUMULATORS	149		BS(16),INIT(0)	
#013.1	MTU_1_STATUS		BIT 0- ACCUMULATOR ONE INDICATOR 1=FAULTED 0=NOT FAULTED	149	CRT	BT	V93X4103X
#013.2	MTU_2_STATUS		BIT 1 - ACCUMULATOR TWO INDICATOR 1 = FAULTED 0 = NOT FAULTED	149	CRT	BT	V93X4105X
#013.3	MTU_3_STATUS		BIT 2-ACCUMULATOR THREE INDICATOR 1=FAULTED 0=NOT FAULTED	149	CRT	BT	V93X4107X
#013.4	GPC_PRIME_TIME_STATUS		BIT 3-GPC PRIME TIME INDICATOR 1=FAULTED 0=NOT FAULTED	149	CRT	BT	V93X4109X
#014	TMP_MTU_I/O_COMPLETION_EVENT	TFCMTUEV	AN EVENT TO SIGNAL I/O_COMPLETION FOR TMP MTU READ	750	750	E,INIT(FALSE)	
#015	MTU_RM_RESULTS	TFCMMTAG	MTU REDUNDANCY MAMAGEMENT RESULTS			A(2),SC,INIT(0)	
#016	MTU_TRANSACTION_STATUS	TFCMMRTS	MTU TRANSACTION STATUS (READ)			SC,INIT(0)	
#017	MTU_1_I/O_BUFFER		MTU 1 I/O BUFFER				
#017.1	MTU_1_TRANSACTION_STATUS	TFCMMSW1	MTU 1 TRANSACTION STATUS			SC,INIT (0)	
#017.2	MTU_1_BUFFER	TFCMMTU1	MTU 1 READ I/O BUFFER	267	955 CRT DL	A(6),I, INIT(64,540)	V91W8850C V91W8852C V91W8854C
#017.3	MTU_1_BITE_WORD	TFCMM1BW	MTU 1 BITE WORD BITS 7-12 AND 16 NOT REFERENCED	149	149 955 CRT DL	BS(16),INIT(0)	V75M3040P
#017.31	CONTROLLING_OSCILLATOR		BIT 1-OSCILLATOR 1=PRIMARY OSCILLATOR 0=SECONDARY OSCILLATOR		CRT	BT	
#017.32	OSCILLATOR_1_		BIT 2-PRIMARY OSCILLATOR AMPLITUDE		955	BT	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	OST	ATTRIBUTES	MML
	AMPLITUDE		STATUS 1=OUT OF LIMITS 0=IN LIMITS				
#17.33	OSCILLATOR_2_ AMPLITUDE		BIT 3-SECONDARY OSCILLATOR 1=OUT OF LIMITS 0=IN LIMITS		955	BT	
#017.34	OSCILLATOR_ FREQUENCY_ DIFFERENCE		BIT 4-OSCILLATOR FREQUENCY STATUS 1=OUT OF LIMITS 0=IN LIMITS		CRT	BT	
#017.35	OSCILLATOR_1_ TEMPERATURE		BIT 5-PRIMARY OSCILLATOR TEMPERATURE STATUS 1=OUT OF LIMITS 0=IN LIMITS		955	BT	
#017.36	OSCILLATOR_2_ TEMPERATURE		BIT 6-SECONDARY OSCILLATOR TEMPERATURE STATUS 1=OUT OF LIMITS 0=IN LIMITS		955	BT	
#017.37	FREQUENCY_DIVIDER_ 1		BIT 13-ACCUMULATOR 1 STATUS 1=FAILED 0=GOOD	149		BT	
#017.38	FREQUENCY_DIVIDER_ 2		BIT 14-ACCUMULATOR 2 STATUS 1=FAILED 0=GOOD	149	CRT	BT	
#017.39	FREQUENCY_DIVIDER_ 3		BIT 15-ACCUMULATOR 3 STATUS 1=FAILED 0=GOOD	149	CRT	BT	
#018	MTU_2_I/O_BUFFER		MTU 2 I/O BUFFER				
#018.1	MTU_2_TRANSACTION_ STATUS	TFCMMSW2	MTU 2 TRANSACTION STATUS			SC,INIT(0)	
#018.2	MTU_2_BUFFER	TFCMMTU2	MTU 2 READ I/O BUFFER	267	955 CRT DL	A(6),I, INIT(64,5#0)	V91W8856C V91W8858C V91W8860C
#018.3	MTU_2_BITE_WORD	TFCMM28W	MTU 2 BITE WORD		149 CRT DL	BS(16),INIT(0)	V75W3140P
#019	MTU_3_I/O_BUFFER		MTU 3 I/O BUFFER				
#019.1	MTU_3_TRANSACTION_ STATUS	TFCMMSW3	MTL 3 TRANSACTION STATUS			SC,INIT(0)	
#019.2	MTU_3_BUFFER	TFCMMTU3	MTL 3 READ I/O BUFFER	267	955	A(6),I,	V91W8862C

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	YML
					CRT	INIT(64,5*0)	V91W8864C
					DL		V91W8866C
#019.3	MTU_3_BITE_WORD	TFCMM3BW	MTU 3 BITE WORD		149	BS(16),INIT(0)	V75M3240P
					CRT		
					DL		
#020	MTU_WRITE_TRANSACTION_STATUS	TFCMMWTS	MTU TRANSACTION STATUS (WRITE)			SC,INIT(0)	
#021	MTU_WRITE_I/O_BUFFER		MTU WRITE I/O BUFFER				
#021.1	MTU_WRITE_STATUS_WORD	TFCM4WSW	MTU WRITE STATUS WORD			SC,INIT(0)	
#021.2	MTU_WRITE_BUFFER	TFCMWMTU	MTU WRITE I/O BUFFER			A(4),I,INIT(0)	
#022	MTU_UPDATE_TIME		MTU UPDATE TIME BUFFER	148	140		
					142		
#022.1	MTU_UPDATE_HALF_HOUR_PORTION	TFCMMTUH	THE 30 MINUTE ELAPSED TIME PORTION OF UPDATE TIME	148	140	DI,INIT(0)	
					142		
#022.2	MTU_UPDATE_HALF_HOUR_COUNT	TFCMMTUM	THE COUNT OF ELAPSED HALF-HOURS OF UPDATE TIME	148	140	I,INIT(0)	
					142		
#023	SOFTWARE_CLOCK_OFFSET	TFCMOFST	SOFTWARE CLOCK OFFSET	148	140	DI,INIT(0), UM(MICRO-SECONDS)	
					148		
#024	BTU_PORT_MASKS	TFCMBPM	COMPOSITE OF DATA PATH MASKS FOR ALL GPC'S	290	205	A(7),I,INIT(0)	V91U7201C
					290		V91U7218C
					355		V91U7235C
					368		V91U7252C
					DL		V91U7269C
					CRT		V91U7286C
							V91U7303C
							V91X7400X
							V91X7404X
							V91X7408X
							V91X7412X
							V91X7416X
							V91X7418X
							V91X7420X
							V91X7422X
							V91X7424X
							V91X7426X
							V91X7428X
							V91X7430X
			BIT 0-BUS 10				
			BIT 1-BUS 11				
			BIT 2-BUS 12				
			BIT 3-BUS 13				
			BIT 4-BUS 14				
			BIT 5-BUS 15				
			BIT 6-BUS 16				
			BIT 7-BUS 17				
			BIT 8-BUS 18				
			BIT 9-BUS 19				
			BIT10-BUS 20				
			BIT11-BUS 21				
			BIT12-BUS 22				
			BIT13-BUS 23				
#025	LAST_EXPIRED_TQE_TIME		TIME OF LAST EXPIRED TQE		140		V91W1999C
					142		

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
					144		
					149		
#025.1	LAST_TQE_HALF_HOUR_PORTION	TFCMLTQH	THE HALF-HOUR PORTION OF TIME OF THE LAST EXPIRED TQE	142 149	140 144 246 DL	F,INIT(0) UM(MICRO-SECONDS)	V91W1999C
#025.2	LAST_TQE_HALF_HOUR_COUNT	TFCMLTQM	THE HALF-HOUR COUNTER PORTION OF TIME OF THE LAST EXPIRED TQE	142 149	140 144 DL	I,INIT(0) UM(HALF HOURS)	V91W1999C
#026	DATA_PATH_MASKING_THRESHOLD_COUNT	TFCMNCTR	THRESHOLD COUNT FOR DATA PATH MASKING AND ANNUNCIATION	ILD MRW		I,INIT(3)	V92Q4000C
#027	BCE_ELEMENT_BYPASS_THRESHOLD_COUNT	TFCMKCTR	THRESHOLD COUNT FOR BCE ELEMENT BYPASSING AND ANNUNCIATION	MRW ILD	DL	I,INIT(3)	V91Q2235C
#028	MULTIPLE_FC_FAILURE_THRESHOLD_COUNT	TFCMPCTR	THRESHOLD COUNT FOR MULTIPLE FF OR FA BUS FAILURES	ILD MRW	DL CPT	I,INIT(1)	V91M2217P
#029	ELIGIBLE_SET_MASK	TFCMESM	BITS 0-19 ARE NOT USED.BITS 20-31 CONTAIN THE ELIGIBLE SET MASK USED BY THE SYNC ROUTINES (SYNC DISCRETE FORMAT)			BS(32),INIT(0)	
#030	BCE_ELEMENT_BYPASS_INDICATOR_1	TFCMBCEBI/FIOBCEBI	BCE ELEMENT BYPASS INDICATOR FOR COMFAULT 1	220 244 330 355	220 244 330 DL	BS(39),INIT(0)	V91M2252P
#031	MTU_COMFAULT1_WORD2	FIOBCSI2	MTU COMFAULT DATA	205	149 DL	BS(32),INIT(0)	V91M2303P
#032	BCE_ELEMENT_BYPASS_INDICATOR_2	TFCMBCBI2/FIOBCBI2	BCE ELEMENT BYPASS FOR COMFAULT2	220 244 330	220 244 330 355 DL	BS(32),INIT(0)	V91M2269P
#200	N_COUNTER_TABLE	#PFIDECT	THIS TABLE CONTAINS DATA PATH ERROR COUNTERS (N_COUNTERS) AND CONTROL INFORMATION NEEDED TO REFERENCE THEM.(RESIDES IN #PFIDECT)			ST	
#200.1	N_COUNTER_POINTERS	FIOBUS06	A TABLE OF POINTERS TO DATA_PATH_ERROR COUNTERS ARRANGED BY BUS NUMBER (BUS 6-21)		355	A(34) V, INITIAL(EACH ENTRY CONTAINS A POINTER TO A DATA PATH ERROR	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
						COUNTER)	
#200.2	LDB/PMU_N_COUNTER_POINTERS	FIOBUS22	A TABLE OF POINTERS TO DATA_ERROR_COUNTERS ARRANGED BY BUS NUMBER (BUS 22-24)			A(15),Y, INITIAL (EACH ENTRY CONTAINS A POINTER TO A DATA PATH ERROR COUNTER)	
#200.3	DATA_PATH_ERROR_COUNTER_TABLE	FIODPECT	A TABLE CONTAINING POINTERS TO N_COUNTER_POINTERS OR LDB/PMU_N_COUNTER_POINTERS AND THE NUMBER OF POINTERS IN THE TABLE	355		(Y,X)(25), INITIAL (THE FIRST SIX ENTRIES ARE UNUSED, ENTRIES 7-25 CONTAIN A POINTER TO A TABLE OF POINTERS AND A COUNT OF THE NUMBER OF POINTERS)	
#200.4	DATA_PATH_ERROR_COUNTERS	FIOB0011	A COUNT OF THE NUMBER OF CONSECUTIVE INPUT ERRORS WHICH OCCURRED ON A DATA PATH	355	268	A(6),F	
#200.5	IUA_11_N_COUNTERS	FIOBJ611				A(8),F,INIT(0)	
#200.6	IUA_12_N_COUNTERS	FIOB0012		244 268	268	A(11),F,INIT(0)	
#200.7	IUA_13_N_COUNTERS	FIOB0013		244 268	268	A(8),F,INIT(0)	
#200.8	IUA_14_N_COUNTERS	FIOB0014				A(8),F,INIT(0)	
#200.6	IUA_17_N_COUNTERS	FIOB0017				A(4),F,INIT(0)	
#201	WIX_TABLE	FIOWIXTB	FOR EACH VALID LISTEN INDEX (0<=N<=254,WHERE N IS AN EVEN NUMBER SENT BY A LISTEN COMMAND)AN ENTRY IN THIS TABLE IS INITIALIZED TO POINT TO A BCE PROGRAM.(RESIDES IN #PFIOFCT)	205 220 390	260 261 262 263 264 265 266 267 268 282	A(128),AC	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
@001	APPLICATION_LOCAL_DATA_AREA		LOCAL DATA AREA SUBSET WHICH CONTAINS STATUS INFORMATION NEEDED BY FCOS			ST	
@001.1	BLOCK_ID	TLDABLCK	IDENTIFIER FOR BLOCK WITHIN UNIT OF COMPILATION			I	
@001.2	BLOCK_INFORMATION	TLDAEINF	BLOCK INFORMATION			BS(16)	
@001.3	UPDATE/EXCLUSIVE_INDICATOR		TLDAEINF BIT 0: 1=BLOCK IS AN UPDATE BLOCK OR AN EXCLUSIVE PROCEDURE 0=NOT AN UPDATE BLOCK OR EXCLUSIVE PROCEDURE		108 182 184	BT	
@001.4	NUMBER_OF_ERROR_VECTORS		TLDAEINF BITS 1-6 CONTAINS THE NUMBER OF ERROR VECTORS EXISTING FOR THIS BLOCK		182	BS(6)	
@001.5	ERROR_VECTOR_DISPLACEMENT		TLDAEINF BITS 7-15 CONTAINS THE DISPLACEMENT FROM THE START OF THE STACK FRAME TO THE ERROR VECTORS		182	BS(9)	
@001.6	RESERVE_SVC_PARAMETER_LIST	TLDARESP	RESERVE/RELEASE SVC PARAMETER LIST IF TLDAEINF BIT 0=1		108 182 184	I	
@001.7	RELEASE_SVC_PARAMETER_LIST	TLDARELP	RESERVE/RELEASE SVC PARAMETER LIST IF TLDAEINF BIT 0=1		108 182 184	I	
@001.8	LOCK_GROUP_INDICATORS	TLDALOCK	RESERVE/RELEASE SVC PARAMETER LIST IF TLDAEINF BIT 0=1		108 182 184	BS(16) OP ✓	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
@002	STACK_FRAME	TFTSA	THE STACK (OR TEMPORARY STORAGE AREA) IS USED BY APPLICATION PROCESSES FOR REGISTER SAVING ACROSS CALLS, DATA STORAGE AREAS, AND ERROR ENVIRONMENT DEFINITION. THE CONTENTS OF THE STACK FRAME VARIES DEPENDING ON THE PROCESS. THE FOLLOWING IS THE BASIC STACK FRAME LAYOUT	101	182 290	ST	
@002.1	PSW_ADDRESS_FIELD	TTSASTRT	THE ADDRESS FIELD OF THE PSW TO BE IN EFFECT UPON RETURN TO THE CALLING ROUTINE.(THIS FIELD IS 0 IN THE FIRST LEVEL STACK FRAME)			V	
@002.2	PSW_PROGRAM_MASK_FIELD	TTSAPGMM	THE SECOND HALF OF THE FIRST WORD OF THE PSW TO BE IN EFFECT UPON RETURN TO THE CALLING ROUTINE.(FIRST LEVEL STACK FRAME CONTAINS INITIAL PROGRAM MASK FOR THE PROCESS)	101	182	I	
@002.3	PREVIOUS_STACK_FRAME_ADDRESS	TTSAPREV	ADDRESS OF THE STACK FRAME OF THE CALLING BLOCK (THIS FIELD IS 0 IN THE FIRST LEVEL STACK FRAME)	101	108 182 184	V	
@002.4	PREVIOUS_STACK_FRAME_SIZE		SIZE OF THE STACK FRAME OF THE CALLING BLOCK			I	
@002.5	APPLICATION_REGISTER_1_SAVE		CONTENTS OF GENERAL PURPOSE REGISTER 1 OF THE CALLING BLOCK			I	
@002.6	PROGRAM_LEVEL_DATA_ADDRESS	TTSAPLDA	ADDRESS OF THE PROGRAM LEVEL DATA AREA FOR THE CURRENT BLOCK		182	V	
@002.7	APPLICATION_REGISTER_2_SAVE		CONTENTS OF GENERAL PURPOSE REGISTER 2 OF THE CALLING BLOCK			U	
@002.8	APPLICATION_REGISTER_3_SAVE		CONTENTS OF GENERAL PURPOSE REGISTER 3 OF THE CALLING BLOCK			I	
@002.9	LOCAL_DATA_AREA_ADDRESS	TTSALCDA	ADDRESS OF THE CURRENT BLOCK'S LOCAL DATA AREA	101	108 182 184 290	V	
@002.10	APPLICATION_REGISTER_4_SAVE		CONTENTS OF GENERAL PURPOSE REGISTER 4 OF THE CALLING BLOCK			U	
@002.11	APPLICATION_REGISTER_5_SAVE		CONTENTS OF GENERAL PURPOSE REGISTER 5 OF THE CALLING BLOCK			U	

STACK FRAME LAYOUT

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	OST	ATTRIBUTES	MML
@002.12	APPLICATION_REGISTER_6_SAVE		CONTENTS OF GENERAL PURPOSE REGISTER 6 OF THE CALLING BLOCK			F	
@002.13	APPLICATION_REGISTER_7_SAVE		CONTENTS OF GENERAL PURPOSE REGISTER 7 OF THE CALLING BLOCK			F	
@002.14	MISCELLANEOUS_PROGRAM_UNIQUE_INFORMATION		VARIOUS DATA INFORMATION DEPENDING ON THE PROCESS			VARIABLE	
@002.15	ERROR_VECTORS		FULLWORD FIELDS CONTAINING INFORMATION INDICATING THE ERROR ENVIRONMENT FOR THIS BLOCK		182	BS(32)	
@002.16	ERROR_ACTION_CODE		ERROR_VECTORS BITS 0-3 0000-GO TO SPECIFIED ADDRESS XX01-SYSTEM XX11-IGNORE ERROR 00XX-NO EVENT ACTION 01XX-SET EVENT SPECIFIED 10XX-RESET EVENT SPECIFIED 11XX-SIGNAL EVENT SPECIFIED		182	BS(4)	
@002.17	ERROR_CODE		ERROR_VECTORS BITS 4-9: 63=ENTIRE GROUP		182	BS(6)	
@002.18	ERROR_GROUP		ERROR_VECTORS BITS 10-15: 63=ALL GROUPS 0=NULL ENTRY		182	BS(6)	
@002.19	ERROR_VARIABLE_FIELD		ERROR_VECTORS BITS 16-31: ADDRESS IF GO TO SPECIFIED OR EVENT ADDRESS IF EVENT ACTION SPECIFIED		182	Y	

STACK FRAME LAYOUT

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
@003	IDLE_TIME	FPMIDTIM	IDLE TIME ACCUMULATION USED IN DUTY CYCLE COMPUTATION:(RESIDES IN FPMIDTIM)	142 191	142 191	SC,INIT(0)	
@004	GST_ADDRESS_TABLE	FPMGSTAD	TABLE OF ADDRESSES TO THE GPC STATUS TABLE INDEXED BY GPC ID (RESIDES IN FCMCBLKS)	368	149 290 351 355 368	A(5),V,INIT(0 ADR OF GST FOR GPC1, ADDR OF GST FOR GPC 2, ADDR OF GST FOR GPC 3, ADDR OF GST FOR GPC4)	
@005	SVC_ENTRY_POINT_TABLE	FPMSVCEP	THE SVC_ENTRY_POINT_TABLE IS A TABLE OF ADDRESSES TO THE SVC SERVICE ROUTINES. THE TABLE IS INDEXED BY THE SVC NUMBER CINTAINED IN THE SVC PARAMETER LIST		100	A(43),V,INIT (SEE APPENDIX H)	
@006	MTU_BUFFER_TABLE	TFMTU	A THREE HALF WORD BUFFER CONTAINING THE TIME IN MTU FORMAT				
@006.1	MTU_DAYS_HOURS	TMTUDYHR	THIS HALF WORD CONTAINS DAYS AND HOURS	148	148 165	BS(16)	
@006.2	MTU_DAYS		BIT NO. DAYS 0 200 1 100 2 80 3 40 4 20 5 10 6 8 7 4 8 2 9 1	148	148 165	BS(10)	
@006.3	MTU_HOURS		BIT NO. HOURS 10 20 11 10 12 8 13 4 14 2 15 1	148	148 165	BS(6)	
@006.4	MTU_MINUTES_SECONDS	TMTUMNSC	THIS HALFWORD CONTAINS MINUTES AND SECONDS	148	148 165	BS(16)	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
@006.5	MTU_MINUTES		BIT NO. MINUTES 0 40 1 20 2 10 3 8 4 4 5 2 6 1	148	148 165	BS(7)	
@006.6	MTU_SECONDS		BIT NO. SECONDS 7 40 8 20 9 10 10 8 11 4 12 2 13 1 14 UNUSED 15 UNUSED	148	148 165	BS(9)	
@006.7	MTU_MILLISECONDS	TMTUMSEC	THIS HALFWORD CONTAINS .125 MILLISECONDS	148	148 165	BS(16)	
@006.8	MTU_.125 MILLISECONDS		BIT NO. .125 MILLISECONDS 0-2 UNUSED 3 4096 4 2048 5 1024 6 512 7 256 8 128 9 64 10 32 11 16 12 8 13 4 14 2 15 1	148	148 165	BS(16)	
@007	EVENT_VARIABLE		THE FOLLOWING DESCRIBES ALL EVENT VARIABLES	182	170 171 172 173 182	BS(16)	
@007.1	EVENT_USED_ INDICATOR		EVENT_VARIABLE BITS 0_14: INDICATES IF A PROCESS' EXECUTION IS		107 108 109	BS(15)	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
@007.2	EVENT_STATE		DEPENDENT ON THE EVENT STATE. THERE ARE THREE POSSIBLE SETTINGS FOR THESE BITS: 1. ZERO-NO PROCESS IS DEPENDENT ON THE EVENT 2. EVENT_QUEUE_ELEMENT(Q006)- ONE PROCESS IS DEPENDENT ON THE EVENT 3. COUNT OF USES-THE EVENT APPEARS MORE THAN ONCE IN EVENT_ QUEUE_ELEMENTS(Q006) EVENT_VARIABLE BIT 15-STATE OF THE EVENT (TRUE=1, FALSE=0)	101 107 108 109 142 182	110 170 171 172 173 320	BT	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
0001	APPLICATION_OVERRIDE_TABLE		AN INITIALIZATION CONTROL TABLE WHICH CONTAINS DEFAULT VALUES OF PARAMETERS NEEDED TO INITIATE SYSTEM EXECUTION (RESIDES IN FCMAOT)	300	340	ST	
0001.1	BOOTSTRAP_PDE_ADDRESS	TAOTPDE	ADDRESS OF THE APPLICATION BOOTSTRAP PROGRAM PDE		300 305	A(4),X,INIT(01F 80001)	
0001.2	BOOTSTRAP_PRIORITY	TAOTPRIO	PRIORITY OF THE APPLICATION BOOTSTRAP PROGRAM		300 305	I,INIT(50)	
0002	MASS_MEMORY_PHASE_TABLE	FCMMMPT	A DESCRIPTION OF EACH MASS MEMORY PHASE.THIS IS READ FROM THE MASS MEMORY (RESIDES IN FCMCBLKS)		320	ST	
0002.1	NUMBER_OF_PHASE_TABLE_RECORDS		NUMBER OF PHASE TABLE RECORDS.ALSO USED TO INDICATE IF TABLE HAS BEEN READ	280 320	280 320	I,INIT(1)	
0002.2			NOT USED			I	
0002.3	PHASE_INFORMATION		PHASE INFORMATION FOR EACH OF THE N PHASES. EACH ENTRY IS DESCRIBED BELOW IN 0002.4,0002.5		320	ST(N)	
0002.4	INDEX_TO_PHASE_SEGMENTS		INDEX TO THE START OF SEGMENT ENTRIES FOR THIS PHASE		320		
0002.5	COUNT_OF_SEGMENTS		NUMBER OF SEGMENTS IN THIS PHASE		320	I	
0002.6	SEGMENT_INFORMATION	TMMPSTRT	A TABLE DESCRIBING EACH SEGMENT IN THIS PHASE. N IS THE COUNT DESCRIBED BY 0002.5. THE TABLE ENTRIES ARE DESCRIBED BELOW		320	ST(N)	
0002.7	SEGMENT_MAIN_MEMORY_ADDRESS	TMMPADD	MAIN MEMORY ADDRESS FOR THIS SEGMENT		320	Y	
0002.8	SEGMENT_PROTECT_STATUS	TMMPPBD	PROTECT STATUS AND ADDITIONAL ADDRESS INFORMATION DESCRIBED BELOW		320	I	
0002.9	SEGMENT_PROTECT_FLAG		TMMPPBD BIT 0 1=PROTECTED 0=UNPROTECTED		320	BT	
0002.10			TMMPPBD BITS 1-7 UNUSED			BS(7)	
0002.11	SEGMENT_ADDRESS_BSR		TMMPPBD BITS 8-11 THE BSR USED IN THE SEGMENTS MAIN MEMORY ADDRESS		320	BS(4)	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
0002.12	SEGMENT_ADDRESS_DSR		THMPPBD BITS 12-15 THE DSP USED IN THE SEGMENTS MAIN MEMORY ADDRESS			BS(4)	
0002.13	SEGMENT_LENGTH	TMMPLNG	LENGTH OF THE SEGMENT (NUMBER OF HALFWORDS)		320	I	
0002.14	SEGMENT_MASS_MEMORY_ADDRESS	TMMPMMA	MASS MEMORY ADDRESS OF THIS SEGMENT BITS 0,1-UNUSED BITS 2-4-FILE ADDRESS BITS 5-7-TRACK ADDRESS BITS 8-10-SUBFILE ACDRESS BITS 11-15-STARTING BLOCK NUMBER		320	I	
0003	BUS_MANAGEMENT_WORK_AREA	TBMWSTRT	WORK AREA USED IN PERFORMING BUS MANAGEMENT FUNCTIONS (RESIDES IN FCMCBLKS)			ST	
0003.1	MASK_OF_BUSES_BEING_RECONFIGURED	TBMWMBRR	MASK OF BUSES BEING RECONFIGURED BIT 0-UNUSED BIT 1=BUS 1 BIT 2=BUS 2 . . . BIT 24=BUS 24 BITS 25-31 UNUSED	350	350 351	F	
0003.2	OLD_REDUNDANT_SET_MASK	TBMWRMSK	REDUNDANT SET MASK WHEN IT IS BEING MODIFIED BITS 0-10 UNUSED BIT-11=GPC1 BIT-12=GPC2 BIT-13=GPC3 BIT-14=GPC4 BIT-15=GPC5	368	368 392	I	
0003.3	OLD_COMMON_SET_MASK	TBMWCMSK	COMMON SET MASK WHEN MASK IS BEING MODIFIED (SAME AS 0003.2)			I	
0003.4	CURRENT_LDB_BUS	TBMWCLDB	BUS NUMBER BEING USED IN LDB TRANSACTIONS	351	351	I,INIT(22)	
0003.5	INTERNAL_PARAMETER_LIST_BUFFER	TBMWBDPM	CONTAINS THE BUS/DATA PATH MASK MANAGEMENT PARAMETER LIST (SEE S035 AND S036)	351 368		I(4)	
0003.6	CURRENT_PL_BUS_MODE	TBMWCPL	CURRENT PAYLOAD MODING NUMBER	351	351	I,INIT(6)	
0004	OVERLAY_I/O_SVC_	FCMOVIOS	THE BUFFER USED FOR THE MASS	320		ST	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
	PARAMETER_LIST		MEMORY I/O REQUESTS ISSUED BY PROGRAM OVERLAY.(RESIDES IN FCMCBLKS)				
0004.1	OVERLAY_I/O_SVC		SVC NUMBER FOR MM I/O			X,INIT (0018)	
0004.2	OVERLAY_I/O_FLAGS		I/C FLAGS FOR OVERLAY I/O: WAIT TYPE I/O,PROTECTED TRANSACTION, STATUS REQUESTED			X,INIT (0070)	
0004.3	OVERLAY_DEVICE_ID		DEVICE ID FOR MASS MEMORY			I,INIT (10)	
0004.4	OVERLAY_OPERATION_CODE		OPERATION CODE FOR READ	320	320	I,INIT (3)	
0004.5			UNUSED			I	
0004.6	OVERLAY_WORD_COUNT		WORD COUNT FOR READ	320		I,INIT (0)	
0004.7	OVERLAY_BUFFER_ADDRESS		ADDRESS OF THE TARGET MAIN MEMORY	320		I,INIT (0)	
0004.8			UNUSED			I	
0004.9	OVERLAY_MM_ADDRESS		MASS MEMORY STARTING BLOCK ADDRESS	320		I,INIT (0)	
0005	ICC_BUFFER_ADDRESS_TABLE	FCMICCAD	TABLE OF ADDRESSES TO THE ICC BUFFERS INDEXED BY GPC ID. (RESIDES IN FCMCBLKS)		142 149	A(5),Y, INIT(ADDR OF ICC BUFFER 1, ADDR OF ICC BUFFER 2, ADDR OF ICC BUFFER 3, ADDR OF ICC BUFFER 4)	
0006	BCE_ELEMENT_MODIFICATION_TABLE	FCMBMT	THE BCE_ELEMENT_MODIFICATION_TABLE IS USED BY THE BCE_ELEMENT_BYPASS_PROCESSOR TO BYPASS OR RESTORE ELEMENTS OF BCE CHAINS		330	ST(49)	
0006.1	BCE_ELEMENT_ADDRESS		CONTAINS THE ADDRESS OF A BCE ELEMENT WHICH IS TO BE BYPASSED OR THE CONTENTS AT THE ADDRESS WILL BE A BCE BRANCH INSTRUCTION IF THE ELEMENT IS BEING BYPASSED AND A SET OF BCE DELAY INSTRUCTIONS IF THE ELEMENT IS BEING RESTORED		330	AC	
0006.2	BCE_DELAY_INSTRUCTIONS		CONTAINS A SET OF BCE DELAY INSTRUCTIONS TO OVERLAY A BCE BRANCH INSTRUCTION POINTED TO BY BCE_ELEMENT_ADDRESS		330	I(2),INIT(TWO BCE DELAY INSTRUCTIONS)	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SEC	OST	ATTRIBUTES	MML
0006.3	ORIGINAL_BCE_CODE		CONTAINS A BCE BRANCH INSTRUCTION WHICH WILL OVERLAY A SET OF BCE DELAY INSTRUCTIONS POINTED TO BY BCE_ELEMENT_ADDRESS		330	I(2),INIT(A BCE BRANCH INSTRUCTION)	
0007	SYSTEM_SOFTWARE_LOADER_PHASE_TABLE	FCMSSLPT	SYSTEM SOFTWARE MEMORY CONFIGURATION MAP		300	ST	
0007.01	NUMBER_OF_PHASE_TABLE_ENTRIES		NUMBER OF ENTRIES IN THE TABLE (EACH ENTRY DESCRIBES WHERE A BLOCK OF INFORMATION IS ON MASS MEMORY, WHERE IT IS TO GO ON MAIN MEMORY AND PROTECT INFORMATION)		300	I	
0008	FCMINSSL_LOCAL_DATA	FCMDATA	SOFTWARE_SYSTEM_LOADER LOCAL DATA	300	300	ST	
0008.01	FCOS_BSR_DSR	FCMZCPSW	HALFWORD 0=ADDRESS OF PROCESSOR DISPATCHER BITS 16-19=RESERVED BITS 20-23=BSR BITS 24-31=DSR		300	Z	
0008.02	FCOS_PROGRAM_MASK	FCMPM	BITS 0-3=0 PSW_RESERVED_FIELD-2 BIT 4=1 PSW_REGISTER_SET BIT 5=0 PSW_MACHINE_CHECK_MASK BIT 7=0 PSW_WAIT_STATE_INDICATOR BIT 8=0 PSW_RUN_STATE_CONTROL BIT 9-15=0 1ST HALF OF PSW_INTERRUPT_CODE		300	BS(15)	
0008.03	ENTER_WAIT_STATE_PSW	FCMWTPSW	A PSW WITH PSW_INSTRUCTION_ADDRESS=ADDRESS OF MSC_INTERRUPT_ENTRY_POINT PSW_PROGRAM_MASK=X'08' PSW_BRANCH_SECTOR_REGISTER=0 PSW_DATA_SECTOR_REGISTER=0 PSW_SYSTEM_MASK=0 PSW_RESERVED_FIELD_2=0 PSW_REGISTER_SET=1 PSW_MACHINE_CHECK_MASK=1 PSW_WAIT_STATE_INDICATOR=1 PSW_RUN_STATE_CONTROL=0 PSW_INTERRUPT_CODE=0		300	A(3),F	
0008.04	BCE_MAX_TIMEOUT_VALUE	FCMTMCDE	MAXIMUM BCE TIMEOUT VALUE X'0003FFFF'=4.32 SECONDS	300	300	X	
0008.05	BCE_POSITION_TAPE_COMMAND_TABLE	FCMBCEPT	BCE POSITION TAPE COMMAND WORD TABLE	300		A(2),F	
0008.06	BCE_BASE_1	FCMBCEB1	BCE BASE REGISTER TABLE 1	300	302	A(2),F	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC #	DST #	ATTRIBUTES	MML
0008.07	BCE_BASE_2	FCMBCEB2	BCE BASE REGISTER TABLE 2	300	302	A(2),F	
0008.08	BCE_EXTEND_BLOCK_COMMAND_TABLE	FCMBCEEB	EXTEND BLOCK COMMAND TABLE-MASS MEMORY	300		A(2),F	
0008.09	BCE_WORD_COUNT_PARTIAL	FCMBCEC1	WORD COUNT TABLE FOR PARTIAL BLOCK	300		A(2),F	
0008.10	BCE_WORD_COUNT_WHOLE	FCMBCEC2	WORD COUNT TABLE FOR WHOLE BLOCKS	300		A(2),F	
0008.11	BCE_READ_COMMAND	FCMBCERD	READ COMMAND TABLE-MASS MEMORY	300		A(2),F	
0008.12	CHECKSUM_ERROR_COUNT	FCMECNT	MASS MEMORY CHECKSUM ERROR COUNTER	300	300	1	
0008.13	IPL_MM_BCE_MASK	FCMIMSM	MASK CONTAINING MM BCE INDICATORS	300	301	BS(32)	
0008.14	DOUBLE_SINGLE_RECEIVE_BRANCH_TABLE	FCMBCEBT	TABLE CONTAINING ADDRESS OF PROPER PATH FOR READ TAPE PORTION OF (302) SSL_MM_BCE_PROCESSOR (FCMINBCE)	300	302	A(2),F	
0009	MAX_WAIT_TIME	FCMICNT	CONTAINS THE MAXIMUM WAIT TIME ALLOWABLE FOR THE RAW INSTRUCTION		301	6	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
Q001	COMMUNICATIONS_ VECTOR_TABLE	TFCVT	THE CVT IS THE PRIMARY FCOS CONTROL TABLE WHICH CONTAINS ADDRESSES TO THE START OF FCOS QUEUES, POINTERS TO UNUSED CONTROL BLOCKS AND OTHER CONTROL INFORMATION (RESIDES IN FCMPSA)			ST	
Q001.1	PCT_RUN_QUEUE_ ADDRESS	TCVTPCT	ADDRESS OF THE TOP PCT IN THE PCT RUN QUEUE	131 132 305	106 110 131 132 133 300	Y,INIT(ADDRESS OF BOOTSTRAP PCT)	
Q001.2	ACTIVE_PCT_ADDRESS	TCVTOLD	ADDRESS OF THE ACTIVE PCT	103	100 102 103 104 105 107 108 109 142 180 181 182 185 200 201 220 280 320 351	Y,INIT(ADDR. OF BOOTSTRAP PCT)	
Q001.3	NEXT_TO_EXECUTE_ PCT_ADDRESS	TCVTNEW	ADDRESS OF THE NEXT TO EXECUTE PCT	102 103 104 105 106 133 142 200 201 280 305 320	100 102 103 140 142 180 181 220	Y,INIT(ADDR. OF BOOTSTRAP PCT)	
Q001.4	TOP_TQE_ADDRESS	TCVTTTQE	ADDRESS OF THE TOP TQE ON THE TIMER QUEUE.	142 143 166 305	140 142	Y,INIT(ADDR OF DUMMY TQE)	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
Q001.5	I/O_ACTIVE_QUEUE_ ADDRESS	TCVTIOA	ADDRESS OF THE TOP IOQE ON THE I/O ACTIVE QUEUE	205 220 305	210 225 242 361	Y,INIT(0)	
Q001.6	I/O_WAIT_QUEUE_ ADDRESS	TCVTIOW	ADDRESS OF THE TOP IOQE IN THE I/O WAIT QUEUE	200 201 220 225 305	200 201 220 225 351 361	Y,INIT(0)	
Q001.7	TOP_ACTIVE_IOQE_ 18-BIT_ADDRESS	TCVTMIOA	FULLWORD(18-BIT)ADDRESS OF THE TOP IOQE IN THE ACTIVE QUEUE	205 220 305	250 251	AC,INIT(0)	
Q001.8	CURRENT_MSC_ ITERATION_COUNT	TCVTMITC	CURRENT MSC ITERATION COUNT			F,INIT(0)	
Q001.9	PCT_FREE_POOL_ ADDRESS	TCVTPCTP	ADDRESS OF THE POOL OF UNUSED PCT'S	131 132 305	101 280 320	Y,INIT(ADDR. OF FIRST PCT IN FREE POOL)	
Q001.10	EQE_FREE_POOL_ ADDRESS	TCVTEQEP	ADDRESS OF THE FIRST UNUSED EQE IN THE CHAIN OF EQE'S	173 175	101 104 174	Y,INIT(ADDR.OF FIRST EQE)	
Q001.11	TQE_FREE_POOL_ ADDRESS	TCVTTQEP	ADDRESS OF THE POOL OF UNUSED TQE'S	142 143 166 305	101 104 107 140 148 174 280	Y,INIT(ADDR.OF THE FIRST TQE IN THE FREE POOL)	
Q001.12	I/O_FREE_POOL_ ADDRESS	TCVTIOFP	ADDRESS OF THE POOL OF UNUSED IOQ'S	200 220 225 280 305	225	Y,INIT(ADDRESS OF THE FIRST IOQE IN THE FREE POOL)	
Q001.13	STACK_POOL_ADDRESS	TCVTSTOR	ADDRESS OF THE POOL OF TEMPORARY STORAGE AREAS USED BY FCOS WHEN APPLICATION PROCESS NEEDS TO BE ASSIGNED AN AREA		101 300 305	Y,INIT(ADDR.OF SMALLEST STACK)	
Q001.14	TOP_EQE_ADDRESS	TCVTTEQE	ADDRESS OF THE FIRST EQE IN THE POOL OF EQE'S.THIS FIELD IS NEVER CHANGED		174 175	Y,INIT(ADDR. OF FIRST EQE IN POOL)	
Q001.15	TEMPORARARY PROCESS_EVENT_	TCVTPECH	ADDRESS OF THE TEMPORARY CHAIN OF PROCESS EVENT ADDRESSES USED	174	174	I,INIT(0)	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
	CHAIN_ADDRESS		DURING EVENT EVALUATION				
Q001.16	MASS_MEMORY_ACTIVE_QUEUE_ADDRESS	TCVTMMA	ADDRESS OF THE ACTIVE MASS MEMORY IOQE(S)	280 305	225 280	Y,INIT(0)	
Q001.17	BCE_BUSY/IDLE_INDICATORS	TCVTBCEB	MASK INDICATING WHICH BCE'S ARE BUSY. BIT 0 IS UNUSED CORRESPOND TO BCE'S 1-24: 1=BCE IS BUSY 0=BCE IS IDLE	200 201 205 220 280 305 350	200 201 280 350 355	F,INIT(0)	
Q001.18	START_I/O_BCE_MASK	TCVTSIOM	COMMUNICATION FROM THE CPU TO THE MSC OF THE "START I/O" BCE MASK	205 280 305	250	F,INIT(0)	
Q001.19	LOCK_GROUP_CONTROL_TABLE_ADDRESS	TCVTLOCK	ADDRESS OF THE LOCK GROUP CONTROL		105 106	Y,INIT(ADDR. OF LOCK_GROUP TABLE)	
Q001.20	ERROR_LOG_TABLE_MAXIMUM_INDEX	TCVTLMAX	THE LARGEST INDEX POSSIBLE FOR ENTRIES INTO THE GPC ERROR LOG TABLE		183	I,INIT(15)	
Q001.21	MAXIMUM_PROGRAM_CHECKS_PER_PROCESS	TCVTIMAX	THE MAXIMUM NUMBER OF NON-SEVER PROGRAM CHECKS ALLOWED PER PROCESS (OR PROCESS CYCLE IF CYCLIC) BEFORE FORCE CLOSE OF PROCESS	ILD	182	I,INIT(3)	
Q001.22	INITIALIZATION_FLAGS	TCVTIFLG	HALFWORD FLAG FIELD	300 305 368	305 361 362	I,INIT(0)	
Q001.23	IPL_IN_PROGRESS_INDICATOR		TCVTIFLG BIT 0: 1=IPL IN PROGRESS, 0=IPL NOT IN PROGRESS	300		BT	
Q001.24	NORMAL_INITIALIZATION_INDICATOR		TCVTIFLG BIT 1: 1=NORMAL INITIALIZATION IN PROGRESS, 0=NORMAL INITIALIZATION NOT IN PROGRESS	305		BT	
Q001.25			UNUSED			BT	
Q001.26			UNUSED			BT	
Q001.27			UNUSED			BT	
Q001.28	FAIL_SYNC_FLAG		TCVTIFLG BIT 5: 1=SELF'S COMMON SET HAS BEEN	300 305	305 361	BT	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
			REDUCED TO 0 AS A RESULT OF FAILING MORE THAN 1 GPC 0=SELF HAS NOT FAILED AS ABOVE	368	362		
Q001.29			UNUSED			BS(11)	
Q001.39	OVERLAY_BUFFER_RESERVATION	TCVTODBR	OVERLAY PHASE TABLE BUFFER RESERVATION FLAG: 0=BUFFER NOT IN USE 1=BUFFER BEING USED BY PROGRAM OVERLAY 2=BUFFER BEING USED BY MM I/O CHECKSUM	280 305 320	280 320	I,INIT(0)	
Q001.40	IPR_IOQE_ADDRESS	TCVTIOIC	ADDRESS OF THE IOQE	220 305	220 225	Y,INIT(0)	
Q001.41	SET_CAM_MASK	TCVTCAMS	MASK USED TO SET CAM LIGHTS	340 375	376	F,INIT(0)	
Q001.42	RESET_CAM_MASK	TCVTCAMR	MASK USED TO RESET CAM LIGHTS	340 375	376	F,INIT(0)	
Q001.43	DUTY_CYCLE_LIMIT	TCVTDCLM	DUTY CYCLE THRESHOLD USED IN DETERMINING WHETHER CURRENT DUTY CYCLE IS HIGH		142	SC,INIT (95.0) ILD	
Q001.44	BUS RECONFIGURATION_BUSY/IDLE_INDICATORS	TCVTSBIS	SPECIAL BUSY/IDLE INDICATOR MASK USED BY BUS RECONFIGURATION WHEN THERE IS A NEED TO WAIT FOR I/O TO QUIESCE ON BUSES NEEDED FOR RECONFIGURATION	220 280 305 350	200 280	F,INIT(0)	
Q001.45	TOP_BRQE_ADDRESS	TCVTBRQ	ADDRESS OF THE FIRST QUEUE ELEMENT ON THE BUS RECONFIGURATION QUEUE	305 350	350	Y,INIT(0)	
Q001.46	BRQE_FREE_POOL_ADDRESS	TCVTBRQF	ADDRESS OF THE POOL OF UNUSED BRQE'S	305 350	350	Y,INIT(ADDR.OF FIRST UNUSED BRQE)	
Q001.47	REDUNDANT_SET_SYNC_MASK	TCVTRSSM	REDUNDANT SET MASK IN FORMAT NEEDED BY SYNC ROUTINES.(SYNC DISCRETE FORMAT).	305 390	142 280 362 363 364 368	F,INIT(0)	
Q001.48	COMMON_SET_SYNC_MASK	TCVTCSSM	COMMON SET MASK IN FORMAT NEEDED BY SYNC ROUTINES.(SYNC DISCRETES	305 390	361 362 368	F,INIT(0)	
Q001.49	SVC_SYNC_IN_	TCVTSVCS	SVC SYNC IN PROGRESS INDICATOR:	220	142	I,INIT(0)	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
	PROGRESS INDICTATOR		0- SVC SYNC NOT IN PROGRESS NON-0-SVC SYNC IS IN PROGRESS	305 363	220		
Q001.50	I/O_SYNC_IN_PROGRESS_INDICATOR	TCVTIOS	I/O SYNC IN PROGRESS INDICATOR: 0=I/O SYNC NOT IN PROGRESS NON-0=I/O SYNC IN PROGRESS	305 362	142 280	I,INIT(0)	
Q001.51	LAST_I/O_SYNC_CODE_ISSUED	TCVTIOSC	LAST I/O SYNC CODE ISSUED	362	362	I,INIT(0)	
Q001.52	CAM_COUNTER_TABLE_ADDRESS	TCVTCNTA	ADDRESS OF GPC SELF'S CAM CENTER TABLE	300	375	Y,INIT(0)	
Q001.53	I/O_ERROR_LOG_TABLE_ADDRESS	TCVTIERA	ADDRESS OF SELF'S ENTRY FOR LATEST I/O ERROR INFORMATION IN THE I/O ERROR LOG TABLE	300	246	Y,INIT(0)	
Q001.54	GPC_ERROR_LOG_TABLE_ADDRESS	TCVTGERA	ADDRESS OF SELF'S ENTRY FOR LATEST GPC ERROR INFORMATION IN THE GPC ERROR LOG TABLE	300		Y,INIT(0)	
Q001.55	CAM_COUNTER_ICC_INDICATOR_MASK	TCVTCNTM	MASK NEEDED TO SET CORRECT ICC INDICATOR TO CAUSE SELF'S CAM COUNTER TABLE TO BE ICC'D	300	375	I,INIT(0)	
Q001.56	I/O_ERROR_LOG_ICC_INDICATOR_MASK	TCVTIERM	MASK NEEDED TO SET CORRECT ICC INDICATOR TO CAUSE SELF'S LAST I/O ERROR LOG ENTRY TO BE ICC'D	300	246	I,INIT(0)	
Q001.57	GPC_ERROR_LOG_ICC_INDICATOR_MASK	TCVTGERM	MASK NEEDED TO SET CORRECT ICC INDICATOR TO CAUSE SELF'S LAST ERROR LOG ENTRY TO BE ICC'D	300	183	I,INIT(0)	
Q001.58	GPC_SELF_STATUS_TABLE_ADDRESS	TCVTGST	ADDRESS OF GPC SELF'S GPC STATUS TABLE	300	142 205 220 244 310 320 351 355 360 361 368 375 390 392	Y,INIT(ADDR.OF STATUS TABLE OF GPC1)	
Q001.59	IPR_SYNC_CODE	TCVTIPRC	IPR SYNC CODE			X,INIT(000000F0)	
Q001.60	I/O_SYNC_CODE	TCVTIOCC	I/C SYNC CODE		362	X,INIT(000000FF)	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
Q001.61	SSIP_SYNC_CODE	TCVTSIPC	SSIP SYNC CODE		225 360 361	X,INIT(00000F00)	
Q001.62	TIMER_SYNC_CODE	TCVTTIMC	TIMER SYNC CODE		364	X,INIT(00000F0F)	
Q001.63	SVC_SYNC_CODE	TCVTSVCC	SVC SYNC CODE		363	X,INIT(00000FF0)	
Q001.64	NULL_SYNC_CODE	TCVTNULC	NULL SYNC CODE			X,INIT(00000FFF)	
Q001.65	LEADING_EDGE_DETECTION_MASK	TCVTKM	LEADING EDGE DETECTION MASK		360 361 362 363 364	X,INIT(00000F00)	
Q001.66	NEW_GPC_DETECTION_MASK	TCVTLM	NEW GPC DETECTION MASK		360 361	X,INIT,(0000088 9)	
Q001.67	IPR_CODE_INVERSE_SENDING_PATTERN	TCVTIPRT	INVERSE SENDING PATTERN OF THE IPR SYNC CODE		362	X,INIT(00000808)	
Q001.68	I/O_CODE_INVERSE_SENDING_PATTERN	TCVTVIOCI	INVERSE SENDING PATTERN OF THE I/O SYNC CODE		362	X,INIT(00000800)	
Q001.69	SSIP_CODE_INVERSE_SENDING_PATTERN	TCVTSIPI	INVERSE SENDING PATTERN OF THE SSIP SYNC CODE		361	X,INIT(00000088)	
Q001.70	TIMER_CODE_INVERSE_SENDING_PATTERN	TCVTTIMI	INVERSE SENDING PATTERN OF THE TIMER SYNC CODE		142 280	X,INIT(00000080)	
Q001.71	SVC_CODE_INVERSE_SENDING_PATTERN	TCVTSVCI	INVERSE SENDING PATTERN OF THE SVC SYNC CODE		363	X,INIT(00000008)	
Q001.72	COMMON_SET_NON_MEMBER_MASK	TCVTCSMI	INVERSE OF THE COMMON SET SYNC MASK	390	361	X,INIT(00000FFF)	
Q001.73	NULL_SYNC_CODE_SENDING_PATTERN	TCVTNULS	SENDING PATTERN FOR THE NULL SYNC CODE		142 220 244 361 362 363 364	X,INIT(00000888)	
Q001.74	IPR_STATUS	TCVTIPRS	IPR STATUS	280 362	220 280	I,INIT(0)	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
Q001.75	REDUNDANT_SET_GPC_COUNT	TCVTCGRS	COUNT OF GPC'S IN SELF'S REDUNDANT SET	390 305	220 368	I,INIT(1)	
Q001.76	COMMON_SET_GPC_COUNT	TCVTCGCS	COUNT OF GPC'S IN SELF'S COMMON SET	390 305	220 368	I,INIT(1)	
Q001.77	COMMON_SET_MAJORITY	TCVTMGCS	MAJORITY OF GPC'S IN SELF'S COMMON SET	390 305	380	I,INIT(1)	
Q001.78	I/O_COMPLETE_IOQE_ADDRESS	TCVTIOQ	ADDRESS OF I/O COMPLETE IOQE AND FLAG FIELD	220 250 305	210 220	AC,INIT(0)	
Q001.79	ERROR_IN_I/O_COMPLETE_IOQE		TCVTIOQ BIT 0: 1=BCE,NO/GO OR MSC TIMEOUT ERROR OCCURRED 0=THERE WAS NO BCE NO/GO OR MSC TIMEOUT		220	BT	
Q001.80			UNUSED			BS(12)	
Q001.81	COMPLETED_IOQE_ADDRESS		FULLWORD ADDRESS OF THE COMPLETED IOQE. (TCVTIOQ BITS 14-31)	220	210 220	BS(12),BS(18)	
Q001.82	MASS_MEMORY_PHASE_TABLE_ADDRESS	TCVTMMPT	MASS MEMORY ADDRESS OF THE MASS MEMORY PHASE TABLE		320	X,INIT(2C00)	
Q001.83	GPC_SELF_ID	TCVTCID	GPC IDENTIFICATION (INTEGER BILLED =1,2,3,OR 4)		244 300	I,INIT(0)	
Q001.84	YEAR_OF_LIFT_OFF	TCVTLOY	YEAR OF LIFT OFF		144	I,INIT(76)	
Q001.85	DAY_OF_YEAR_OF_LIFT_OFF	TCVTLOD	DAY OF YEAR OF LIFT OFF		144	I,INIT(0)	
Q001.86	GPC_SOFTWARE_CLOCK	TCVTSWCH,TCVTSWCM	GPC SOFTWARE CLOCK	141 142 149 305	149 162 163		
Q001.87	SOFTWARE_CLOCK_30_MINUTE_PORTION	TCVTSWCH	THIRTY MINUTE PORTION OF THE GPC SOFTWARE CLOCK	141 142 149 305	162 163	F,INIT(0) UM(MICROSECONDS)	
Q001.88	SOFTWARE_CLOCK_HALF_HOUR_COUNTER	TCVTSWCM	HALF HOUR COUNTER OF GPC SOFTWARE	141 142 149 305	149 162 242	I,INIT(0)	
Q001.89	MET_REFERENCE_TIME	TCVTMRTH,TCVTMRTM	REFERENCE TIME FOR MISSION ELAPSED TIME	142 149	144		

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
				205 305			
Q001.90	MET_REFERENCE_30_MINUTE_PORTION	TCVTMRTH	THIRTY MINUTE PORTION OF THE MET REFERENCE TIME	142 149 205 305	144 205	F,INIT(0) UM(MICROSECONDS)	
Q001.91	MET_REFERENCE_HALF_HOUR_COUNTER	TCVTMRTM	HALF HOUR COUNTER PORTION OF MET REFERENCE TIME	142 149 205 305	142 144 149 205	I,INIT(0)	
Q001.92	TIME_REMAINING_FOR_TOP_IOQE	TCVTMTTG	TIME REMAINING FOR TOP ACTIVE IOQE IN 33 MICROSECOND UNITS	210	251	I,INIT(0)	
Q001.93	MSC_I/O_MONITOR_FLAG	TCVTMNTR	MSC I/O MONITOR FLAG: ZERO= MONITOR TOP IOQE, NON-ZERO-DO NOT MONITOR I/O	210	251	F,INIT(0)	
Q001.94	SYNC_FAIL_RESET_CAM_MASK	TCVTRESC	MASK USED TO RESET CAM DISCRETES BY SYNC_FAIL_PROCESSOR,(368)	368	369	F,INIT(0)	
Q001.95	SYNC_FAIL_SET_CAM_MASK	TCVTSETC	MASK USED TO SET CAM DISCRETES BY SYNC_FAIL_PROCESSOR(368)	368	369	F,INIT(0)	
Q001.96	SYSTEM_RESET_EMULATION_FLAG_WORD	TCVTSREF	SYSTEM RESET EMULATION FLAG WORD: ZERO=STANDARD NORMAL INITIALIZATION, NON-ZERO=SYSTEM RESET EMULATION REQUEST	142 305	142 305	I,INIT(0)	
Q001.97	MSC_STATUS_INDICATOR	TCVTMSC	MSC STATU INDICATOR: 0=MSC IS IDLE LT 0=MSC IS BUSY/INTERRUPTABLE GT 0=MSC IS BUSY/AND NOT INTERRUPTABLE	210 251 310	210	F,INIT(0)	
Q001.98	CPU_TO_MSC_STATUS	TCVTCPU	STATUS INDICATOR FROM THE CPU TO THE MSC: GE 0=THE CPU HAS A REQUEST FOR MSC LT 0=CPU HAS NO REQUEST FOR THE MSC	210 310	251	F,INIT(0)	
Q002	PROCESS_DIRECTORY_ENTRY	TFPDE	THE PDE IS A HAL/S-FC GENERATED TABLE WHICH CONTAINS INFORMATION		174		

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
			CONCERNING THE PROCESS NEEDED BY FCOS IN THE CONTROL OF APPLICATION PROCESSES				
Q002.1	PROCESS_EVENT	TPDEVENT	THE PROCESS EVENT VARIABLE ASSOCIATED WITH THIS PROCESS BIT 31: 0=PROCESS IS NOT CURRENTLY SCHEDULED 1=PROCESS IS CURRENTLY SCHEDULED	101 107 108 133 149 174	101 107 108 109 110 142	E,INIT(0)	
Q002.2	ASSIGNED_PCT_ADDRESS	TPDEPCT	ADDRESS OF THE PCT WHICH CONTAINS CONTROL INFORMATION CONCERNING THIS PROCESS IF IT IS SCHEDULED OR ZERO IF THE PROCESS IS NOT SCHEDULED	101 133	101 108 109 110	Y,INIT(0)	
Q002.3	PROCESS_ENTRY_POINT	TPDEP	FULLWORD INDIRECT ADDESS CONSTANT (ZCON) OF PROCESS	101	101 107	Z,INIT (ZCON ADDRESS OF PROCESS WITH DATA SECTOR POINTER REFERRING TO DATA CSECT OF PROCESS.)	
Q002.4	STACK_INFORMATION	TPDESTAK	INFORMATION CONCERNING THE TEMPORARY STORAGE AREA (STACK) NEEDED BY THIS PROCESS ACCORDING TO THE STACK SIZE INDICATOR OF PDE_FLAGS.		101 107	I,INIT (SEE S002.5)	
Q002.5	PDE_FLAGS	TPDEFLGS	PDE FLAG FIELD			BS(16)	
Q002.6	STACK_SIZE/ADDRESS_INDICATOR		TPDEFLGS BIT 0: 1=TPDESTAK CONTAINS ADDRESS OF PRE-ALLOCATED STACK 0=TPDESTAK CONTAINS SIZE OF STACK NEEDED BY PROCESS (FCOS ASSIGNS STACK)		101	BT	
Q002.7			UNUSED			BS(11)	
Q002.8	PROCESS_MAJOR_FUNCTION_ID		TPDEFLGS BITS 12-15 CONTAINING THE MAJOR FUNCTION ID OF THIS PROCESS		110		

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
Q003	PROCESS_CONTROL_TABLE	TFPCT	THE PCT CONTAINS CONTROL AND STATUS INFORMATION CONCERNING A SCHEDULED PROCESS.A PCT EXISTS FOR EACH SCHEDULED PROCESS AND REMAINS ON THE PCT RUN QUEUE UNTIL THE PROCESS COMPLETES, NORMALLY OR ABNORMALLY.(RESIDES IN FCMCBLKS)	300		ST(30)	
Q003.1	NEXT_PCT_ADDRESS	TPCTNXT	ADDRESS OF THE NEXT PCT IN THE PCT QUEUE	101 131 132	102 103 105 110 131 132 320	Y	
Q003.2	PCT_PRIORITY	TPCTPRI	PRIORITY LEVEL OF THE PROCESS	101 105 106	102 105 131 142 320	I	
Q003.3	FCOS_ASSIGNED_STACK_ADDRESS	TPCTSTOR	ADDRESS OF THE TEMPORARY STORAGE AREA FOR THIS PROCESS IF ASSIGNED BY FCOS	101	107 133	Y	
Q003.4	PDE_ADDRESS	TPCTPDE	ADDRESS OF THE PROCESS DIRECTORY ENTRY OF THIS PROCESS	101	107 108 110 133 142 174	Y	
Q003.5	INTERRUPT_PSW_OF_PROCESS	TPCTPSW	THE INITIAL PSW OR THE PSW AT THE TIME OF THE LAST INTERRUPTION OF THIS PROCESS	100 101 105 107 182 184 220 320	103 180 181 183 220	A(21,F	
Q003.6	PROCESS_INTERRUPT_REGISTER_0	TPCTGPR0	CONTENTS OF GENERAL PURPOSE REGISTER 0 WHEN THE PROCESS IS INTERRUPTED FOR ANOTHER PROCESS TO EXECUTE	101 103 107 182	103 108 182	F	
Q003.7	PROCESS_INTERRUPT_REGISTER_1	TPCTGPR1	SAME AS Q003.6 FOR GENERAL PURPOSE REGISTER 1.	103 182 320	103 320	F	
Q003.8	PROCESS_INTERRUPT_	TPCTGPR2	SAME AS Q003.6 FOR GENERAL PURPOSE	103	103	F	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
	REGISTER_2		REGISTER 2	182 320	320		
Q003.9	PROCESS_INTERRUPT_ REGISTER_3	TPCTGPR3	SAME AS Q003.6 FOR GENERAL PURPOSE REGISTER 3	103 182 320	103 320	F	
Q003.10	PROCESS_INTERRUPT_ REGISTER_4	TPCTGPR4	SAME AS Q003.6 FOR GENERAL PURPOSE REGISTER 4	103 320	103 320	F	
Q003.11	PROCESS_INTERRUPT_ REGISTER_5	TPCTGPR5	SAME AS Q003.6 FOR GENERAL PURPOSE REGISTER 5	103 320	103 320	F	
Q003.12	PROCESS_INTERRUPT_ REGISTER_6	TPCTGPR6	SAME AS Q003.6 FOR GENERAL PURPOSE REGISTER 6	103	103	F	
Q003.13	PROCESS_INTERRUPT_ REGISTER_7	TPCTGPR7	SAME AS Q003.6 FOR GENERAL PURPOSE REGISTER	103	103	F	
Q003.14	INTERRUPT_FLOATING_ POINT_REGISTER_0	TPCTFPR0	SAME AS Q003.6 FOR FLOATING POINT REGISTER 0	103	103	SC	
Q003.15	INTERRUPT_FLOATING_ POINT_REGISTER_1	TPCTFPR1	SAME AS Q003.6 FOR FLOATING POINT REGISTER 1	103	103	SC	
Q003.16	INTERRUPT_FLOATING_ POINT_REGISTER_2	TPCTFPR2	103F AS Q003.6 FOR FLOATING POINT REGISTER 2	103		SC	
Q003.17	INTERRUPT_FLOATING_ POINT_REGISTER_3	TPCTFPR3	SAME AS Q003.6 FOR FLOATING POINT REGISTER 3	103	103	SC	
Q003.18	INTERRUPT_FLOATING_ POINT_REGISTER_4	TPCTFPR4	SAME AS Q003.6 FOR FLOATING POINT REGISTER 4	103	103	SC	
Q003.19	INTERRUPT_FLOATING_ POINT_REGISTER_5	TPCTFPR5	SAME AS Q003.6 FOR FLOATING POINT REGISTER 5	103	103	SC	
Q003.20	INTERRUPT_FLOATING_ POINT_REGISTER_6	TPCTFPR6	SAME AS Q003.6 FOR FLOATING POINT REGISTER 6	103	103	SC	
Q003.21	INTERRUPT_FLOATING_ POINT_REGISTER_7	TPCTFPR7	SAME AS Q003.6 FOR FLOATING POINT REGISTER 7	103	103	SC	
Q003.22	REPEAT_DELTA_TIME		REPEAT DELTA TIME IF THE PROCESS IS CYCLIC		140		
Q003.23	DELTA_TIME_30_ MINUTE_PORTION	TPCTRPTH	THE 30_MINUTE PORTION OF THE REPEAT DELTA TIME	101	140	F,UM(MICROSECON DS)	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
Q003.24	DELTA_TIME_HALF_HOUR_COUNTER	TPCTRPTM	THE COUNT OF MULTIPLES OF 30 MINUTES IN THE REPEAT DELTA TIME	101	140	I	
Q003.25	OUTSTANDING_I/O_COUNT	TPCTIORQ	COUNT OF OUTSTANDING I/O REQUESTS	101 200 201	133	I	
Q003.26	ORIGINAL_PRIORITY	TPCTOPRI	THE ORIGINAL PRIORITY THE PROCESS IF IT HAS BEEN PRIORITY PROMOTED	101 105 106 320	106 131	I	
Q003.27	LAST_ERROR_GROUP_CODE	TPCTERR	THE ERROR GROUP CODE OF THE LAST ERROR WHICH OCCURRED FOR THIS PROCESS	101 107 183	185	I	
Q003.28	PROGRAM_INTERRUPT_COUNT	TPCTECNT	THE COUNT OF PROGRAM INTERRUPTS WHICH HAVE OCCURRED IN THIS PROCESS	101 107	182	I	
Q003.29	PCT_FLAGS	TPCTFLGS	PCT FLAG BITS. SEE FOLLOWING ITEMS FOR DEFINITION	101 142 320	107 133 140 225	BS(16)	
Q003.30	CANCELLED_PCT_INDICATOR		TPCTFLGS BIT 0 1=CANCELLED 0=NOT CANCELLED	101 109 142	101 107 110 174 225	BT	
Q003.31	FCOS_PCT_INDICATOR		TPCTFLGS BIT 1: 1=FCOS PROCESS 0=NOT AN FCOS PROCESS	320		BT	
Q003.32	OUTSTANDING_EQE_INDICATORS		TPCTFLGS BITS 2,3: 00=NO OUTSTANDING EQE'S 01=INITIATION EQE OUTSTANDING 10=CANCELLATION EQE OUTSTANDING	133 173 174 175	133 140 173 174 175	BS(2)	
Q003.33	TERMINATED_PCT_INDICATOR		TPCTFLGS BIT 4: 0=PROCESS NOT TERMINATED 1=PROCESS TERMINATED	108 142	225	BT	
Q003.34	CYCLE_OVERRUN_INDICATOR		TPCTFLGS BIT 5: 0=CYCLE OVERRUN HAS NOT OCCURRED 1=CYCLE OVERRUN OCCURRED DURING CYCLE	142		BT	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
Q003.35	CANCELLATION_ CRITERIA		TPCTFLGS BITS 6,7: 00=NONE 01=UNTIL TIME 10=WHILE EVENT 11=UNTIL EVENT	101	101	BS(2)	
Q003.36	REPEAT_OPTIONS		TPCTFLGS BITS 8,9: 00=NONE 01=REPEAT 10=REPEAT EVERY 11=REPEAT AFTER	101	101 107 133 174	BS(2)	
Q003.37			TPCTFLGS BIT 10 IS UNSUPPORTED (INDICATES DEPENDENCY)			BT	
Q003.38	SSIP_PCT_INDICATOR		TPCTFLGS BIT 11: 0=NOT SSIP 1=SSIP PROCESS	101	101 142	BT	
Q003.39	INITIAL_CONDITIONS		TPCTFLGS BITS 12,13: 00=NONE 01=AT 10=IN 11=ON	101 174	101 174	BS(2)	
Q003.40			UNUSED			BT	
Q003.41			TPCTFLGS BIT 15 IS UNSUPPORTED (TASK INDICATOR)			BT	
Q003.42	PCT_WAIT_ INDICATORS	TPCTWAIT	PCT WAIT OPTIONS FLAGS, SEE FOLLOWING ITEMS FOR DEFINITION	101 104 107 142 200 201 320	101 103 105 107 174	BS(16)	
Q003.43			TPCTWAIT BITS 0-10 NOT USED		103 105	BS(11)	
Q003.44	ABSOLUTE_TIME_WAIT_ INDICATOR		TPCTWAIT BIT 11: 1=WAITING FOR A SPECIFIC ABSOLUTE TIME 0=NOT WAITING FOR ABSOLUTE TIME	104	103 105	BT	
Q003.45	I/O_WAIT_INDICATOR		TPCTWAIT BIT 12: 1=WAITING FOR I/O 0=NOT WAITING FOR I/O	200 201	103 105	BT	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
Q003.46	EVENT_WAIT_INDICATOR		TPCTWAIT BIT 13: 1=WAITING FOR EVENT 0=NOT WAITING FOR EVENT	104 105	103 105	BT	
Q003.47	DELTA_TIME_WAIT_INDICATOR		TPCTWAIT BIT 14: 1=WAITING FOR DELTA TIME 0=NOT WAITING FOR DELTA TIME	104	103 105	BT	
Q003.48	SCHEDULED_WAIT_INDICATOR		TPCTWAIT BIT 15: 1=WAITING FOR FIRST OR NEXT CYCLE 0=NOT WAITING FOR INITIAL CONDITIONS OR NEXT CYCLE	101 107 142	103 105 109 110 174	BT	
Q003.49	TIME_INITIATED_I/O_PARAMETER_LIST_ADDRESS	TPCTIOPP	ADDRESS OF THE TIME INITIATED I/O PARAMETER LIST ASSOCIATED WITH THIS PROCESS	101	140	Y	
Q004	LOCK_GROUP_CONTROL_TABLE	FPMLOKCW	THE LGT IS THE TABLE CONTAINING CONTROL WORDS FOR EACH OF THE DATA LOCK GROUPS. THESE CONTROL WORDS AND THE MASK OF USED GROUPS INDICATE WHICH PROCESS IS IN CONTROL OF EACH LOCK GROUP. (RESIDES IN FCMCBLKS)				
Q004.1	MASK_OF_USED_LOCK_GROUPS		MASK INDICATING WHICH LOCK GROUPS ARE CURRENTLY IN USE (BIT 1 CORRESPONDS TO LOCK GROUP 15,... BIT 15 CORRESPONDS TO LOCK GROUP 1): 0=LOCK GROUP NOT IN USE 1=LOCK GROUP IN USE	105 106	105 106	BS(16)	
Q004.2	LOCK_GROUP_CONTROL_WORDS		FIFTEEN ADDRESSES,ONE FOR EACH LOCK GROUP STARTING WITH GROUP 1, INDICATING THE ADDRESS OF THE PCT FOR THE PROCESS USING THE LOCK GROUP	105	105 106	A(15,Y	
Q005	TIMER_QUEUE_ELEMENT	TFTQE	FCOS CONTROL BLOCK CONTAINING: POINTER TO THE NEXT TQE, ADDRESS OF THE PARENT PCT, TIME OF EXPIRATION AND FLAGS DESCRIBING THE TQE TYPE				
Q005.1	NEXT_TQE_ADDRESS	TTQENXT	POINTER TO NEXT TQE IN CHAIN	142 143 166	142 143 166	Y	
Q005.2	TQE_PARENT_PCT_ADDRESS	TTQEPCT	ADDRESS OF THE PARENT PCT	101 104	140 142	Y	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
				107 140 166	143 148 174		
Q005.3	TQE_TIME_OF_EXPTRATION	TTQETOXH,TTQETOXM	TIME OF EXPIRATION FOR THIS TQE	140 142 148 167	140 142 148 163 166 167	F,H	
Q005.4	TIME_OF_EXPIRATION_HALF_HOUR	TTQETOXH	HALF HOUR PORTION IN MICROSECONDS	140 142 148 167	140 142 148 163 166 167	F	
Q005.5	TIME_OF_EXPIRATION_HALF_HOUR_MULTIPLES	TTQETOXM	HALF HOUR MULTIPLES OF TIME OF EXPIRATION	140 142 148 167	140 142 148 163 166 167	H	
Q005.6	TQE_FLAGS	TTQEFLGS	TQE FLAGS	104 107 140 142	140 143 148 174	BS(16)	
Q005.7	TQE_STATUS		TTQEFLGS BIT 0: 1=CANCELLED TQE 0=ACTIVE TQE	101 143	142	BT	
Q005.8	NULL_TQE_INDICATOR		TTQEFLGS BIT 1: 1=NULL TQE 0=NOT NULL TQE	101 142	140	BT	
Q005.9	TQE_REQUEST_TYPE		TTQEFLGS BIT 2: 1=INITIAL TQE 0=NOT INITIAL TQE	101 104 107 140	140	BT	
Q005.10	TQE_TYPE_INDICATOR		TTQEFLGS BITS 3-15: 0001=SSIP TQE 0002=WAIT TQE 0003=WAIT UNTIL TQE 0004=SCHEDULE AT TQE 0005=SCHEDULE IN TQE 0006=SCHEDULE UNTIL TQE 0007=REPEAT AFTER TQE 0008=REPEAT EVERY TQE	101 104 107	140 142 148 174	BS(13)	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
			0009=I/O TQE 000A=MTU RUNTIME UPDATE TQE 100A=GPC PRIME UPDATE TQE 000B=MET UPDATE TQE 000C=INIT.GMT UPDATE TQE				
Q006	EVENT_QUEUE_ELEMENT	TEQE	A CONTROL BLOCK WHICH CONTAINS INFORMATION NECESSARY TO ACTIVATE EVENT DEPENDENT REQUESTS (RESIDES IN FCMCBLKS)			ST(30)	
Q006.1	NEXT_EQE_ADDRESS	TEQENXT	ADDRESS OF NEXT EQE IN CHAIN	173 174 175	173 174	Y	
Q006.2	EQE_PARENT_PCT_ADDRESS	TEQEPCT	ADDRESS OF THE PCT	101 104 175	173 174 175	Y	
Q006.3	OPERATOR_FIELD	TEQEOPS	A COUNT OF OPERATORS AND THE OPERATORS FOR EVENT EXPRESSION STACK EVALUATION	101 104	173 174	F	
Q006.4	COUNT_OF_OPERATORS		TEQEOPS BITS 0-3 COUNT OF OPERATORS	101 104	173	BS(4)	
Q006.5	EVENT_EXPRESSION_OPERATORS		TEQEOPS BITS 4-31: 28 BITS OF OPERATORS EACH TWO BITS IN LENGTH DEFINED AS FOLLOWS: 01_LOGICAL OR 10_LOGICAL NOT 11_LOGICAL AND 00_PUSH OPERATOR	101 104		BS(28)	
Q006.6	EVENT_VARIABLE_1	TEQEVAR1	ADDRESS OF EVENT VARIABLE 1	101 104 173 175	173 174 175	Y	
Q006.7	EVENT_VARIABLE_2	TEQEVAR2	ADDRESS OF EVENT VARIABLE 2	101 104 173	174 175	Y	
Q006.8	EVENT_VARIABLE_3	TEQEVAR3	ADDRESS OF EVENT VARIABLE 3	101 104	173 174 175	Y	
Q006.9	EVENT_VARIABLE_4	TEQEVAR4	ADDRESS OF EVENT VARIABLE 4	101 104	173 174 175	Y	
Q006.10	EVENT_VARIABLE_5	TEQEVAR5	ADDRESS OF EVENT VARIABLE 5	101	173	Y	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
				104	174 175		
Q006.11	EQE_TYPE	TEQETYPE	A FLAGWORD-SEE FOLLOWING FOR THE DEFINITION	101 104 173 175	173 174 175	I	
Q006.12	OPTION_EVENT_TYPE		TEQETYPE BITS 0-3: 0000=WAIT 1000=SCHEDULE ON 0100=SCHEDULE WHILE 0010=SCHEDULE UNTIL 0001=NULL EQE	101 104	173 174	BS(4)	
Q006.13	EXPRESSION_TYPE		TEQETYPE BITS 4-7: 1000=SINGLE EVENT TRUE STATE 0100=SINGLE EVENT FALSE STATE 0010=OR OPERATORS	104 173	173 174	BS(4)	
Q006.14			TEQETYPE BITS 8-12 UNUSED	104		BS(5)	
Q006.15	EQE_SATISFIED_FLAG		TEQETYPE BIT 13: 1=EQE SATISFIED 0=EQE NOT SATISFIED	104		BT	
Q006.16	ACTIVE_EQE_FLAG		TEQETYPE BIT 14: 1=EQE ACTIVE 0=EQE NOT ACTIVE	104		BT	
Q007	I/O_QUEUE_ELEMENT	TFIOQ	THE IOQE IS THE CONTROL BLOCK THAT CONTAINS THE INFORMATION NECESSARY TO INITIATE AND COMPLETE AN I/O REQUEST (RESIDES IN FCMCBLKS)			ST (30)	
Q007.1	ADDRESS_OF_NEXT_IOQE_IN_CHAIN	TIOQNXT	ADDRESS OF THE NEXT IOQE IN THIS PARTICULAR CHAIN	200 201 225	205 225	Y	
Q007.2	ADDRESS_OF_REQUESTING_PROCESS_PCT	TIOQPCT	ADDRESS OF THE PCT FOR THE PROCESS WHICH ISSUED THIS I/O REQUEST	200 201	225	Y	
Q007.3	ADDRESS_OF_THIS_IOQE	TIOQSELF	THE ADDRESS OF THIS IOQE IN A FORMAT SUITABLE FOR IOP PROCESSING (18 BITS)			AC	
Q007.4	MASK_OF_BUSES_TO_MONITOR	TIOQMNTM	THE MASK OF BUSES TO BE MONITORED FOR I/O COMPLETION FOR THIS REQUEST	200 201 351	200 205 251	BS(32)	
Q007.5	ADDRESS_OF_I/O_BUFFER	TIOQBUFA	THE ADDRESS OF WHERE TO SEND OR RECEIVE THE FIRST DATA WORD FOR	200	149 205	Y	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	OST	ATTRIBUTES	MML
			THIS REQUEST				
Q007.6	IOQE_FLAGS_FIELD_1	TIOQFLG1	VARIOUS FLAG BITS USED BY THIS I/O REQUEST	200 201 225	201 225	BS(16)	
Q007.7	MM_HALT/RESET_ ISSUED	TIOQHIMM	TIOQFLG1 BIT 0: 1=A MASS MEMORY HALT OR RESET HAS BEEN ISSUED 0=MASS MEMORY HALT/RESET HAS NOT BEEN ISSUED	200 201		BT	
Q007.8	MM_HALT/RESET_ RECEIVED	TIOQHRMM	TIOQFLG1 BIT 1: 1=A MASS MEMORY HALT OR RESET HAS BEEN RECEIVED 0=A MASS MEMORY HALT/RESET HAS NOT BEEN RECEIVED	200		BT	
Q007.9			TIOQFLG1 BIT 2 UNUSED			BT	
Q007.10	PARENT_PCT_ TERMINATED	TIOQTRMM	TIOQFLGL BIT 3: 1=PCT ASSOCIATED WITH I/O REQUEST HAS BEEN TERMINATED 0=PCT ASSOCIATED WITH I/O REQUEST HAS NOT BEEN TERMINATED	200	225	BT	
Q007.11	MM_TAPE_POSITION	TIOQPTMM	TIOQFLG1 BIT 4: 1=A MASS MEMORY TAPE POSITION COMMAND HAS BEEN ISSUED 0=MASS MEMORY TAPE POSITION COMMAND NOT BEING ISSUED	200		BT	
Q007.12	LAST_MM_ TRANSACTION	TIOQLTMM	TIOQFLG1 BIT 5: 1=LAST MM OPERATION FOR THIS REQUEST 0=NOT A MM REQUEST OR NOT THE LAST MM OPERATION FOR REQUEST	200		BT	
Q007.13	I/O_COMPLETE	TIOQIOCM	TIOQFLG1 BIT 6: 1=I/O IS COMPLETE FOR THIS TRANSACTION 0=I/O NOT COMPLETE FOR TRANSACTION	200		BT	
Q007.14	PRE-INITIALIZED_ IOQE	TIOQPREI	TIOQFLG1 BIT 7: 1=PRE-INITIALIZED IOQE 0=NOT A PRE-INITIALIZED IOQE	200 201	225	BT	
Q007.15	ERROR_IN_ TRANSACTION	TIOQTRER	TIOQFLG1 BIT 8: 1=ERROR OCCURRED DURING TRANSACTION 0=NO ERROR HAS OCCURRED	200	205	BT	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
Q007.16	INVALID_PARENT_PCT	TIOQIPCT	TIOQFLG1 BIT 9: 1=PCT ASSOCIATED WITH THIS REQUEST IS INVALID 0=PARENT PCT IS VALID	200	225	BT	
Q007.17	ICC_RETRIED	TIOQIRTY	TIOQFLG1 BIT 10: 1=I/O HAS BEEN RETRIED FOR ICC FOR THIS REQUEST 0=ICC NOT RETRIED	200		BT	
Q007.18	I/O_ERROR_PROCESSING_DONE	TIOQERPR	TIOQFLG1 BIT 11: 1=I/O ERROR PROCESSING HAS ALREADY BEEN DONE FOR REQUEST 0=I/O EPROP PROCESSING NOT DONE	200		BT	
Q007.19			TIOQFLG1 BITS 12-15 UNUSED	200	200	BS(4)	
Q007.23	IOQE_FLAGS_FIELD_2	TIOQFLG2	VARIOUS FLAG BITS USED BY THIS I/O REQUEST	200	200 201	BS(16)	
Q007.24			TIOQFLG2 BITS 0-4 UNUSED			BS(5)	
Q007.29	I/O_REQUEST_SYNC_TYPE	TIOQSYNT	TIOQFLG2 BIT 5: 0=REDUNDANT SET 1=COMMON SET	200		BT	
Q007.30	COMFAULT_STATUS_NEEDED	TIOQCOMF	TIOQFLG2 BIT 6: 1=COMFAULT REQUESTED 0=COMFAULT NOT REQUESTED	200	205	BT	
Q007.31	ICC_REQUEST	TIOQICCR	TIOQFLG2 BIT 7: 1=ICC REQUEST 0=NOT ICC REQUEST	200		BT	
Q007.32	TIME_TAG_REQUESTED	TIOQTTAG	TIOQFLG2 BIT 8: 1=TIME TAGGING REQUESTED 0=NO TIME TAGGING REQUESTED	200		BT	
Q007.33	WAIT_FOR_I/O	TIOQWAIT	TIOQFLG2 BIT 9: 1=WAIT TYPE I/O 0=NO WAIT I/O	200	200 201	BT	
Q007.34	PROTECTED_TRANSACTION	TIOQPRTR	TIOQFL2 BIT 10: 1=TRANSACTION IS PROTECTED(REQUIRES ICC UPON COMPLETION) 0=TRANSACTION NOT PROTECTED	200		BT	
Q007.35	STATUS_REQUESTED	TIOQSTRQ	TICQFLG2 BIT 11: 1=STATUS REQUESTED REPORTING ERRORS	200		BT	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
			IF ANY 0=STATUS NOT REQUESTED				
Q007.36	OUTPUT_TYPE_REQUEST	TIOQOTYT	TIOQFLG2 BIT 12: 1=OUTPUT I/O TRANSACTION 0=NOT OUTPUT I/O TRANSACTION	200		BT	
Q007.37	MAJOR_FUNCTION_ID	TIOQMFID	TIOQFLG2 BITS 13-15 CONTAIN THE MAJOR FUNCTION ID OF THE MAJOR FUNCTION ASSOCIATED WITH THIS REQUEST	200		BS(3)	
Q007.40	IOQE_PRIORITY	TIOQPRI	THE PRIORITY OF THIS REQUEST	200 205	200 201 205	I	
Q007.41	ITERATIONS_COUNT_FOR_REQUEST	TIOQICNT	THE NUMBER OF ITERATIONS FOR WHICH THE I/O WILL BE MONITORED	205 210	205	I,UM=(16 MICROSECONDS)	
Q007.42	IOQE_COMPLETION_EVENT_ADDRESS	TIOQEVNT	THE ADDRESS OF THE EVENT VARIABLE TO BE SET UPON COMPLETION OF THIS I/O REQUEST	200	201	Y	
Q007.43	IOQE_DEVICE_ID	TIOQDVID	THE UNIQUE DEVICE OR TRANSACTION IDENTIFICATION USED FOR THIS REQUEST. SEE APPENDIX G FOR A COMPLETE LIST	200	205 251 351	I	
Q007.44	IOQE_OPERATION_CODE	TIOQOPCD	THE OPERATION TO BE PERFORMED,READ, WRITE, STATUS, ETC.	200	205	I	
Q007.45	IOQE_WORD_COUNT	TIOQWDCD	THE NUMBER OF WORDS TO BE RECEIVED OR TRANSMITTED BY THIS REQUEST	200	205	I	
Q007.46	IOQE_STARTING_ADDRESS	TIOQSTAD	A FIELD WHICH CONTAINS ONE OF THE FOLLOWING: 1)PMU RAM ADDRESS FOR THE DI/PL DATA RAM READ, 128 KBPS PROGRAM READ,128 KBPS FORMAT PROGRAM WRITE OR A 64 KBPS FORMAT WRITE 2)MASS MEMORY STARTING BLOCK ADDRESS 3)TEST CONTROL SUPERVISOR INPUT DATA 4)LDB INPUT DATA	200	205	Y,Y,X,X	
Q008	BUS RECONFIGURATION_QUEUE_ELEMENT	TFBRQE	CONTAINS INFORMATION FROM THE BUS RECONFIGURATION PARAMETER LIST NEEDED TO SERVICE THE REQUEST (RESIDES IN FCMCBLKS)			ST(5)	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
Q008.1	NEXT_BRQE_ADDRESS	TBRQENXT	POINTER TO THE NEXT BRQE IN THE QUEUE	350	350	Y	
Q008.2	BRQE_TYPE	TBRQETYP	A HALFWORD DEFINING THE TYPE OF RECONFIGURATION REQUEST (SEE SC36.1 FOR DEFINITION)	350 351	350 351	I	
Q008.3	REQUEST_VARIABLE_1	TBRQEV1	THE VARIABLE 1 FIELD FROM THE BUS RECONFIGURATION PARAMETER LIST (SEE SC36.3 FOR DEFINITION)	350 351	350 351	I	
Q008.4	REQUEST_VARIABLE_2	TBRQEV2	THE VARIABLE 2 FIELD FROM THE BUS RECONFIGURATION PARAMETER LIST (SEE SC36.4 FOR DEFINITION)	350 351	350 351	I	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
S001	SCHEDULE_PARAMETER_LIST		SCHEDULE SVC PARAMETER LIST		101		
S001.1	PROCESS_PRIORITY	TSCDPRIO	PRIORITY VALUE IN BITS 0-7 OF TSCDPRIO		101	I	
S001.2	SCHEDULE_SVC_NUMBER		SVC NUMBER (1) IN BITS 8-15 OF TSCDPRIO		100	BS(8),INIT (01)	
S001.3	PROCESS_SCHEDULE_OPTIONS	TSCDFLGS	PROCESS SCHEDULE OPTIONS IN BITS 6-15,0-5 UNUSED		101	BS(16)	
S001.4	SCHEDULE_CANCELLATION_CRITERIA		CANCEL OPTIONS-BITS 6-7: 00- NONE 01- UNTIL TIME 10- WHILE EVENT 11- UNTIL EVENT		101	BS(2)	
S001.5	SCHEDULE_REPEAT_OPTIONS		REPEAT OPTIONS - BITS 8-9: 00- NONE 01- REPEAT 10 REPEAT EVERY 11 REPEAT AFTER		101	BS(2)	
S001.6	UNSUPPORTED_DEPENDENT_INDICATOR		BIT 10 INDICATES DEPENDENT PROCESS NOT SUPPORTED		101	BT	
S001.7			BIT 11 NOT USED				
S001.8	SCHEDULE_INITIAL_CONDITIONS		INITIAL CONDITIONS- BITS 12-13: 00- NONE 01- AT 10- IN 11- ON		101	BS(2)	
S001.9			BIT 14 NOT USED			BT	
S001.10	UNSUPPORTED_TASK_INDICATOR		BIT 15 INDICATES A TASK-NOT SUPPORTED		101	BT	
S001.11	PROCESS_PDE_ADDRESS	TSCDPDE	THE ADDRESS OF THE PROCESS DIRECTORY ENTRY FOR THE PROCESS TO BE SCHEDULED.		101	V	
S001.12	ON_EVENT_EXPRESSION_ADDRESS	TSCDEXP1	THE ADDRESS OF AN EVENT EXPRESSION IF THE ON OPTION WAS SPECIFIED.		101	Y	
S001.13	CANCELLATION_EVENT_EXPRESSION_ADDRESS	TSCDEXP2	THE ADDRESS OF AN EVENT EXPRESSION IF THE WHILE OR UNTIL OPTIONS SPECIFIED AN EVENT EXPRESSION		101	Y	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
S002	TERMINATE_PARAMETER_LIST		TERMINATE SVC PARAMETER LIST				
S002.1	TERMINATE_PROCESS_COUNT		BITS 0-7 CONTAIN THE COUNT OF PROCESSES SPECIFIED IN THE PARAMETER LIST		108	BS(8)	
S002.2	TERMINATE_SVC_NUMBER		BITS 8-15 CONTAIN THE SVC NUMBER: 02- NO LABEL SPECIFIED 03- LABEL SPECIFIED		100	BS(8)	
S002.3	TERMINATE_PROCESS_LIST		A LIST OF ADDRESSES WHICH POINT TO THE DIRECTORY ENTRIES OF THE PROCESSES WHICH ARE TO BE TERMINATED.		108	A (N),Y	
S004	CANCEL_PARAMETER_LIST		CANCEL SVC PARAMETER LIST				
S004.1	CANCEL_PROCESS_COUNT		COUNT OF PROCESSES SPECIFIED IN THE PARAMETER LIST		109	BS(8)	
S004.2	CANCEL_SVC_NUMBER		SVC NUMBER: 04- NO LABEL SPECIFIED 05- LABEL SPECIFIED		100 109	BS(8)	
S004.3	CANCEL_PROCESS_LIST		A LIST OF ADDRESSES WHICH POINT TO THE PROCESS DIRECTORY ENTRIES OF THE PROCESSES WHICH ARE TO BE CANCELLED		109	A(N),Y	
S006	WAIT_PARAMETER_LIST		WAIT SVC PARAMETER LIST				
S006.1	WAIT_SVC_NUMBER		BITS 0-7 ARE NOT USED. BITS 8-15 CONTAIN THE SVC NUMBER: 06- A DELTA TIME VALUE WAS SPECIFIED 07- THE UNTIL OPTION WAS SPECIFIED 08- AN EVENT EXPRESSION WAS SPECIFIED		100 104	BS(16)	
S006.2	WAIT_EVENT_EXPRESSION_ADDRESS		ADDRESS OF AN EVENT EXPRESSION IF THE OPTION FOR WAS SPECIFIED.		104	Y	
S012	SIGNAL_EVENT_PARAMETER_LIST		SIGNAL SVC PARAMETER LIST		172		
S012.1	SIGNAL_SVC_NUMBER		BITS 0-7 ARE NOT USED. BITS 8-15 CONTAIN THE SVC NUMBER-12.		100	BS(16)INIT(12)	
S012.2	SIGNAL_EVENT_ADDRESS		ADDRESS OF THE EVENT VARIABLE WHICH IS TO BE SIGNALLED		172	Y	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	OST	ATTRIBUTES	MML
S013	SET_EVENT_PARAMETER_LIST		SET SVC PARAMETER LIST				
S013.1	SET_SVC_NUMBER		SET SVC NUMBER-13		100	BS(16),INIT(13)	
S013.2	SET_EVENT_ADDRESS		ADDRESS OF THE EVENT VARIABLE WHICH IS TO BE SET.		170	Y	
S014	RESET_EVENT_PARAMETER_LIST		RESET SVC PARAMETER LIST		171		
S014.1	RESET_SVC_NUMBER		BITS 0-7 ARE NOT USED. BITS 8-15 CONTAIN THE SVC NUMBER- 14		100	BS(16),INIT(14)	
S014.2	RESET_EVENT_ADDRESS		ADDRESS OF THE EVENT VARIABLE WHICH IS TO BE RESET		171	Y	
S015	UPDATE/EXCLUSIVE_RESERVE_RELEASE_PARAMETER_LIST		SVC PARAMETER LIST FOR REQUESTS TO USE/RELEASE LOCK GROUPS OR USE/RELEASE EXCLUSIVE PROCEDURES/FUNCTIONS				
S015.1	DATA_READ/WRITE_INDICATOR		A BIT INDICATING THE FOLLOWING: 1- DATA VARIABLES ARE TO BE READ ONLY 0- DATA VARIABLES ARE TO BE WRITTEN OR THIS IS A RESERVE/RELEASE SVC FOR AN EXCLUSIVE PROCEDURE OR FUNCTION		105 106	BT	
S015.2			NOT USED			BS(7)	
S015.3	RESERVE_SVC_NUMBER		THE SVC NUMBER FOR RESERVE AS FOLLOWS: 15- CODE BLOCK RESERVE 16- DATA BLOCK RESERVE		105	BS(8)	
S015.4			NOT USED			BS(8)	
S015.5	RELEASE_SVC_NUMBER		THE SVC FOR RELEASE AS FOLLOWS: 17- CODE BLOCK RELEASE 18- DATA BLOCK RELEASE		106	BS(8)	
S015.6	RESOURCE_IDENTIFICATION		AN IDENTIFIER INDICATING CODE BLOCK OR DATA AREAS TO BE RESERVED/RELEASED AS FOLLOWS: Y(ADDRESS)- ADDRESS OF THE WORD IN THE SECTOR O CSECT GENERATED		105 106	Y OR BS(16)	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
			FOR THE REQUESTED PROCEDURE/FUNCTION OR A BIT PATTERN INDICATING WHICH LOCK GROUPS ARE TO BE RESERVED OR RELEASED WHERE BIT 15 CORRESPONDS TO LOCK GROUP ONE AND BIT 1 CORRESPONDS TO LOCK GROUP 15.				
S019	OVERLAY_PARAMETER_LIST		OVERLAY SVC PARAMETER LIST				
S019.1	OVERLAY_SOURCE	TOPLSORC	INDICATION OF OVERLAY SOURCE (BITS 0-7): NE 0- ANOTHER GPC 0- MASS MEMORY 320			I	
S019.2	OVERLAY_SVC_NUMBER		BITS 8-15 CONTAIN THE SVC NUMBER -19		100	BS(8),INIT(19)	
S019.3	FUNCTION_BASE_PHASE_NUMBER	TOPLFNBS	THE PHASE NUMBER ASSOCIATED WITH THE FUNCTION BASE TO BE READ IN OR ZERO IF NO FUNCTION BASE IS REQUESTED.		320	I	
S019.4	PROGRAM_OVERLAY_PHASE_NUMBER	TOPLPOVL	THE PHASE NUMBER ASSOCIATED WITH THE OPS OVERLAY		320	I	
S019.5	OVERLAY_COMPLETION_EVENT_ADDRESS	TOPLEVNT	THE ADDRESS OF THE EVENT VARIABLE TO BE SET WHEN THE OVERLAY REQUEST IS COMPLETELY SERVICED		320	Y	
S019.6	OVERLAY_I/O_STATUS_ADDRESS	TOPLSTAT	THE ADDRESS OF FULLWORD WHERE STATUS IS TO BE PLACED IF THERE IS ON I/O ERROR		320	Y	
S020	SEND_ERROR_PARAMETER_LIST		SEND ERROR SVC PARAMETER LIST				
S020.1	SEND_ERROR_SVC_NUMBER		BITS 0-7 ARE NOT USED.BITS 8-15 CONTAIN THE SVC NUMBER-20		100	BS(16),INIT(20)	
S020.2	SEND_ERROR_GROUP_NUMBER		THE GROUP NUMBER CORRESPONDING TO THE ERROR CONDITION		182	BS(8)	
S020.3	SEND_ERROR_CODE_NUMBER		THE CODE NUMBER CORRESPONDING TO THE ERROR CONDITION		182	BS(8)	
S021	CLOSE_PARAMETER_LIST		CLOSE SVC PARAMETER LIST				
S021.1	CLOSE_SVC_NUMBER		BITS 0-7 ARE NOT USED. BITS 8-15 CONTAIN THE SVC NUMBER-21		100	BS(16),INIT(21)	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
S022	TIME/DATE_REQUEST_PARAMETER_LIST		TIME AND DATE SVC PARAMETER LIST		144		
S022.1	TIME/DATE_REQUEST_TYPE		BITS 0-7 CONTAIN THE REQUEST TYPE AS FOLLOWS: 01- RUNTIME 02- CLOCKTIME 03- DATE 04- MET		144	BS(8)	
S022.2	TIME/DATE_REQUEST_SVC_NUMBER		THE SVC NUMBER-22		100	BS(8),INIT (22)	
S022.3	MET_BUFFER_ADDRESS		THE ADDRESS WHERE MET TIME IS TO BE STORED		144	Y	
S023	PROCESS_STATUS_PARAMETER_LIST		ERRGRP/ERRNUM SVC PARAMETER LIST				
S023.1	STATUS_REQUEST_TYPE		BITS 0-7 CONTAIN THE REQUEST TYPE AS FOLLOWS: 01- ERRGRP 02- ERRNUM		185	BS(8)	
S023.2	STATUS_SVC_NUMBER		THE SVC NUMBER- 23		100	BS(8),INIT(23)	
S024	I/O_SVC_PARAMETER_LIST	TFIOS				ST	
S024.1	I/O_SVC_NUMBER	TIOSSVC	SVC NUMBER			I,INIT(24)	
S024.2	I/O_SVC_PARAMETER_LIST_FLAGS	TIOSFLGS	FLAGWORD		200	BS(16)	
S024.3			BITS 0-4 UNUSED			BS(5)	
S024.8	SYNC_TYPE_FLAG	TICSSYNT	BIT 5-TYPE OF SYNC REQUIRED FOR I/O REQUEST. 0= REDUNDANT SET, 1= COMMON SET			BT	
S024.9	COMFAULT_STATUS_NEEDED_FLAG	TIOSCOMF	BIT 6-COMFAULT TYPE STATUS NEEDED FOR THIS I/O REQUEST			BT	
S024.10	ICC-TYPE_REQUEST_FLAG	TIOSICCR	BIT 7-THIS IS AN ICC TYPE REQUEST			BT	
S024.11	TIME_TAG_REQUESTED_FLAG	TIOSTTAG	BIT 8-TIME TAGGING OF THE DATA HAS BEEN REQUESTED			BT	
S024.12	WAIT_FOR_I/O_COMPLETE_FLAG	TIOSWAIT	BIT 9-THE REQUESTOR WOULD LIKE FOR I/O TO COMPLETE BEFORE RESUMING EXECUTION			BT	

SVC PARAMETER LISTS

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
S024.13	PROTECTED_TRANSACTION_FLAG	TIOSPRTR	BIT 10-THIS TRANSACTION IS PROTECTED AND WILL REQUIRE INTER-COMPUTER COMMUNICATION UPON I/O COMPLETE.			BT	
S024.14	STATUS_REQUESTED_FLAG	TIOSSTRQ	BIT 11- REQUESTOR HAS ASKED THAT IT BE NOTIFIED IF AN ERROR OCCURED DURING THE TRANSACTION			BT	
S024.15	OUTPUT_TYPE_REQUEST_FLAG	TIOSOTYT	BIT 12-THIS IS AN OUTPUT TYPE I/O TRANSACTION			BT	
S024.16	MAJOR_FUNCTION_ID_NUMBER	TIOSMFID	BITS 13-15--FOR MASS MEMORY I/O, INDICATES THE MAJOR FUNCTION OF THE REQUESTING PROCESS. 000=PAYLOAD,001=GN&C,010=SM,011=OPS-0			BS(3)	
S024.17	DEVICE_ID_NUMBER	TIOSDVID	THE UNIQUE DEVICE OR TRANSACTION IDENTIFICATION USED FOR THIS REQUEST. SEE APPENDIX G FOR A COMPLETE LIST.	148	200	I	
S024.18	OPERATION_CODE	TIOSOPCD	THE I/O OPERATION TO BE PERFORMED. READ,WRITE,STATUS,ETC.	148	200	I	
S024.19	I/O_PRIORITY	TIOSPRI	THE PRIORITY OF THIS REQUEST.		200	I	
S024.20	WORD_COUNT_NUMBER	TIOSWDCT	THE NUMBER OF WORDS TO BE RECEIVED OR TRANSMITTED BY THIS REQUEST.		200	I	
S024.21	I/O_BUFFER_ADDRESS	TIOSBUFA	THE ADDRESS OF WHERE TO SEND OR RECEIVE THE FIRST DATA WORD FOR THIS REQUEST	148	200	Y	
S024.22	EVENT_ADDRESS	TIOSEVNT	THE ADDRESS OF THE EVENT TO BE POSTED UPON COMPLETION OF THE I/O.	148	200	Y	
S024.23	MISCELLANEOUS_COMMUNICATION	TIOSSTAD	A FIELD WHICH MAY CONTAIN ONE OF THE FOLLOWING: 1)PMU RAM ADDRESS FOR THE OI/PL DATA RAM READ, 128 KBPS PROGRAM READ, 128 KBPS FORMAT PROGRAM WRITE, 64 KBPS PROGRAM READ OR 64 KBPS FORMAT PROGRAM WRITE. 2) MASS MEMORY STARTING BLOCK ADDRESS 3)TEST CONTROL SUPERVISOR INPUT DATA 4) LDB INPUT DATA		200	1) X 2) X 3) X 4) X	
S025	CONFIGURATION_	TFCMS	CONFIGURATION MANAGEMENT SVC		340	ST(N)	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
	MANAGEMENT_SVC_LIST		PARAMETER LIST				
S025.1	CM_SVC_NUMBER	TFCMSVC	SVC NUMBER		100	!,INIT(25)	
S025.2	CM_SVC_REQUEST	TFCMSID	REQUEST ID OF CM SERVICE 1-READ/WRITE DISCRETES 2-CHANGE CAM LIGHT SETTING 3-SET/RESET TERMINATION LATCHES.		340	I	
S025.3	CM_SVC_OPERATION	TFCMOP	AN INDICATOR TELLING WHICH OPERATION IS TO BE PERFORMED BY EACH CM SERVICE REQUEST		340	I	
S025.4	CM_SVC_BUFFER	TFCMSBUF	ADDRESS IN MAIN MEMORY WHERE DATA IS TO BE STORED.		340	Y	
S025.5	FAIL_DISCRETE_OUTPUT_MASK	TFCMSBUF	MASK PASSED IN CM SVC PARAMETER LIST TO SET/RESET FAIL DISCRETES		340	BS(16)	
S025.6	INHIBIT_FAIL_DISCRETE_OUTPUT		BIT 0 INHIBIT OUTPUT FOR TESTING			BT	
S025.7	FAIL_DISCRETE_OUTPUT_1		BIT 1 FAILURE VOTE GPC N+1			BT	
S025.8	FAIL_DISCRETE_OUTPUT 2		BIT 2 FAILURE VOTE GPC N+2			BT	
S025.9	FAIL_DISCRETE_OUTPUT_3		BIT 3 FAILURE VOTE GPC N+3			BT	
S015.10	FAIL_DISCRETE_OUTPUT_4		BIT 4 FAILURE VOTE GPC N+4			BT	
S025.11	TEST_VOTER_DISCRETES_INPUT_MASK	TFCMSBUF	MASK PASSED IN CM SVC PARAMETER LIST TO PROVIDE INPUT TO VOTER LOGIC (BITS 0-10 NOT USED)		340	BS(16)	
S025.12	VOTER TEST CONTROL		BIT 11-INHIBITS NORMAL VOTER OPERATION			BT	
S025.13	VOTER_TEST_INPUT_1		BIT 12_TEST VOTE INPUT GPC N+1			BT	
S025.14	VOTER_TEST_INPUT_2		BIT 13_TEST_VOTER_DISCRETES_INPUT_MASK			BT	
S025.15	VOTER_TEST_INPUT_3		BIT 14_ TEST_VOTER_DISCRETES_INPUT_MASK			BT	
S025.16	VOTER_TEST_INPUT_4		BIT 15-TEST_VOTER_DISCRETES_INPUT_MASK			BT	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
S025.17	TERMINATION_CONTROL_LATCH_MASK	TFCMSBUF	MASK TO SET/RESET TERMINATION CONTROL LATCHES (BITS 0-13 NOT USED) ON= SET OR RESET		340	BS(32)	
S025.18	TIMER_TERMINATION_LATCH_FLAG		BIT 14 OF TERMINATION_CONTROL_LATCH_MASK 1= SET OR RESET			BT	
S025.19	VOTER_TERMINATION_CONTROL_LATCH_FLAG		BIT 15 OF TERMINATION_CONTROL_LATCH MASK			BT	
S025.20	DISCRETE_OUTPUT_MASK	TFCMSBUF	MASK TO SET/RESET DISCRETES		340	BS(32)	
S025.21			BIT0 D000 SPARE			BT	
S025.22			BIT1 D001 SPARE			BT	
S025.23			BIT2 D002 SPARE			BT	
S025.24			BIT3 D003 SPARE			BT	
S025.25			BIT4 D004 SPARE			BT	
S025.26			BIT5 D005 SPARE			BT	
S025.27			BIT6 D006 SPARE			BT	
S025.28	GPC_I/O_ACTIVE_INDICATOR	TFCMSBUF	BIT7 D007 1= READY TO PERFORM DATA BUS TRANSMISSION 0= GPC IS POWERED OFF OR IOP IS IN RESET STATE			BT	
S025.29			BIT8 D008 SPARE			BT	
S025.30	GPC_READY_INDICATOR		BIT9 D009 1= READY 0= NOT READY FOR MCDS KEYBOARD TRANSACTIONS			BT	
S025.31			BIT10 D010 SPARE			BT	
S025.32			BIT11 D011 SPARE			BT	
S025.33	GPC_MML_RESET		BIT12 D012 1=TERMINATE CURRENT MASS MEMORY TRANSACTION			BT	
S025.34			0=CONTINUE CURRENT TRANSACTION			BT	
S025.35	GPC_MML_RESET		BIT13 D013 SAME AS D012 EXCEPT FOR MASS MEMORY 2			BT	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
S025.36			BIT14 D014 SPARE			BT	
S025.37			BIT15 D015 SPARE			BT	
S025.38			BIT16 D016 SPARE			BT	
S025.39			BIT17 D017 SPARE			BT	
S025.40			BIT18 D018 SPARE			BT	
S025.41			BIT19 D019 SPARE			BT	
S025.42	GPC_STANDBY_MODE		BIT20 D020 1=GPC IS IN THE CREW COMMAND STANDBY MODE 0=NOT IN STANDBY MODE				
S025.43			BIT21 D021 SPARE			BT	
S025.44			BIT22 D022 SPARE			BT	
S025.45			BIT23 D023 SPARE			BT	
S025.46	GPC_RUN_MODE		BIT24 D024 1=IN RUN MODE 0=NOT IN RUN MODE			BT	
S025.47			BIT25 D025 SPARE			BT	
S025.48			BIT26 D026 SPARE			BT	
S025.49			BIT27 D027 SPARE			BT	
S025.50	GPC_SYNC		BIT28 D028 1=GPC SYNC SIGNAL SET 0=NOT GPC SYNC SIGNAL SET			BT	
S025.51			BIT29 D029 SPARE			BT	
S025.52	GPC_IDENTIFICATION_SOURCE		BIT30 D030 1=VOLTAGE SOURCE FOR GPC IDENTIFICATION DISCRETES INPUTS D10,D11,D12 IS APPLIED 0=VOLTAGE SOURCE NOT APPLIED			BT	
S025.53	GPC_IPL_INDICATOR		BIT31 D031 1=COMMANDS THE GPC COMPUTER IPL INDICATOR ON D&C PANEL TO BE ILLUMINATED 0=NOT ILLUMINATED			BT	
S026	PROGRAM_		PROGRAM MODIFICATION SVC PARAMETER				

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
	MODIFICATION_PARAMETER_LIST		LIST				
S026.1	PROGRAM_MODIFICATION_FLAG_FIELD	TMODFLGS	PROGRAM MODIFICATION FLAG FIELD INDICATES TYPE OF REQUEST IN BITS 0-7		325	BS(16)	
S026.2		TMODHX80	BIT 0 CANNOT BE USED			BT	
S026.3	INTEGER_DATA	TMODINT	BIT 1 ON INDICATES INTEGER DATA		325	BT	
S026.4	INTEGER_DOUBLE_DATA	TMODINTD	BIT 2 ON INDICATES INTEGER DOUBLE DATA		325	BT	
S026.5	SCALAR_DATA	TMODSLR	BIT 3 ON INDICATES SCALAR DATA		325	BT	
S026.6	SCALAR_DOUBLE_DATA	TMODSLRD	BIT 4 ON INDICATES SCALAR DOUBLE DATA		325	BT	
S026.7		TMODHX04	BIT 5 NOT USED			BT	
S026.8	READ/WRITE_REQUEST_TYPE	TMODREAD	BIT 6 ON INDICATES A READ REQUEST OFF INDICATES A WRITE REQ		325	BT	
S026.9	SET_STORAGE_PROTECT	TMODSSP	BIT 7 ON INDICATES SET STORAGE PROTECT AT THE THE MAIN MEMORY LOCATION INDICATED BY 'START ADDRESS'		325	BT	
S026.10	PROGRAM_MODIFICATION_SVC_NUMBER		BITS 8-15 CONTAIN SVC NUMBER(26)		100	BS(8),INIT(26)	
S026.11	PROGRAM_MODIFICATION_HALFWORD_COUNT	TMODHWDS	THE COUNT OF 16 BIT WORDS TO BE MODIFIED OR READ		325	I	
S026.12	MAIN_MEMORY_START_ADDRESS	TMODADDR	THE FULL WORD STARTING ADDRESS OF MAIN MEMORY TO BE MODIFIED OR READ		325	AC	
S026.13	INTEGER_DATA_BUFFER_ADDRESS	TMODATI	THE HALFWORD ADDRESS OF THE INTEGER DATA BUFFER		325	Y	
S026.14	INTEGER_DOUBLE_DATA_BUFFER_ADDRESS	TMODATID	THE HALFWORD ADDRESS OF THE INTEGER DOUBLE DATA BUFFER		325	Y	
S026.15	SCALAR_DATA_BUFFER_ADDRESS	TMODATS	THE HALFWORD ADDRESS OF THE SCALAR DATA BUFFER		325	Y	
S026.16	SCALAR_DOUBLE_DATA_BUFFER_ADDRESS	TMODATSD	THE HALFWORD ADDRESS OF THE SCALAR DOUBLE DATA BUFFER		325	Y	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
S030	INITIAL_SSIP_SYNC_PARAMETER_LIST		INITIAL SSIP SYNCRONIZATION SVC PARAMETER LIST				
S030.1	INITIAL_SSIP_SYNC_SVC_NUMBER		BITS 0-7 ARE NOT USED.BITS 8-15 CONTAIN THE SVC NUMBER_30		100	BS(16),INIT(30)	
S031	MTU_UPDATE_PARAMETER_LIST	TFMTUSPL	SVC31 USER ISSUED PARAMETER LIST FOR A TIME UPDATE				
S031.1	MTU_UPDATE_REQUEST_TYPE	TMTURSVC	REQUEST		148	BS(8)	
S031.2	MTU_UPDATE_SVC_NUMBER	TMTURSVC	SVC NUMBER		148	BS(8),INIT(31)	
S031.3	MTU_UPDATE_COMPLETION_EVENT_ADDRESS	TMTUEVNT	COMPLETION EVENT ADDRESS		148	Y	
S031.4	MTU_ACCUMULATOR_NUMBER	TMTUACUM	ACCUMULATOR			H	
S031.5	MTU_UPDATE_COINCIDENT_TIME	TMTUCTIM	COINCIDENT TIME FOR THE UPDATE		148	H	
S031.6	MTU_UPDATE_ABSOLUTE_TIME	TMTUUTIM	ABSOLUTE TIME OF UPDATE		148	DSC	
S031.7	MTU_DELTA_UPDATE_TIME	TMTUDTIM	DELTA UPDATE VALUE		148	E	
S032	MTU_RM_PARAMETER_LIST	TFSVCPL	SVC32 PARAMETER LIST FOR MTU REDUNDANCY MANAGEMENT		149		
S032.1	MTU_RM_REQUEST_TYPE	TSVCRTSN	THE FIRST EIGHT BITS OF THIS HALF WORD CONTAIN THE REQUEST TYPE NUMBER		149	H	
S032.2	MTU_RM_SVC_NUMBER	TSVCRTSN	THE SECOND HALF OF THIS HALFWORD CONTAINS THE SVC NUMBER		149	H	
S032.3	MTU_RM_COMPLETION_EVENT	TSVCEVNT	ADDRESS OF THE PCT WAITING ON THE COMPLETION OF THIS EVENT		149	Y	
S033	OPS_CANCEL_PARAMETER_LIST	TFOPSCPL	SVC33 OPS CANCEL PARAMETER LIST				
S033.1	OPS_CANCEL_COUNT	TOPSCSVC	THE FIRST EIGHT BITS CONTAINING A COUNT OF PCTS NOT TO BE CANCELLED		110	H	
S033.2	OPS_CANCEL_SVC_NUMBER		TOPSCSVC BITS 8-15 CONTAIN THE SVC NUMBER		110	BS(8),INIT(33)	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
S033.3	OPS_MAJOR_FUNCTION_ID	TOPSMFID	MAJOR FUNCTION ID		110	H	
S033.4	OPS_PROCESS_LIST_ADDRESS	TOPSPGMA	ADDRESS OF A LIST OF PROCESSES THAT ARE NOT TO BE CANCELLED		110	H	
S034	NORMAL_INITIALIZATION_EMULATION_EMULATION_PARAMETER_LIST		NORMAL INITIALIZATION EMULATOR SVC PARAMETER LIST				
S034.1	NORMAL_INITIALIZATION_EMULATION_SVC_NUMBER		BITS 0-7 ARE NOT USED.BITS 8-15 CONTAIN THE SVC NUMBER_34		100	BS(16),INIT(34)	
S035	MASK_MANAGEMENT_PARAMETER_LIST		BUS/DATA PATH MASK MANAGEMENT SVC PARAMETER LIST		355 368		
S035.1	MASK_MANAGEMENT_REQUEST_TYPE		TYPE OF BUS/DATA PATH MASK MANAGEMENT REQUESTED AS FOLLOWS: '81'-SET BUS MASK FOR SPECIFIED BUS '82'-SET BUS MASKS FOR BUSES COMMANDED BY A GPC '83'-SET BUS MASKS FOR BUSES ON A SPECIFIED STRING '84'-SET BUS MASKS FOR BFCS ENGAGE '86'-SET BUS MASKS FOR BUSES INDICATED IN A MASK '01'-RESET BUS AND DATA PATH MASKS OF SPECIFIED BUS '02'-RESET BUS AND DATA PATH MASKS OF BUSES COMMANDED BY A GPC '03'-RESET BUS AND DATA PATH MASKS FOR BUSES ON A SPECIFIED STRING '04'-RESET BUS AND DATA PATH MASKS FOR BFCS DISENGAGE '05'-RESET ALL BUS AND DATA PATH MASKS(EXCEPT DEU) '06'-RESET BUS AND DATA PATH MASKS FOR BUSES IN SPECIFIED MASK '41'-RESET DATA PATH MASK OF SPECIFIED BUS '43'-RESET DATA PATH MASKS FOR BUSES ON A SPECIFIED STRING '45'-RESET ALL DATA PATH MASKS (EXCEPT DEU)	351	355 368	A(2),X	
S035.2	MASK_MANAGEMENT_SVC_NUMBER		SVC NUMBER		100 368	BS(8),INIT(35)	

SVC PARAMETER LISTS

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
S035.3	MASK_MANAGEMENT_VARIABLE_FIELD		VARIABLE DATA WHICH CAN BE ANY OF THE FOLLOWING: BUS#-INTEGER BUS NUMBER FOR TYPES HEX'81','01',OR HEX '41' GPCID-INTEGER GPCID FOR TYPES HEX '82', OR HEX'02' STRING II- INTEGER STRING NUMBER FOR TYPES HEX'83',HEX'03' OR HEX '43' MASK-32 BIT MASK OF BUSES FOR TYPES HEX'86' AND HEX'06' WHERE BITS 1-24 CORRESPOND TO BUSES 1-24 IGNORED-TYPES HEX'84','HEX'04' HEX '05',AND HEX '45'	355 368	I OR BS(32)		
S036	BUS_RECONFIGURATION_PARAMETER_LIST		BUS, RECONFIGURATION SVC PARAMETER LIST	350 368			
S036.1	BUS_RECONFIGURATION_REQUEST_TYPE	TBRPTYPE	BITS 0-7 CONTAIN THE TYPE OF BUS RECONFIGURATION REQUEST AS FOLLOWS: 1-REMOVE A BUS FROM COMMAND MODE 2-PLACE A BUS IN COMMAND MODE 3-PLACE A BUS IN LISTEN MODE 4-DISABLE A BUS 5-REMOVE A STRING FROM COMMAND MODE 6-RELEASE ALL STRINGS COMMANDED BY A GPC 7-PLACE A STRING IN COMMAND MODE 8-MODE A STRING 9-CONFIGURE ACCORDING TO THE NOMINAL STRING ASSIGNMENT TABLE	350 368	I		
S036.2	BUS_RECONFIGURATION_SVC_NUMBER		BITS 8-15 CONTAIN THE SVC NUMBER-36	100 368	BS(8),INT(36)		
S036.3	LOR_TOGGLING_INDICATOR	TBRPSTAT	THE ADDRESS OF THE SYNC STATUS FIELD IF THE REQUEST IS AN LOR TOGGLING REQUEST	350 368	Y		
S036.4	RECONFIGURATION_VARIABLE_1_FIELD	TBRPVAR1	VARIABLE DATA WHICH CAN BE ANY OF THE FOLLOWING: BUS#-INTEGER BUS# FOR TYPES 1,2,3, OR 4 STRING#-INTEGER STRING # FOR TYPES 5,7 OR 8 GPCID-INTEGER GPCID OF GPC TO BE RECONFIGURED FOR TYPE 6 OVERLAY#-INTEGER OVERLAY NUMBER IF IT IS A NOMINAL STRING ASSIGNMENT	350 368	I		

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
			TABLE CONFIGURATION REQUEST				
S036.5	RECONFIGURATION_VARIABLE_2_FIELD	TBRPVAR2	VARIABLE DATA WHICH CAN BE ANY OF THE FOLLOWING: 0-ZERO FOR TYPES 1,2,6 AND 9 MODE#-MODE NUMBER FOR TYPE 8 GPCID-GPCID OF THE GPC BEING GIVEN COMMAND OF THE STRING/BUS FOR TYPES 3,4,5 OR 7		350 368	I	
S037	BCE_ELEMENT_BYPASS_PARAMETER LIST		BCE ELEMENT BYPASS SVC PARAMETER LIST				
S037.1	BCE_ELEMENT_BYPASS_SVC_NUMBER		BITS 0-7 ARE NOT USED.BITS 8-15 CONTAIN THE SVC NUMBER-37		100	BS(16),INIT(37)	
S037.2	BCE_ELEMENT_NUMBER		A NUMBER WHICH CORRESPONDS TO THE BCE ELEMENT(S) TO BE BYPASSED OR RESTORED IN I/O TRANSACTIONS		330	I	
S038	SYNC_MASK_GENERATION_PARAMETER LIST		SYNC MASK GENERATION SVC PARAMETER LIST				
S038.1	SYNC_MASK_GENERATION_SVC_NUMBER		BITS 0-7 ARE NOT USED.BITS 8-15 CONTAIN THE SVC NUMBER-38		100	BS(16),INIT(38)	
S039	FORCE_FAIL_TO_SYNC_PARAMETER_LIST		FORCE FAIL-TO-SYNC SVC PARAMETER LIST				
S039.1	FORCE_FAIL_TO_SYNC_SVC_NUMBER		BITS 0-7 ARE NOT USED.BITS 8-15 CONTAIN THE SVC NUMBER-39		100	BS(16),INIT(39)	
S040	PREINITIALIZED_I/O_SVC_PARAMETER LIST		I/O SVC PARAMETER LIST FOR PREINITIALIZED I/O SVC REQUESTS		201		
S040.1	PREINITIALIZED_I/O_I/O_SVC_NUMBER		BITS 0-7 ARE NOT USED.BITS 8-15 CONTAIN THE SVC NUMBER_40		100	BS(18),INIT(40)	
S040.2	PREINITIALIZED_IOQE_NUMBER		A NUMBER USED TO INDEX INTO THE PREINITIALIZED_IOQE_ADDRESS_TABLE TO LOCATE THE PREINITIALIZED IOQE BEING REQUESTED		201	I	
S041	SELECT_I/O_PARAMETER_LIST	SVCIN	SVC41 TIMER INITIATED PARAMETER LIST			H,INIT(41)	
S041.1	SELECT_I/O_SVC_NUMBER	SVCSVCNO	SVC NUMBER				
S041.2	SELECT_I/O_PDE_	SVCPDE	PROGRAM DIRECTORY ENTRY ADDRESS		120	Y	

SVC PARAMETER LISTS

ID	ITEM ASSEMBLER NAME	DESCRIPTION	SRC	OST	ATTRIBUTES MML
	ADDRESS				
S041.3	I/O_PARAMETER_LIST_ADDRESS	SVCID			
		ADDRESS OF THE I/O PARAMETER LIST	120	V	
S043	TIME_CONVERSION_PARAMETER_LIST	SVC43 TIME CONVERSION PARAMETER LIST	144 150		
S043.1	TIME_CONVERSION_REQUEST_TYPE	BITS 0-7 CONTAIN THE REQUEST TYPE 0 REQUEST TYPE CONVERTS FCOS FIXED POINT TIME TO FLOATING POINT FORMAT 1 REQUEST TYPE CONVERTS MTU TIME TO FLOATING POINT FORMAT	144 150	BS(8)	
S043.2	TIME_CONVERSION_SVC_NUMBER	BITS 0-7 CONTAIN THE SVC NUMBER 43	100 150	BS(8),INIT(43)	
S043.3	TIME_CONVERSION_INPUT_ADDRESS	ADDRESS OF TIME TO BE CONVERTED	144 150	H	
S043.4	TIME_CONVERSION_OUTPUT_ADDRESS	ADDRESS TO PUT TIME THAT HAS BEEN CONVERTED TO FLOATING POINT	144 150	H	

SVC PARAMETER LISTS

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
Y001	PREINITIALIZED_IOQE_ADDRESS_TABLE	FIOIAT	THIS TABLE CORRELATES I/O NUMBERS TO PREINITIALIZED IOQE'S (RESIDES IN FCMCBLKS)		201	A(13),Y,INIT(0,12#(A POINTER TO A UNIQUE IOQE FOR EACH I/O NUMBER))	
Y002	IUA_TABLE	FIOIUATB	THIS TABLE IS USED BY THE IOP DISPATCHER TO DETERMINE IF DATA PATHS ARE MASKED. THE ENTRIES ARE ORDERED BY DEVICE ID. IF THE DEVICE ID REFERENCES ONLY ONE IUA, THAT IUA IS FOUND AT THE RESPECTIVE DISPLACEMENT.FOR MULTIPLE IUA REFERENCES,THE ENTRY IS A POINTER TO A LIST OF IUA'S. THE IUA'S USED ARE FOUND IN THE INDIVIDUAL COMMAND WORDS LOCATED IN FIGURE 5.2.3-2.(RESIDES IN FCMCBLKS).		205	A(39),Y	
Y003	BCE_DEVICE_MASK_TABLE	FIOBCD	THIS TABLE ASSOCIATES DEVICE ID'S WITH A MASK OF BUSES NEEDED TO SERVICE AN I/O OPERATION. (SEE APPENDIX G.)(RESIDES IN FCMCBLKS)	351 390	200 201 220 244	A(39),BS(32)	
Y004	BCE_PROGRAM_COUNTER_POINTER_TABLE	FIOPCP	A TABLE INDEXED BY DEVICE ID CONTAINING POINTERS TO BCE PROGRAM COUNTER TABLES. A ZERO ENTRY INDICATES SPECIAL LOGIC IS REQUIRED TO LOCATE THE PROPER BCE PROGRAM COUNTER TABLE.(RESIDES IN FCMCBLKS).		205	A(39),Y	
Y005	BCE_BASE_REGISTER_TABLE	FIOBRE	TABLE INDEXED BY BUS NUMBER CONTAINING BUFFER ADDRESSES FOR SINGLE ELEMENT NON-SELF-INITIALIZING BCE PROGRAMS. (RESIDES IN FCMCBLKS).	205 280	261 262 263 264 282 283	A(25),AC	
Y006	BCE_WORD_COUNT_TABLE	FIOWCE	TABLE INDEXED BY BUS NUMBER CONTAINING WORD COUNTS FOR SINGLE ELEMENT NON-SELF-INITIALIZING BCE PROGRAMS.(RESIDES IN FCMCBLKS)	205	261 263 264 282	A(24),F	
Y007	BCE_COMMAND_WORD_TABLE	FIOCWE	TABLE INDEXED BY BUS NUMBER CONTAINING COMMAND WORDS FOR SINGLE ELEMENT NON-SELF-INITIALIZING BCE PROGRAMS (RESIDES IN FCMCBLKS)	205 280	264 282 283	A(25),BS(32)	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
Y008	PMU_BASE_REGISTER_TABLE	FIOPBRE	TABLE INDEXED BY BUS NUMBER CONTAINING BUFFER ADDRESSES FOR PMU BCE PROGRAMS. (RESIDES IN FCMCBLKS)	205		A(47),AC	
Y009	PMU_WORD_COUNT_TABLE	FIOPWCE	TABLE INDEXED BY BUS NUMBER CONTAINING WORD COUNTS FOR PMU BCE PROGRAMS. (RESIDES IN FCMCBLKS)	205		A(49),F	
Y010	PMU_COMMAND_WORD_TABLE	FIOPCWE	TABLE INDEXED BY BUS NUMBER CONTAINING COMMAND WORDS FOR PMU BCE PROGRAMS. (RESIDES IN FCMCBLKS)	205		A(49),BS(32)	
Y012	MASS_MEMORY_EXTEND_BLOCK_COUNT_TABLE	FIOEBM	CONTAINS TWO COMMAND DATA WORDS TO EXTEND THE MASS MEMORY MAXIMUM BLOCK COUNT.	280 282	282 283		
Y012.1	EXTENDED_BLOCK_TABLE_MASS_MEMORY_1	TEBMMM1	COMMAND DATA WORD FOR MASS MEMORY 1.	280	282 283	H	
Y012.2	EXTENDED_BLOCK_TABLE_MASS_MEMORY_2	TEBMMM2	COMMAND DATA WORD FOR MASS MEMORY 2. (RESIDES IN FCMCBLKS)	280	282 283	H	
Y013	MASS_MEMORY_ENABLE_WRITE_TABLE	FIOEWM	TO PROVIDE MM BCE 'WRITE' PROGRAMS THE COMMAND DATA TO OVERRIDE WRITE PROTECTION OF A REQUESTED MM TRACK (RESIDES IN FCMCBLKS)	280 282	282 283		
Y013.1	MASS_MEMORY_ENABLE_WRITE_MM1	TEWMMM1	COMMAND WORD TO BE USED TO OVERRIDE WRITE PROTECTION ON MASS MEMORY 1 (BUS 18)	280	282 283	H	
Y013.2	MASS_MEMORY_ENABLE_WRITE_MM2	TEWMMM2	COMMAND WORD TO BE USED TO OVERRIDE WRITE PROTECTION ON MASS MEMORY 2(BUS 19).	280	282 283	H	
Y014	ICC/IPR_BASE_REGISTER_TABLE	FIOIBRE	TABLE INDEXED BY BUS NUMBER TO OBTAIN ADDRESS OF IPR ICC BUFFERS. (RESIDES IN FCMCBLKS)	205	262	A(5),AC	
Y015	FF/FA_BITE_ACQUISITION_BASE_REGISTER_TABLE	FIOBBRE	A TABLE INDEXED BY BUS NUMBER TO DETERMINE THE BUFFER LOCATIONS INTO WHICH TO READ FF AND FA BITE INFORMATION. (RESIDES IN FCMCBLKS)	205	264	A(18),AC	
Y016	INITIAL_ITERATION_COUNT_TABLE	FIOITC	THIS TABLE IS INDEXED BY DEVICE ID TO DETERMINE THE TIME, IN 16 MICROSECOND UNITS, THAT AN I/O OPERATION WILL TAKE. ENTRY ONE IS NOT USED. A ZERO ENTRY INDICATES SPECIAL FCOS PROCESSING WILL OCCUR	205		A(39),I,INIT (0,42,42,625,0, 0,0,0,212,0,0, 38,78,194,91,21 2 69,15,14,14,14,	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
			TO DETERMINE THE TIME. (RESIDES IN FCMCBLKS)			39,0,0,0,0,0,0,0,0,37,37,37,29,0,0,0,0,0)	
Y017	PMU_INITIAL_ITERATION_COUNT_TABLE	FIOPMUIC	EACH ENTRY REPRESENTS THE TIME, IN 16 MICROSECOND UNITS, THAT A PMU I/O OPERATION WILL TAKE. ENTRY ONE IS NOT USED. A ZERO ENTRY INDICATES THE TIME WILL VARY WITH EACH REQUEST AND MUST BE RECOMPUTED. (RESIDES IN FCMCBLKS).	205		A(11),I,INIT(0,1193,1193,1209,1209,13,7,7,0,0)	
Y018	DEU_INITIAL_ITERATION_COUNT_TABLE	FIODEUIC	EACH TABLE ENTRY CONTAINS A NUMBER REPRESENTING THE TIME,IN 16 MICROSECOND UNITS,THAT A DEU I/O OPERATION WILL TAKE. THE FIRST ENTRY IS NOT USED.(RESIDES IN FCMCBLKS)	205		A(7),I,INIT(0,2244,344,438,1313,375,488)	
Y018.1	DEU_FILL_INITIAL_ITERATION_COUNT	FIODFLIC	ENTRY NUMBER TWO IN THE DEU INITIAL_ITERATION_COUNT_TABLE	205		I	
Y018.2	DEU_RESET_SCRATCH_PAD_LINE_INITIAL_ITERATION_COUNT	FIODRSIC	ENTRY NUMBER SEVEN IN THE DEU_INITIAL_ITERATION_COUNT_TABLE	205		I	
Y019	LDB_INITIAL_ITERATION_COUNT_TABLE	FIOLDBIC	EACH TABLE ENTRY CONTAINS A NUMBER REPRESENTING THE TIME,IN 16 MICROSECOND UNITS,THAT LDB I/O OPERATION WILL TAKE. THE FIRST TABLE ENTRY IS NOT USED. ZERO ENTRIES INDICATE THE TIME VARIES WITH EACH REQUEST AND IS RECOMPUTED AS THE REQUEST IS PROCESSED. (RESIDES IN FCMCBLKS)	205		A(7),I,INIT(0,39,41,0,0,50,10)	
Y020	BASE_REGISTER/ELEMENT_TABLE	FIOBRTBL	THIS TABLE IS USED BY THE LEVEL C I/O_ERROR_INTERRUPT_HANDLER TO CORRELATE EACH BCE BASE REGISTER ADDRESS WITH A UNIQUE ELEMENT NUMBER IN ORDER TO BYPASS A BCE ELEMENT. INDEXING IS USED TO COMPARE EACH ADDRESS IN THE TABLE WITH A BASE REGISTER ADDRESS OBTAINED BY READING BCE LOCAL STORE LOCATION C3 FOR THE BCE IN ERROR. THE ADDRESSES ARE WITHIN THE CGL_GNC COMPOOL AND ARE DEFINED VIA THE HAL EQUATE FACILITY. (RESIDES IN FCMCBLKS)		244	A(32),AC,INIT(0,TFCMEI11,TFCMEI12,TFCMEI13,TFCMEI14,TFCMDIS1,TFCMEI21,TFCMADT1,TFCMSBL1,TFCMIMU1,TFCMMTU1,TFCMDIS2,TFCMEI22,TFCMADT2,TFCMSBL2,	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
						TEQMIMU2, TEQMMTU2, TEQMDIS3, TEQMFI23, TEQMADT3, TEQMSRL3, TEQMIMU3, TEQMMTU3, TEQMDIS4, TEQMFI24, TEQMADT4, TEQMEAI1, TEQMEAI2, TEQMEAI3, TEQMEAI4, TEQMEAI5, TEQMEAI6)	
Y021	MASS_MEMORY_UTILITY_BASE_REGISTER_TABLE	FIO3RU	THIS TABLE COMMUNICATES 1 TO 31 BUFFER ADDRESSES TO THE 'LOAD BASE REGISTER' INSTRUCTIONS IN THE MASS_MEMORY_UTILITY_WRITE_BCE_ PROCESSOR (283) INDEXED BY BLOCK NUMBER STARTING WITH BLOCK 2. AND CONTAINS TWO ENTRIES FOR EACH BLOCK-ONE ENTRY PER MM UNIT. (RESIDES IN FIOMUWPG)	280	283	A(31,2),F	
Y022	MASS_MEMORY_MANAGEMENT_SEARCH_COMPLETE_WORD_BUFFER	FIOMSCWB	THIS BUFFER IS USED TO RECEIVE THE MASS MEMORY SEARCH COMPLETE WORD PRIOR TO TRANSMITTING DATA. (RESIDES IN FCMCBLKS)	282 283	282 283	H	
Y022.1	MASS_MEMORY_BCE_BRANCH_TABLE	FIOBBM	THIS TABLE COMMUNICATES TO THE MASS_MEMORY_UTILITY_WRITE_BCE_ PROCESSOR (283) THE ADDRESS OF THE SECOND MM BLOCK TRANSMIT SEQUENCE. THE TABLE CONTAINS TWO ENTRIES-ONE /MM UNIT. RESIDES IN FCMCBLKS.	280	283	A(31,2),F	
Y023	MSC_DATA_CONSTANTS			280	283		
Y023.1	MSC_COMPLETED_I/O_REQUEST	FIOMIOQE	18 BIT ADDRESS OF IOQE FOR WHICH AN I/O COMPLETE INTERRUPT WILL BE GENERATED. IF BIT 0=1 THERE WAS AN ERROR ON THE REQUEST.	251	251	BS(32)	
Y023.2	MSC_SSIP_ICC_DEVICE_ID	FIOSSIP	SSIP ICC DEVICE ID		251	I	
Y023.3	MSC_I/O_MONITOR_WINDOW	FIOMWNDS,FIOMWNDC	THIS IS THE AMOUNT OF TIME THE MSC WILL MONITOR THE BUSES IN AN I/O TRANSACTION FOR I/O COMPLETE USING		251	I	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
			THE @ RAH INSTRUCTION. (FIOMWNDS IS FOR SSIPICC I/O. FIOMWNDO IS FOR ALL OTHER I/O)				
Y023.4	MSC_MONITOR_COUNT		THIS IS A FICTITIOUS CONSTANT USED IN THE FLOWCHARTS FOR CLARIFICATION PURPOSES. IN THE ACTUAL CODE THE VALUE RESIDES IN A REGISTER.	251	251	I	
Y024	MASK_OF_BUSES_TO_BE_RESET	FIOMSBMK	MASK OF BUSES WHICH SHOULD BE RESET TO A WIX INSTRUCTION	244	252	BS(32)	
Y025	TIME_INITIATED_I/O_TABLE	FPMTIOT	TABLE USED TO ASSOCIATE I/O WITH A PARTICULAR PROCESS UNTIL THE PROCESS IS SCHEDULED. AFTER THE PROCESS HAS BEEN SCHEDULED THE ASSOCIATION OF THE I/O TO THE PROCESS IS CONTAINED IN THE PCT OF THE PROCESS.	101 120	101 120	ST(2)	
Y025.1	TIME_INITIATED_I/O_PARENT_PDE_ADDRESS	TTIOTPDE	THE ADDRESS OF THE PDE OF THE PROCESS ASSOCIATED WITH THE I/O DEFINED IN THE PARAMETER LIST ADDRESSED BY THE PROCESS_I/O_PARAMETER_LIST_ADDRESS (Y025.2) OR ZERO IF THIS ENTRY IN THE TABLE IS NOT CURRENTLY IN USE.	101 120	101 120	Y,INIT(0)	
Y025.2	PROCESS_I/O_PARAMETER_LIST_ADDRESS	TTIOTIO	THE ADDRESS OF THE I/O PARAMETER LIST FOR THE I/O (PRE-INITIALIZED IOCE TYPE I/O) TO BE ASSOCIATED WITH THE PROCESS IDENTIFIED BY THE TIME_INITIATED_I/O_PARENT_PDE_ADDRESS (Y025.1)	120	101	Y	
Y026	GPC_AGE_RESTART_FLAG	FIORSUME	FLAG WORD USED TO CONTROL GPC RESTART FOLLOWING AN AGE INTERRUPT. (RESIDES IN FCMC8LKS)	242	242	I,INIT(0)	
Y027	AGE_INTERRUPT_GMT	FIOAGEFG	AGE INTERRUPT FLAG WORD CONTAINING THE HALF HOUR PORTION OF GMT AT THE TIME OF THE INTERRUPT. (RESIDES IN FCMC8LKS)	242		I,INIT(0)	
Y028	AGE_INTERRUPT_Q_POINT_FLAG	FIOAGFL2	FLAG WORD USED TO INDICATE A Q-POINT VALID AGE INTERRUPT HAS OCCURRED. (RESIDES IN FCMC8LKS)	242	242	I,INIT(0)	
Y029	TRANSACTION_	FIOTCT	A TABLE INDEXED BY DEVICE ID	220	220	A(40),I,INIT(0)	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
	COUNTER_TABLE		CONTAINING K COUNTERS FOR EACH TRANSACTION. (RESIDES IN FCMCBLKS).	244 330	244 330		
Y030	DDU1_OUTPUT_BUFFER	CGDD_ADI_CONTROL (FIODDUB1)	OUTPUT BUFFER FOR DDU1	260	260	A(36),I	
Y031	DDU2_OUTPUT_BUFFER	CGDD_ADI_CONTROL (FIODDUB2)	OUTPUT BUFFER FOR DDU2	260	260	A(36),I	
Y032	ICC_PROGRAM_ COUNTER_TABLE	FIOICCPC	A TABLE INDEXED BY BUS NUMBER CONTAINING THE ICC BCE PROGRAM COUNTER VALUES. (RESIDES IN FIOADCNS)	205		A(6),AC	
Y033	RS_IPR_PROGRAM_ COUNTER	FIOIPRRS	THE REDUNDANT SET IPR BCE PROGRAM COUNTER VALUE. (RESIDES IN FIOADCNS)		205	AC	
Y034	CS_IPR_PROGRAM_ COUNTER	FIOIPRCS	THE COMMON SET IPR BCE PROGRAM COUNTER VALUE. (RESIDES IN FIOADCNS)		205	AC	
Y035	DEU_FILL_PROGRAM_ COUNTER_TABLE	FIOPDFLP	A TABLE INDEXED BY BUS NUMBER CONTAINING THE DEU FILL BCE PROGRAM COUNTER VALUES. (RESIDES IN FIOADCNS)		205	A(9),AC	
Y036	DEU_POLL_PROGRAM_ COUNTER_TABLE	FIOPDPOL	A TABLE INDEXED BY BUS NUMBER CONTAINING THE DEU POLL BCE PROGRAM COUNTER VALUES. (RESIDES IN FIOADCNS)		205	A(9),AC	
Y037	DEU_DUMP_PROGRAM_ COUNTER_TABLE	FIOPDDPP	A TABLE INDEXED BY BUS NUMBER CONTAINING THE DEU DUMP BCE PROGRAM COUNTER VALUES. (RESIDES IN FIOADCNS)		205	A(9),AC	
Y038	DEU_RESET_SCRATCH_ PAD_LINE_PROGRAM_ COUNTER_TABLE	FIOPDRSP	A TABLE INDEXED BY BUS NUMBER CONTAINING THE DEU RESET SCRATCH PAD LINE BCE PROGRAM COUNTER VALUES. (RESIDES IN FIOADCNS)		205	A(9),AC	
Y039	DEU_KEYBOARD_ REQUEST_PROGRAM_ COUNTER_TABLE	FIOPDKEY	A TABLE INDEXED BY BUS NUMBER CONTAINING THE DEU KEYBOARD REQUEST BCE PROGRAM COUNTER VALUES (RESIDES IN FIOADCNS)		205	A(9)	
Y040	DEU_BITE_STATUS_ PROGRAM_COUNTER_ TABLE	FIOPDRBS	A TABLE INDEXED BY BUS NUMBER CONTAINING THE DEU BITE STATUS BCE PROGRAM COUNTER VALUES		205	A(9),AC	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
			(RESIDES IN FIOADCNS)				
Y041	8_WORD_SSIP_ICC_PROGRAM_COUNTER	FIOSIP08	THE 8 WORD SSIP ICC BCE PROGRAM COUNTER VALUE (RESIDES IN FIOADCNS)		205	AC	
Y042	16_WORD_SSIP_ICC_PROGRAM_COUNTER	FIOSIP16	THE 16 WORD SSIP ICC BCE PROGRAM COUNTER VALUE (RESIDES IN FIOADCNS)		205	AC	
Y043	32_WORD_SSIP_ICC_PROGRAM_COUNTER	FIOSIP32	THE 32 WORD SSIP ICC BCE PROGRAM COUNTER VALUE (RESIDES IN FIOADCNS)		205	AC	
Y044	64_WORD_SSIP_ICC_PROGRAM_COUNTER	FIOSIP64	THE 64 WORD SSIP ICC BCE PROGRAM COUNTER VALUE (RESIDES IN FIOADCNS)		205	AC	
Y045	128_WORD_SSIP_ICC_PROGRAM_COUNTER	FIOSP128	THE 128 WORD SSIP ICC BCE PROGRAM COUNTER VALUE (RESIDES IN FIOADCNS)		205	AC	
Y046	LDB_INTERROGATE_W/O_DATA_PROGRAM_COUNTER_TABLE	FIOLIBCE	A TABLE INDEXED BY BUS NUMBER CONTAINING THE LDB INTERROGATE WITHOUT DATA BCE PROGRAM COUNTER VALUES. (RESIDES IN FIOADCNS)		205	A(24),AC	
Y047	LDB_INTERROGATE_W/DATA_PROGRAM_COUNTER_TABLE	FIOLDBPC	A TABLE INDEXED BY BUS NUMBER CONTAINING THE LDB INTERROGATE WITH DATA BCE PROGRAM COUNTER VALUES. (RESIDES IN FIOADCNS)		205	A(24),AC	
Y048	LDB_TRANSMISSION_ENABLE_PROGRAM_COUNTER_TABLE	FIOLDBTC	A TABLE INDEXED BY BUS NUMBER CONTAINING THE LDB TRANSMISSION ENABLE BCE PROGRAM COUNTER VALUES. (RESIDES IN FIOADCNS)		205	A(24),AC	
Y049	LDB_STATUS_REQUEST_PROGRAM_COUNTER_TABLE	FIOLRBCE	A TABLE INDEXED BY BUS NUMBER CONTAINING THE LDB STATUS REQUEST BCE PROGRAM COUNTER VALUES. (RESIDES IN FIOADCNS)		205	A(24),AC	
Y050	LDB_STATUS_PROGRAM_COUNTER_TABLE	FIOLCBCE	A TABLE INDEXED BY BUS NUMBER CONTAINING THE LDB STATUS BCE PROGRAM COUNTER VALUES. (RESIDES IN FIOADCNS)		205	A(24),AC	
Y051	TCS_STANDARD_CONTROL_BCE_PROGRAM_COUNTER_TABLE	FIOACBCE	A TABLE INDEXED BY BUS NUMBER CONTAINING THE TCS STANDARD CONTROL BCE PROGRAM COUNTER VALUES (RESIDES IN FIOADCNS)		205	A(24),AC	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
Y052	TCS_STANDARD_READ_BCE_PROGRAM_COUNTER_TABLE	FIOARBCE	A TABLE INDEXED BY BUS NUMBER CONTAINING THE TCS STANDARD READ BCE PROGRAM COUNTER VALUES (RESIDES IN FIOADCNS)		205	A(24),AC	
Y053	TCS_STANDARD_WRITE_BCE_PROGRAM_COUNTER_TABLE	FIOAWBCE	A TABLE INDEXED BY BUS NUMBER CONTAINING THE TCS STANDARD WRITE BCE PROGRAM COUNTER VALUES (RESIDES IN FIOADCNS)		205	A(24),AC	
Y054	PMU_WRITE_COMPUTER_DATA_RAM_PROGRAM_COUNTER_TABLE	FIOPCWCD	A TABLE INDEXED BY BUS NUMBER CONTAINING THE PMU WRITE COMPUTER DATA RAM BCE PROGRAM COUNTER VALUES (RESIDES IN FIOADCNS)		205	A(25),AC	
Y055	PMU_WRITE_128/64_KBPS_PROGRAM_COUNTER_TABLE	FIOPCWDF	A TABLE INDEXED BY BUS NUMBER CONTAINING THE PMU 128/64 KBPS FORMAT WRITE BCE PROGRAM COUNTER VALUES (RESIDES IN FIOADCNS)		205	A(25),AC	
Y056	PMU_READ_128/64_KBPS_PROGRAM_COUNTER_TABLE	FIOPCRDF	A TABLE INDEXED BY BUS NUMBER CONTAINING THE PMU 128/64 KBPS FORMAT READ BCE PROGRAM COUNTER VALUES (RESIDES IN FIOADCNS)		205	A(25),AC	
Y057	PMU_FORMAT_SELECT_PROGRAM_COUNTER_TABLE	FIOPCWFR	A TABLE INDEXED BY BUS NUMBER CONTAINING THE PMU FORMAT SELECT BCE PROGRAM COUNTER VALUES (RESIDES IN FIOADCNS)		205	A(25),AC	
Y058	PMU_READ_BITE_PROGRAM_COUNTER_TABLE	FIOPCBIT	A TABLE INDEXED BY BUS NUMBER CONTAINING THE PMU BITE READ BCE PROGRAM COUNTER VALUES (RESIDES IN FIOADCNS)		205	A(25),AC	
Y059	PMU_SM_OI/PL_DATA_RAM_READ_PROGRAM_COUNTER_TABLE	FIOPCOID	A TABLE INDEXED BY BUS NUMBER CONTAINING THE PMU SM OI/PL DATA RAM READ BCE PROGRAM COUNTER VALUES (RESIDES IN FIOADCNS)		205	A(39),AC	
Y060	PMU_TCS_OI/PL_DATA_RAM_READ_PROGRAM_COUNTER_TABLE	FIOPCOII	A TABLE INDEXED BY BUS NUMBER CONTAINING THE PMU TCS OI/PL DATA RAM READ BCE PROGRAM COUNTER VALUES (RESIDES IN FIOADCNS)		205	A(25),AC	
Y061	MTU_WRITE_PROGRAM_COUNTER_TABLE	FIOMTUWT	A TABLE INDEXED BY BUS NUMBEP CONTAINING THE MTU WRITE BCE PROGRAM COUNTER VALUES (RESIDES IN FIOADCNS)		205	A(13),AC	
Y062	MTU_READ_PROGRAM_COUNTER_TABLE	FIOMTURD	A TABLE INDEXED BY BUS NUMBER CONTAINING THE MTU READ BCE		205	A(13),AC	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
			PROGRAM COUNTER VALUES (RESIDES IN FICADCNS)				
Y063	DATA_ACQUISTION_REQUEST_TABLE		A TABLE OF DART ENTRIES SUPPLIED BY SM APPLICATION CONTAINING CONTROL INFORMATION USED BY FCOS TO CONSTRUCT THE BCE CONTROL TABLES FOR A PMU SM DATA RAM READ REQUEST. THE ADDRESS OF THIS TABLE IS PASSED IN THE I/O PARAMETER LIST			A(N),BS(32) N IS SUPPLIED BY THE USER	
Y063.01	DART_RESERVED		BITS 0-3 - NOT USED		205	BS(4)	
Y063.02	DART_RAM_ADDRESS		PMU RAM STARTING ADDRESS		205	BS(12)	
Y063.03	DART_WORD_COUNT		NUMBER OF WORDS TO READ FROM THE RAM (MAXIMUM OF 32)		205	BS(6)	
Y063.04	DART_BUFFER_DISPLACEMENT		DISPLACEMENT FROM I/O BUFFER ADDRESS SPECIFIED IN THE I/O PARAMETER LIST WHERE THE RAM DATA READ IS TO BE PLACED		205	BS(10)	
Y064	WRITE_COMPUTER_DATA_RAM_REQUEST_TABLE		A TABLE SUPPLIED BY THE SM APPLICATION CONTAINING CONTROL INFORMATION USED BY FCOS TO CONSTRUCT THE BCE CONTROL TABLES FOR A PMU WRITE COMPUTER DATA RAM REQUEST				
Y064.01	COMPUTER_DATA_RAM_COMMAND_WORD		COMMAND WORDS REQUIRED BY BCE PROGRAM TO EFFECT DATA TRANSFER		205	A(5),BS(32)	
Y064.02	COMPUTER_DATA_RAM_BUFFER_ADDRESS		ADDRESS OF DATA BUFFERS TO BE TRANSFERRED TO THE PMU		205	A(4),Y	
Y064.3	COMPUTER_DATA_RAM_WORD_COUNT		THE NUMBER OF 16 BIT WORDS MINUS ONE (-1) CONTAINED IN THE ASSOCIATED I/O BUFFER		205	A(4),I	
Y065	MSC_ITERATION_COUNT_BIAS	FIODELTA	BIAS VALUE USED IN COMPUTING THE MSC ITERATION COUNT (RESIDES IN FCMCBLKS)		205	I,INIT(41) UM(16 MICRO-SECONDS)	
Y066	SYNC_TIMEOUT_SAVE_AREA	FIOSYNCT	SAVE AREA USED BY FICPDISP TO SAVE THE STANDARD SYNC TIMEOUT VALUE FOR SSIP ICC REQUESTS (RESIDES IN FCMCBLKS)	205		ID,INIT(0)	
Y067	SSIP_SYNC_TIMEOUT_COUNTER	FIOSIPTO	ALTERNATE SSIP SYNC TIMEOUT VALUE FOR ICC REQUESTS		205	ID,INIT(5000)	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
			(RESIDES IN FCMCBLKS)				
Y068	LDB_SKELETON_COMMAND_WORD	FIODLCMW	SKELETON COMMAND WORD FOR LOB I/O REQUESTS (RESIDES IN FCMCBLKS)		205	BS(32)	
Y069	MTU_READ_BCE_JITTER_BIAS	FPMKBIAS	BCE JITTER CORRECTION FOR MTU READS (RESIDES IN FCMCBLKS)		205	ID,INIT(849) UM(MICROSECONDS)	
Y070	FA_INPUT_BUFFER	CGEB_FA1_DATA$(1,1)(TFCMFAI1) CGE3_FA1_DATA$(2,1)(TFCMFAI2) CGEB_FA1_DATA$(3,1)(TFCMFAI3) CGEB,FA1_DATA$(4,1)(TFCMFAI4)	&1 IN GNC COMPOOL	267		A(4),I	
Y071	FA_INPUT_ATTACK_POINTS_VOLTS_BUFFER	CGEB_FA2_DATA$(1,1)(TFCMFAI5) CGEB_FA2_DATA$(2,1)	&52 IN GNC COMPOOL	267		A(2),I	
Y072	FA_OUTPUT_BUFFER	CGEB_FA_DATA$(1,1)(TFCMFAO1) CGEB_FA_DATA$(2,1)(TFCMFAO2) CGEB_FA_DATA$(3,1)(TFCMFAO3) CGEB_FA_DATA$(4,1)(TFCMFAO4)	&234 IN GNC COMPOOL		267	A(4),I	
Y073	FF_INPUT_1_BUFFER	CGEB_FF1_DATA$(1,1)(TFCMFI11) CGEB_FF1_DATA$(2,1)(TFCMFI12) CGE3_FF1_DATA$(3,1)(TFCMFI13) CGEB_FF1_DATA$(4,1)(TFCMFI14)	&65 IN GNC COMPOOL	267		A(4),I	
Y074	FF_INPUT_2_DISCRETES	CGE3_FF2_DISC_WRD1$(1,)(TFCMDIS1) CGEB_FF2_DISC_WRD1$(2,)(TFCMDIS2) CGE3_FF2_DISC_WRD1$(3,)(TFCMDIS3) CGEB_FF2_DISC_WRD1$(4,)(TFCMDIS4)	IN GNC COMPOOL	267		A(4),BS(16)	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
Y075	FF_INPUT_2_TACAN	CGEB_FF2_DATA$(1,1:1)(TFCMFI21) CGEB_FF2_DATA$(2,1:1)(TFCMFI22) CGEB_FF2_DATA$(3,1:1)(TFCMFI23) CGEB_FF2_DATA$(4,1:1)(TFCMFI24)	&146 IN GNC COMPOOL	267		A(14),BS(16)	
Y076	FF_INPUT_2_ADTA	CGEB_ADTA_MODE_STAT$(1:)(TFCMADT1) CGEB_ADTA_MODE_STAT$(2:)(TFCMADT2) CGEB_ADTA_MODE_STAT$(3:)(TFCMADT3) CGEB_ADTA_MODE_STAT$(4:)(TFCMADT4)	&152 IN GNC COMPOOL	267		A(4),BS(16)	
Y077	FF_INPUT_2_MSBLS	CGEB_MSBLS_AZ$(1,)(TFCMSBL1) CGEB_MSBLS_AZ$(2,)(TFCMSBL2) CGEB_MSBLS_AZ$(3,)(TFCMSBL3)	&222 IN GNC COMPOOL	267		A(3),BS(16)	
Y078	FF_INPUT_2_IMU	CGEB_BITE$(1,)(TFCMIMU1) CGEB_BITE$(2,)(TFCMIMU2) CGEB_BITE$(3,)(TFCMIMU3)	&180 IN GNC COMPOOL	267		A(3),BS(16)	
Y079	FF_OUTPUT_BUFFER	CGEB_SPI_CMD$(1,)(TFCMFF01) CGEB_SPI_CMD$(2,)(TFCMFF02) CGEB_SPI_CMD$(3,)(TFCMFF03) CGEB_SPI_CMD$(4,)(TFCMFF04)	&269 IN GNC COMPOOL		267	A(4),!	
Y080	NOSEWHEEL_STEERING	CGEB_NSWHEL_CMD(TFCMFNST)	&310 IN GNC COMPOOL		264	!	
Y081	PFI_PAYLOAD_DISCRETES	PFI_OUT TFCMPFD1	(IN SM COMPOOL) PAYLOAD DISCRETES OUTPUT BUFFERS		265	ST(1) A(8),BS(16)	

ID	ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
YC82	PF2_PAYLOAD_DISCRETES	PF2_OUT TFCMPFD1	(IN SM COMPOOL) PAYLOAD DISCRETES OUTPUT BUFFERS	265	ST(1) A(8),AS(16)		
Y083	APU_OUTPUT_BUFFER	DLA_APU_WRITE TFCMAPUT	IN SM COMPOOL	264	ST(3),I		
Y086	DEU_FILL_COMMAND_WORD	FICPDFLC	SKELETON COMMAND WORD FOR DEU FILL REQUESTS (RESIDES IN ECMCBLKS)	205	AS(32)		

IO ITEM	ASSEMBLER NAME	DESCRIPTION	SRC	DST	ATTRIBUTES
YO82 PF2_PAYLOAD_DISCRETES	PF2_OUT_TFCMPFC1	(IN SM COMPOOL) PAYLOAD DISCRETES OUTPUT BUFFERS		265	ST(1) A(8),BS(16)
YO83 APU_OUTPUT_BUFFER	DLA_APU_WRITE_TFCMAPUI	IN SM COMPOOL		264	ST(3),I

OPS Dependent Modules

Appendix F

(To Be Provided)

Appendix G

I/O Tables

DEVICE	DEVICE ID	OP CODE	SVC NO.	SYNC TYPE	COMFAULT STATUS	ICC TYPE	TIME TAG	WAIT	PROT.	STATUS	INPUT OUT-PUT	MAJ. FUNC. ID	I/O PRIOR-ITY	WORD COUNT	BUFFER	EVENT	BUS NOMINAL
IPR VIA ICC (Redundant Set)	1	N/A	N/A	RS	NO	YES	NO	NO	YES	NO	N/A		225	7	FIOIBUF1 FIOIBUF2 FIOIBUF3 FIOIBUF4		1 2 3 4
IPR VIA ICC (Common Set)	2	N/A	N/A	CS	NO	YES	NO	NO	YES	NO	N/A		255	7	FIOIBUF1 FIOIBUF2 FIOIBUF3 FIOIBUF4		1 2 3 4
SSIP ICC	3	N/A	24	CS	NO	YES	NO	YES	YES	YES	N/A		225	8,16, 32,64, or 128	CZ2VICC1 CZ2VICC2 CZ2VICC3 CZ2VICC4		1 2 3 4
DATA INITIALI-ZATION VIA ICC	4	N/A	N/A	CS	NO	YES	NO	YES	YES	YES	N/A		225	0			1-4
DEU1	5	1(FILL) 2(POLL) 3(KYBD) 4(DUMP) 5(RBS) 6(RSPL)	24 24 24 24 24 24	RS RS RS RS RS RS	NO NO NO NO NO NO	NO NO NO NO NO NO	NO NO NO NO NO NO	YES YES YES YES YES YES	NO YES YES NO YES NO	YES YES YES YES YES YES	O I I I I O		160 160 160 130 160 160	1 42 5 0	FIODBF1P FIODBF1P FIODBF1B		6 6 6 6 6 6
DEU2	6	1(FILL) 2(POLL) 3(KYBD) 4(DUMP) 5(RBS) 6(RSPL)	24 24 24 24 24 24	RS RS RS RS RS RS	NO NO NO NO NO NO	NO NO NO NO NO NO	NO NO NO NO NO NO	YES YES YES YES YES YES	NO YES YES NO YES NO	YES YES YES YES YES YES	O I I I I O		160 160 160 130 160 160	1 42 5 0	FIODBF2P FIODBF2P FIODBF2B		7 7 7 7 7 7
DEU3	7	1(FILL) 2(POLL) 3(DYBD) 4(DUMP) 5(RBS) 6(RSPL)	24 24 24 24 24 24	RS RS RS RS RS RS	NO NO NO NO NO NO	NO NO NO NO NO NO	NO NO NO NO NO NO	YES YES YES YES YES YES	NO YES YES NO YES NO	YES YES YES YES YES YES	O I I I I O		160 160 160 130 160 160	1 42 5 0	FIODBF3P FIODBE3P FIODBF3B		8 8 8 8 8 8
DDU	8	1	40	RS	NO	NO	NO	YES	NO	YES	O		235	36 36	FIODDUH1 FIODDUH2		10-13 10-13

DEVICE	DEVICE ID	OP CODE	SVC NO.	SYNC TYPE	COMFAULT STATUS	ICC TYPE	TIME TAG	WAIT	PROT.	STATUS	INPUT OUT-PUT	MAJ. FUNC. ID	I/O PRIOR-ITY	WORD COUNT	BUFFER	EVENT	BUS NOMINAL
PMU	9	1(WRITE CMPTR DATA RAM)	24	RS	NO	NO	NO		NO		O		200				24
		2(WRT 128 KBPS PGM)	24	RS	NO	NO	NO		NO		O		180	512			24
		3(WRT 64 KBPS PGM	24	RS	NO	NO	NO		NO		O		180	512			24
		4(READ 128 KBPS PGM)	24	RS	NO	NO	NO		NO		I		190	512			24
		5(READ 64 KBPS PGM	24	RS	NO	NO	NO		YES		I		190	512			24
		6(BITE READ)	24	RS	NO	NO	NO		YES		I		180	1			24
		7(HARD FMT SEL)	24	RS	NO	NO	NO		NO		O		150	0			24
		8(PROG FMT SEL)	24	RS	NO	NO	NO		NO		O		150	0			24
		9(SM RDS OI/PL RAM)	24	RS	NO	NO	NO		YES		I		235				24
		10(TCS RDS OI/PL RAM)	24	RS	NO	NO	NO		YES		I		235				24

DEVICE	DEVICE ID	OP CODE	SVC NO.	SYNC TYPE	COMFAULT STATUS	ICC TYPE	TIME TAG	WAIT	PROT.	STATUS	INPUT OUT-PUT	MAJ. FUNC. ID	I/O PRIOR-ITY	WORD COUNT	BUFFER	EVENT	BUS NOMINAL
MMU	10	1(WRT W/CKSUM)	24	RS	NO	NO	NO		NO		O		N/A				18,19
		2(WRT W/O CKSUM)	24	RS	NO	NO	NO		NO		O		N/A				18,19
		3(RDS W/CKSUM)	24	RS	NO	NO	NO		YES		I		N/A				18,19
		4(RDS W/O CKSUM)	24	RS	NO	NO	NO		YES		I		N/A				18,19
		5(MM UTILITY WRITE)	24	CS	NO	NO	NO		NO		O		N/A				18,19
FF & FA BITE ACQ	11	2	40	RS	NO	NO	NO	NO	YES	NO	I		250	1	CZ2BTFF1	NO	10
														1	CZ2BTFF2		11
														1	CZ2BTFF3		12
														1	CZ2BTFF4		13
														1	CZ2BTFA1		14
														1	CZ2BTFA2		15
														1	CZ2BTFA3		16
														1	CZ2BTFA4		17
FCINPUT1 (FF AND FA)	12	2	40	RS	YES	NO	NO	YES	YES	YES	I		250	20	TFCMFI11	NO	10
														20	TFCMFI12		11
														20	TFCMFI13		12
														20	TFCMFI14		13
														15	TFCMFAI1		14
														15	TFCMFAI2		15
														15	TFCMFAI3		16
														8	TFCMFAI4		17
FCINPUT2 (FF AND FA)	13	2	40	RS	YES	NO	YES	YES	YES	YES	I		250	7	TFCMDIS1	NO	10
														8	TFCMFI21		11
														6	TFCMADT1		
														3	TFCMSBL1		
														14	TFCMMU1		
														7	TFCMDIS2		
														8	TFCMFI22		
														6	TFCMADT2		
														3	TFCMSBL2		
														14	TFCMMU2		
														7	TFCMDIS3		12
														8	TFCMFI23		

DEVICE	DEVICE ID	OP CODE	SVC NO.	SYNC TYPE	COMFAULT STATUS	ICC TYPE	TIME TAG	WAIT	PROT.	STATUS	INPUT OUT-PUT	MAJ FUNC. ID	I/O PRIOR-ITY	WORD COUNT	BUFFER	EVENT	BUS NOMINAL
FCINPUT2 (FF AND FA)	13	2	40	RS	YES	NO	YES	YES	YES	YES	I		250	6 3 14 7 8 6 7 7	TFCMAUT3 TFCMSBL3 TFCMIMU3 TFCMDIS4 TFCMF124 TFCMADT4 TFCMFAI5 TFCMFAI6	NO	12 13 14 15
FCOUTPUT1 (FF AND FA)	14	1	40	RS	NO	NO	NO	NO	NO	NO	O		250	13 12 12 4 16 16 16 14	TFCMFPO1 TFCMFPO2 TFCMFPO3 TFCMFPO4 TFCMFAO1 TFCMFAO2 TFCMFAO3 TFCMFAO4	NO	10 11 12 13 14 15 16 17
FCOUTPUT2 (DDU AND FA)	15	1	40	RS	NO	NO	NO	NO	NO	NO	O		240	36 36 16 16 16 14	FIODDUB1 FIODDUB2 TFCMFAO1 TFCMFAO2 TFCMFAO3 TFCMFAO4	NO	10-13 14 15 16 17
FAOUTPUT	16	1	40	RS	NO	NO	NO	NO	NO	NO	O		240	16 16 16 14	TFCMFAO1 TFCMFAO2 TFCMFAO3 TFCMFAO4	NO	14 15 16 17
NW STEERING	17	1	40	RS	NO	NO	NO	NO	NO	NO	O		245	1	TFCMFWS1	NO	10
MTU1 (REDUNDANT SET)	18	1(WRT)	24	RS	NO	NO	NO	NO	NO	NO	O		245	4	FPMTUMRU, FPMTUGRS, or FPMTUGUD	NO	10-12
		2(MET RESET)	24	RS	NO	NO	NO	NO	NO	NO	O		245	4	FPMTUMRU	YES	10-12
MTU2 (REDUNDANT SET)	19	1(WRT)	24	RS	NO	NO	NO	NO	NO	NO	O		245	4	FPMTUMRU, FPMTUGRS, or FPMTUGUD	NO	10
		2(MET RESET)	24	RS	NO	NO	NO	NO	NO	NO	O		245	4	FPMTUMTU	YES	11

DEVICE	DEVICE ID	OP CODE	SVC NO.	SYNC TYPE	COMFAULT STATUS	ICC TYPE	TIME TAG	WAIT	PROT.	STATUS	INPUT OUT-PUT	MAJ. FUNC. ID	I/O PRIOR-ITY	WORD COUNT	BUFFER	EVENT	BUS NOMINAL
MTU3 (REDUNDANT SET)	20	1(WRT)	24	RS	NO	NO	NO	NO	NO	NO	O		245	4	FPMTUMRU, FPMTUGRS, or FPMTUGUD	NO	12
		2(MET RESET)	24	RS	NO	NO	NO	NO	NO	NO	O		245	4	FPMTUMPU	YES	10-12
MTUALL(COMMON SET)	21	2	24	CS	YES	NO	NO	NO	YES	YES	I		255	7 7 7	TFCMMTU1 TFCMMTU2 TFCMMTU3	YES	10 11 12
TCS FF1	22	1	24	RS	NO	NO	NO		NO		O		210				10
		2	24	RS	NO	NO	NO		YES		I		210				10
TCS FF2	23	1	24	RS	NO	NO	NO		NO		O		210				11
		2	24	RS	NO	NO	NO		YES		I		210				11
TCS FF3	24	1	24	RS	NO	NO	NO		NO		O		210				12
		2	24	RS	NO	NO	NO		YES		I		210				12
TCS FF4	25	1	24	RS	NO	NO	NO		NO		O		210				13
		2	24	RS	NO	NO	NO		YES		I		210				13
TCS FA1	26	1	24	RS	NO	NO	NO		NO		O		210				14
		2	24	RS	NO	NO	NO		YES		I		210				14
TCS FA2	27	1	24	RS	NO	NO	NO		NO		O		210				15
		2	24	RS	NO	NO	NO		YES		I		210				15
TCS FA3	28	1	24	RS	NO	NO	NO		NO		O		210				16
		2	24	RS	NO	NO	NO		YES		I		210				16
TCS FA4	29	1	24	RS	NO	NO	NO		NO		O		210				17
		2	24	RS	NO	NO	NO		YES		I		210				17
PF BITE ACQ	30	2	40	RS	NO	NO	NO	NO	NO	NO	I		210	1	CZ2BTPF1	NO	20
PF1 DISCRETES	31	1	40	RS	NO	NO	NO	NO	NO	NO	O		210	4	CZ2BTPF2 TFCMPFD1	NO	21 20
PF2 DISCRETES	32	1	40	RS	NO	NO	NO	NO	NO	NO	O		210	4	TFCMPFD2	NO	21
PF2 APU	33	1	40	RS	NO	NO	NO	NO	NO	NO	O		210	3	TFCMAPU1	NO	21
TCS PF1	34	1	24	RS	NO	NO	NO		NO		O		210				20
		2	24	RS	NO	NO	NO		YES		I		210				20
TCS PF2	35	1	24	RS	NO	NO	NO		NO		O		210				21
		2	24	RS	NO	NO	NO		YES		I		210				21
LDB	36	1(INT W/O DATA 2(INT W/ DATA) 3(GO AHEAD) 4(TRANS. ENABLE)	24	RS	NO	NO	NO	NO	YES		I		230	1	TFCMLI11		22,23
			24	RS	NO	NO	NO		YES		I		230				22,23
			24	RS	NO	NO	NO		YES		I		230				22,23
			24	RS	NO	NO	NO		YES		O		230				22,23

DEVICE	DEVICE ID	OP CODE	SVC NO.	SYNC TYPE	COMFAULT STATUS	ICC TYPE	TIME TAG	WAIT	PROT.	STATUS	INPUT OUT-PUT	MAJ FUNC. ID	I/O PRIOR. ITY	WORD COUNT	BUFFER	EVENT	BUS NOMINAL
LDB	36	5(STATUS REQUEST)	24	RS	NO	NO	NO		YES		I		230	1	TFCMLSR1		22,23
		6(STATUS)	24	RS	NO	NO	NO		YES		O		230				22,23
TCS LF1	37	1	24	RS	NO	NO	NO		NO		O		210				22
		2	24	RS	NO	NO	NO		YES		I		210				22
TCS LA1	38	1	24	RS	NO	NO	NO		NO		O		210				23
		2	24	RS	NO	NO	NO		YES		I		210				23

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
IPR via ICC (Redundant Set)	1		1		00101 00000 00000 00000 00110	FIOIBUF1			7	IPR - Transmit and receive 7 words
			2			FIOIBUF2				
			3			FIOIBUF3				
						FIOIBUF4				
IPR via ICC (Common Set)	2		1		00101 00000 0000 00000 00110	FIOIBUF1			7	IPR - Transmit and receive 7 words
			2			FIOIBUF2				
			3			FIOIBUF3				
			4			FIOIBUF4				
SSIP ICC	3		1		00101 00000 0000 00000 00111	CZ2VICC1			8	SSIP - Transmit and receive 8 words
			2			CZ2VICC2				
			3			CZ2VICC3				
			4			CZ2VICC4				
			1		00101 00000 0000 00000 01111	CZ2VICC1			16	SSIP - Transmit and receive 16 words
			2			CZ2VICC2				
			3			CZ2VICC3				
			4			CZ2VICC4				

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
SSIP ICC	3		1		00101 00000 0000 00000 11111	C22VICC1			32	SSIP - Transmit and receive 32 words
			2			C22VICC2				
			3			C22VICC3				
			4			C22VICC4				
			1			C22VICC1			64	SSIP - Transmit and receive 64 words
			2			C22VICC2				
			3			C22VICC3				
			4			C22VICC4				
			1		00101 00000 0000 00011 11111	C22VICC1			128	SSIP - Transmit and receive 128 words
			2			C22VICC2				
			3			C22VICC3				
			4			C22VICC4				
Data Initialization via ICC	4		1-4						0	

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
DEU1	5	1	6		01011 00000 00000 00000 0000	FIODBF1P			1	DEU1 FILL
		2			01011 00001 00000 0000 00000	FIODBF1P			1/2	DEU1 POLL
		3								DEU1 KYBD
		4								DEU1 DUMP
		5			01011 00010 00000 0000 00000	FIODBF1P			5	DEU1 REQ. BITE
										STATUS
		6			01011 00100 00000000000000000				0	DEU1 RESET S.P.L.
DEU2	6	1	7							DEU2 FILL
		2			01011 00000 00000 0000 00000	FIODBF2P			1	DEU2 POLL
		3			01011 00001 00000000000000	FIODBF2P			1/2	DEU2 KYBD
		4								DEU2 DUMP
		5			01011 00010 00000 00000 0000	FIODBF2B			5	DEU2 REQ. BITE
										STATUS
		6			01011001000000000000000000				0	DEU2 RESET S.P.L.
DEU3	7	1	8							DEU3 FILL
		2			01011000000000000000000000	FIODBF3P			1	DEU3 POLL
		3			01011000010000000000000000	FIODBF3P			1/2	DEU3 KYBD
		4								DEU3 DUMP
		5			01011000100000000000000000	FIODBF3B			5	DEU3 REQ. BITE
										STATUS
		6			01011001000000000000000000				0	DEU3 RESET S.P.L.

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
DDU	8	1	10						36	Output to DDU
				1	01101 10000 000000001 01110	FIODDUB1			14	ADI
				2	01101 10000 000001000 00110	FIODDUB1+14			6	AMI
				3	01101 10000 000000100 00110	FIODDUB1+20			6	AVI
				4	01101 10000 000000010 01010	FIODDUB1+26			10	HSI
			10						36	Output to DDU
				5	01110 10000 000000001 01110	FIODDUB2			14	ADI
				6	01110 10000 000001000 00110	FIODDUB2+14			6	AMI
				7	01110 10000 000000100 00110	FIODDUB2+20			6	AVI
				8	01110 10000 000000010 01010	FIODDUB2+26			10	HSI

Device	Device I/D	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
DDU	8	1	11	1	01101 10000 000000001 01110	FIODDUB1			36	Output to DDU
				2	01101 10000 000001000 00110	FIODDUB1+14			14	ADI
				3	01101 10000 000000100 00110	FIODDUB1+20			6	AVI
				4	01101 10000 000000010 01010	FIODDUB1+26			6	AVI
			11	5	01110 10000 000000001 01110	FIODDUB2			10	HSI
				6	01110 10000 000001000 00110	FIODDUB2+14			36	Output to DDU
				7	01110 10000 000000100 00110	FIODDUB2+20			14	ADI
				8	01110 10000 000000010 01010	FIODDUB2+26			6	AVI
									6	AVI
									10	HSI

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
DDU	8	1	12	1	01101 10000 000000001 01110	FIODDUB1			36	Output to DDU
				2	01101 10000 000001000 00110	FIODDUB1+14			14	ADI
				3	01101 10000 000000100 00110	FIODDUB1+20			6	AMI
				4	01101 10000 000000010 01010	FIODDUB1+26			6	AVI
			12						10	HSI
				5	01110 10000 000000001 01110	FIODDUB2			36	Output to DDU
				6	01110 10000 000001000 00110	FIODDUB2+14			14	ADI
				7	01110 10000 000000100 00110	FIODDUB2+20			6	AMI
				8	01110 10000 000000010 01010	FIODDUB2+26			6	AVI
									10	HSI

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
DDU	8	1	13	1	01101 10000 000000001 01110	FIODDUB1			36	Output to DDU
				2	01101 10000 000001000 00110	FIODDUB1+14			14	ADI
				3	01101 10000 000000100 00110	FIODDUB1+20			6	AMI
				4	01101 10000 000000010 01010	FIODDUB1+26			6	AVI
									10	HSI
			13	5	01110 10000 000000001 01110	FIODDUB2			36	Output to DDU
				6	01110 10000 000001000 00110	FIODDUB2+14			14	ADI
				7	01110 10000 000000100 00110	FIODDUB2+20			6	AMI
				8	01110 10000 000000010 01010	FIODDUB2+26			6	AVI
									10	HSI

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
PMU	9	1	24							Write computer data RAM
		2							512	Write 128 KBPS PGM
		3							512	Write 64 KBPS PGM
		4							512	Read 128 KBPS PGM
		5							512	Read 64 KBPS PGM
		6			01111000000000000000				1	Bite read
		7							0	Hard format select
		8							0	Prog. format select
		9								SM RDS OI/PL RAM
		10								TCS RDS OI/PL RAM

Device	Device ID	GP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
MMU	10	1	18 19							Write with checksum
		2	18 19							Write without checksum
		3	18							Read with checksum
			19							
		4	18 19							Read without checksum
		5	18 19							MM Utility write

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FF & FA BITE ACQ	11	2	10		01011 01010 000000000 00000	CZ2BTFF1			1	FF1 Bite Read
			11			CZ2BTFF2			1	FF2 Bite Read
			12			CZ2BTFF3			1	FF3 Bite Read
			13			CZ2BTFF4			1	FF4 Bite Read
			14		01100 01010 000000000 00000	CZ2BTFA1			1	FA1 Bite Read
			15			CZ2BTFA2			1	FA2 Bite Read
			16			CZ2BTFA3			1	FA3 Bite Read
			17			CZ2BTFA4			1	FA4 Bite Read

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCINPUT1 (FF and FA)	12	2	10	1	01011 00010 00001 0000 10011	TFCMF111			20	Input from MDM
										TFO1
							14	6	1	RH RHC CMD A roll
							14	7	1	RH RHC CMD A
										Pitch
							14	1	1	RH RHC CMD A yaw
							1	6	1	LH RHC CMD A Roll
							1	7	1	LH RHC CMD A
										Pitch
							1	1	1	LH RHC CMD A Yaw
							14	5	1	RH SBTC CMD A
							1	5	1	LH SBTC CMD A
							14	3	1	RH RPTA CMD A
							1	3	1	LH RPTA CMD A
							7	1	1	Hydr Sys 1 Sup
										Pres A
							4	0,1,2	3	Discretes
							12	0,1	2	Discretes
							6	0	1	Discretes
							15	0	1	Discretes
							14	0	1	Norm Accel 1
										signal
							14	2	1	Lat accel 1
										signal

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCINPUT1 (FF and FA)	12	2	11	1	01011 00010 00001 0000 10011	TFCMFI12			20	Input from MDM FF 02
							14	6	1	RH RHC CMD B Roll
							14	7	1	RH RHC CMD B Pitch
							14	1	1	RH RHC CMD B Yaw
							1	6	1	LH RHC CMD B Roll
							1	7	1	LH RHC CMD B Pitch
							1	1	1	LH RHC CMD B Yaw
							14	5	1	RH SBTC CMD B
							1	5	1	LH SBT CMD B
							14	3	1	RH RPTA CMD B
							1	3	1	LH RPTA CMD B
							7	1	1	Hydr. Sys 2 Sup Pres A
							4	0,1,2	3	Discretes & Fill
							12	0,1	2	Discretes
							6	0	1	Discretes
							15	0	1	Discretes
							14	0	1	Vorm Accel 2 Signal
							14	2	1	Lat Accel 2 Signal

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCINPUT1 (FF and FA)	12	2	12	1	01011 00010 00001 0000 10011	TFCMFI13			20	Input from MDM FF03
							14	6	1	YH RHC CMD C Roll
							14	7	1	YH RHC CMD C Pitch
							14	1	1	RH RHC CMD C Yaw
							1	6	1	LH RHC CMD C Roll
							1	7	1	LH RHC CMD C Pitch
							1	1	1	LH RHC CMD C Yaw
							14	5	1	RH SBTC CMD C
							1	5	1	LH SBTC CMD C
							14	3	1	RH RPTA CMD C
							7	1	1	Hydr Sys 3 Sup Pres A
							4	0,1,2	3	Discretes
							6	0	1	Discretes
							15	0	1	Discretes
							14	0	1	Norm Accel 3 signal
							14	2	1	Lat Accel 3 signal

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCINPUT1 (FF and FA)	12	2	13	1	01011 00010 00001 0000 10011	TFCMFI14			20	Input from MDM
								6	1	FF04
							14	7	1	Fill
							14	1	1	Fill
							14	6	1	Fill
							1	7	1	Fill
							1	1	1	Fill
							1	5	1	Fill
							14	5	1	Fill
							1	3	1	Fill
							14	3	1	Fill
							1	1	1	Fill
							7	0,1,2	3	Discretes and
							4			Fill
							12	0,1	2	Discretes
							6	0	1	Fill
							15	0	1	Fill
							14	0	1	Fill
							14	2	1	Fill

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCINPUT1 (FF and FA)	12	2	14	1	01100 00010 000010000 01110	TFCMFA11			15	Input from MDM FA01
							1	0	1	Rudder Posn Fdbk - 1
							1	1	1	Speed Brake Posn Fdbk - 1
							1	2	1	Body Flap Posn - 1
							1	3	1	Rt Inbd Elevon Posn Fdbk - 1
							1	5	1	Lt Inbd Elevon Posn Fdbk - 1
							1	6	1	Rt Outbd Elevon Posn Fdbk - 1
							1	7	1	Lt Outbd Elevon Posn Fdbk - 1
							11	0	1	Discretes
							15	0	1	Discretes
							14	0	1	LH Atch Pt Cap Volts - 1A
							14	1	1	RH Atch Pt Cap Volts - 1A
							14	2	1	Hydr Sys 1 Supply Press B
							14	21	1	Gyro 1 Roll Rate
							14	20	1	Gyro 1 Pitch Rate
							14	23	1	Gyro 1 Yaw Rate

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCINPUT1 (FF and FA)	12	2	15	1	00010 00001 0000 01110 01100	TFCMFAI2			15	Input from MDM FA02
							1	0	1	Rudder Posn Fdbk - 2
							1	1	1	Speed Brake Posn Fdbk - 2
							1	2	1	Body Flap Posn - 2
							1	3	1	Rt Inbd Elevon Posn Fdbk - 2
							1	5	1	Lt Inbd Elevon Posn - 2
							1	6	1	Rt Outbd Elevon Posn - 2
							1	7	1	Lt Outbd Elevon Posn - 2
							11	0	1	Discretes
							5	0	1	Discretes
							14	0	1	LH Atch Pt Cap Volts - 1B
							14	1	1	RH Atch Pt Cap Volts - 1B
							14	2	1	Sydr Sys 2 Supply Press B
							14	21	1	Gyro 2 Roll Rate
							14	20	1	Gyro 2 Pitch Rate
							14	23	1	Gyro 2 Yaw Rate

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCINPUTL (FF And FA)	12	2	16	1	00010 00001 0000 01110 01100	TFCMPAI3			15	Input from MDM FA03
							1	0	1	Rudder Posn Fdbk - 3
							1	1	1	Speed Brake Posn Fdbk - 3
							1	2	1	Body Flap Posn - 3
							1	3	1	Rt Inbd Elevon Posn Fdbk - 3
							1	5	1	Lt Inbd Elevon Posn Fdbk - 3
							1	6	1	Rt Outbd Elevon Posn Fdbk - 3
							1	7	1	Lt Outbd Elevon Posn Fdbk - 3
							11	0	1	Discretes
							5	0	1	Fill
							14	0	1	Fill
							14	1	1	Hydr Sys 3 Supply Press B
							14	2	1	Gyro 3 Roll Rate
							14	21	1	Gyro 3 Pitch Rate
							14	20	1	Gyro 3 Yaw Rate
							14	23	1	

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCINPUT1 (FF and FA)	12	2	17	1	01100 0010 000010000 0011	TFCMFAI4			8	Input from MDM FA04
							1	0	1	Rudder Posn Fdbk-4
							1	1	1	Speed Brake Posn Fdbk-4
							1	1	1	Speed Brake Posn Fdbk-4
							1	2	1	Body Flax Posn-4
							1	3	1	Rt dnbd Elevon Posn Fdbk-4
							1	5	1	Lt dnbd Eleven Posn Fdbk-4
							1	6	1	Rt outbd Eleven Posn Fdbk-4
							1	7	1	Lt Outbd Eleven Posn Fdbk-4
							11	0	1	Discretes

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCINPUT2 (FF and FA)	13	2	10	1	01011 00010 000011011 00110	TFCMDIS1	4	0,1,2	7	Input from MDM FF01
							12	0,1	3	Discretes
							6		2	Discretes
							15	0	1	Discretes
									1	Discretes
				2	01011 00010 000101010 00111	TFCMFI21			8	Input from MDM FF01
							7	0	1	Fwd Atch Pt Cap
							0	0	1	Volts - A
										TACAN 1 Bearing Word A
							0	1	1	TACAN 1 Bearing Word B
							0	2	1	TACAN 1 Range Word A
							0	3	1	TACAN 1 Range Word B
							0	4	1	RADAR ALT 1 Parent Word
							0	5	1	MDM FF01 Dscr In
							0	6	1	MDM FF01 Dscr In
				3	01011 00010 000110010 00000	TFCMADT1	11	1	6	ADTA 1
				4	01011 00010 000110100 00000	TFCMSBL1	3	0	3	MSBLS 1
				5	01011 00010 000110011 00000	TFCMIMU1	11	0	14	IMU 1

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCINPUT2 (FF and FA)	13	2	11	1	01011 00010 000011011 00110	TFCMDIS2			7	Input from MDM FFO2
							4	0,1,2	3	Discretes and Fill
							12	0,1	2	Discretes
							6	0	1	Discretes
							15	0	1	Discretes
				2	01011 00010 000101010 00111	TFCMFI22			8	Input from MDM FFO2
							7	0	1	Fill
							0	0	1	TACAN 2 Bearing Word A
							0	1	1	TACAN 2 Bearing Word B
							0	2	1	TACAN 2 Range Word A
							0	3	1	TACAN 2 Range Word B
							0	4	1	RADAR ALT 2 Parent Wor1
							0	5	1	MDM FFO2 Dscr In
							0	6	1	MDM FFG2 Dscr In
				3	01011 00010 000110010 00000	TFCMADT2	11	1	6	ADTA 2
				4	01011 00010 000110100 00000	TFCMSBL2	3	0	3	MSBLS 2
				5	01011 00010 000110011 00001	TFCMIMU2	11	0	14	IMU 2

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCINPUT2 (FF and FA)	13	2	12	1	01011 00010 000011011 00110	TFCMDIS3	4	0,1,2	7	Input from MDM FF03
							12	0,1	3	Discretes
							6	0	2	Discretes
							15	0	1	Discretes
									1	Discretes
				2	01011 00010 000101010 00111	TFCMF123			8	Input from MDM
										FF03
							7	0	1	Fwd Atch Pt Cap
										Volts-B
							0	0	1	TACAN 3 Bearing
										Word A
							0	1	1	TACAN 3 Bearing
										Word B
							0	2	1	TACAN 3 Range
										Word A
							0	3	1	TACAN 3 Range
										Word B
							0	4	1	Fill
							0	5	1	MDM FF03 Descr In
							0	6	1	MDM FF03 Descr In
				3	01011 00010 000110010 00000	TFCMADT3	11	1	6	ADTA 3
				4	01011 00010 000110100 00000	TFCMSBL3	3	0	3	MSBLS 3
				5	01011 00010 000110011 00000	TFCMIMU3	11	0	14	IMU 3

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCINPUT2 (FF and FA)	13	2	13	1	01011 00010 000011011 00110	TFCMDIS4			7	Input from MDM
										FFO4
							4	0,1,2	3	Discretes and Fill
							12	0,1	2	Discretes
							6	0	1	Fill
							15	0	1	Fill
				2	01011 00010 000101010 00111	TFCMFI24			8	Input from MDM
							7	0	1	FFO4
							0	0	1	Fill
							0	1	1	Fill
							0	2	1	Fill
							0	3	1	Fill
							0	4	1	Fill
							0	5	1	Fill
							0	6	1	Fill
				3	01011 00010 000110010 00000	TFCMADT4	11	1	6	ADTA 4

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCINPUT2 (FF and FA)	13	2	14	1	01100 01001 111000000 00110	TFCMFAI5	14	0	7	Input from MDM
									1	LH Atch Pt Cap Volts-1A
							14	1	1	RH Atch Pt Cap Volts-1A
							14	2	1	Hydr Sys 1 Supply Press B
							14	3	1	LH Atch Pt Cap Volts-2A
							14	4	1	RH Atch Pt Cap Volts-2A
							14	5	1	LH Atch Pt Cap Volts-3A
							14	6	1	RH Atch Pt Cap Volts-3A
	13	2	15	1	01100 01001 111000000 00110	TFCMFAI6			7	Input from MDM FA02
							14	0	1	LH Atch Pt Cap Volts-1B
							14	1	1	RH Atch Pt Cap Volts-1B
							14	2	1	Hydr Sys 2 Supply Press 3
							14	3	1	LH Atch Pt Cap Volts-2B
							14	4	1	RH Atch Pt Cap Volts-2B
							14	6	1	RH Atch Pt Cap Volts-2B

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCOUTPUT1	14	1	10						13	Output to MDM
(FF and FA)										FF01
				1	01011 01000 100001010 00000	TFCMFF01	8	10	1	Write Speed Brake Posn Cmd (SPI)
				2	01011 01000 001010000 00000	TFCMFF01+2	2	0	1	Discretes Set
				3	01011 01000 001000000 00000	TFCMFF01+4	2	0	1	Discretes Reset
				4	01011 01000 101010000 00000	TFCMFF01+6	10	0	1	Discretes Set
				5	01011 01000 101000000 00000	TFCMFF01+8	10	0	1	Discretes Reset
				6	01011 01000 010110000 00000	TFCMFF01+10	5	0	1	Discretes Set
				7	01011 01000 010100000 00000	TFCMFF01+12	5	0	1	Discretes Reset
				8	01011 01000 110110000 00000	TFCMFF01+14	13	0	1	Discretes Set
				9	01011 01000 110100000 00000	TFCMFF01+16	13	0	1	Discretes Reset
				10	01011 01000 000000110 00001	TFCMFF01+18	0	6	2	TACAN
				11	01011 01000 100000110 00001	TFCMFF01+20	11	0	2	IMU

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCOUTPUT1 (FF and FA)	14	1	11						12	Output to MDM FF02
				1	01011 01000 001010000 00000	TFCMFF02+2	2	0	1	Discretes Set
				2	01011 01000 001000000 00000	TFCMFF02+4	2	0	1	Discretes Reset
				3	01011 01000 101010000 00000	TFCMFF02+6	10	0	1	Discretes Set
				4	01011 01000 101000000 00000	TFCMFF02+8	10	0	1	Discretes Reset
				5	01011 01000 010110000 00000	TFCMFF02+10	5	0	1	Discretes Set
				6	01011 01000 010100000 00000	TFCMFF02+12	5	0	1	Discretes Reset
				7	01011 01000 110110000 00000	TFCMFF02+14	13	0	1	Discretes Set
				8	01011 01000 110100000 00000	TFCMFF02+16	13	0	1	Discretes Reset
				9	01011 01000 000000110 00000	TFCMFF02+18	0	6	2	TACAN
				10	01011 01000 100000110 00001	TFCMFF02+20	11	0	2	IMU

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCOUTPUT1 (FF and FA)	14	1	12						12	Output to MDM
				1	01011 01000 001010000 00000	TFCMFF03+2	2	0	1	FF03
				2	01011 01000 001000000 00000	TFCMFF03+4	2	0	1	Discretes Set
				3	01011 01000 101010000 00000	TFCMFF03+6	10	0	1	Discretes Reset
				4	01011 01000 101000000 00000	TFCMFF03+8	10	0	1	Discretes Set
				5	01011 01000 010110000 00000	TFCMFF03+10	5	0	1	Discretes Reset
				6	01011 01000 010100000 00000	TFCMFF03+12	5	0	1	Discretes Set
				7	01011 01000 110110000 00000	TFCMFF03+14	13	0	1	Discretes Reset
				8	01011 01000 110100000 00000	TFCMFF03+16	13	0	1	TACAN
				9	01011 01000 000000110 00001	TFCMFF03+18	0	6	2	IMU
				10	01011 01000 100000110 00001	TFCMFF03+20	11	0	2	

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCOUTPUT1 (FF and FA)	14	1	13						4	Output to MDM FF04
				1	01011 01000 101010000 00000	TFCMFF04+6	10	0	1	Discretes Set
				2	01011 01000 101000000 00000	TFCMFF04+8	10	0	1	Discretes Reset
				3	01011 01000 110110000 00000	TFCMFF04+14	13	0	1	Discretes Set
				4	01011 01000 110100000 00000	TFCMFF04+16	13	0	1	Discretes Reset

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCOUTPUT1 (FF and FA)	14	1							16	Output to MDM FA01
				1	01100 01000 000000001 01011	TFCMFA01	0	1	12	ASA
				2	01100 01000 001010000 00000	TFCMFA01+12	2	0	1	Discrete Set
				3	01100 01000 001000000 00000	TFCMFA01+14	2	0	1	Discrete Reset
				4	01100 01000 101010000 00000	TFCMFA01+16	10	0	1	Discrete Set
				5	01100 01000 101000000 00000	TFCMFA01+18	10	0	1	Discrete Reset

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCOUTPUT1 (FF and FA)	14	1	15	1	01100 01000 000000001 01011	TFCMFA02	0	1	16	Output to MDM FA02
				2	01100 01000 001010000 00000	TFCMFA02+12	2	0	12	AGA
				3	01100 01000 001000000 00000	TFCMFA02+14	2	0	1	Discrete Set
				4	01100 01000 101010000 00000	TFCMFA02+16	10	0	1	Discrete Reset
				5	01100 01000 101000000 00000	TFCMFA02+18	10	0	1	Discrete Set
										Discrete Reset

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCOUTPUT1 (FF and FA)	14	1	16	1	01100 01000 000000001 01011	TFCMFA03	0	1	16	Output to MDM FA03
				2	01100 01000 001010000 00000	TFCMFA03+12	2	0	12	ASA
				3	01100 01000 001000000 00000	TFCMFA03+14	2	0	1	Discrete Set
				4	01100 01000 101010000 00000	TFCMFA03+16	10	0	1	Discrete Set
				5	01100 01000 101000000 00000	TFCMFA03+18	10	0	1	Discrete Reset

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCOUTPUT1 (FF and FA)	14	1	17	1	01100 01000 000000001 01011	TFCMFA04	0	1	14	Output to MDM FA04
				2	01100 01000 001010000 00000	TFCMFA04+12	2	0	12	ASA
				3	01100 01000 001000000 00000	TFCMFA04+14	2	0	1	Discrete Set
									1	Discrete Reset

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCOUTPUT2 (DDU and FA)	15	1	10	1	01101 10000 000000001 01110	FIODDUB1			36	Output to DDU
				2	01101 10000 000001000 00110	FIODDUB1+14			14	ADI
				3	01101 10000 000000100 00110	FIODDUB1+20			6	A/E
				4	01101 10000 000000010 01010	FIODDUB1+26			6	AVI
			10						10	HSI
				5	01110 10000 000000001 01110	FIODDUB2			36	Output to DDU
				6	01110 10000 000001000 00110	FIODDUB2+14			14	ADI
				7	01110 10000 000000100 00110	FIODDUB2+20			6	A/E
				8	01110 10000 000000010 01010	FIODDUB2+26			6	AVI
									10	HSI

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCOUTPUT2 (DDU AND FA)	15	1	11	1	0110110000000000001011l0	FIODDUB1			36	Output to DDU
				2	0110110000000000100000110	FIODDUB1 + 14			14	ADI
				3	0110110000000000110000110	FIODDUB1 + 20			6	AMI
				4	0110110000000000101001010	FIODDUB1 + 26			6	AVI
									10	HSI
			11	5	0111010000000000001011l0	FIODDUB2			36	Output to DDU
				6	0111010000000000100000110	FIODDUB2 + 14			14	ADI
				7	0111010000000000110000110	FIODDUB2 + 20			6	AMI
				8	0111010000000000101001010	FIODDUB2 + 26			6	AVI
									10	HSI

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCOUTPUT2 (DDU AND FA)	15	1	12						36	Output to DDU
				1	0110110000000000001011110	FIODDUB1			14	ADI
				2	0110110000000000100000110	FIODDUB1 + 14			6	AMI
				3	0110110000000000010000110	FIODDUB1 + 20			6	AVI
				4	0110110000000000001001010	FIODDUB1 + 26			10	HSI
			12							
				5	0111010000000000001011110	FIODDUB2			36	Output to DDU
				6	0111010000000000100000110	FIODDUB2 + 14			14	ADI
				7	0111010000000000010000110	FIODDUB2 + 20			6	AMI
				8	0111010000000000001001010	FIODDUB2 + 26			6	AVI
									10	HSI

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCOUTPUT2 (DDU and FA)	15	1	13						36	Output to DDU
				1	01101 10000 000000001 01110	FIODDUB1			14	ADI
				2	01101 10000 000001000 00110	FIODDUB1+14			6	AMI
				3	01101 10000 000000100 00110	FIODDUB1+20			6	AVI
				4	01101 10000 000000010 01010	FIODDUB1+26			10	HSI
			13						36	Output to DDU
				5	01110 10000 000000001 01110	FIODDUB2			14	ADI
				6	01110 10000 000001000 00110	FIODDUB2+14			6	AMI
				7	01110 10000 000000100 00110	FIODDUB2+20			6	AVI
				8	01110 10000 000000010 01010	FIODDUB2+26			10	HSI

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCOUTPUT2 (DDU and FA)	15	1	14	1	01100 01000 000000001 01011	TFCMFA01	0	1	16	Output to MDM FA01
				2	01100 01000 001010000 00000	TFCMFA01+12	2	0	12	ASA
				3	01100 01000 001000000 00000	TFCMFA01+14	2	0	1	Discrete Set
				4	01100 01000 101010000 00000	TFCMFA01+16	10	0	1	Discrete Set
				5	01100 01000 101000000 00000	TFCMFA01+18	10	0	1	Discrete Set

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
PCOUTPUT2 (DDU and FA)	15	1	15						16	Output to MDM FA02
				1	01100 01000 000000001 01011	TPCMFA02	0	1	12	ASA
				2	01100 01000 001010000 00000	TPCMFA02+12	2	0	1	Discrete Set
				3	01100 01000 001000000 00000	TPCMFA02+14	2	0	1	Discrete Reset
				4	01100 01000 101010000 00000	TPCMFA02+16	10	0	1	Discrete Set
				5	01100 01000 101000000 00000	TPCMFA02+18	10	0	1	Discrete Reset

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCOUTPUT2 (DDU and FA)	15	1	16	1	01100 01000 0000000001 01011	TFCMFA03	0	1	16	Output to MDM FA03
				2	01100 01000 0010010000 00000	TFCMFA03+12	2	0	12	ASA
				3	01100 01000 0010000000 00000	TFCMFA03+14	2	0	1	Discrete Set
				4	01100 01000 1010100000 00000	TFCMFA03+16	10	0	1	Discrete Reset
				5	01100 01000 1010000000 00000	TFCMFA03+18	10	0	1	Discrete Set
										Discrete Reset

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FCOUTPUT2 (DDU and FA)	15	1	17						14	Output to MDM FA04
				1	01100 01000 000000001 01011	TFCMFA04	0	1	12	ASA
				2	01100 01000 001010000 00000	TFCMFA04+12	2	0	1	Discrete Set
				3	01100 01000 001000000 00000	TFCMFA04+14	2	0	1	Discrete Reset

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FAOUTPUT	16	1	14						16	Output to MDM FA01
				1	01100 01000 0000000001 01011	TFCMFA01	0	1	12	ASA
				2	01100 01000 0010100000 00000	TFCMFA01+12	2	0	1	Discrete Set
				3	01100 01000 0010000000 00000	TFCMFA01+14	2	0	1	Discrete Reset
				4	01100 01000 1010100000 00000	TFCMFA01+16	10	0	1	Discrete Set
				5	01100 01000 1010000000 00000	TFCMFA01+18	10	0	1	Discrete Reset

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FAOUTPUT	16	1	15	1	01100 01000 000000001 01011	TFCMFA02	0	1	16	Output to MDM FA02
				2	01100 01000 001010000 00000	RFCMFA02+12	2	0	12	ASA
				3	01100 01000 001000000 00000	TFCMFA02+14	2	0	1	Discrete Set
				4	01100 01000 101010000 00000	TFCMFA02+16	10	0	1	Discrete Reset
				5	01100 01000 101000000 00000	TFCMFA02+18	10	0	1	Discrete Set Discrete Reset

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FAOUTPUT	16	1	16	1	01100 01000 000000001 01011	TFCMFA03	0	1	16	Output to MDM FA03
				2	01100 01000 001010000 00000	TFCMFA03+12	2	0	12	ASA
				3	01100 01000 001000000 00000	TFCMFA03+14	2	0	1	Discrete Set
				4	01100 01000 101010000 00000	TFCMFA03+16	10	0	1	Discrete Reset
				5	01100 01000 101000000 00000	TFCMFA03+18	10	0	1	Discrete Set
										Discrete Reset

Device	Device ID	OP Code	BCF Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
FAOUTPUT	16	1	17	1	01100 01000 000000001 01011	TFCMFAO4	0	1	14	Output to MDM FA04
				2	01100 01000 001010000 00000	TFCMFA04+12	2	0	12	ASA
				3	01100 01000 001000000 00000	TFCMFA04+14	2	0	1	Discrete Set
									1	Discrete Reset

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
NW Steering	17	1	10		01011 01000 100000110 00000	TPCMFNS1	8	6	1	Nose Wheel Steering Computer CMD

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
MTU1 (Redundant Set)	18	1	10		01011 01000 001100001 00011	FPMTUMRU, FPMTUGRS, or FPMTUGUD	3	1	4	Write to MTU 1
		2	10		01011 01000 001100001 00011	FPMTUMRU	3	1	4	Met Reset

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
MTU2 (Redundant Set)	19	1	11		01011 01000 001100001 00011	FPMTUMRU, FPMTUGRS, or FPMTUGUD	3	1	4	Write to MTU 2
		2	11		01011 01000 001100001 00011	FPMTUMRU	3	1	4	Set Reset

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
MTU3 (Reduandant Set)	20	1	12		01011 01000 001100001 00011	FPMTURMRU, FPMTUGRS, Or FPMTUGUD	3	1	4	Write to MTU 3
		2	12		01011 01000 001100001 00011	FPMTUMRU	3	1	4	Met Reset

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
MTUALL (Common Set)	21	2	10		01011 01001 001100001 00110	TFCMMTU1	3	1	7	READ MTU1
			11		01011 01001 001100001 00110	TFCMMTU2	3	1	7	READ MTU2
			12		01011 01001 001100001 00110	TFCMMTU3	3	1	7	READ MTU3

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
TCS FF1	22	1	10							Output to MDM FF01
		2	10							Input from MDM FF01
TCS FF2	23	1	11							Output to MDM FF02
		2	11							Input from MDM FF02
TCS FF3	24	1	12							Output to MDM FF03
		2	12							Input from MDM FF03
TCS FF4	25	1	13							Output to MDM FF04
		2	13							Input from MDM FF04
TCS FA1	26	1	14							Output to MDM FA01
		2	14							Input from MDM FA01
TCS FA2	27	1	15							Output to MDM FA02
		2	15							Input from MDM FA02
TCS FA3	28	1	16							Output to MDM FA03
		2	16							Input from MDM FA03
TCS FA4	29	1	17							Output to MDM FA04
		2	17							Input from MDM FA04

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
PF BITE ACQ	30	2	20		01011 01010 000000000 00000	C22BTPF1			1	PF1 Bite Read
			21			C22BTPF2			1	PF2 Bite Read
PF1 Discretes	31	1	20						4	Output to MDM PF01
				1	01011 01000 101010000 00000	TFCMPFD1	10	0	1	Discretes Set
				2	01011 01000 101010000 00000	TFCMPFD1+2	10	0	1	Discretes Reset
				3	01011 01000 001010000 00000	TFCMPFD1+4	2	0	1	Discretes Set
				4	01011 01000 001000000 00000	TFCMPFD1+6	2	0	1	Discretes Reset
									4	Output to MDM PF02
PF2 Discretes	32	1	21							
				1	01100 01000 101010000 0000	TFCMPFD2	10	0	1	Discretes Set
				2	01100 01000 101010000 0000	TFCMPFD2+2	10	0	1	Discretes Reset
				3	01100 01000 101010000 0000	TFCMPFD2+4	2	0	1	Discretes Set
				4	01100 01000 101010000 0000	TFCMPFD2+6	2	0	1	Discretes Reset

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
PP2 APU	33	1	21	1	01100 01000 11000011 00000	TPCMAPU1	12	3	3	Output to MDM PF02
				2	01100 01000 11001000 00000	TPCMAPU1+2	12	8	1	APU 3 Fuel Quantity
				3	01100 01000 11000101 00000	TPCMAPU1+4	12	10	1	APU 1 Fuel Quantity
										APU 2 Fuel Quantity
TCS PF1	34	1	20							Output to MDM PF01
		2	20							Input from MDM PF01
TCS PF2	35	1	21							Output to MDM PF02
		2	22							Input from MDM PF02

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
LDB	36	1	22		10001 00110 00000000000 00000	TFCMLI11			1	Interrogate W/O GPC Data
			23						1	
		2	22							Interrogate with GPC Data
			23							
		3	22							Go Ahead
			23							
		4	22							Transmission Enable
			23							
		5	22		10001 00100 00000000000 00000	TFCMLSR1			1	Status Request
			23						1	
		6	22							Status
			23							

Device	Device ID	OP Code	BCE Number	BCE Element Number	Command	Buffer Name	Card	Channel	Word Count	Description
TCS LF1	37	1	22							Output to MDM LF01
		2	22							Input from MDM LF01
TCS LA1	38	1	23							Output to MDM LA01
		2	23							Input from MDM LA01

Appendix H

Supervisor Calls

Supervisor Call Number	Name of Service	FCOS Service Routine	Explanation
1	SCHEDULE	FPMSCHED	Initialize a process for execution
2	TERMINATE	FPMTERM	Immediately terminate the current process
3	TERMINATE	FPMTERM	Immediately terminate the specified processes
4	CANCEL	FPMCANCL	Terminate the specified processes after each one's current cycle is complete
5	CANCEL	FPMCANCL	Terminate the specified processes after each one's current cycle completes
6	WAIT	FPMWAIT	Wait for a specified time period to elapse
7	WAIT	FPMWAIT	Wait until an absolute time.
8	WAIT	FPMWAIT	Wait until the specified event expression is true
10	unsupported		
11	unsupported		
12	SIGNAL	FPMSIGNL	Modify an event variable
13	SET	FPMSET	Set an event variable to TRUE
14	RESET	FPMRESET	Set an event variable to FALSE
15	RESERVE	FPMRES	Reserves an EXCLUSIVE procedure or function for use by the current process
16	RESERVE	FPMRES	Reserves a data area for use by the current process (UPDATE Block)

Supervisor Call Number	Name of Service	FCOS Service Routine	Explanation
17	RELEASE	FPMREL	Release a previously reserved EXCLUSIVE procedure or function
18	RELEASE	FPMREL	Release a previously reserved data area (UPDATE Block)
19	OVERLAY	FCMPOVLY	Phase Overlay
20	SEND ERROR	FPMSDERR	Pass an error to the FCOS for processing in accordance with the ON ERROR statement in effect
21	CLOSE	FPMCLOSE	End an execution cycle of process
22	RUNTIME	FPMTMHAL	Obtain current GMT (Non Redundant)
	CLOCKTIME	FPMTMHAL	Obtain current GMT (Redundant)
	DATE	FPMTMHAL	Obtain GMT date
	MET	FPMTMHAL	Obtain current MET
23	ERRGRP	FPMSTAT	Obtain latest error code
	ERRNUM	FPMSTAT	Obtain latest error code
	Unsupported		
24	I/O	FIOSVC	Perform I/O service
25	CM	FCMSVC	Read GPC Discretes
26	PFMMOD	FCMPMOD	Perform program modifications
27	Unsupported		
28	HALTMM	FIOHLTMM	HALT transaction on specified Mass Memory Unit
29	(Reserved)		
30	ISR	FCMASYNC	Initial SSIP Sync
31	UPDATE_MTU	FPMUPMTU	Perform MTU Update

Supervisor Call Number	Name of Service	FCOS Service Routine	Explanation
32	TIME_MGT	FPMMTURM	Supports MTU sync and time tagging
33	OPSCANCL	FPMOPSCN	Perform cancellation necessary for OPS transitions
34	Unused		
35	DO_MSG_MGMT	FCMBMASK	Set/Reset Bus/Data Path Masks
36	RECONFIG	FCMBMAN	Perform bus reconfiguration
37	BCEMOD	FCMBCEMD	User modification of BCE chains
38	SMR	FCMSMASK	Sync mask build
39	FRC_FAIL	FCMSFAIL	Force Fail to Sync
40	PIO	FIOSVCP	Pre-initialize IOQE I/O Request
41	SIO	FPMSIO	Select Timer Initiated I/O
43	MTU_FMT_CONV FXFLCVT	FPMTMCVT FPMTMCVT	MTU Time Conversion FCOS Time Conversion

APPENDIX I

I/O PROFILE

*PREINITIALIZED CODE

DEVICE	DEVICE ID	ALT DEVICE IDs OPERATION CODE 1	2	3	4	5	6	7	8	9	10
IPR Via ICC (Redundant Set)	1	Send 3 words	Send 10 words								
IPR Via ICC (Common Set)	2	"	"								
SSIP ICC	3										
Data Initialization Via ICC	4										
DEU #1	5	Fill	Poll	KYBD	Dump	Request BITE Status	Reset Scratch Pad Line				
DEU #2	6	"	"	"	"	Request BITE Status	Reset Scratch Pad Line				
DEU #3	7	"	"	"	"	"	"				
*DDU	8	WRT									
PMU	9	WRT GPC DATA RAM	WRT 128 KBPS PGM	WRT 64 KBPS PGM	RDS 128 KBPS PGM	RDS 64 KBPS PGM	RDS Bite	Hard FMT SEL	Programmable FMT SEL	SM RDS OI/PL RAM	TCS RDS OI/PL RAM
MMU	10	WRT WITH CKSM	WRT W/O CKSM	RDS With CKSM	RDS W/O CKSM	MM Utility WRT					
*FF & FA Bite ACQ	11	-	RDS								
*FCINPUT 1 (FF and FA)	12	-	RDS				All FF Analogs and Discretes All FA Inputs except LH,RH Attach PTS 2A,3A,2B,3B				
*FCINPUT 2 (FF and FA)	13	-	RDS				All FF Serial Digitals and Discretes plus TAC/RA PA LH,RH attach Pts 2A,3A,2B,3B				
*FCOUTPUT 1 (FF and FA)	14	WRT	-								
*FCOUTPUT 2 (DDU and FA)	15	WRT	-								

*PRELIMINARY

ALT DEVICE IDS
OPERATION CODE

DEVICE	DEVICE ID	1	2	3	4	5	6	7	8	9	10
* FAOUTPUT	16	WRT	-								
* NW STEERING	17	WRT	-								
MTU 1 (Redundant set)	18	WRT	MET Reset								
MTU 2 (Redundant set)	19	WRT	MET Reset								
MTU 3 (Redundant set)	20	WRT	MET Reset								
MTUALL(Common Set)	21	-	RDS								
TCS FF1	22	WRT	RDS								
TCS FF2	23	"	"								
TCS FF3	24	"	"								
TCS FF4	25	"	"								
TCS FA1	26	"	"								
TCS FA2	27	"	"								
TCS FA3	28	"	"								
TCS FA 4	29	"	"								
*FF BITE ACQ	30	-	RDS								

*PREINITIALIZED IOQE

ALT DEVICE IDs

DEVICE	DEVICE ID	OPERATION CODE 1	2	3	4	5	6	7	8	9	10
*PF1 Discretes	31	WRT	-								
*PF2 Discretes	32	WRT	-								
*PP2 APU	33	WRT	-								
TCS PF1	34	WRT	RDS								
TCS PP2	35	"	"								
LDB	36	Interrogate W/O Data	Initiate With Data	Go Ahead	Transmission Enable	Status Report	Status				
TCS LF1	37	WRT	RDS								
TCS LA1	38	"	"								

PART 2

USER INTERFACE

SDDS

User Interface

Table of Contents

TABLE OF CONTENTS (cont'd)

1. INTRODUCTION

This document presents the User Interface detail design descriptions of modules to be used for the Approach and Landing Test (ALT). The design presented reflects requirements as specified in the Computer Program Development Specification Level A (CPDS), Book 1 (Software), dated July 23, 1976 and all approved Change Requests.

The remainder of the INTRODUCTION, Section 1, discusses the document's purpose and Section 2 explains the format used to describe each module.

The MODULE DESCRIPTION Section, Section 3, contains the ALT detailed design for each module in User Interface.

Appendices contain additional information concerning the User Interface design:

Appendix A contains the User Interface Requirements - Detailed Design Cross Reference.

Appendix B contains the User Interface resource allocation summary.

Appendix C contains the User Interface module list.

Appendix D contains a table of error conditions and the default system action to be performed in User Interface.

Appendix E contains the data descriptors table.

Appendix F contains a description of the ICC messages.

Appendix G contains a keyboard message time line.

Appendix H contains an explanation of control segment grammar.

A detailed knowledge of the HAL/S Realtime Statements and a general knowledge of the AP-101 flight computer are assumed. The following list of documents contain information regarding thes topics:

- HAL/S Language Specification
- Interface Control Document: HAL/FCOS, IBM File #6246156A
- Advanced System/4 Pi Model AP-101 Computer Software Systems Manual, IBM File #62280045
- User Interface User's Guide, Release 4.

1.1 PURPOSE

The Detail Design Specification provides explicit module descriptions of the implementation of the User Interface requirements. The detailed breakout of the modules, the detailed logic flows and the detailed data descriptions provide the final level of design description preceding the actual program listing.

1.2 SCOPE

User Interface comprises a part of the permanently resident System Software that provides software services and controls for the applications processing. User Interface provides software services for external users of the Orbiter Flight Computer Software, both onboard and ground based. It also provides the interface between the applications processing and these external system users. It can be divided into three major areas:

Command Input Processing provides the flight software with user inputs from the MCDS, messages from the ground through the launch data bus, and inter-computer channel communications from other GPC's.

Operations Control performs initiations, sequencing, and termination of application and system services processing as requested by command inputs and/or application services.

Output Message Processing and Coordination provides flight software with the capability of presenting displays and messages to the user through the MCDS, of commanding the C and W indicator alarm, the alert light and the alert tone and of communicating with the ground by way of the downlist and the launch data bus.

2. FORMAT EXPLANATION

The individual detail design description of each module consists of a Module Writeup which contains interfaces with other modules, a process description and any limitations the processing may have. The input and output to the module is presented in the Module Data Table and Module I/O Table. Finally, the detailed design is presented in the form of a structured control flow in the Module Control Flow.

2.1 MODULE IDENTIFIERS

Several methods for referring to a program have been defined. The program is identified by its HAL/Assembler name, English_Equivalent_Name or its three digit ID.

2.1.1 English_Equivalent_Names

An English_Equivalent_Name is provided for each HAL/S or assembly language program. Each name conforms to the following standards.

a. It shall be uniquely defined.

b. It shall not contain other than accepted English abbreviations, acronyms defined in the Level A CPDS, or the acronyms Appendix to the SDDS.

c. It shall be concise.

d. It shall be indicative of its use.

e. Underscore connectors shall be used between words within the English name. Each new word will begin with a capital letter.

Example:

1. Error_Logging_Routine

2. LDB_Message_Router

Note that the same standards apply to the "ITEM" name of I/O parameters (see Appendix E).

2.1.2 Module Identification (ID) Codes

All programs, procedures, and segments (where segments are convenient modularizations of complex control flows) are identified by a numerical ID of the format XXX.XX. These codes are utilized in the Source/Destination section of the Data Table (see Section 2.3) and in the appropriate parts of the Control Flow (see Section 2.5).

The module ID's conform to the following standards:

a. A unique three digit code shall be invented for each Program and Procedure. The numbers are assigned in the following way:

FCOS	100-199	Process Management
	200-299	I/O Management
	300-399	Configuration Management
UI	400-699	
SC	700-999	

b. Segment ID codes shall consist of the same three digit prefix as the module it is a part of, followed by a decimal and a two digit number (e.g., 112.01). Segments are numbered consecutively even if they are referenced only by another segment.

c. All ID codes are used consistently throughout the entire document.

2.2. MODULE WRITEUPS

The title at the beginning of each section consists of the section number, followed by the module English_Equivalent_Name, the module HAL/S or assembly language name and the module ID.

Example:

X.X.X.X. English_Equivalent_Name (MODULE1) (XXX)

Each module writeup is divided into the parts described below.

Function - This paragraph provides a very general description of the functions performed by the module.

a. Control Interface - This paragraph identifies all known users of the module and its form of invocation. Priority and rate parameters are specified as symbolic parameters defined in Software Awareness Memo (SAM) 10.

Control Interface
1. Form of Invocation
2. Known Users

Example:

1. SCHEDULE DMI_MCDS_IN PRIORITY (PRIO_DMI)
   REPEAT EVERY TIME_DMI;

2. (a) CALLed by (ID) English_Equivalent_Name
   (HAL/S_NAME)

   (b) SCHEDULEd by (ID) English_Equivalent_Name_2
   (HAL/S NAME2)

(c) Event English_Equivalent_Name (HAL/S NAME)
set by (ID) English_Equivalent_Name_3
(HAL/S_NAME3)

b. Input - In most cases this is a reference to the module data table. However, if special interfaces need to be described they are described here. For a reference to a table "See Table X.X.X.X.-1" is used.

c. Process Description - This is a narrative which functionally describes the processing performed by the module as depicted in its Control Flow. If data parameters are referenced, they are referenced by their English_Equivalent_Names. It includes a reference to the Control Flow Figure number of the form: "The control flow for this module is shown in Figure X.X-X".

d. Outputs - In most cases this is a reference to the module Data Table. However, if special interfaces need to be described, they are described here. For a reference to the table "See Table X.X.X.X-1" is used.

e. Module References - This section identifies all other modules which this module references. Each module is identified by a line consisting of the three digit ID, English_Equivalent_Name and HAL/S or Assembly language name. Modules are CALLed or SCHEDULEd. It also includes the FCOS service routine name of FCOS Supervisor Calls that are invoked.

Example:

1. (001) Reset_Event_Processor (FPMRESET) is CALLed.
2. (002) Lights_and_Alarms_Processing (DLA_LIGHT_ALARM_PROC) is CALLed.

f. Module Attributes

This section specifies the HAL compilation type of the module, whether it is a program, external or internal procedure, function, or task. A program is the highest level unit of compilation and can only be invoked by a SCHEDULE. An external or internal procedure is a subroutine outside or inside the program scope, respectively, and can only be invoked by a CALL. A task is a block of code which is invoked by a SCHEDULE. A function is an operation invoked by the appearance of the name in an expression.

g. Template References - Specifies all templates referenced in the module. Each template is identified by an English_Equivalent_Name and a HAL/S name.

1. English_Name_1 (HALS_NAME_1)
2. English_Name_2 (HALS_NAME_2)

h. Error Handling

This section describes the error handling techniques employed by this module. For each error detected by the module an explanation of the error and the action taken by the module is provided. If the module outputs an error message via the Annunciation Function the error is briefly discussed and the error message number (see Appendix D) is enclosed in parenthesis.

i. Constraints and Assumptions - This section describes the programming limitations to which this module is subject and delineates the assumptions which are reflected in the design of this module.

j. Detailed Implementation - This section is used to provide a text extension of the logic statements which are usually enclosed within a processing block of a Control Flow. These logic statements may be in HAL/S form, but assembler language code is not allowed. The process block contains a reference number which relates to a number in this section of the module writeup.

2.3 MODULE DATA TABLE

All data input and output to a module or the hardware and internal parameters necessary for the understanding of a processing event (i.e., all data referenced in the control flow, narrative, or detailed implementation section) are itemized in a tabular format (see Figure 2.3-1). Each item in the table is explained below.

1 Name - The module English_Equivalent_Name along with the HAL/S or assembler name in parenthesis.

Example:

MCDS_Input_Processor (DMI_MCDS_IN)
Configuration_Management_SVC_Service_Routine (FCMSVC)

2 Item Number - Items are sequentially numbered in the general order of their appearance in the associated control flow.

3 Item - The English_Equivalent_Name assigned to the data in the Data Descriptors Table (see Appendix E).

4 Description ID - The ID assigned to the data in the Data Descriptors Table (see Appendix E).

5 Activity Type - This column denotes the usage of the parameter. Codes are:

I - An external memory location referenced by the module.
O - An external memory location changed by the module.
L - Local variable
Z - COMPOOL constant
T - Temporary Variable (Variables used only within a HAL DO group)

Any number of activity types that apply are specified in this column.

6 Source - A listing of the three digit ID's of all modules updating the parameter or hardware source of the parameter. If more than four modules update the parameter, a reference to the Data Descriptors Appendix is inserted instead of the list. The three digit ID's are listed in numerical order.

7 Destination - A listing of the three digit ID's of all modules referencing the parameter or the hardware source to which the parameter is being written. If more than four modules reference the parameter, a reference to the Data Descriptors Appendix is inserted instead of the list. The three digit ID's are listed in numerical order.

8 HAL/S - The HAL name is used in coding. (Include the HAL equate name in parenthesis where applicable). If no name is assigned the column is blank.

9 MML - The Master Measurements List number in the requirements source. If not specified the column is left blank.

10 D - An 'X' is placed in the column if the parameter is available to be downlisted.

11 C - An 'X' is placed in the column if the parameter is available to be displayed on a CRT.

2.4 MODULE I/O TABLE

The module writeup contains a table specifying the data passed to FCOS in each I/O request. The format of the table and explanation of the parameters is found in Appendix G in Part 1 (FCOS) which contains a table of the I/O request definitions for each device and OP code. If the I/O is of the pre-initialized type the table is not included with the module writeup. Reference to this table or FCOS Appendix G is included in the control flow where the I/O SVC is issued.

2.5 MODULE CONTROL FLOW

2.5.1 Flow Charts

System Software modules are documented by Structured Control Flows. Where possible/practical, the structure of the flow follows the structure of the program listing. Each page of flowcharting presents a complete picture of a primary processing objective with subsequent expansions of detail appropriately referenced. The expansion is accomplished either through a segment that is presented in detail on later pages of the control flow or through a reference to the Detailed Implementation section of the module writeup. The first page of each flow contains one elliptical block specifying "Enter" and a block specifying "Return". Logic flow is generally from top to bottom and left to right thus no arrowheads are required on connecting lines. See Figure 2.5-1 for the Control Flow legend.

The initial page of the control flow is titled with the same title as the module writup; that is English_Equivalent_Name and HAL/S or assembler name.

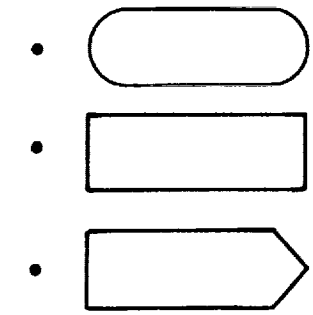
Example:

Figure X.X.X.X - English_Equivalent_Name (HAL_NAME)

Subsequent pages of a flow have the same title as the initial page without the HAL/S or assembler name. Additionally, they have a subtitle consisting of an English_Equivalent_Name for the segment and the ID defined in the process box which references the segment.

Example:

Figure X.X.X.X-2 English_Equivalent_Name
English_Equivalent_Segment_Name (XXX.1)

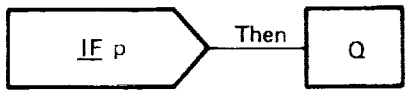


Terminal - identifies Enter/RETURN point (such as HAL/S executable units of compilation and internal procedures and functions)

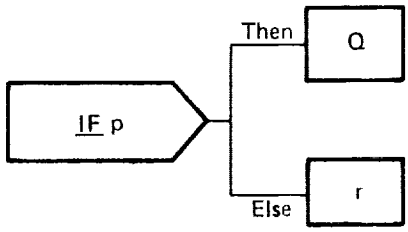
Mathematical statement/process control block which sometimes references another flow for a lower level of detail

Decision statement block

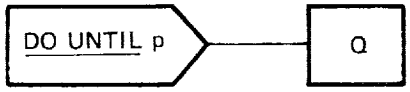
Documentation Examples



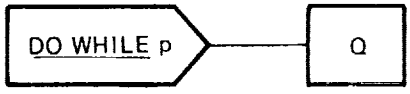
If p true then do Q, and return in-line*.
(The word THEN is required.)



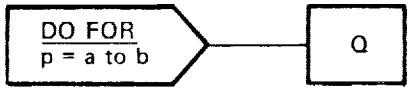
If p true then do Q and return in-line*, else do r, and return in-line*.



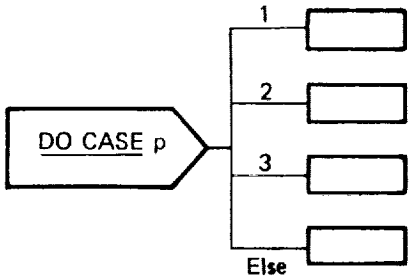
Do Q until p true, then return in-line*.
Note: The decision on p is made after doing Q.



Do Q while p true, then return in-line*.
Note: The decision on p is made before doing Q.



Do Q while p increments from a to b, and return in-line*.



Do one of specified processes based on value of p, and return in-line.* NOTE: Some identification of each case choice is given on the line leading into box, such as an index number, or the word ELSE (for the branch taken if the case index is outside its valid range).

* - Return in-line means return to the statement immediately below the decision. If there is no such statement, retrace path to the last previous decision and repeat this logic.

Figure 2.5-1. Control Flow Legend

3. USER INTERFACE PROGRAM DESCRIPTIONS

User Interface, like FCOS, is a permanently resident set of modules designed to support users of flight software. User Interface is that part of the Space Shuttle software which provides the interface between the user or ground and the various application software. User Interface software provides the user and ground with the capability to monitor and control the software functions. This is accomplished through the various command inputs and support of application control segments through the use of control segment grammar. (See Appendix H.) User Interface software provides the exclusive medium for two-way communication with the crew via CRT displays and keyboard entries, for communication with the ground via the LDB and Downlist, and for data exchange between computers via the ICC. Figures 3-1 and 3-2 are high level structured flows of User Interface.

Figure 3-3 illustrates the data and processing flow through User Interface with inputs from the ICC, LDB, and Keyboard Interfaces. The chart depicts the major areas in User Interface and their relationship to one another in the processing of inputs to present displays on the CRT, to support the applications, and to transfer data to the ground.

User Interface is divided into three major areas (Figure 3-4) and the remainder of the functional overview discussion addresses the following:

a. Command Input Processing provides the flight software with user inputs from the MCDS, messages from the ground through the launch data bus, and intercomputer channel communications from other GPC's.

b. Operations Control performs initiation, sequencing, and termination of application and system services processing as requested by command inputs and/or application services.

c. Output Message Processing and Coordination provides flight software with the capability of presenting displays and messages to the user through the MCDS, of commanding the C and W indicator alarm, the alert light and the alert tone and of communicating with the ground by way of the downlist and launch data bus.

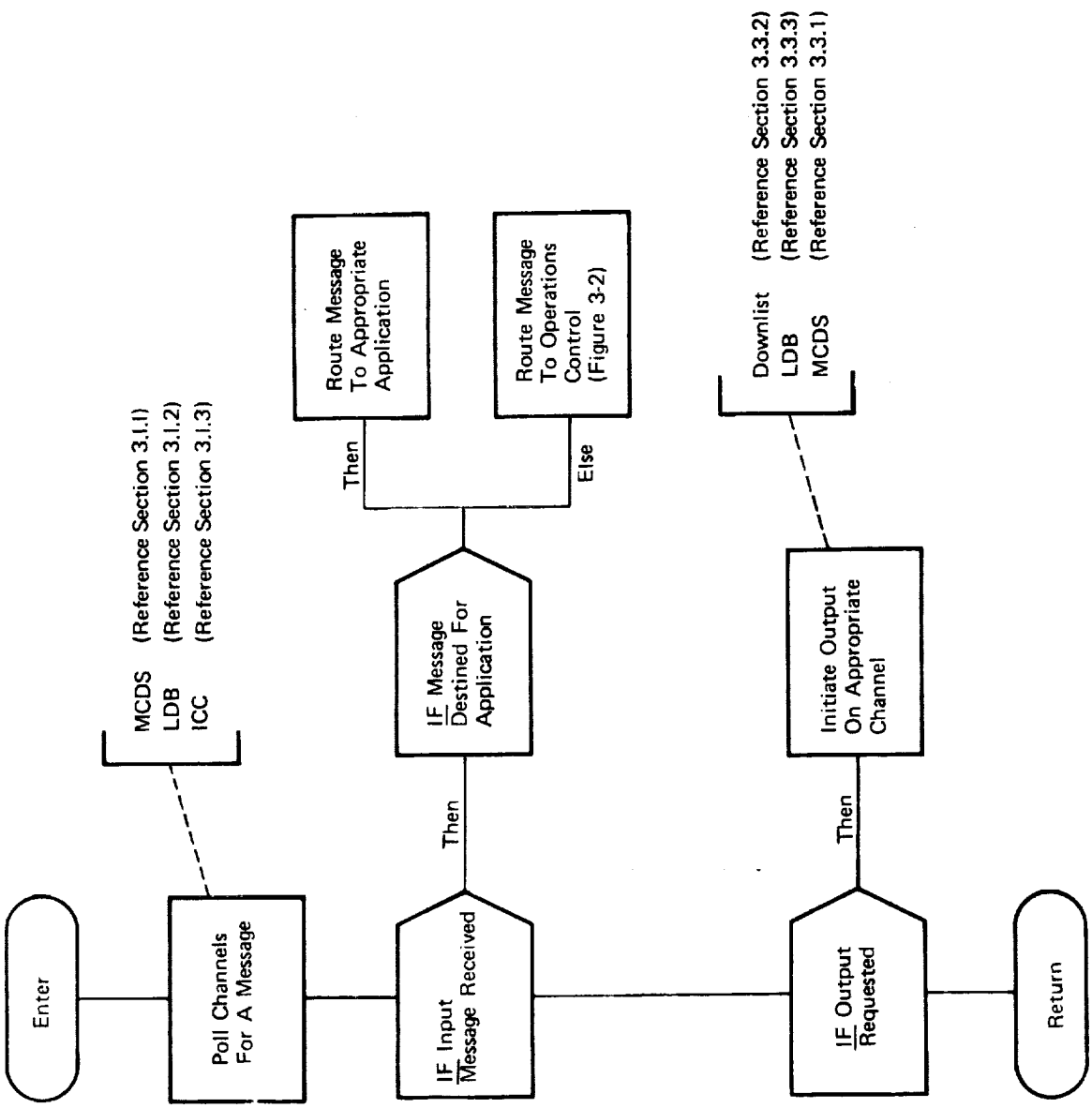


Figure 3-1. User Interface

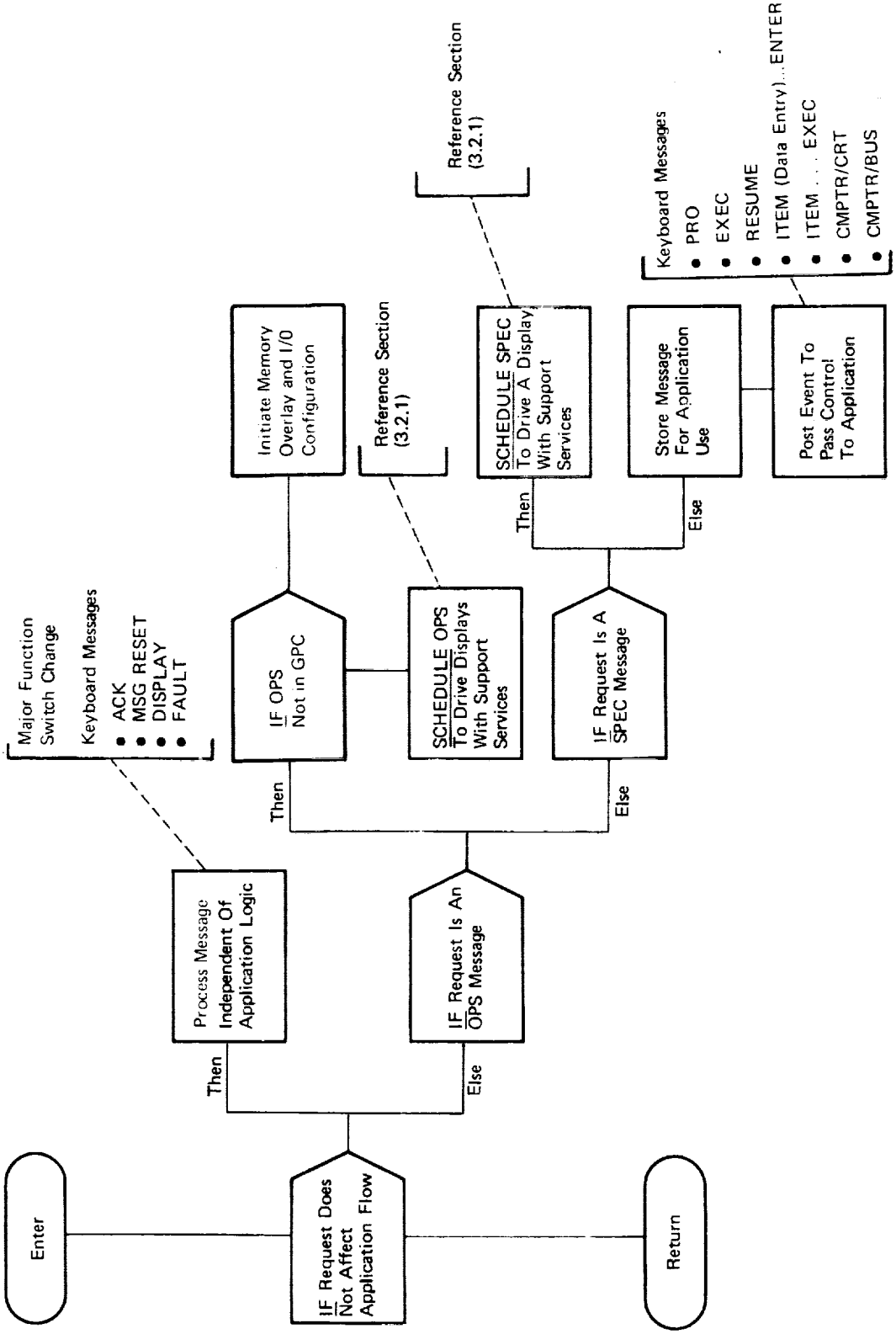


Figure 3-2. Operations Control

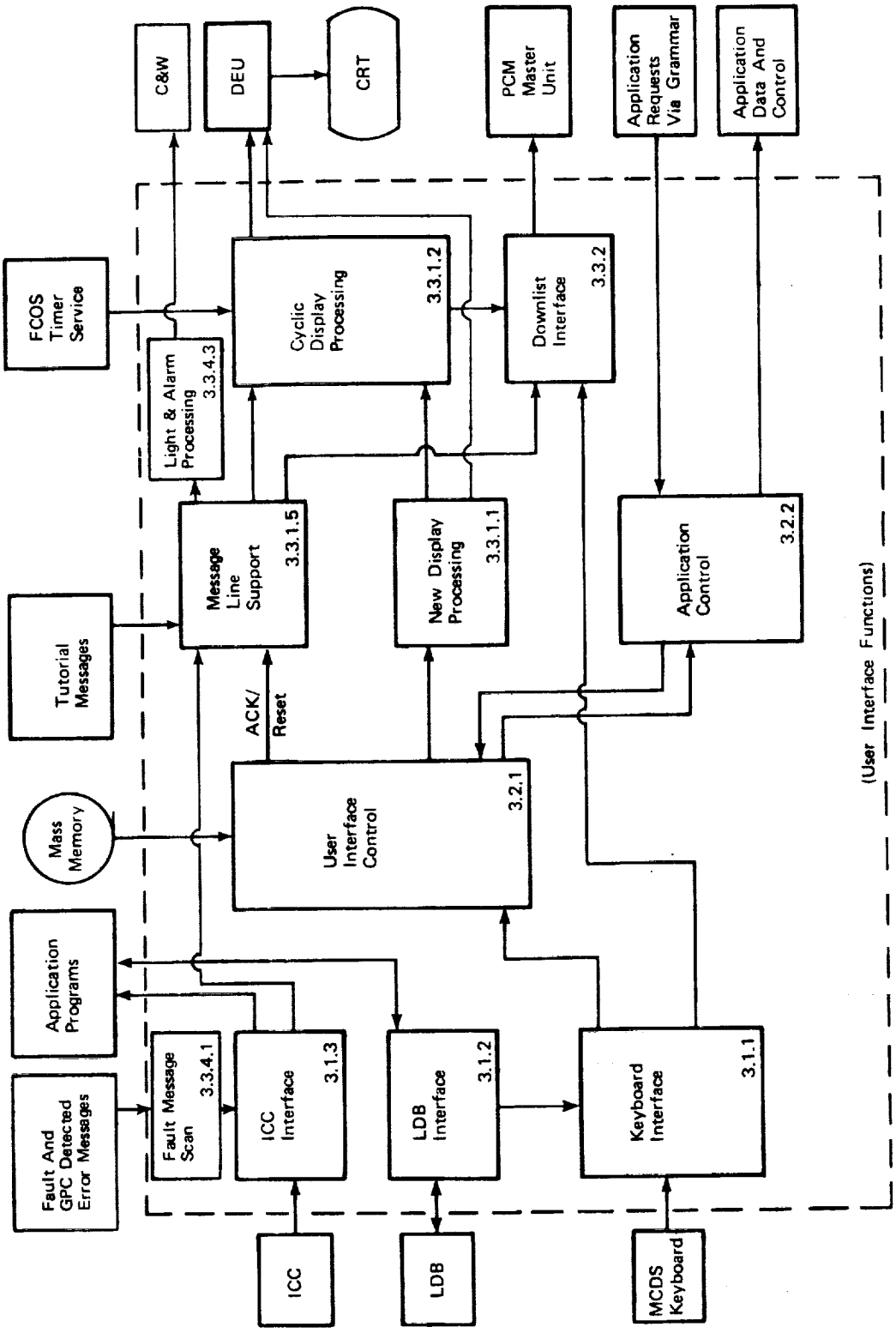


Figure 3-3. User Interface Processing Flow

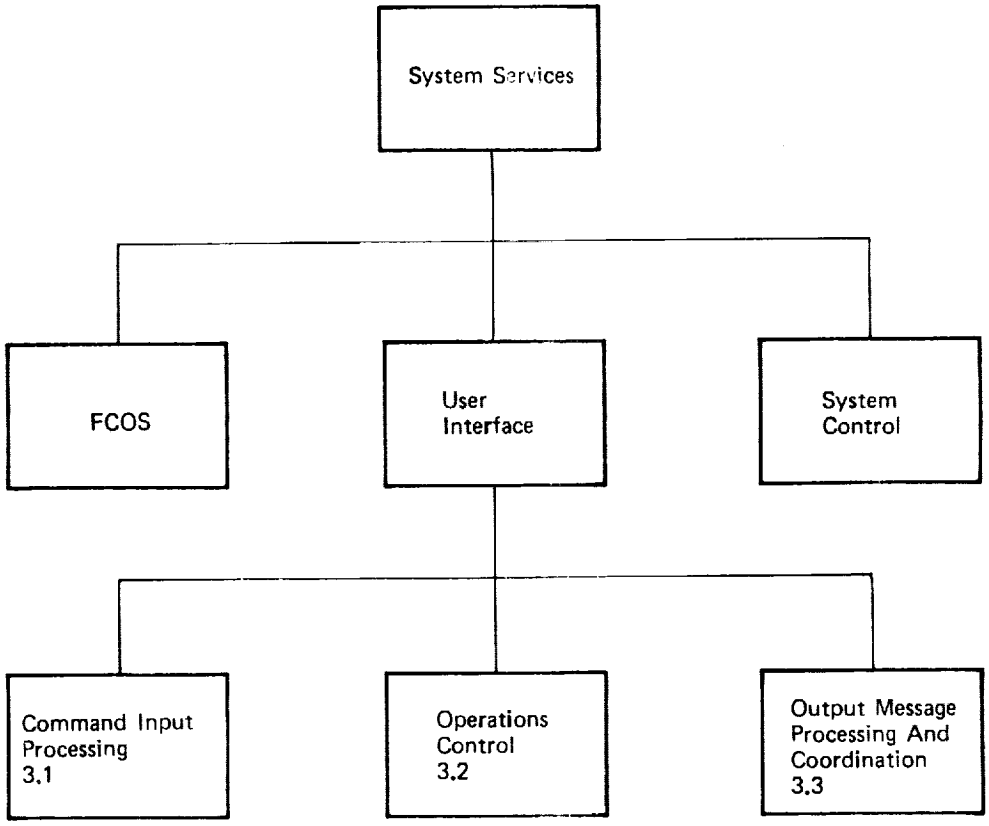


Figure 3-4. Functional Overview Hierarchy Diagram

3.1 Command Input Processing

The Command Input Processing software conducts the communications necessary to obtain inputs from the multi-functional CRT Display System (MCDS), the Launch Data Bus (LDB), and the Inter-Computer Channel (ICC). In addition to the acquisition of these inputs, the routing of commands and data to target GPC applications software is provided. See Figure 3.1-1.

- Keyboard Interface software provides input communications with the MCDS and routes the data acquired from it to Operation Control for further processing and response.

- The LDB Interface controls the I/O communications with the ground support equipment through the Launch Data Bus and routes the acquired data to the appropriate applications software or to Operations Control for processing and response.

- ICC Interface provides common support for ICC messages required to be transferred among the common set of GPC's at common sync time. a service is provided to accumulate messages between System Interface Program (SIP) cycles, to merge them into a single message for transfer to all other GPC's in the common set, and to unpack the messages from all GPC's and make them available to the processes for which they are destined.

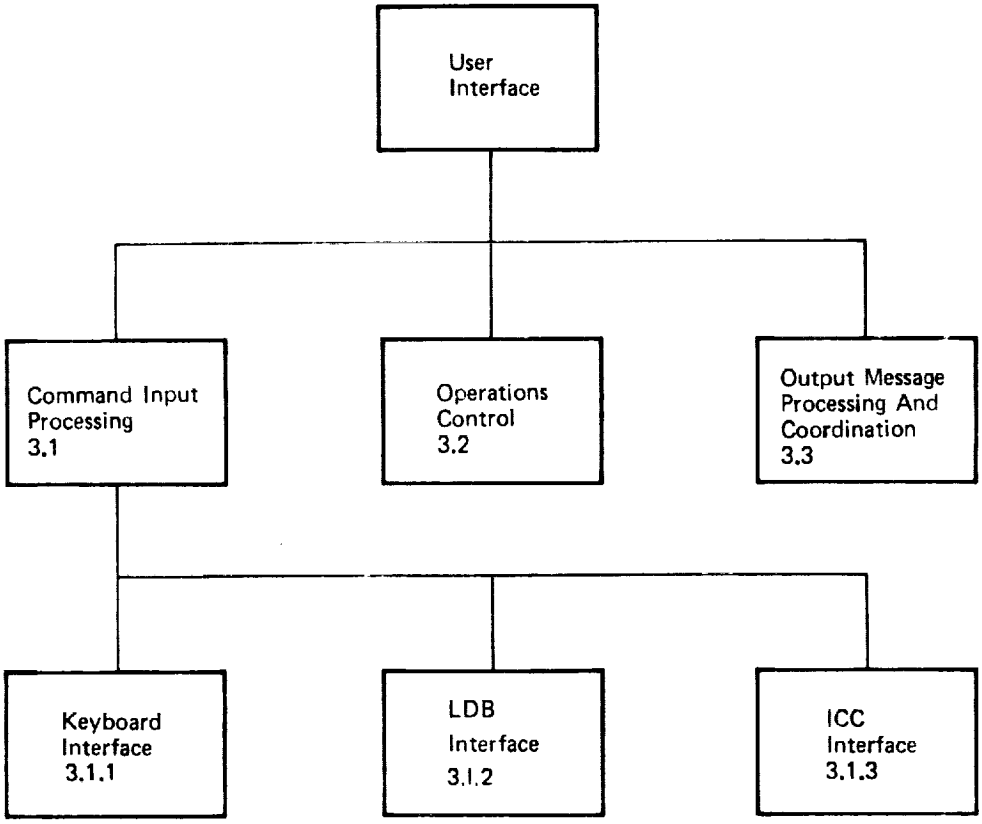


Figure 3.1-1. Command Input Processing Hierarchy Diagram

3.1.1 Keyboard Interface

The Keyboard Interface software is responsible for acquiring user's MCDS keyboard inputs and MCDS status. Any MCDS messages obtained from the ground via the LDB Interface software are also accepted and filtered by major function and routed to the appropriate software module for processing. Figure 3.1.1-1 presents the hierarchial diagram of Keyboard Interface.

a. MCDS_Input_Processor (DMI_MCDS_IN) (400) provides the logic to control and receive all inputs from the MCDS's. This is managed by cyclicly polling each MCDS and acting on the responses accordingly.

g. MCDS-Message_Processor (DMM_MCDS_PROCESS) (405) receives all MCDS messages regardless of organization, monitors the major function switch, initiates DEU bus changes based on the major function switch and passes the keyboard messages to User Interface Control for additional processing.

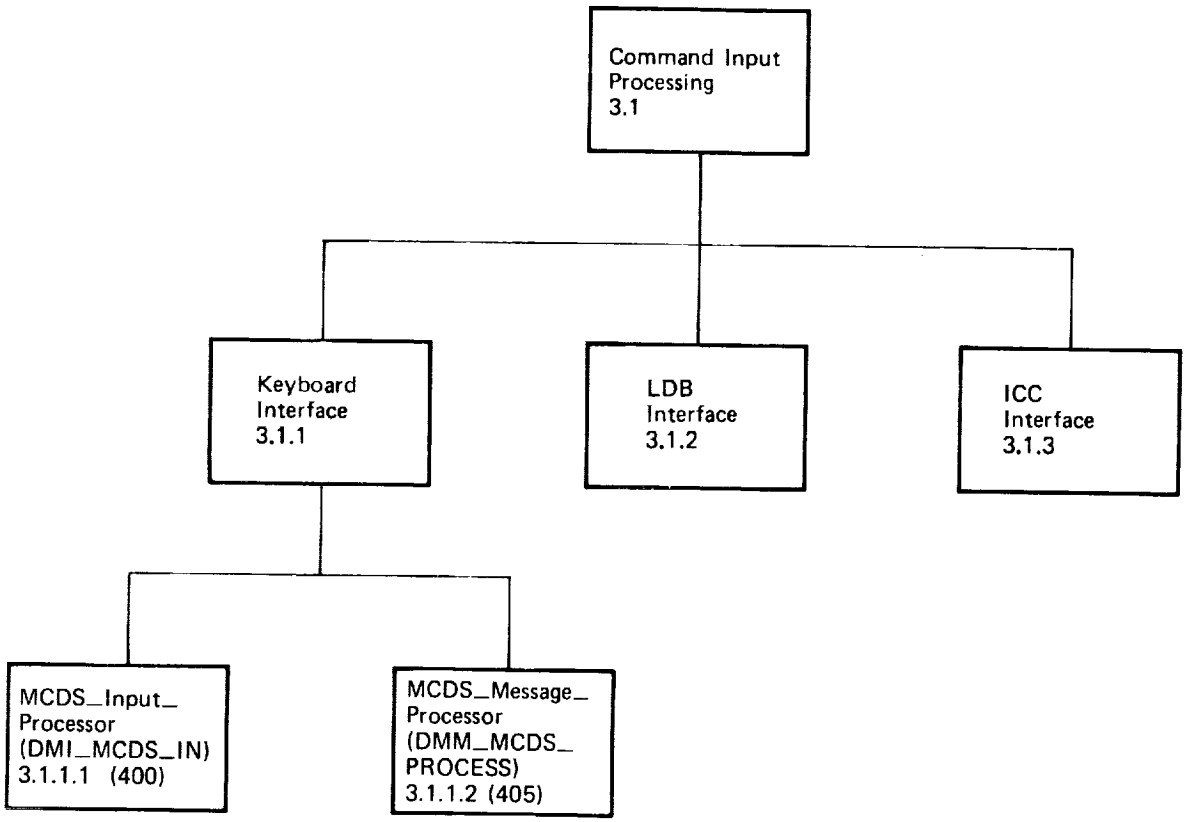


Figure 3.1.1-1 Keyboard Interface Hierarchy Diagram

3.1.1.1 MCDS_INPUT_PROCESSOR (DMI_MCDS_IN) (400)

The function of this module is to receive all inputs from the MCDS by cyclicly polling each DEU whose buses are in command or listen mode. Successful inputs are then passed to the MCDS_Message_Processor (405).

a. Control Interface:

1. SCHEDULE DMI_MCDS_IN PRIORITY (PRIOR_DMI)
   REPEAT EVERY TIME_DMI

2. SCHEDULED by (710) GPC_STARTUP (AIR_GPC_STARTUP)

b. Input - See Table 3.1.1.1-1

c. Process Description - The following logic applies for each DEU. For any DEU input to be processed the DEU data buses must be in command or listen mode and the DEU_Poll_Flag must be set. If these two conditions are met, the DEU_Poll_Flag is disabled so this module has control of the inputs, the device ID for the DEU is determined, and a mode status request with wait is issued using FCOS macro DIO. The processing depends on the transmission status.

   If there is an error on transmission, an internal flag (Polling_Error_Flag) is set for use during the next polling cycle. I/O is suppressed and MCDS_Message_Processor releases control of the input by resetting the DEU_Poll_Flag. Processing cycles to the top for the next DEU.

   For successful transmission, the Polling_Error_Flag status is checked. If on, the I/O must now be unsupressed. Also, BITE available and freeze must be forced by setting it in the Message_Header_Mask. This signals Cyclic_Display_Processor to perform a refresh of any display frozen as a result of the I/O suppression. The Polling_Error_Flag is reset to zero. Regardless of the Polling_Error_Flag status, MCDS_Message_Processor is called to further process the inputs.

   Upon return from MCDS_Message_Processor the module cycles to the next DEU. The control flow for this module is shown in Figure 3.1.1.1-1.

d. Output - See Table 3.1.1.1-1.

e. Module References:

1. (405) MCDS_Message_Processor (DMM_MCDS_PROCESS) is called.

2. (200) I/O_SVC_Service_Processor (FIOSVC) is invoked via DIO macro

3. (815) Force_OPS_0(AOP_FORCE_OPS_0) is called.

f. Module Attributes - Program

g. Template References:

1.	UI_FCOS_Shared_Compool	(CZ2_COMMON)
2.	UI_Section_of_Common_Compool	(CZ1_COMMON)
3.	UI_General_Compool	(CDM_UI_COMPOOL)
4.	FCOS_Compool	(FCMCOM)
5.	MCDS_Message_Processor	(DMM_MCDS_PROCESS)

h. Error Handling - When a transmissions error or time out is detected on the mode status response, I/O is suppressed by setting CRT_Status_Flag 2 and DEU_Poll_Flag is turned back on (set to 1).

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.1.1.1-1

NAME MCDS_Input_Processor (DMI_MCDS_IN)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	MCDS_Input_Busy	I680	O	400	See App. E	CZ2V_MCDS_IN			
2	DMI_Local_DEU_Number	L340	L	400	400,405	DM1VL_DEU			
3	IOP_Transmitter_State	I010.06	I	355,351, 830	See App. E	CZ2B_ACT_XMITRI	See Appendix E		
4	IOP_Receiver_State	I010.08	I	335,355	See App. E	CZ2B_ACT_RECVR			
5	DEU_Poll_Flags	J062	IO	See App. C	See App. C	CZIB_D_DEU_POLL_FLAG			
6	DIT_Status	J060.01	I	400,440	400,660 665	CZIV_D_DIT_STAT			
7	DEU_Message_Header	J060.02	I,O	See Appendix E		CZIB_D_DIT_MSG_ HEADR	V93M20018 V93M2101P		
							V91M2201P		
8	Polling_Error_Flag	L341	L	400	400	DMIV_POLL_ERR			
9	CRT_Status_Flags	B010.40	O	See App. E	See App. E	CDMB_MAT_CRT_STAT			
10	Number_Of_DEUS	J004	I		See App. E	CZIK_D_NUMBER_DEUS			

Table 3.1.1.1-2

DEVICE	DEVICE ID	OP CODE	SVC NO.	SYNC TYPE	COMFAULT STATUS	ICC TYPE	TIME TAG	WAIT	PROT.	STATUS	INPUT OUT-PUT	MAJ. FUNC. ID	I/O PRIOR-ITY	WORD COUNT	BUFFER	EVENT	BUS NOMINAL
DEU1	5	2	24	RS	NO	NO	NO	YES	YES	YES	I		160	1	FIODBF1P		6
DEU2	6	2	24	RS	NO	NO	NO	YES	YES	YES	I		160	1	FIODBF2P		7
DEU3	7	2	24	RS	NO	NO	NO	YES	YES	YES	I		160	1	FIODBF3P		8

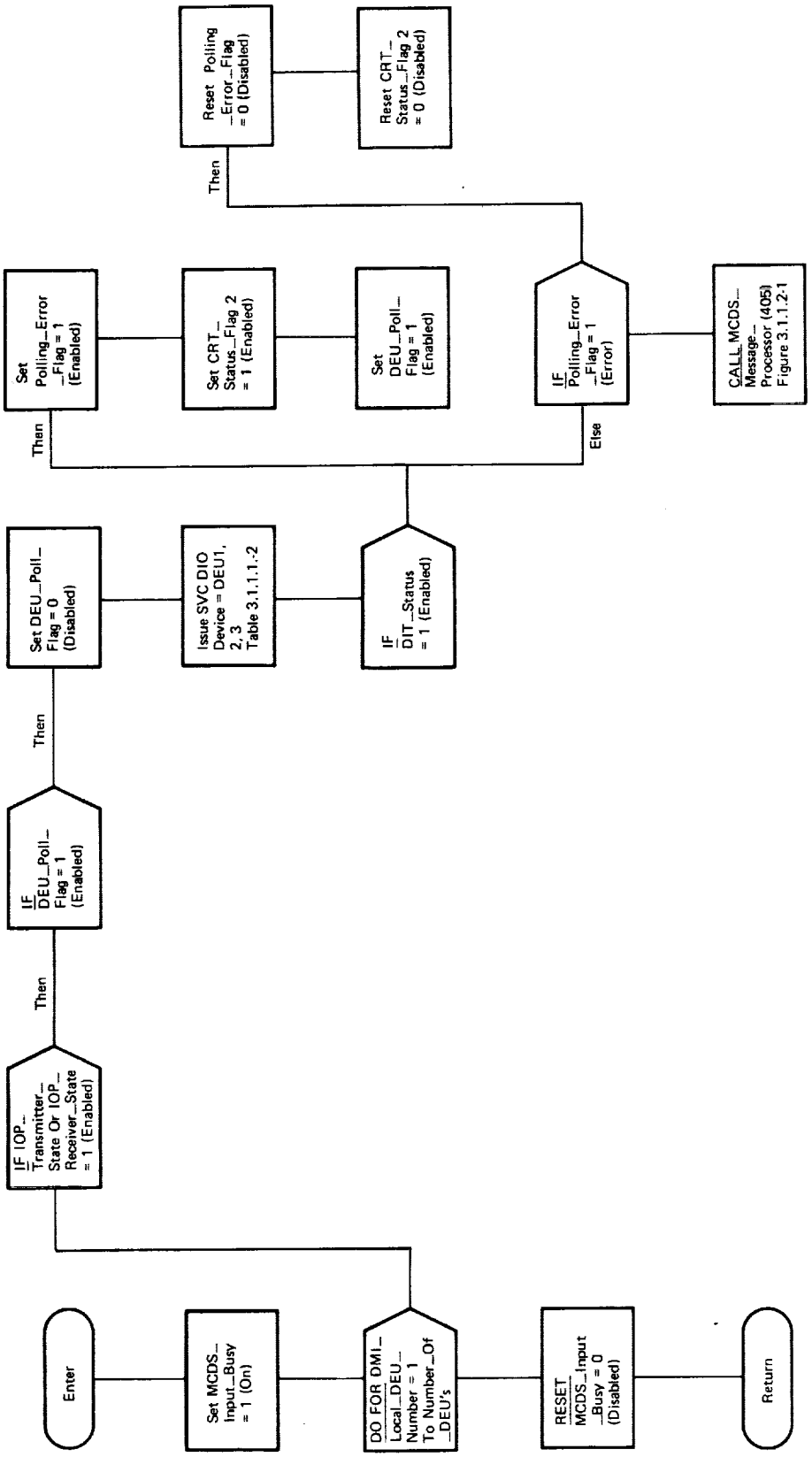


Figure 3.1.1.1-1. MCDS_Input_Processor (DMI_MCDS_IN)

3.1.1.2 MCDS_Message_Processor (DMM_MCDS_Processor) (405)

This module interrogates the mode status response obtained by MCDS_Input_Process (400) and processes the following conditions indicated by the response:

1. DEU initialization
2. BITE availability for downlisting
3. Stand-alone-self-test in progress
4. Keyboard message pending
5. Freeze DEU inputs

The module also checks for major function switch setting and configures the DEU buses based on the switch setting.

a. Control Interface -

   1. CALL DMM_MCDS_PROCESS
   2. CALLed by (400) MCDS_Input_Processor (DMI_MCDS_IN)

b. Input - See Table 3.1.1.2-1

c. Process Description - The Header_Message_Type is checked for a mode status response. If it is then the following processing takes place. A test is made to determine if the DEU requested initialization by checking Header_Initialization. If this bit is on then the memory configuration (MC) for the GPC is checked. If the memory configuration is proper, either MC0 (OPS 000) or MC3 (OPS 201), and the DEU_Loader is not currently active, then the DEU_Load_Table is turned on and the DEU_Loader is scheduled. If the DEU is currently active, DEU_POLL_FLAG is turned on and processing within this module is complete for this DEU.

   If the GPC is not in the proper memory configuration then the Initialization_Error_Flag is checked. If off, the initialization error had not been annunciated and is done so at this time. Also, the Initialization_Error_Flag is turned on to prevent successive annunciation of the same error. Regardless of this flag's setting, DEU_Poll_Flag is turned on reenabling polling, and processing is complete for this DEU in this module.

If Header_Initialization is off, other bits within the Message_Header_Mask are checked to determine the processing path. If Stand-alone-self-test is indicated, that is, Header_SAST set on, then DEU_Poll_Flag is set on and processing is complete. If Header_SAST is off, the Header_DEU_BITE is checked. If on, the Same_DEU_BITE is checked. If this flag is off, the same DEU BITE error had not occurred on the previous poll so it is annunciated at this time and the SAME_DEU_BITE is set on to prevent subsequent annunciation if error occurs during the next cycle. Regardless of the Same_DEU_BITE setting, a BITE status request is sent to the DEU with a wait by using the FCOS service (DIO) macro.

If DEU_Bite_Transmission_Status is set, a transmission error has occurred and an error annunciating this fact is set. If DEU_Bite_Transmission_Status is successful (reset to zero) DEU_Power_Transient is checked. If on, a power transient has occurred and the CRT screen is blanked. To correct this, Message_Freeze_Process is CALLed to unfreeze the screen. If DEU_Power_Transient is off or by continuing from the power transient logic above, DEU_BTU_BITE has bits set corresponding to the BITE errors occurring within the DEU registers. Regardless of the transmission status, DEU Poll_Flag is turned on completing the BITE processing as well as this modules processing of this DEU.

If the Header_DEU_BITE is off, the following major function (MF) switch processing takes place. First, the Same_DEU_BITE is reset indicating BITE error annunciation will not be suppressed the next time a BITE error occurs. Next, the new MF setting is taken from the Message_Header_Mask and is placed in Current_MF_Setting. Bypass_Logic is turned on. This flag will be reset when the major function has not changed between polls and at least three cycles have gone by. Next, Current_MF_Setting is checked for an invalid setting of three. If it is not three, Current_MF_Setting is compared with Last_Major_Function to determine if there has been a change. If there has, MF_Timing_Needed is set to start timing three cycles. MF_3cycle_Timer is set (to one) indicating this is the first cycle in which a major function switch change has been detected. Also, Last_MF is set to reflect the current MF setting.

If there has not been a MF change detected, then MF_Timing_Needed is checked. If on, one is added to the count in MF_3cycle_Timer. Next, if the count is greater or equal to three the new change is ready to be processed in the following manner. First, DEU_Number_Of_Keystrokes is set to two and Keyboard_Message is set to twelve to signal User_Interface_Control_Supervisor (500) of the major function change. Next, ICC_Acceptance_Status reset to zero is the criteria for determining when the ICC_Message_Collector (480) is no longer CALLed within a DO loop to send the MF change to the collector to reconfigure the buses. That is, the ICC_Message_Collector will set ICC_Acceptance_Status to zero upon successful transmission. Next, the ICC_Status_Flag is set and MF_Timing_Needed is reset. If MF_Timing_Needed was off then Bypass_Logic is turned off.

If the major function had not changed or Current_MF_Setting was an invalid setting of three (meaning switch was in between one of the three positions) or by continuing from the MF logic above, the message router processing begins.

If Bypass_Logic is zero meaning the current cycle is not part of any major function change cycle, then the following is done. First, the current freeze status is taken from the Message_Header_Mask and placed in Current_Freeze_Bit_Setting. This status is compared with the previous freeze status stored in Previous_Freeze_Bit_Setting. If the two are different, Freeze_Process is CALLed. Otherwise, Header_Keyboard is checked. If on, the GPC sends a keyboard request to pick up the keyboard message using the FCOS macro (DIO) with a wait for response. If there is an I/O error on transmission of the response, an indicator to annunciate this error is set and DEU_Poll_Flag is set. If there is no I/O error the internal procedure Convert_Keys is CALLed.

If Bypass_Logic was set to one and either MF_Timing_Needed on or Current_MF_Setting set to three then DEU_Poll_Flag is set on reenabling polling.

If the Header_Message_Type is not a mode status response, this indicates the caller was the LDB_Message_Router and the only processing needed is the keyboard message processing. This is done by calling the internal procedure Convert_Keys.

The internal procedure Convert_Keys takes each keyboard message keystroke and converts it from a five bit code into an internal code from zero to thirty-one. The number of keystrokes is stored in DEU_Number_Of_Key_Strokes, the five bit code in Hex_Keystroke_Code, and the internal codes are placed in Keyboard_Message. After the final keystroke is converted, the DEU_Message_Ready Flag is set to one, and the Keyboard_Message Ready_Event is set indicating to the User_Interface_Control_Supervisor (500) the message is ready for further processing.

The internal procedure Freeze_Process indicates a freeze or unfreeze condition to the User_Interface_Control_Supervisor (500) by setting DEU_Number_Of_Keystrokes=2, Keyboard Message = display and Keyboard_Message_Ready_Flag and the DEU_Message_Ready_Flag on. Also, the current freeze condition is set equal Current_Freeze_Bit_Setting. The control flow for this module is shown in Figure 3.1.1.2-1.

d. Output - See Table 3.1.1.2-2

e. Module References -

1. (480) ICC_Message_Collector (DIM_ICC_COLLECTOR) is CALLed.
2. (790) DEU Loader (AIG_DEU_LOADER) is CALLed

f. Module Attributes - external procedure.

g. Template References -

1. UI_General_Compool (CDM_UI_COMPOOL)

2. Annunciation_Compool (CDL_ANNUN)

3. UI_Section_Of_Common_Compool (CZ1_COMMON)

4. UI/FCOS_Shared_Compool (CZ2_COMMON)

h. Error Handling - The following errors are detected by MCDS_Message_Processor:

1. Detection of a further BITE status available indication in the mode status response causes the issuance of a Bite status request command. The BITE status registers are made available for down-listing and an error message (DU001, DU002, or DU003 corresponding to DEU1, 2 or 3) is output.

2. Detection of a transmission error or a time out (no response) from a DEU on a keyboard message request command or a BITE status request command causes the output of an error message (DU004, DU005, DU006 corresponding to DEU1, 2 or 3).

3. A test is made to verify that either the IPL overlay (MC=0) or SM-8 (MC=3) is resident when Header_Initialization is turned on. If neither is resident, then the DEU Loader is not scheduled and an error message (SC008) is output.

i. Constraints and Assumptions - The keyboard message data obtained from LDB_Message_Router is assumed to have previously been processed for syntax verification and is assumed to be in the same format as that obtained from the MCDS_Input_Processor. It is also assumed that only keyboard messages will be input from the LDB_Message_Router and that those keyboard messages are addressed only to the DEU's whose buses are enabled for this GPC. It is also assumed that the major function switch setting at the addressed DEU is the same as the major function switch setting specified from the LDB to prevent the ground from overriding the setting via LDB equivalent message.

DATA TABLE 3.1.1.2-1

NAME MCDS_Message_Processor (DMM_MCDS_PROCESS)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	Message_Header_Mask	L002	L	405	405	DMMV_MSG_HDR			
2	DEU_Message_Header	J060.02	I	See Appendix E		CZIB_D_DIT_MSG_HEADR	V93M20018 V93M2101P		
							V93M2201P		
3	Memory_Configuration_Number	I010.05	I	405,820 880	See App. E	CZ2V_MC	See Appendix E		
4	MC_Zero	L089	L	405,500	405,600	IPL_MC			
5	MC_Three	L090	L	405	405	SM_8_MC			
6	Initialization_Error_Flag	L015	L	405	405	DMMV_INIT_ERR			
7	DEU_Poll_Flags	J062	IO	See App. E	400,405 815	CZIB_D_DEU_POLL_FLAG			
8	DEU_Load_Table	J068	O	405,790	790	CZIB_D_DLT			
9	Same_DEU_Bite	L014	L	405	405	DMMB_SAME_BITE			
10	DMI_Local_DEU_Number	L091	L	400,405	405	DMMVL_DEUNO			
11	DEU_Bite_Transmission_Status	J070.01	O	400,405 790	400,405 790	CZ1V_D_XMIS_STAT			
12	DEU_Bite_Status_Message	J070.21	I	405,790	405,790	CZIV_D_BITE_STAT			
13	Keyboard_Message	J060.20	O	See App. E	See App. E	CZIV_D_DIT_KYBD_MSG			
14	CRT_Status_Flags	B010.40	O	See App. E	See App. E	CDMB_MAT_CRT_STAT			
15	DEU_BTU_Bite	I100.19	O	405, 820,880	660,665	CZ2B_BTU_BITE_DEU	See Appendix E	X	X
16	Current_MF_Setting	L019	L	405	405	DMMV_MF_NEW			
17	Bypass_Logic	L018	L	405	405	DMMV_BYPASS			
18	MF_Timing_Needed	L016	L	405	405	DMMB_MF_TIMING			
19	MF_3Cycle_Timer	L017	L	405	405	DMMB_MF_TIMR			

DATA TABLE 3.1.1.2-1 (Cont'd)

NAME MCDS_Message_Processor (DMM_MCDS_PROCESS) 405

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
20	Last_MF	I450	O	See App. E	405	CZ2V_MF_OLD			
21	DEU_Number_Of_Keystrokes	J060.03	I	405	405	CZIV_D_DIT_NUMOFKEYS_			
22	MF_Change_For_ICC	L0022	L	405	405	DMMB_MF_ICC			
23	ICC_Acceptance_Status	L0024	L	405	405	DMMB_ICC_ST			
24	Current_MF	I270	O	405	405,485 580	CZ2V_DEU_MF			
25	ICC_Status_Flags	I320	O	See Appendix E		CZ2B_ICC_FLAG			
26	Current_Freeze	L0021	L	405	405	DMMV_NEW_FRZ			
27	Old_Freeze	L0020	L	405	405	DMMV_OLD_FRZ			
28	Keystroke_Storage	L0003	L	405	405	DMMVL_KEY			
29	Keyboard_Message_Ready_Event	J020	O	405		CZIE_D_MCDS_EVENT			
30	Temporary_I_Loop	L0092	T	405	405	DMMVL_I			
31	Temporary_J_Loop	L0093	T	405	405	DMMVL_J			
32	HEX_Keystroke_Code	L0005	L	405	405	DMMKL_KEY_5BIT			
33	DIT_Status	J060.01	I	405	405	CZ1V_D_DIT_Status			
34	DEU_Message_Ready_Flag	J050	I	405	405	CZIV_D_DIT_MSG_READY			
35	Error_Message_Line_Ready			405	405	CDMB_MAT_CRT_STATUS(2)			
36	DEU_Device_No	L0094	I	405	405	DDMV_BT_STAT_DDEV			

DATA TABLE 3.1.1.2-1 (Cont'd)

NAME MCDS_Message_Processor (DMM_MCDS_PROCESS)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
37	HEADER_MESSAGE_TYPE	L006	L	405	405	DMMV_MSG_HDR			
38	HEADER_DEU_NO	L007	L	405	405	DMMV_MSG_HDR			
39	HEADER_MF_SETTING	L008	L	405	405	DMMV_MSG_HDR			
40	HEADER_FREEZE	L009	L	405	405	DMMV_MSG_HDR			
41	HEADER_KEYBOARD	L010	L	405	405	DMMV_MSG_HDR			
42	HEADER_SAST	L011	L	405	405	DMMV_MSG_HDR			
43	HEADER_DEU_BITE	L012	L	405	405	DMMV_MSG_HDR			
44	HEADER_INITIALIZATION	L013	L	405	405	DMMV_MSG_HDR			

Table 3.1.1.2-2

DEVICE	DEVICE ID	OP CODE	SVC NO.	SYNC TYPE	COMFAULT STATUS	ICC TYPE	TIME TAG	WAIT	PROT.	STATUS	INPUT OUT-PUT	MAJ. FUNC. ID	I/O PRIOR. ITY	WORD COUNT	BUFFER	EVENT	BUS NOMINAL
DEU 1	5	2	24	RS	No	No	No	Yes	Yes	Yes	I		160	1	FIODBF1P		6
DEU 2	6	2	24	RS	No	No	No	Yes	Yes	Yes	I		160	1	FIODBF2P		7
DEU 3	7	2	24	RS	No	No	No	Yes	Yes	Yes	I		160	1	FIODBF3P		8

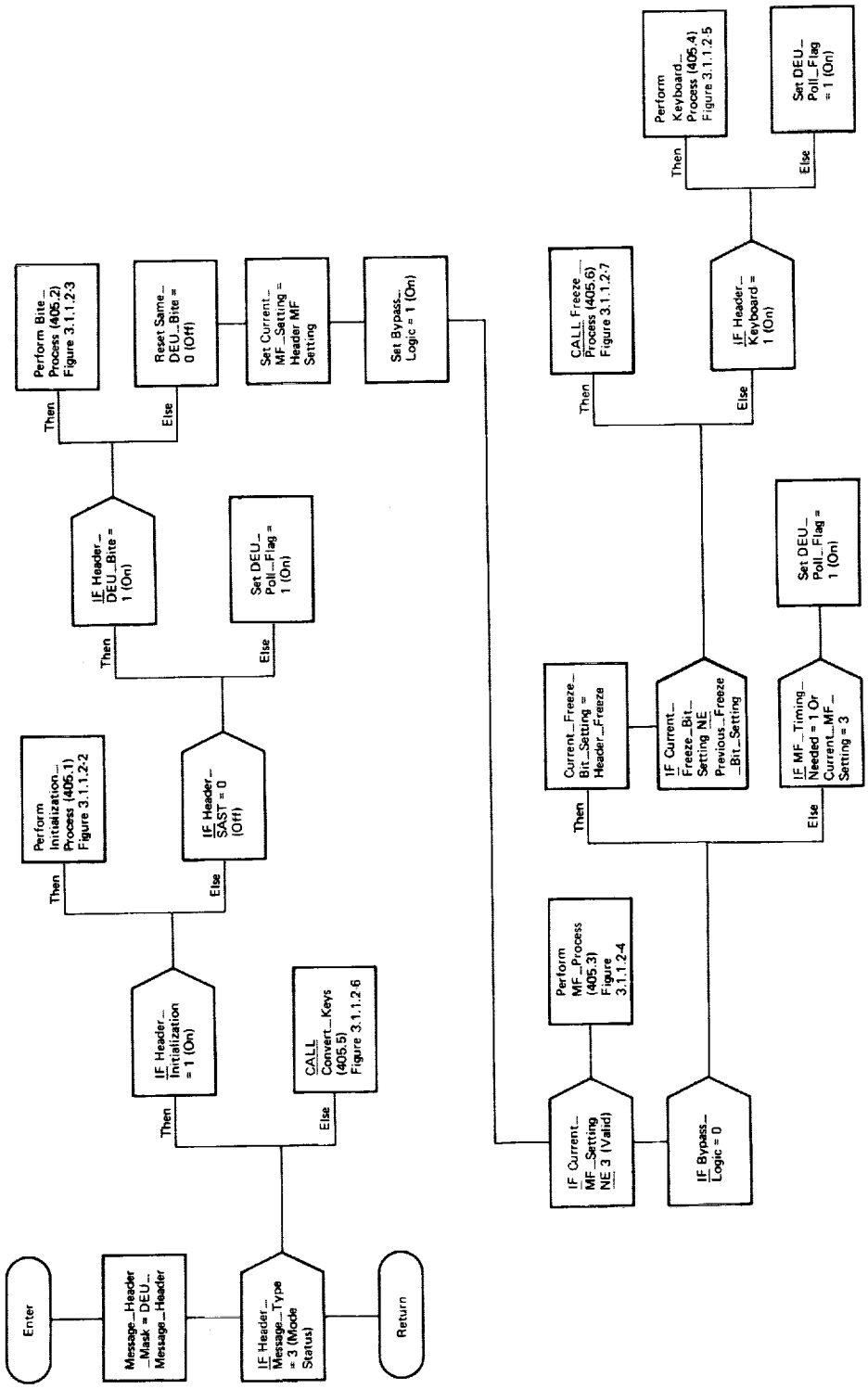


Figure 3.1.1.2-1. MCDS_Message_Processor (DMM_MCDS_Process)

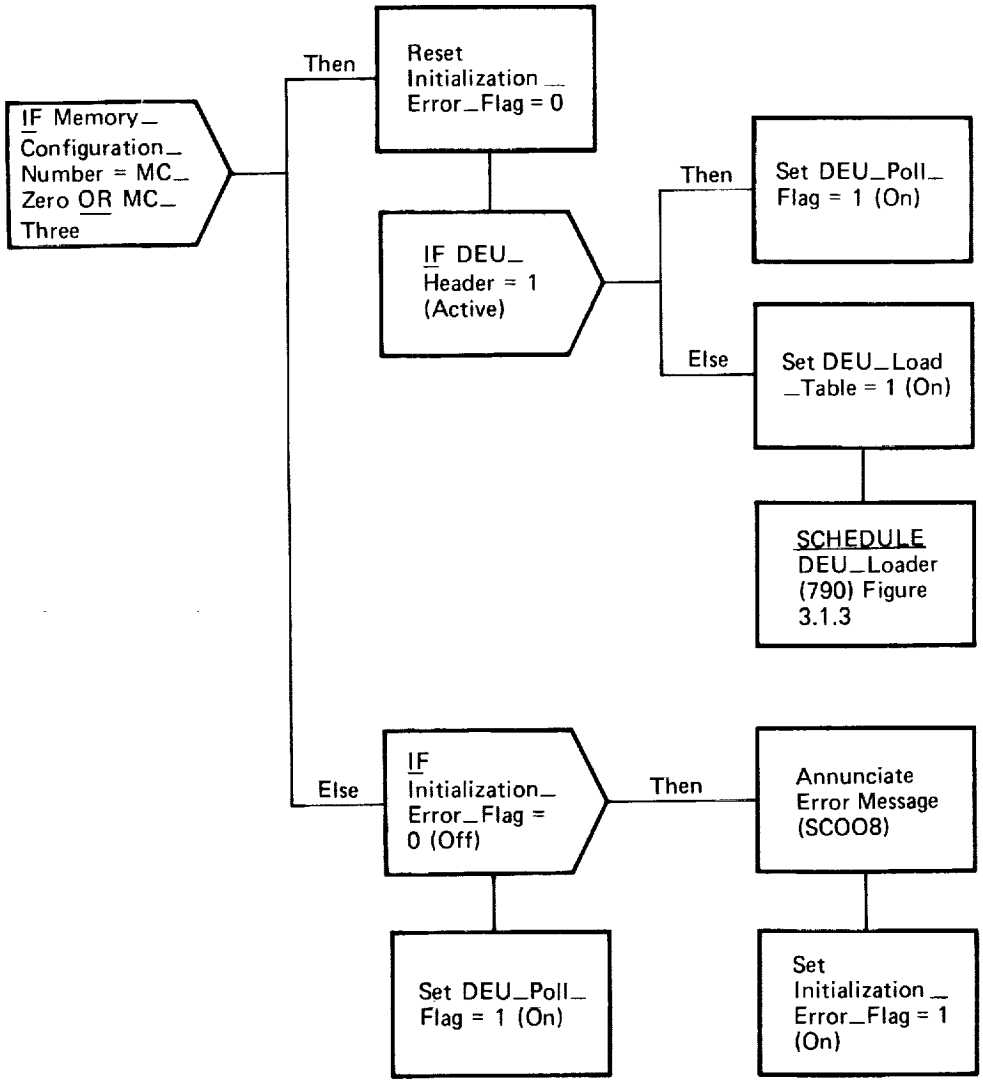


Figure 3.1.1.2-2. MCDS_Message_Processor Initialization_Process (405.1)

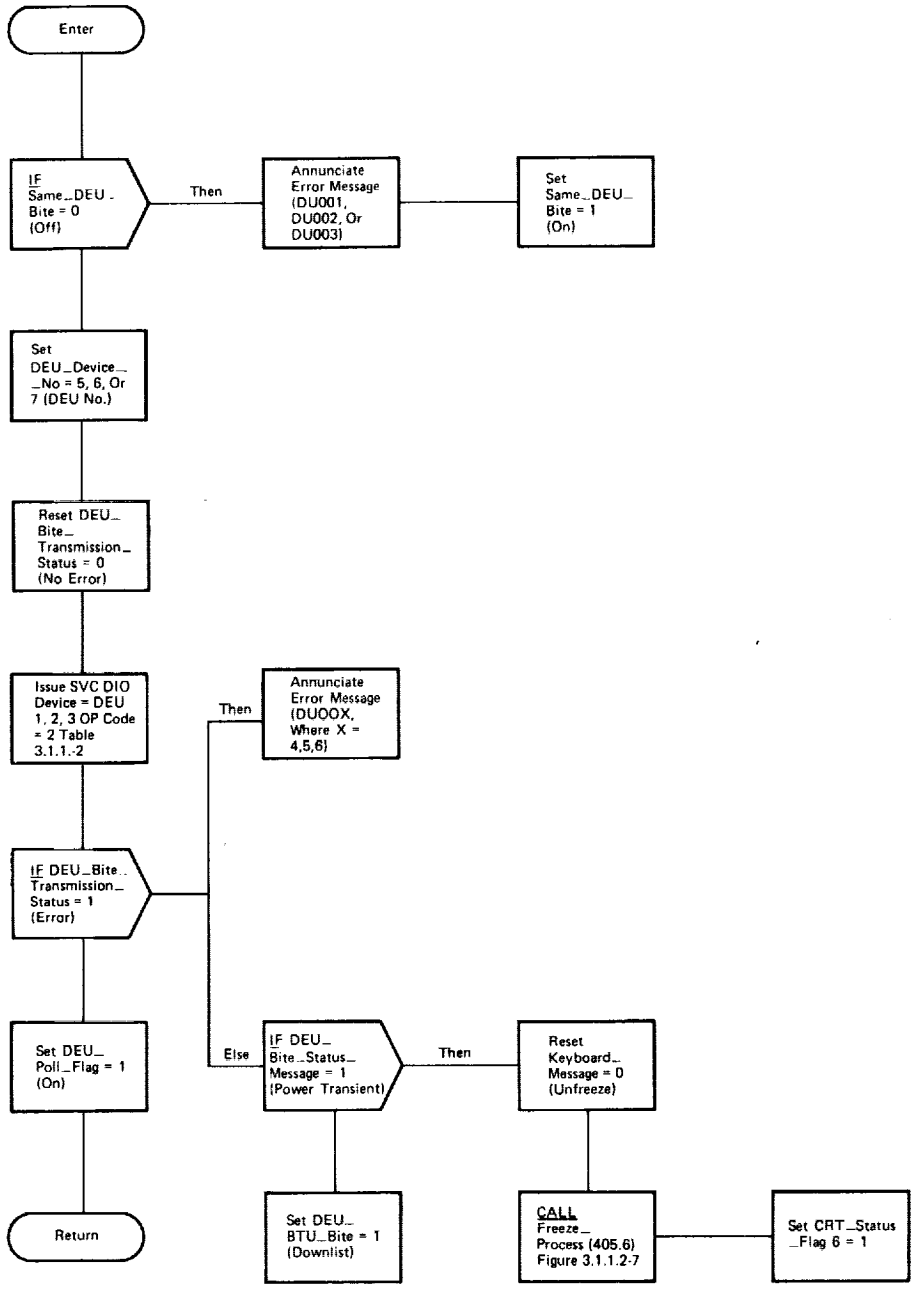


Figure 3.1.1.2-3. MCDS_Message_Processor Bite_Process (405.2)

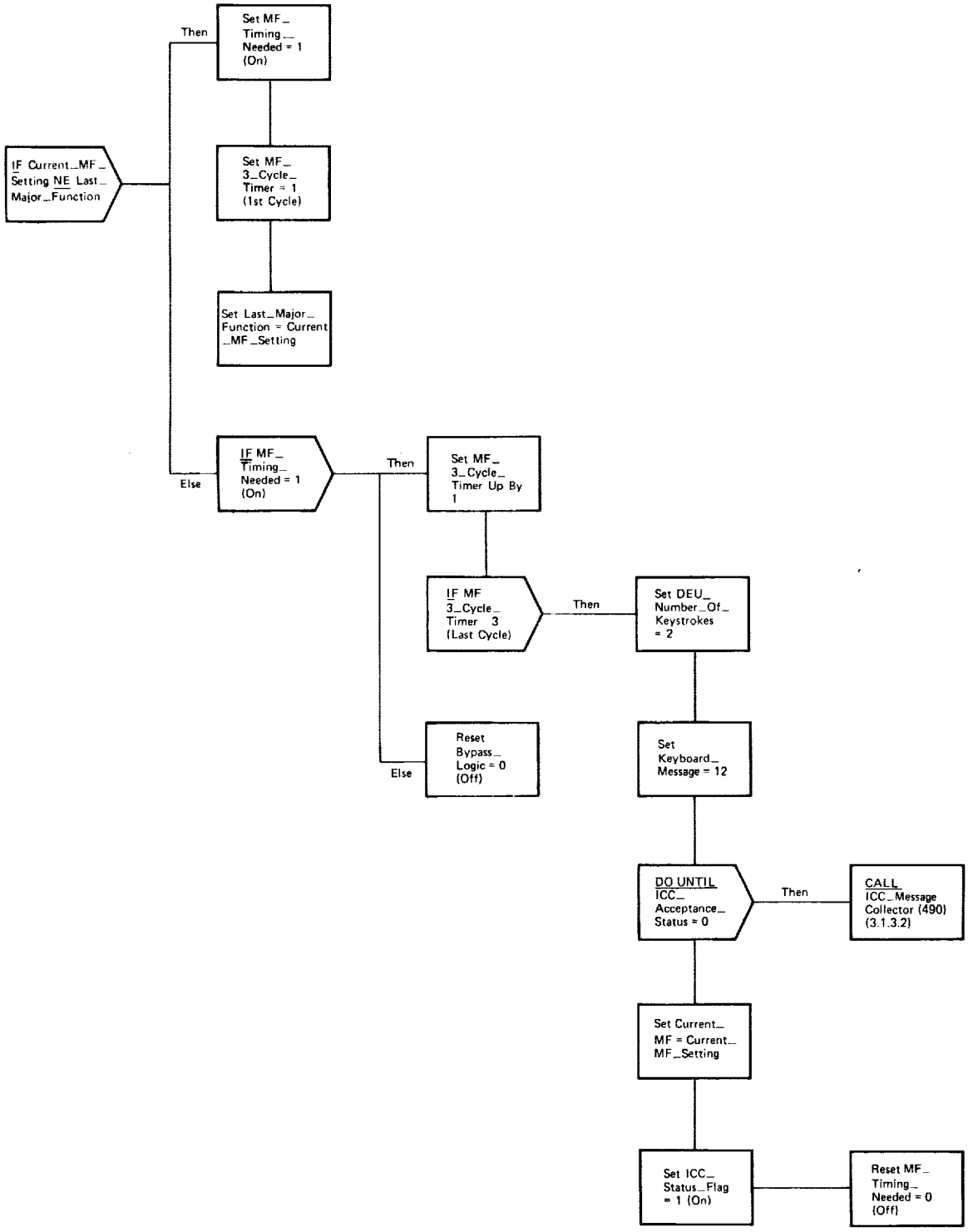


Figure 3.1.1.2-4. MCDS_Message_Processor MF_Process (405.3)

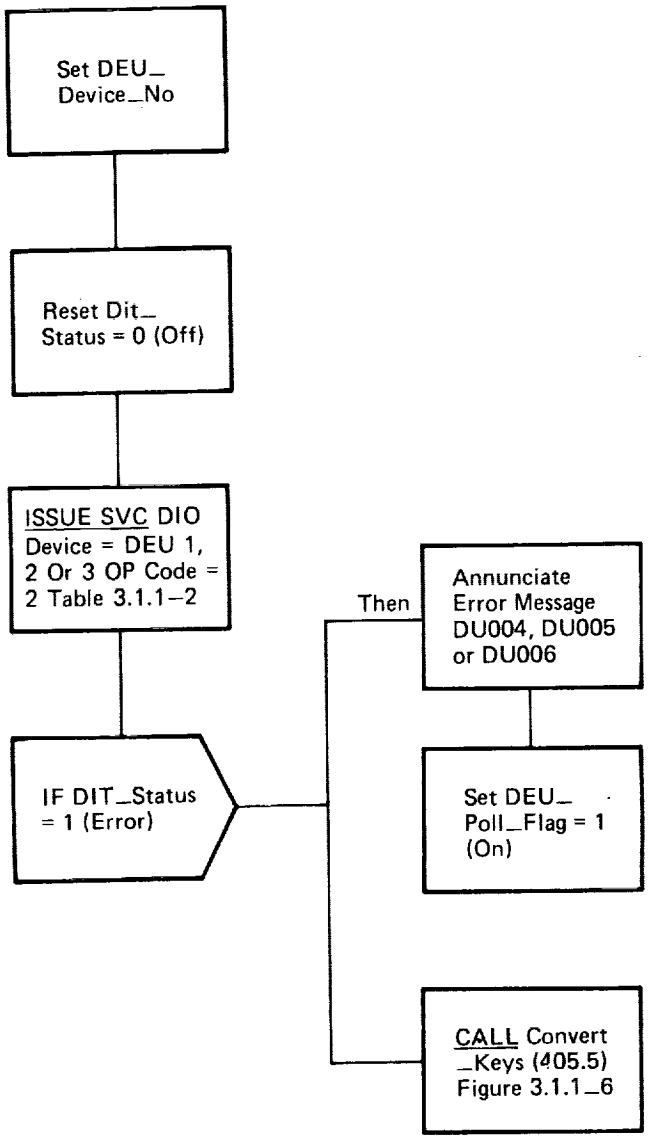


Figure 3.1.1.2-5. MCDS_Message_Processor Keyboard_Process (405.4)

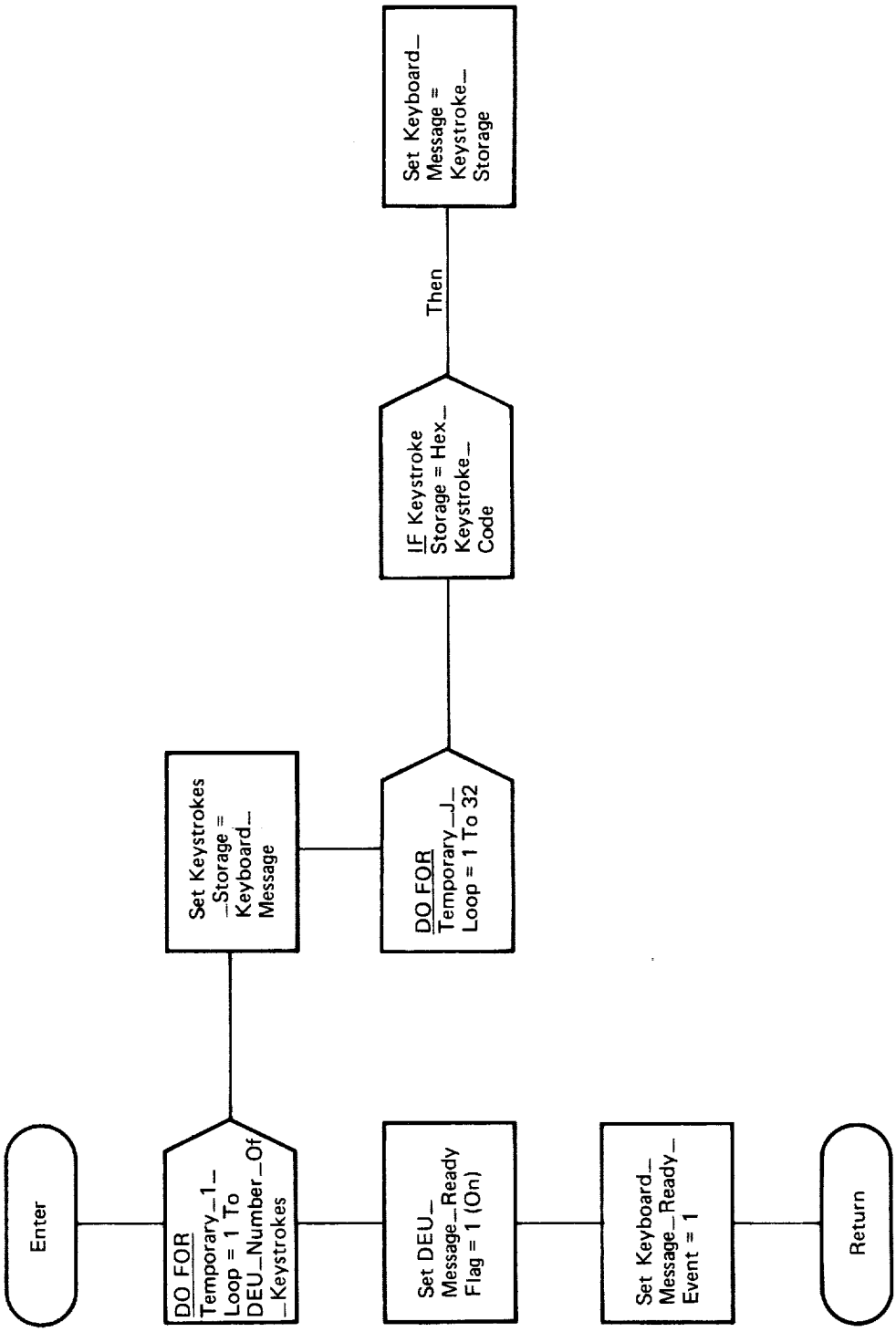


Figure 3.1.1.2-6. MCDS_Message_Processor Convert_Keys (405.5)

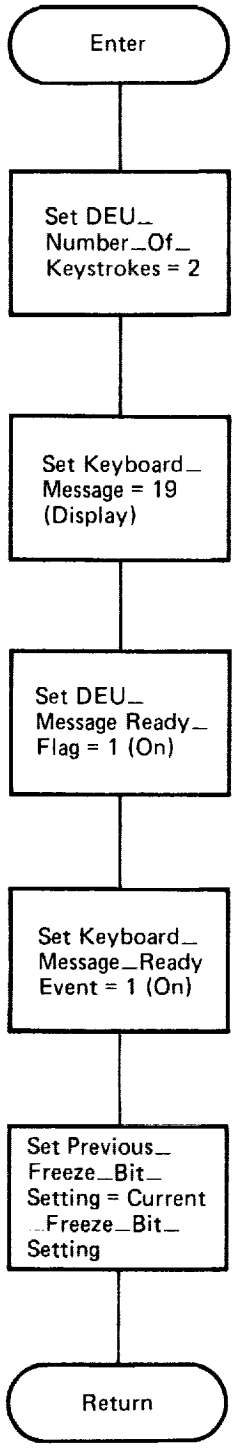


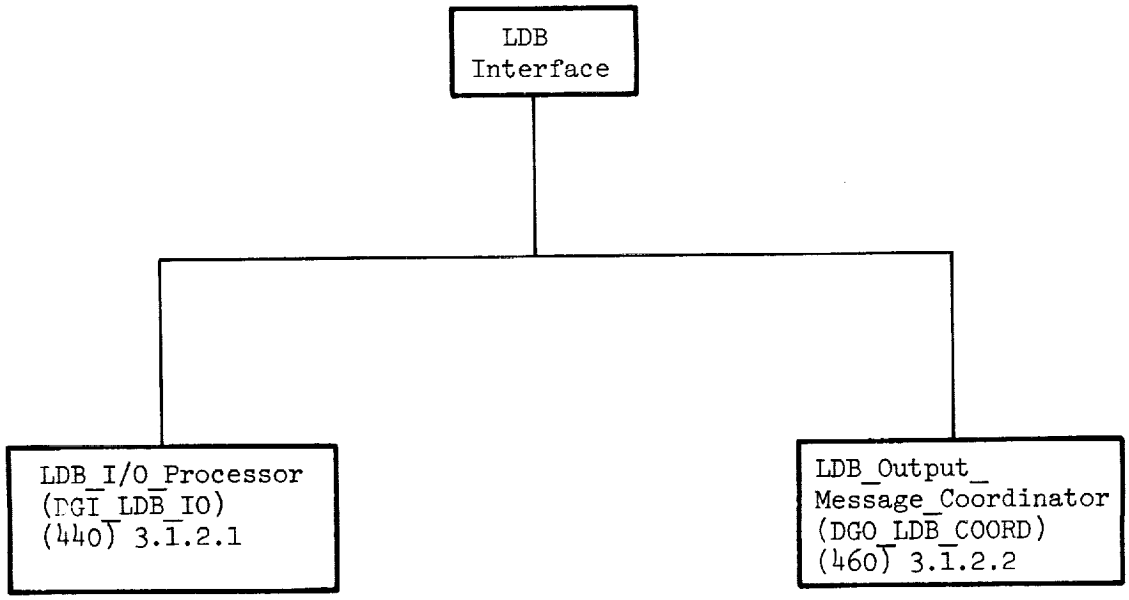
Figure 3.1.1.2-7. MCDS_Message_Processor Freeze_Process (405.6)

3.1.2 LDB INTERFACE

The LDB Interface is a collection of functions which allow Orbiter/ground communications via the Launch Data Bus. See Figure 3.1.2-1.

a. LDB_I/O Processor - Controls all communications transmitted across the Launch Data Bus between the ground and the GPC. Cyclic polling across the LDB establishes the need for communications. Input messages from the ground are received, validated, and routed to various GPC software applications. Messages from the GPC software applications are transmitted when no messages are pending on the ground.

b. LDB_Output_Message_Coordinator - (DGO_LDB_COORD) accepts messages from applications programs, moves them to the LDB Output Buffers and sets a flag to initiate transmission to the ground by the LDB I/O Processor.



LDB Interface Hierarchy
Figure 3.1.2-1

3.1.2.1 LDB_I/O_Processor (DGI_LDB_IO)(440)

The function of this module is to handle all communications transmitted across the launch data bus (LDB) between the ground and the GPC. Cyclic polling across the LDB is the mechanism used to determine the need for GPC/ground communications. Input messages from the ground are received and validated as a result of affirmative response from the ground to the processor's poll for pending messages. Messages from the GPC software applications are transmitted to the ground when no messages are pending on the ground.

a. Control Interface

1. SCHEDULE DGI_LDB_IO PRIORITY (PRIO_DGI) REPEAT EVERY TIME_DGI;
2. SCHEDULEd by (910) (Read/Write Specialist Function)(ASB_RD_WRT)

b. Input - See Table 3.1.2.1-1.

c. Process Description - Upon activation (either the initial SCHEDULE or if made ready by an expiration of a WAIT), a test is made to determine if the GPC has a message to send to the ground. If so, the "input/output poll sequence" is performed. If no outbound traffic is pending, the "input poll sequence" is performed.

The input/output poll sequence first tests to determine if a message is being transmitted. A message will be retransmitted if an error was encountered in a previous attempt to transmit the message. If a repeated transmission, an appropriate indicator is set to communicate this fact to the ground. The poll command, "GPC HAS DATA", is sent to the ground and a test is made to determine if a valid response has been received. If a valid response is received and the ground is ready to receive the GPC data, the message will be transmitted via the "transmit sequence" which is described later. If the ground responds by indicating that a ground message is pending, the GPC will yield and the ground message will be received via the "receive sequence" which is also described later.

The other possible valid response which can be received after the output poll command has been issued is a response which effectively means the ground cannot accept the message. In this case, the communication cycle is terminated and the output message will remain pending.

The "input poll sequence" is executed if the GPC does not have a message pending for transmission. The LDB I/O Processor will issue the "interrogate poll" command which prepares the ground to transmit any message pending. If the ground response is valid and indicates a message is pending, the message is received by performing the "LDB receive message" sequence. Otherwise, if any other valid response is received, the communication cycle is terminated. Presumably the ground does not have message traffic pending.

If in either of the above poll sequences any error occurs, the communication cycle is terminated and the error is processed to determine if LDB bus switching is required.

The "LDB receive sequence" is executed if during the output or input poll sequences, the ground indicated a ground message is pending. A "go ahead"command is issued to the ground and the message is received. A series of error checks are made which include sumcheck, transaction ID, message destination and whether the application can accept the message. (Note - TCS-1, TCS-S, and Mass Memory functional destinations are valid for vehicle checkout software configurations only). If errors are detected appropriate status indications are set which subsequently will be sent to the ground. A final check is made to determine if the incoming message is a DEU equivalent message and, if so, MCDS User Interface polling is disabled for the addressed DEU. This check and resultant action is taken based upon asynchronous arrival of LDB messages and DEU polling processes.

Upon completion of the input message error checks, status is transmitted to the ground. A housekeeping check is then made to determine if the input message contained errors and, if so, the errors will be processed by calling the LDB Error Processor. In this case the input message is not saved and is lost as far as the GPC is concerned. If no errors were encountered, error counters are reset and the LDB Message Router is called to complete message routing and application processing.

The message "transmit sequence" process is executed based upon an affirmative ground response received in the "output poll" sequence. A 'here comes the GPC data' command, message and message sumcheck are transmitted followed by a status request command. Any response from the ground denotes a successful communication cycle. Error counters are reset and the LDG Output_Contention_Event reset. The Output_Convention_Event being reset will allow the LDB_Output_Message_Coordinator to process additional messages destined for ground transmission.

If the GPC fails, after having requested status, to receive a response from the ground, an indicator is set which will cause retransmission of the message on subsequent communications cycles. The error will then be processed by the LDB Error Processor.

The LDB Error Processor which is called at various points in the above sequences, increments the LDB_Error_Count and tests to determine if three consecutive errors have occurred. If so, the LDB_Error_Count is reset and the alternate launch data bus is configured as the active bus. The bus switching is performed by the FCOS bus reconfiguration function.

The control flow for this module is presented in Figure 3.1.2.1-1.

d. Outputs - See table 3.1.2.1 - 1

e. Module References -

1. (200) I/O_SVC_Service_Processor (FIOSVC) is CALLed.
2. (291) Checksum_Generator(#CFIOCGR) is CALLed.
3. (350) Bus_Reconfiguration_Pre-Processor(FCMBMAN) is CALLed.
4. (171) Reset_Event_Processor(FPMRESET) is CALLed.
5. (170) Set_Event_Processor(FPMSET) is CALLed.
6. SACS Input Processor (VSI_SACS_IN) is SCHEDULED.
7. TCS Single Command Processor (V01_TCS_SC) is SCHEDULED.
8. MM Message Processor (DMP_MM_MSG_PROC) is SCHEDULED.
9. TCS Sequence Acquisition Processor (V02-TCS_SA) is SCHEDULED.

f. Module Attributes - Program

g. Template References -

1. MM_Message_Processor (DMP_MM_MSG_PROC)
2. MM_Utility_Compool_1(CDH_MM_Utility)
3. MM_Utility_Compool_2(CDI_MM_Utility)
4. SACS_GMEM_Compool (CVA_SACS_GMEM_COMPOOL)
5. TCS_Compool(CVT_TCS_UNIQUE)
6. SACS_Input_Processor(VSI_SACS_IN)
7. TCS_Single_Command_Processor(V01_TCS_SC)
8. TCS_Sequence_Acquisition_Processor (V02_TCS_SA)
9. LDB_Compool (CDV_COMPOOL)
10. VCO_Shared_Compool(CVS_SHARED_COMPOOL)
11. Checksum generator (#CFIOCGR)
12. UI_Section_Of_Common_Compool(CZ1_COMMON)
13. UI_FCOS_Shared_Compool (CZ2_COMMON)
14. FCOS_Compool(FCMCOM)

h. Error handling - The following error checks are performed on each GND to GPC message:

1. Transmission errors
   (a) Parity
   (b) GND computer address
   (c) Pattern Check
   (d) Message length (time out)

2. Sumcheck error.

3. Invalid operation code (in Interrogate Response Word)

4. Duplicate transaction ID

5. Invalid data request (functional destination not valid for software

configuration or invalid transaction ID on TCS-S continuation)

6. Addressed functional destination cannot accept data (not an error but message is discarded)

7. TCS resolving linkage (not an error, but message is discarded)

A GPC to ground "status" command is sent in response to each of the above errors. In addition, a count of consecutive transmission/sumcheck errors is maintained. When this count reaches three, the FCOS_Bus_Reconfiguration_Pre-processor is invoked to switch the alternate LDB to the active state.

i. Constraints and Assumptions - Messages containing the function destination codes for the EIU interface and LS interface will be treated as an "invalid data request" and appropriate status will be transmitted to the ground. Messages of more than 516 words and messages with undefined destinations will not be transmitted by the ground.

j. Detailed Implementation - None.

DATA TABLE 3.1.2.1-1

NAME LDB_I/O_Processor (DGI_LDB_IO)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	Interrogate_Response_Status	V150	IO	440	440	CDVV_LDB_IRW_TSW			
2	VCO_OPS_Flag	J120	I		440,460	CZ1B_V_VALID_OPS			
3	Input_Buffer_ Address	V210	L	440	440	DGI_IBUF_ADDR			
4	VCO_Input_Buffer	V230	I	440	440	CVSV_LDB_IBUF			
5	IO_Status_Pointer	V250	L	440	440	DGI_IOERR_CODE			
6	VCO_Transaction_Status_Word	V260	IO	440	440	CVSV_LDB_TSW			
7	LDB_Input_Buffer	V040	I	440	440	CDVV_LDB_FSW_IBUF			
8	LDB_Transaction_Status_Word	V010	IO	440	440	CDVV_LDB_FSW_ITSW			
9	Output_Pending_Flag	V190	IO	440,460	440,460	CDVB_XMIT_FLAG			
10	Interrogate_Word_Count	V580	L	440	440	DGI_IWD_PARMS.DDCNT			
11	Output_Message_Length	V100	IO	440,460	440	CDVV_OUT_MSG_LENG			
12	Interrogate_Response_Code	V180.05	I	440	440	CDVV_OPCODE			
13	Input_Message_Status_Word	V120	IO	440	440	CDVV_LDB_STAT			
14	Status_Mask	V590	L	440	440	DGI_STAT_PARMS. DDMASK			
15	VCO_Send_Word_Count	V600	L	440	440	DGI_VCO_TREN_PARMS. DWDCT			
16	VCO_Send_NZB_Count	V610	L	440	440	DGI_VCO_TREN_PARMS. DDCNT			
17	LDB_Send_Word_Count	V620	L	440	440	DGI_TREN_PARMS.DWDCT			
18	LDB_Send_NZB_Count	V630	L	440	440	DGI_TREN_PARMS.DDCNT			
19	Interrogate_Error_Pointer	V170	IO	440	440	CDVV_LDB_IRW_IERR			
20	Error_Data_ Structure	V270	L	440	440	DGI_ERR_STRUC			

DATA TABLE 3.1.2.1-1 (Cont.)

NAME LDB_I/O_Processor (DGI_LDB_IO)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
21	Error_Data_Pointer	V270.05	L	440	440	DGI_ERR_PTR			
22	LDB_Error_Count	V090	IO	440	440	CDVV_ERR_COUNT			
23	Output_Contention_Event	V200	IO	440,460	440,460	CDVE_LDB_ACTV			
24	IO_Status_Pointer	V250	L	440	440	DGI_IOERR_CODE			
25	Input_Message_Length	V110	IO	440	440	CDVV_IN_WDCOUNT			
26	Ground_Message_Word_Count	V180.10	IO	440	440	CDVV_IN_MSG_LENG			
27	VCO_Receive_Word_Count	V640	L	440	440	DGI_VCO_GO_PARMS.DWDCT			
28	LDB_Receive_Word_Count	V650	L	440	440	DGI_GO_PARMS.DWDCT			
29	Checksum_Save_Area	V280	L	440	440	DGI_TEMP_SUMCHECK			
30	End_Message_Pointer	V290	L	440	440	DGI_ENDDATA			
31	Input_Checksum_Count	V300	L	440	440	DGI_SUMCHECK_COUNT			
32	Current_Transaction_ID	V310	L	440	440	DGI_CURR_TRANSACT			
33	Input_Transaction_ID	V040.10	I	440	440	CDVV_TRANSACT_ID			
34	Current_Message_Destination	V320	L	440	440	DGI_FUNC_DEST			
35	Input_Destination	V040.05	I	440	440	CDVV_DEST			
36	Current_DEU_Number	V330	L	440	440	DGI_DEU_ID			
37	LDB_DEU_Number	V040.25	I	440	440	CDVV_LDB_DEU			
38	Old_Transaction_ID	V130	IO	440	440	CDVV_OLD_TRANSACT			
39	Masked_Status_Word	V340	L	440	440	DGI_ERR_TEST			
40	Error_Mask	V350	L		440	DGI_ERR_MASK			

DATA TABLE 3.1.2.1-1 (Cont.)

NAME LDB_I/O_Processor (DGI_LDB_IO)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
41	TCSS_Transaction_ID	V140	IO	440	440	CDVV_TCSS_TRANSACT			
42	GPC_ID	#001	I	300	See App E	TFCMID			
43	GPC_Prime_ID	I050	I	351,700 820,880	See App. E	CZ2V_GPC_P		X	
44	IOP_Transmitter_State	I010.07	I	355,830	440,660 665,830	CZ2V_ACT_XMITR2			
45	LDB_Command_Bus	V660	L	440	440	DGI_CMD_MODE.VAR_DATA1			
46	LDB_Listen_Bus	V670	L	440	440	DGI_LSTN_MODE.VAR_DATA1			
47	LDB_Prime_ID	V680	L	440	440	DGI_LSTN_MODE.VAR_DATA2			
48	Current_OPS	I010.02	I	560, 820,880	See App. E	CZ2V_CURRENT_OPS			
49	IOP_Receiver_State	I010.08	I	335 355	400,440 815,830	CZ2B_ACT_RECVR			
50	TCS_Busy_Flag	V360	IO	440	440	CVSA_BUSY			
51	TCS_Buffer_Ready	V370	I		440	CVAA_READY			
52	TCS_Block_Flag	V380	IO	440	440	CVSA_SAP_INPUT			
53	SACS_Input_Buffer	V390	O	440		CVAV_LDB_DATA			
54	TCS_Input_Buffer	V400	O	440	440 460	CVSA_BUFFER			
55	MM_Header	V410	IO	440	440	CDHV_REQUEST			
56	MM_Patch_Header	V430	O	440,460	440,460	CDHV_PATCH_REQUEST			
57	MM_Operation_Code	V040.15	IO	440	440	CDVV_MM_OPCOD			
58	MM_Input_Block_Index	V500	IO	440	440	DLM_MM_BLOCKID			
59	MM_Block_Number	V040.35	IO	440	440	CDVV_LDB_MM_BLOCK			
60	MM_Read_Write_Buffer	V450	IO	440,460	440,460	CDHV_BLOCKS			

DATA TABLE 3.1.2.1-1 (Cont.)

NAME LDB_I/O_Processor (DGI_LDB_IO)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
61	Initial_MM_Router_Header_Word	V420	O	440	440	CDHB_RHW_ST			
62	MM_Router_Header_Word	V410.05	IO	440	440	CDHV_RHW			
63	MM_Patch_Word_Count	V470	O	440	440	CDHV_LDB_COUNT			
64	MM_Block_Count	V410.15	IO	440	440	CDHB_N_BLK			
65	Equivalent_Header	I720.02	O	440	400	CZ2B_DIT_HDR			
66	Equivalent_DEU_Number	I730	O	440	400	CZ2V_EQ_DEUNO			
67	LDB_Equivalent_DEU_Message_Event	I700	O	400,440	400	CZ2E_LDB_EQ_EVT			
68	TCS_Sequence_Buffer	V510	IO	440	440,460	CVSA_DATA			

Table 3.1.2.1-2

DEVICE	DEVICE ID	OP CODE	SVC NO.	SYNC TYPE	COMFAULT STATUS	ICC TYPE	TIME TAG	WAIT	PROT.	STATUS	INPUT OUT-PUT	MAJ FUNC. ID	I/O PRIOR. ITY	WORD COUNT	BUFFER	EVENT	BUS NOMINAL
LDB	36	1	24	RS	No	No	No	Yes	Yes	Yes	I	N/A	230	1	TPCMLI11 CDVV_LDB_IRW	N/A	22,23
LDB	36	2	24	RS	No	No	No	Yes	Yes	Yes	I	N/A	230	1	CDVV_LDB_IRW	N/A	22,23
LDB	36	3	24	RS	No	No	No	Yes	Yes	Yes	I	N/A	230	VAR	CDVV_LDB_PSW_IBUF,CVSV_LDB_IBUF	N/A	22,23
LDB	36	4	24	RS	No	No	No	Yes	Yes	No	O	N/A	230	VAR	CDVV_LDB_PSW_OBUF, CVSV_LDB_OBUF	N/A	22,23
LDB	36	5	24	RS	No	No	No	Yes	Yes	Yes	I	N/A	230	1	TPCMLSR1 CDVV_LDB_IRW	N/A	22,23
LDB	36	6	24	RS	No	No	No	Yes	Yes	No	O	N/A	230	1	CDVV_LDB_STAT	N/A	22,23

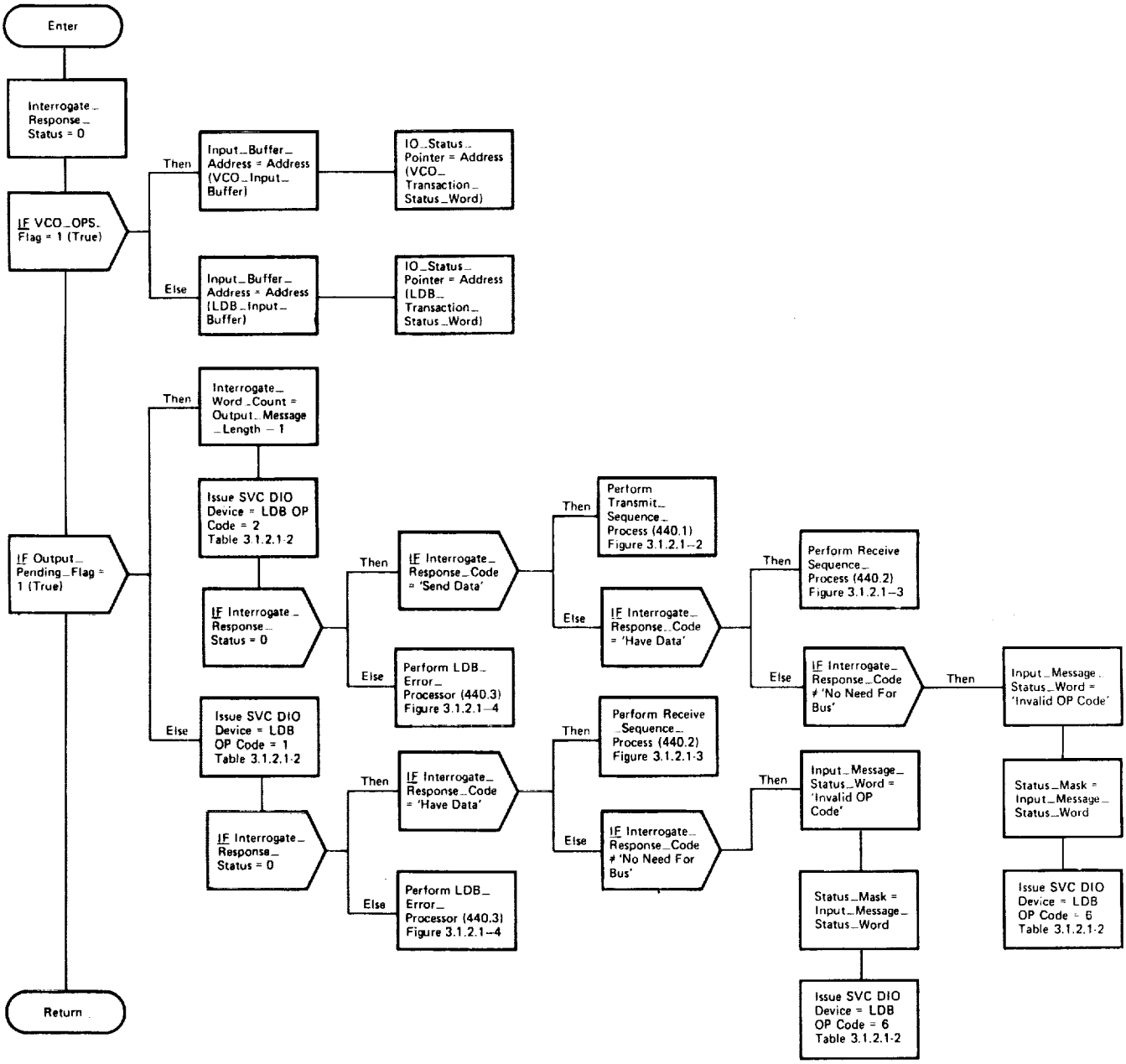


Figure 3.1.2.1-1. LDB_IO_Processor (DGI_LDB_IO)

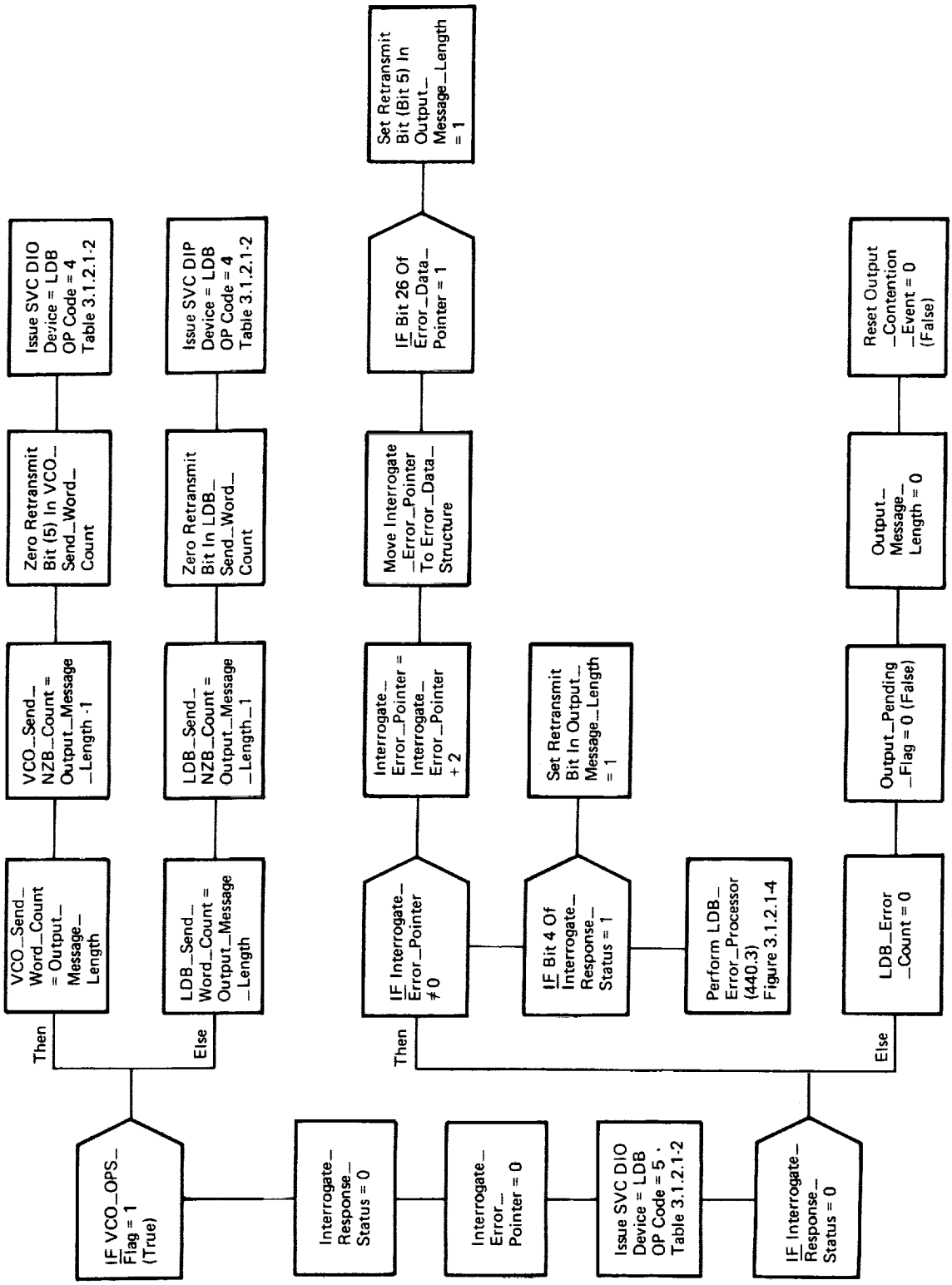


Figure 3.1.2.1-2. LDB_I/O_Processor Transmit_Sequence_Process (440.1)

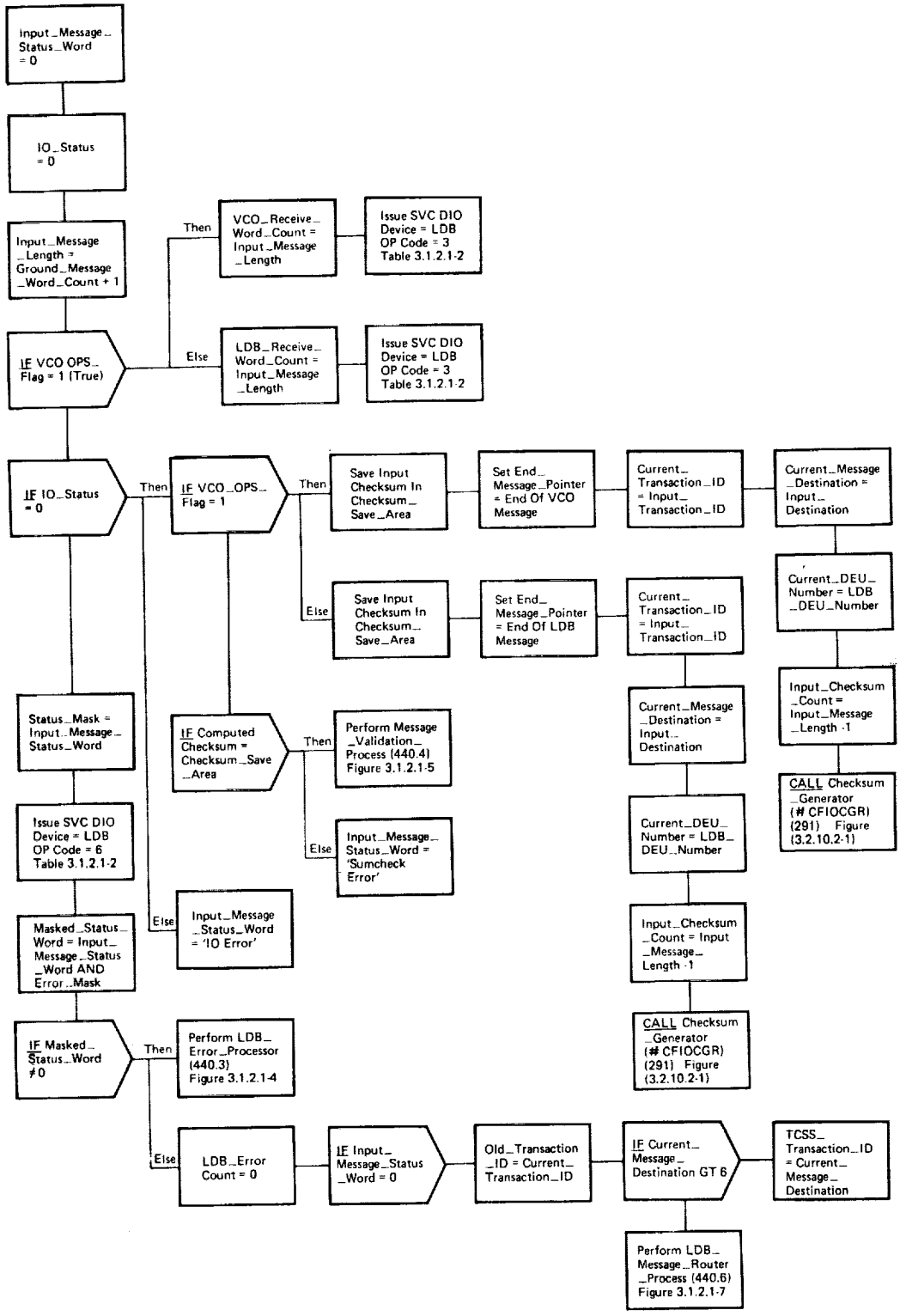


Figure 3.1.2.1-3. LDB_IO_Processor Receive_Sequence_Process (440.2)

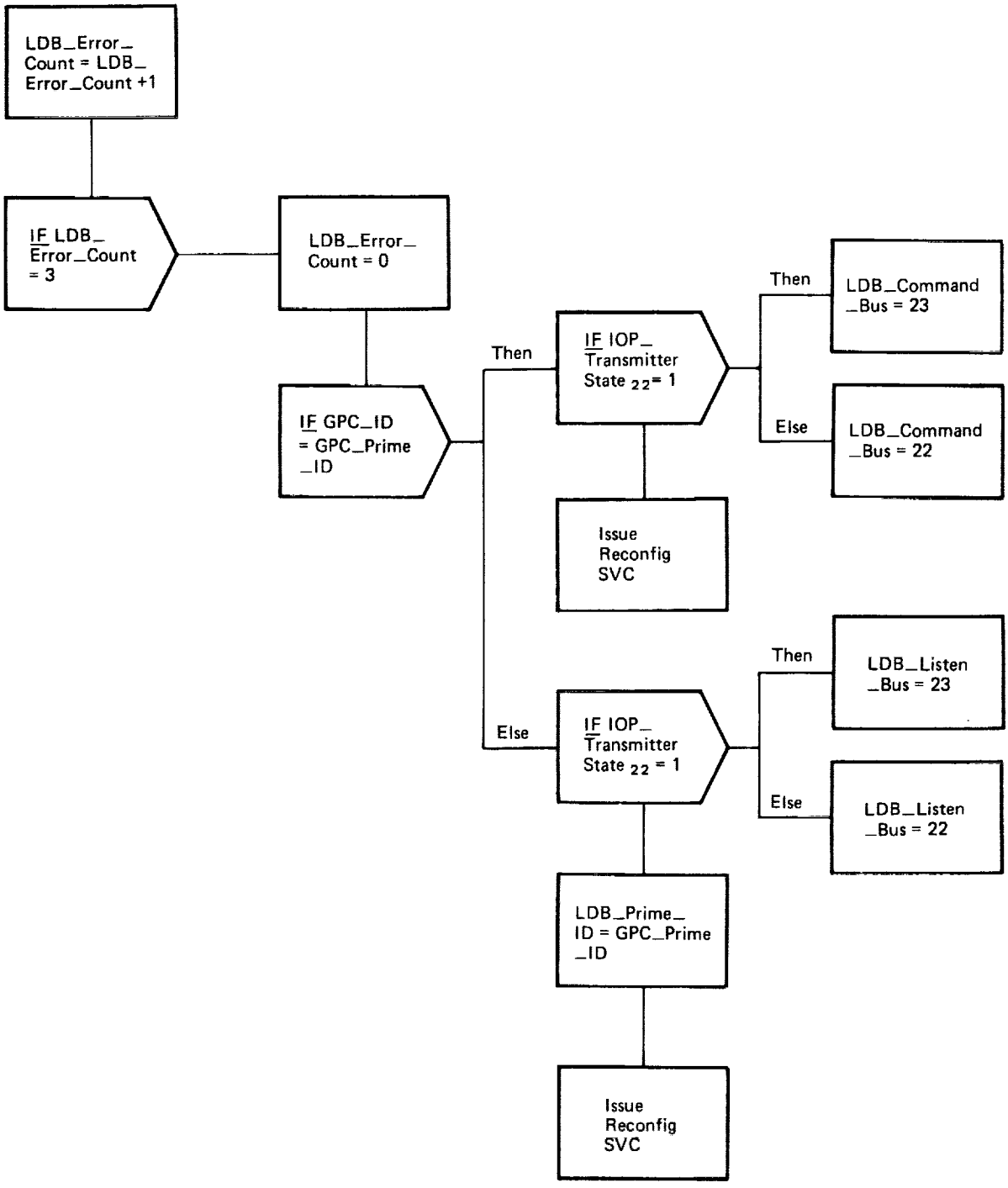


Figure 3.1.2.1-4. LDB_IO_Processor LDB_Error_Processor (440.3)

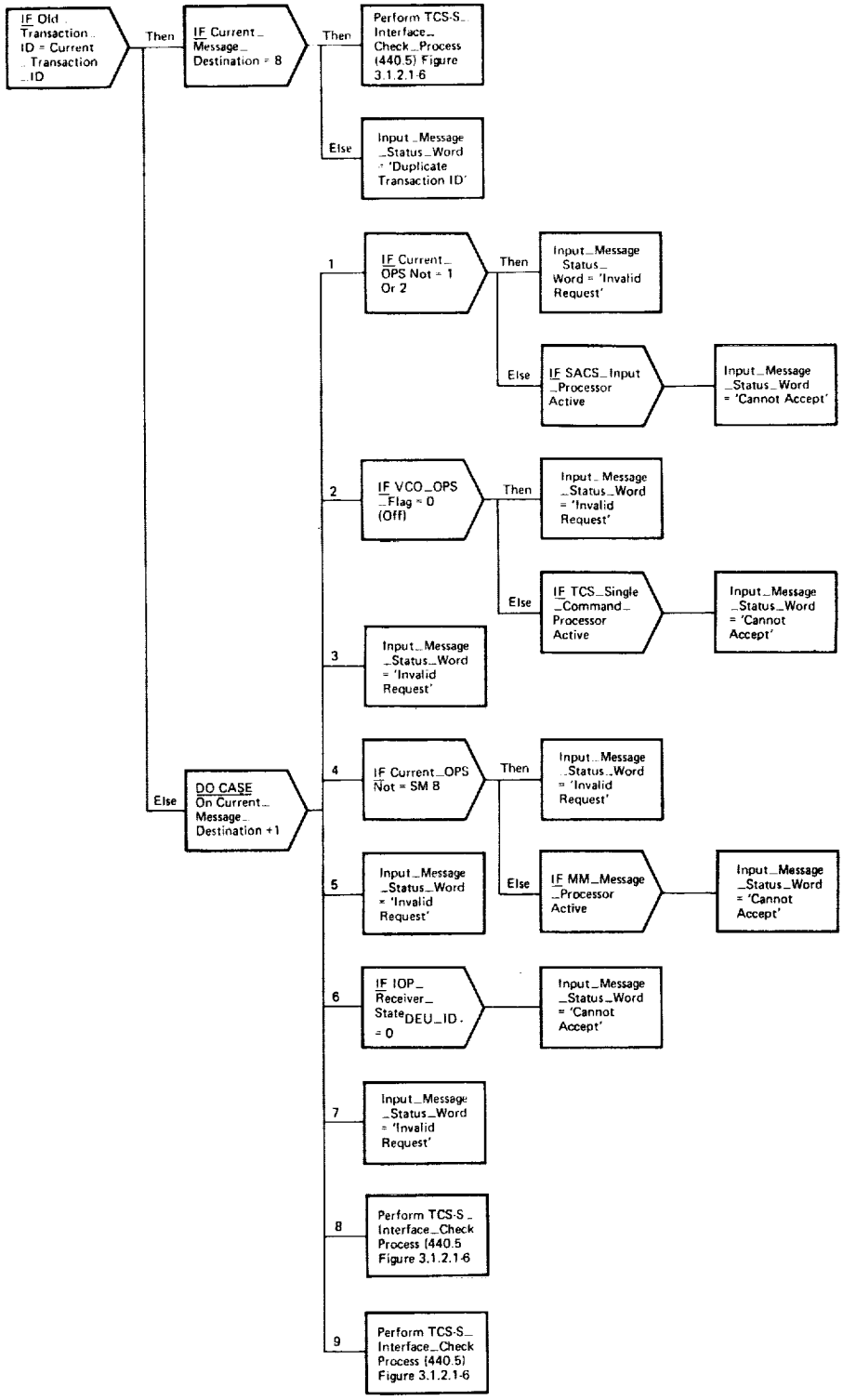


Figure 3.1.2.1-5. LDB_I/O_Processor Message_Validation_Process (440.4)

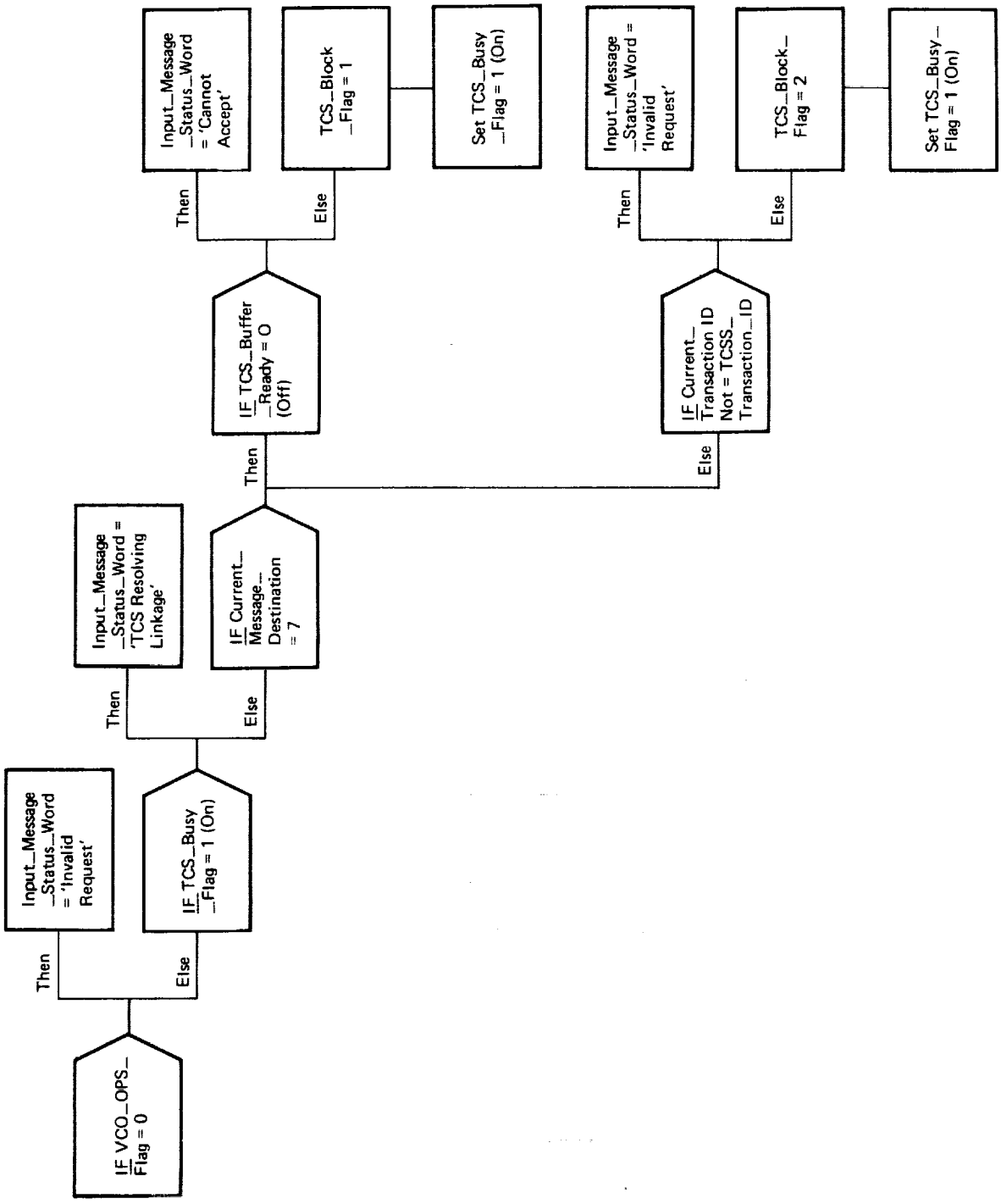


Figure 3.1.2.1-6. LDB_I/O_Processor TCSS_Interface_Check_Process (440.5)

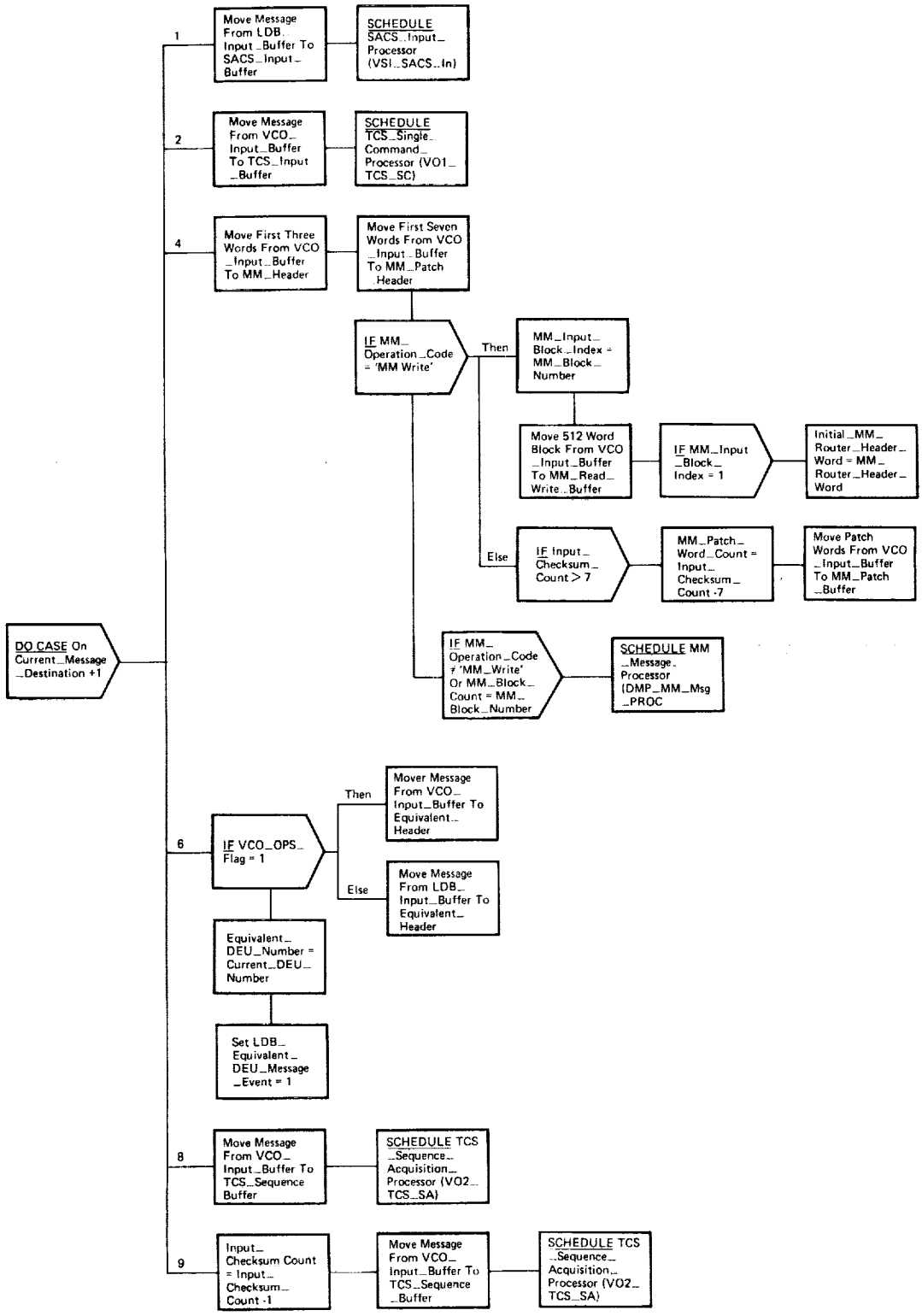


Figure 3.1.2.1-7. LDB_I/O_Processor
LDB_Message_Router_Process (440.6)

3.1.2.2 LDB_Output_Message_Coordinator (DGO_LDB_COORD)(460)

The LDB Output Message Coordinator accepts requests from applications programs to process a message for ground transmission. The message is moved from the applications buffer to the LDB Output Buffer and an event is set to initiate transmission of the message by the LDB I/O Processor.

a. Control Interface
   1. Call DGO_LDB_COORD (ID,Word Count)
   2. CALLed by
      (a) Frequency Response Test Waveform generator (V21_WAVEFORM)
      (b) TCS Single Command Processor (VO1_TCS_SC)
      (c) TCS Sequence Acquisition Processor (VO2_TCS_SA)
      (d) Mass Memory Message Processor (DMP_MM_MSG_PROC)
      (e) SACS_Response_Formatter (VS3_RESP_XMITTER)
      (f) RAMP_Call_Processor (VRP_RAMP_CALL)

b. Input - See Table 3.1.2.2 - 1

c. Process Description - The LDB Output Message Coordinator first WAITS for LDB Output_Contention_Event to be false. When the LDB Output_Contention_Flag becomes false, normal processing continues. The output contention event is set to indicate the LDB_Output_Buffer contains (or will contain) an active message. The message is then moved from the application buffer. A sumcheck is performed after the message is moved. Upon completion of the move and sumcheck, the Output_Pending_Flag is set which advises the LDB I/O Processor that a message is ready for transmission to the ground. The control flow for this processor is presented in Figure 3.1.2.2 - 1

d. Outputs - See Table 3.1.2.2 - 1

e. Module References -

   1. (291) Checksum_Generator (#CFIOCGR) is CALLed.
   2. (104) Wait_Processor (FPMWAIT) is CALLed.
   3. (170) Set_Event_Processor (FPMSET) is CALLed.

f. Module attributes - Exclusive External Procedure.

g. Template References -
   1. LDB_Compool (CDV_COMPOOL)
   2. VCO_Shared_Compool(CVS_SHARED_COMPOOL)
   3. MM_Utility_Compool_1(CDH_MM_UTILITY)
   4. SACS_GMEM_Compool (CVA_SACS_GMEM_COMPOOL)
   5. MM_Utility_Compool_2(CDI_MM_UTILITY)
   6. TCS_Compool(CVT_TCS_UNIQUE)
   7. Ramp_Compool(CVR_RAMP_COMPOOL)

8. UI_Section_Of_Common_Compool(CZ1_COMMON)

h. Error Handling - None.

i. Constraints and Assumptions.
   1. The message length parameter passed will be in the valid range for the CALLing application.
   2. Input buffer ID's passed will be valid.

j. Detailed Implementation - None.

DATA TABLE 3.1.2.2-1

NAME LDB_Output_Message_Coordinator (DGO_LDB_COORD)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	Output_Contention_Event	V200	IO	440,460	440,460	CDVE_LDB_ACTV			
2	Output_Buffer_ID	V520	L		460	DGO_BUFFER_ID			
3	TCS_Input_Buffer	V400	I	440	440,460	CVSA_BUFFER			
4	VCO_Output_Buffer	V240	O	460	440,460	CVSV_LDB_OBUF			
5	SACS_Response	V540	I		460	CVAV_RESPONSE_DATA			
6	Waveform_Generator_Error_Response	V550	I		460	CVAV_V_ERR_RESP			
7	MM_Response	V440	I		460	CDHV_PATCH_RESPONSE			
8	Output_Word_Count	V530	L		460	DGO_WDCOUNT			
9	MM_Request	V410.20	I	440	440,460	CDHB_OPCD			
10	MM_Output_Block_Index	V490	O	460	460	DGO_MM_BLOCK			
11	MM_Input_Block_ID	V410.10	I	440	440,460	CDHB_NDX			
12	MM_Read_Write_Buffer	V450	I	440,460	440,460	CDHV_BLOCKS			
13	Ramp_Error_Response	V560	I		460	CVRV_ERROR_RESPONSE_DATA			
14	LDB_Output_Buffer	V080	IO	460	440,460	CDVV_LDB_OUT_DATA			
15	Output_Checksum_Count	V570	L	460	460	DGO_SUMCHECK_COUNT			
16	VCO_OPS_Flag	J120	I		440,460	CZ1B_V_VALID_OPS			
17	Output_Message_Length	V100	IO	440,460	440,460	CDVV_OUT_MSG_LENG			
18	Output_Pending_Flag	V190	IO	440,460	440,460	CDVB_XMIT_FLAG			

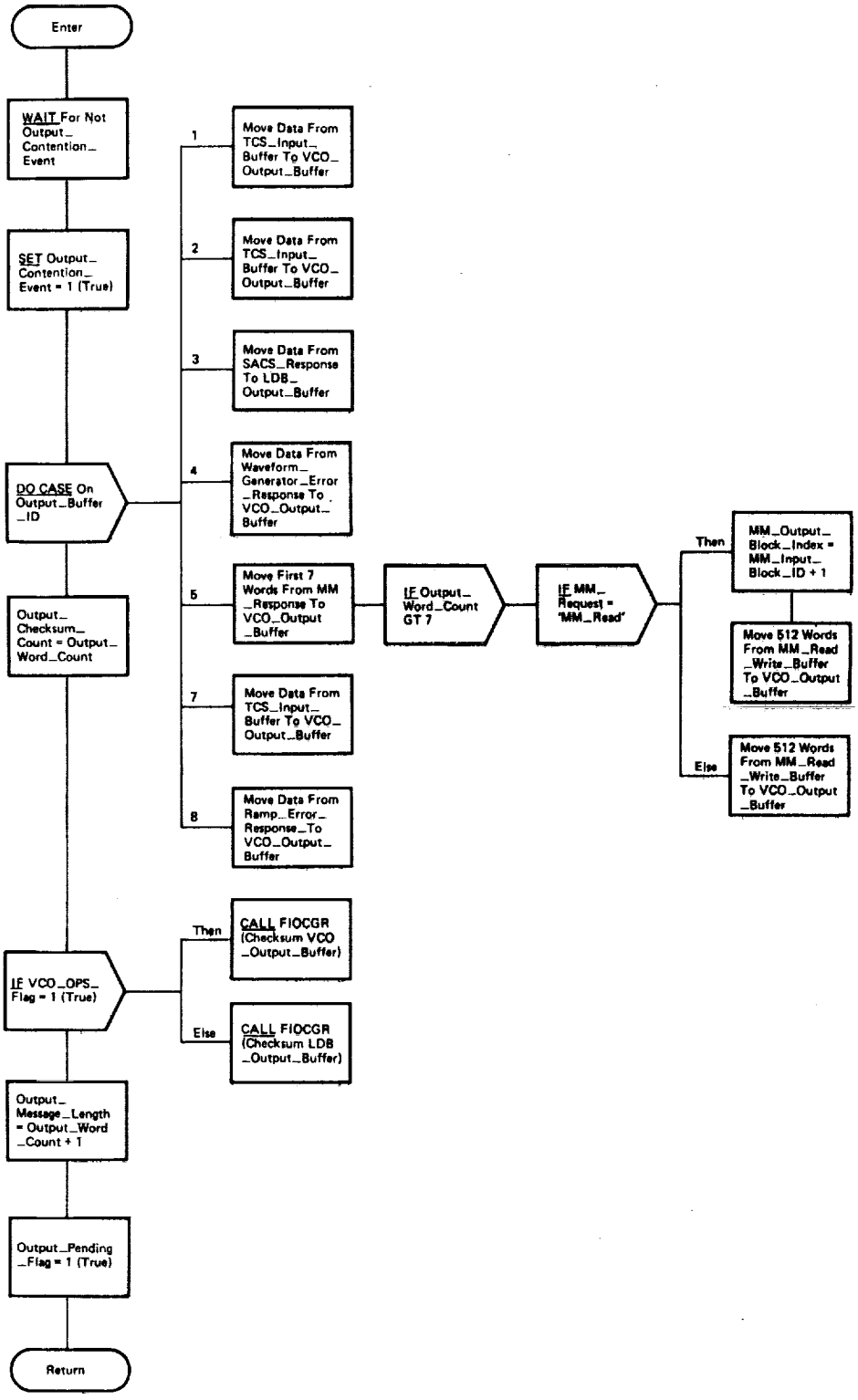


Figure 3.1.2.2-1. LDB_Output_Message_Coordinator (DGO_LDB_COORD)

3.1.3 ICC Interface

These processes are designed to provide common support for ICC messages required to be transferred amoung the common set GPCs at common sync time and in support of System_Interface_Processor (SIP) functions. System Services may communicate across the common set of GPCs only at the time of common sync and as a function of System_Interface_Processor. The exception to this is in the event of GPC redundancy management fault detection and isolation, and I/O problem reporting in the event of I/O anomalies.

A fixed size buffer (64 fullwords) is provided for System Services to buffer their individual messages. Messages accumulated between SIP cycles are merged into a single message. The buffer is transferred to all other GPCs in the common set during SIP activity. Four input buffers (each of size 64 fullwords) are provided in each GPC to receive the ICC messages from the other GPCs. All GPCs write on their ICC channels at the same time (during SIP). Upon I/O completion the ICC message Router unpacks the messages in each buffer and makes them available to the processes for which they are destined.

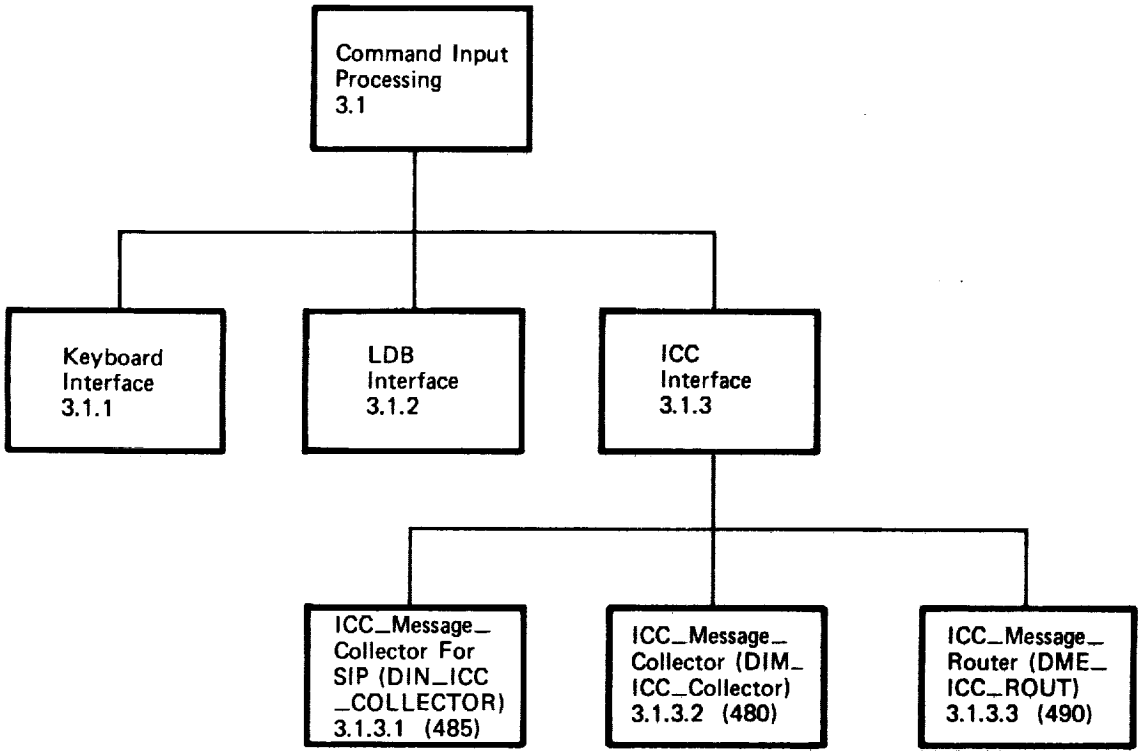


Figure 3.1.3-1 ICC Interface Hierarchy Diagram

3.1.3.1 ICC_Message_Collector_For_SIP (DIN_ICC_SIPCOLL) (485)

This module is called to move ICC messages passed from the Caller to the ICC Message buffer appropriate for its GPC.

a. Control Interface -

1. CALL DIN_ICC_SIPCOLL (PASSED_ARRAY) ASSIGN (ASSIGN_VAL)

2.(a)CALLed by (670) Fault_Message_Scan (DMR_FMS)
(b)CALLed by (750) System_Interface_Processor (AIE_SIP)

b. Input - See Table 3.1.3.1-1

c. Processor Description - The ICC_Message_Collector_For_SIP checks the status of the ICC_Buffer_Freeze_Flag and determines the caller. When the ICC_Buffer_Freeze_Flag specifies a "Frozen" condition and the caller is not the System_Interface_Processor, the Assign_Request_Status is set to HEX '01' and this procedure returns to the caller. Otherwise processing continues. Further Processing consists of testing for initial non-frozen condition, determining if the passed message will fit in the remaining ICC_Messages space, testing the Message_Header for reasonable values, and finally processing the Message.

When Initial_Value is zero, the ICC_Address table structure is loaded with the addresses initialized in the Name Table structure, and the GPC_ICC_Buffer_Name Address is loaded with the address of the proper copy of the ICC_Messages array for this GPC. This provides efficient structure copy indexing

When the passed message will not fit in the remaining ICC_Messages space, the Assign_Request_Status is set to HEX '02' and this procedure returns to the caller.

When the message will fit in the ICC_Message space the passed message parameters are checked for reasonable values. If the Message_Index is not in the range of one (1) to twenty eight (28) or the Message_Size is zero (0), the Assign_Request_Status is set to HEX '03' an this procedure will return to the caller. Otherwise the message is processed.

This procedure processes two type of messages, direct data messages and indirect data messages. Direct data messages are those messages where all data for the message is contained in the array passed by the caller, and indirect messages are those messages where the array passed by the caller contains an index to the data to be moved.

The ICC_Message_Index selected from the first word of the array passed by the caller is used as an index to select the proper entry in the ICC_Message_Table. The ICC_Message Table contains a Routing_Code value and this value determines which type of message is to be processed.

The proper location to store the passed messaged is the ICC_Messages buffer for this GPC (i.e., GPC 1 uses Buffer 1 GPC 2 user Buffer 2 etc) indexed by the value ICC_Buffer_Pointer.

When the Routing_Code is less than 14, the message is a direct data message type and the message contents are moved from the array passed by the caller to the proper ICC_Message location.

When the Routing_Code equals 14 or 15 the message is an indirect data message type and the second word of the passed message contains a value called the Address_Index. The Address_Index is used to select the proper data source address from the ICC_Address_Name table.

When the Routing_Code 14, the Message_Header is moved to the proper ICC_Message location followed by the data as specified by the selected data source address.

When the Routine_Code is 15, the Message_Header and the Address_Index is moved to the proper ICC_Message location followed by the data as specified by the selected data source address.

When message processing is completed, the Assign_Request_Status is set to HEX '00' to indicate message processed status, the ICC_Buffer_Pointer is loaded with the value reflecting where the next data message is to be placed, and this ICC_Message location is loaded with a HEX 'FFFF' to indicate the logical end of the ICC_Messages. This procedure then returns to the caller.

The control flow for this procedure is presented in Figure 3.1.3.1-1 and Figure 3.1.3.1-2.

d. Output - See Table 3.1.3.1-1.

e. Module References - None

f. Module Attribute - External Procedure

g. Template Reference -

1. UI-FCOS_Shared_Compool (CZ2_COMMON)
2. UI_General_Compool (CDM_UI_COMPOOL)
3. FCOS_Compool (FCMCOM)

h. Error Handling

The status of each call to this procedure is returned to the caller in bits 9 to 16 of the Caller_Assign_Value.

When the ICC buffer is set frozen and the caller is not the System_Interface_Processor the status is set to Hex '01', the message is not processed, and this procedure returns to the caller.

When the message passed to this procedure will not fit into the ICC buffer the status is set to Hex '02', the message is not processed, and this procedure returns to the caller.

When the message passed to this proceudre contains an value less than one(1) or greater than twenty-eight (28) in bits 1 to 8 of the first word of the passed message, or contains a value of zero (0) in bits 10 to 16 of the first word of the passed message, the status is set to Hex '03', the message is not processed, and the procedure returns to the caller.

When the message has been processed the status is set to Hex '00' and the procedure returns to the caller.

i. Constraints and Assumptions

This procedure expects callers to identify themselves correctly, that is, the System_Interface_Processor (AIE_SIP) is the only caller that has bits 1 to 8 of its Assign_Value set to Hex '00'.

DATA TABLE 3.1.3.1-1

NAME ICC_Message_Collector_For_SIP (DIN_ICC_SIPCOLL)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	Assign_Value (Passed Parameter...Defined in DIN_ICC_SIPCOLL but exists elsewhere)	L103	L,I	485,670 750	485,670 750	DIN_ASSIGN			
2	Assign_Caller (BITS 1-8 of DIN Assign Val)	L104	L,I	670,750	485,670 750				
3	ICC_Buffer_Freeze_Flag	I420	I	361,490	480,485	CZ2V_ICC_BUF_FRZ			
4	Assign_Request_Status (BITS 9 to 16 of Assign_Value)	L105	L,O	485	485,670 750				
5	Initial_Value	L107	L	485	485	DIN_INITIAL			
6	ICC_Address_Table	I650	O	485	485,490	CZ2V_LNAMES			
7	Name_Table	L102	L	485		DIN_ICC_NAMES			
8	GPC_ICC_Buffer_Name	L112	L	485	485	DIN_NICCBUF			
9	ICC_Messages	I610.05	O	480,485 490	See APP. E	CZ2B_ICC_MSG_BUF			
10	GPC_ID	#001	I	300	See APP. E	TFCMID			
11	Pointer	L108	L	485	485	DIN_BUF_POINTER			
12	ICC_Buffer_Pointer	I430	IO	480,485 490	480,485 750	CZ2V_ICC_BUF_ POINT			
13	Message_Header	L109		485	485	DINB_BITVAL			
14	Passed_Array (Passed parameter...defined in DIN_ICC_SIPCOLL but exists elsewhere)	L106	LI	670,750	485 670 750	DIN_ARRAY			
15	Message_Size	L110	L	485	485	DIN_MSG_SIZE			
16	Message_Index	L111	L	485	485	DIN_MSG_INDEX			
17	Routing_Code	L113	L	485	485	DIN_RCODE			
18	I	L116	T	485	485	I			
19	Address_Index	L114	L	485	485	DIN_ADRINX			
20	Address_Name	L115	L	485	485	DIN_Q			

DATA TABLE 3.1.3.1-1 (cont'd)

NAME ICC_Message_Colletn_For_SIP (DIN_ICC_SIPCOLL)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
21	ICC_Address_Name	I650.01	I	485	485,490	CZ2V_ICC_NM			

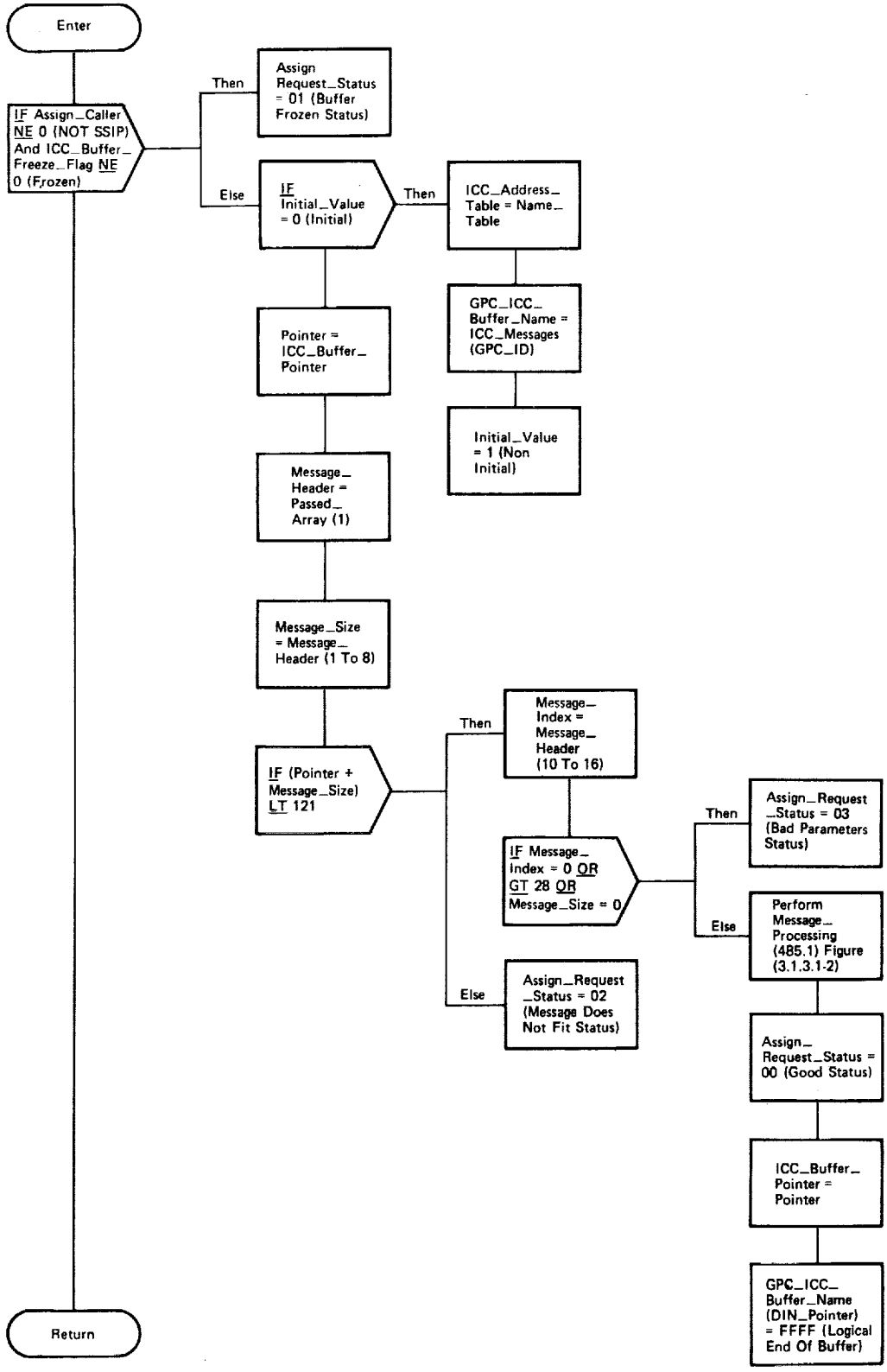


Figure 3.1.3.1-1. ICC_Collector_For_SIP (DIN_ICC_SIPCOLL)

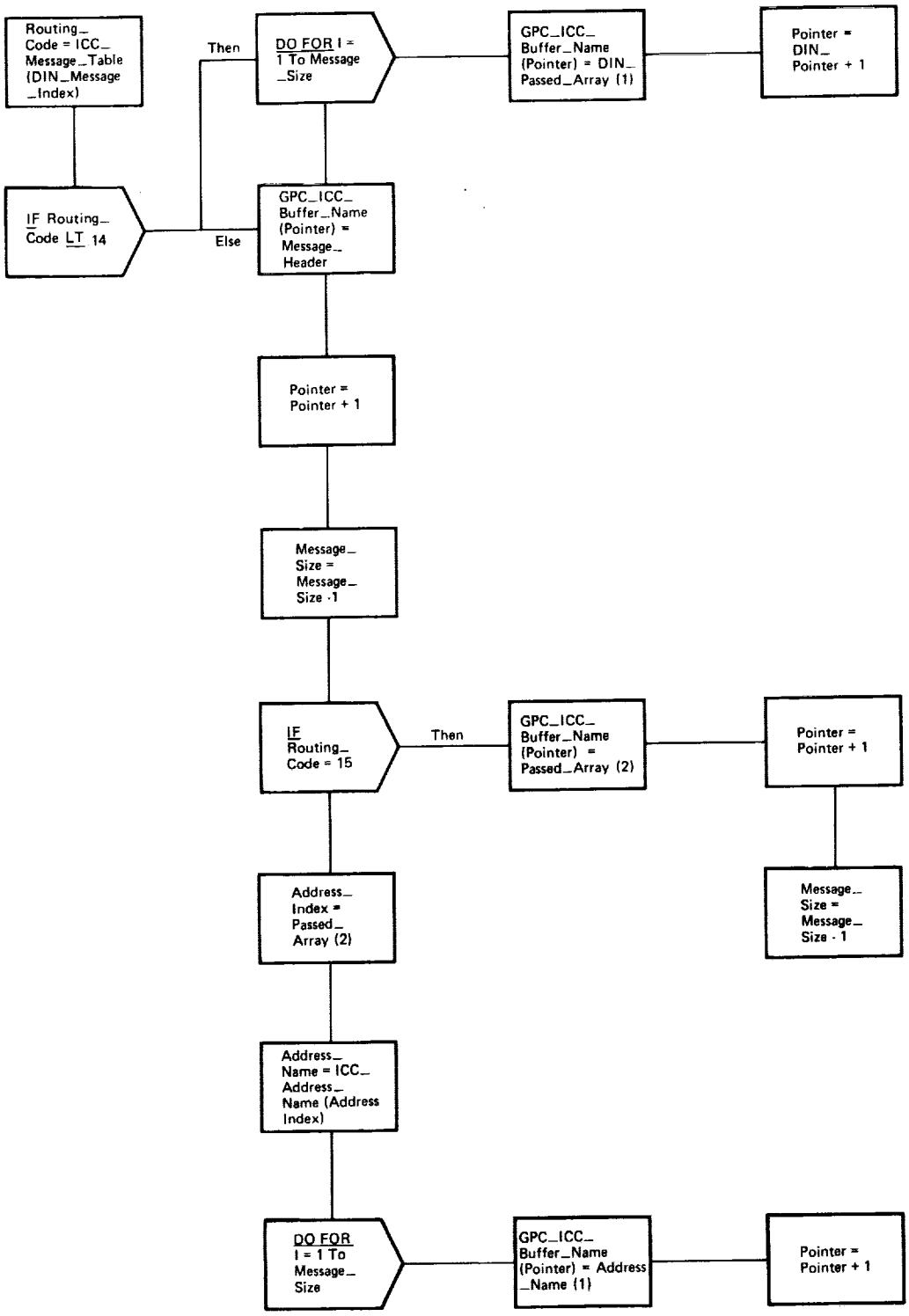


Figure 3.1.3.1-2. ICC_Collector_For_SIP Message_Processing (485.1)

3.1.3.2 ICC_Message_Collector (DIM_ICC_COLLECTOR) (480)

To be provided.

3.1.3.3 ICC_Message_Router (DME_ICC_ROUT)(490)

This module dequeues messages in each ICC Message Buffer by processing each message as specified by the ICC Message Table entry.

a. Control Interface - 1. CALL DME_ICC_ROUT
2. CALLed by (750) System_Interface_Processor (AIE_SIP)

b. Input - See Table 3.1.3.3-1

c. Process Description - In order to understand this module, here is a brief ICC review. Each GPC has 4 buffers, one for each of the ICC receiving channels. Buffer 1 contains the input messages for GPC 1., Buffer 2, for GPC 2 and so on. Each buffer contains up to 120 words of ICC messages. Each ICC message consists of a header and as many data words as required to complete the message. The messages, for the purposes of ICC_Message_Router, are divided into fault messages requiring filtering, non fault requiring filtering and messages requiring no filtering. Filtering means that only one message for a particular message type is processed with all others of that type skipped. The header consists of a ICC message table (IMT) index and an indicator containing the length of the message. Those messages with an IMT index of one are the fault messages requiring filtering. The filtering comparison is the FMPT and major/minor fields of the type one messages. Since there are many unique combinations of these two fields, the occurrence of the message with the lower message time for each of these combinations is saved for processing. The messages with IMT index of two are filtered according to message class. The first occurrence of a particular message class is processed immediately and further occurrences are ignored. Messages with an index of 3, 4, 5, 6, 10, 11, or 12, are filtered by first occurrence. For example, if a message with an index of 3 occurs, that message is processed immediately and a flag is set to inhibit other type 3 messages from being processed. Those messages with IMT_Index of 20, 21, 22 or 23 are filtered by comparing the entire data word of one message with the same word of a previous message (with the same index). Only the messages whose data word is identical to a previous message are skipped. Otherwise the first occurrence of that unique data word for each index is processed immediately. Messages with any other IMT_Index are processed for each message; that is, there is no filtering on these messages.

All the messages for this ICC buffer are processed until the trailer indicator ($FFFF_{16}$) is encountered. In addition to the buffer for the processing GPC, the input buffers of the GPC's in common set with this GPC are processed in a similar matter until all the messages of these buffers have been processed. Now for the process itself.

The Filter_Mask is preset not to filter IMT_Indexes of 7,8,9. By design IMT_Indexes two through twenty-three are assigned to non-fault filterable messages; however, 7, 8, and 9 are not used so they are inhibited by masking them out with Filter-Mask. Next the Fault_Filter_Value and Non-Fault_Filter_Value are preset to zero as well as the Number_Of_Uniques for each ICC buffer. The following process takes place for every ICC receiving buffer.

First, the Processing_GPC value is compared with the Buffer_Number value and the Common_Set_Mask for the Processing_GPC. If equal to either then Buffer_End is reset to zero indicating this buffer should be processed. Next, Buffer_End is checked for zero. If it is zero then the buffer is processed in the following manner. The ICC_Message_Header is obtained from the ICC_Messages and the ICC_Message_Header is checked for a 'FFFF'. If it is, this indicates the logical end of the buffer has been reached so Buffer_End is set to one. Otherwise, processing continues. First, the IMT_Index and Message_Size are obtained from the ICC_Message_Header. The IMT_Index is checked to be within zero to twenty-eight (the valid indexes). If it is then the ICC_Message_Table_Value is obtained from the ICC_Message_Table and Contention_Flag is checked for off. If off, then Filtered_Message_Match is preset to zero and IMT_Index is checked for one. If it is, the following (fault type filtered message) processing occurs.

The match for these type one messages is by the FMPT copy number and the major/minor index. If another message with IMT_Index set to one is found, those two fields are compared with the two fields of all the previous saved type ones. If they match, the Filtered_Message_Match is set to one and the message of the two tested with the smaller message time is saved by storing the ICC_Buffer_Number in Temporary_Buffer_Number and the pointer to the saved message in Temporary_Index.

If there is no match, then this message is considered another unique combination and is saved. The Fault_Filter_Value is incremented by one. Temporary_Buffer_Number and Temporary_Index are set and the Combined_GPC_Mask is set.

If the IMT_Index is not one, it is checked to be two. If it is, the Current_Message_Class_Indicator is obtained from the ICC_Message. If Non_Fault_Filter_Value is greater than zero, the Current_Message_Class_Indicator is compared with the Saved_Message_Class Indicator of all the previous type 2 messages that had unique message class and were saved for that reason. If a match is found Filtered_Message_Match is set to one. If Non_Fault_Filtered_Value is zero or there was not a match from above, the Non_Fault_Filtered_Value is incremented by one and the message class for this saved message is placed in Saved_Message_Class_Indicator and the message is processed immediately by calling Error_Message_Line_Support.

If the IMT_Index is not two, it is checked for 3, 4, 5, 6, 10, 11, or 12. If it is one of these, the Filter_Mask is checked to make sure a message with this index had not previously been processed. If not, then the Filter_Mask is updated to include this index and Routing_Code_Process (490.7) is CALLed.

If the IMT_Index is none of the values checked above, it is tested to be 20, 21, 22 or 23. If it is, the message with the same IMT_Index is filtered by the contents of this one word message. First the Structure_Copy_Number and Saved_Number_Of_Uniques for this index is obtained. If Saved_Number_Of_Uniques is non zero, the message value of the current message is compared with the One_Word_Message_Contents of all the saved messages with the same index. If the contents are equal, Filtered_Message_Match is set to one. If Filtered_Message_Match is zero, the Saved_Number_Of_Uniques is incremented by one, this value is stored in Number_Of_Uniques. The One_Word_Message contents are obtained from the ICC_Message and Routing_Code_Processor (490.7) is called to process this message.

If the IMT_Index is non of the above (1-6, 10-12, 20-23) then no filtering is done and Routing_Code_Process is CALLed.

By continuing processing, or, if Contention_Flag was on, or if it was not a valid IMT_Index, or, if ICC_Message_Header was equal to FFFF, the ICC_Message_Pointer is updated by Message_Size to point to the header of the next message within the buffer. If ICC_Message_Pointer is at the physical end of the buffer (greater than 120 words) Buffer_End is set to one. If not at buffer end, the next buffer is processed. When the last buffer is processed all the unique fault filtered messages are processed by obtaining the Buffer_Number and message pointer from Temporary_Buffer_Number and Temporary_Index. Next, the GPC's to receive the message are masked in ICC_Message and Error_Message_Line_Support is CALLed.

As a final process, the ICC_Buffer_Pointer is set to one to point to the start of the buffer, ICC_Message for the Processing_GPC is set to FFFF and ICC_Buffer_Freeze_Flag is reset to not frozen.

The Routing_Code_Processor processes codes of 3, 14, and 15. The Processing_Value and Routing_Code are obtained from ICC_Message_Table_Value. For Routing_Code = 3 or 14, and, depending on the Processing_Value, the message is processed as follows. If Processing_Value is other than 1, 2, and 3, no further processing is done. If Processing_Value = 1, Reconfiguration_Message_Header is CALLed; for Processing_Value = 2, ICC/IDLE_OPS_Processor is CALLed; and if = 3 Bus_Configuration_Change is CALLed.

If Routing_Code = 15, this means it is a data move action using an index found in the message content. The Data_Move_Index is obtained from the ICC_Message. The Get_Data_From_Index is the second word of ICC_message and the Data_Move_Length is from Message_Size. The message is moved one word at a time to the Receiving_Address. If the Processing_Value is four than BTU_Port_Masking_Routine (290) is CALLed. The flow for this module is found in Figure 3.1.3.3-1.

d. Output - See Table 3.1.3.3-1

e. Module References -

1. (870) GPC_Reconfiguration_Message_Handler (ARG_RECONFIG_MSG) is called
2. (495) ICC/Idle_OPS_Processor (DID_IDLE_OPS) is CALLed
3. (830) Bus_Configuration_Change (ARD_BUS_CHG) is CALLed
4. (290) BTU_Port_Mask_Routine (FIOBPM) is CALLed
5. (680) Error_Message_Line_Support (DMT_ERR_MSG) is CALLed

f. Module Attributes - external procedure

g. Template References -

1. UI/FCOS_Shared_Compool (CZ2_Common)
2. FCOS_Compool (FCMCOM)

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.1.3.3-1

NAME ICC_Message_Router (DME_ICC_ROUT)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	Filter_Mask	L131		490	490	DME_FILX			
2	Fault_Filter_Value	L132		490	490	DME_NFFF1			
3	Non_Fault_Filter_Value	L133		490	490	DME_NFFF2			
4	Number_Of_Uniques	L134		490	490	DME_NBR_UNIQ			
5	Buffer_Number	L135		490	490	DME_I			
6	ICC_Message_Pointer	L136		490	490	DME_A			
7	Buffer_End	L137		490	490	DME_B			
8	Common_Set_Member	L138		490	490	DME_C			
9	Processing_GPC	L164		490	490	DME_GPCI			
10	ICC_Message_Header	L139		490	490	DME_MSGHEAD			
11	IMT_Index	L140		490	490	DME_IMTI			
12	Message_Size	L141		490	490	DME_MSGS			
13	ICC_Message_Table_Value	L142		490	490	DME_MSGTABV			
14	Contention_Flag	L143		490	490	DME_MSGTABV			
15	Filtered_Message_Match	L144		490	490	DME_MATCH			
16	Current_FMPT_Copy_Number	L145		490	490	DME_CURFMPT			
17	Current_Major_Minor_Index	L148		490	490	DME_CURMJMN			
18	Saved_Buffer_Number	L146		490	490	DME_BF			
19	Saved_Index_Number	L147		490	490	DME_IN			
20	Combined_GPC_Mask	L149		490	490	DMEB_GP			

DATA TABLE 3.1.3.3-1 (cont'd)

NAME ICC_Message_Router (DME_ICC_ROUT)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
21	Temporary_Buffer_Number	L150		490	490	DME_FBN			
22	Temporary_Index	L151		490	490	DME_FIN			
23	Current_Message_Class_Indicator	L152		490	490	DME_CURINDMC			
24	Saved_Message_Class_Indicator	L153		490	490	DME_INDMC			
25	Structure_Copy_Number	L154		490	490	DME_YY1			
26	Save_Number_Of_Uniques	L155		490	490	DME_YY2			
27	One_Word_Message_Contents	L156		490	490	DME_MSG_VAL			
28	Processing_Value	L157		490	490	DME_PROCV			
29	Routing_Code	L158		490	490	DME_RCODE			
30	Data_Move_Index	L160		490	490	DME_LOCI			
31	Get_Data_Via_Index	L159		490	490	DME_J			
32	Data_Move_Length	L161		490	490	DME_T			
33	Receiving_Address	L162		490	490	DME_Q			
34	Move_Loop	L163		490	490	DME_X			
35	ICC_Buffer	I610		490,750 800 830	See APP. E	CZ2V_ICC_BUF			
36	ICC_Messages	I610.18		480,485 490	See APP. E	CZ2B_ICC_MSG_BUF			
37	ICC_Message_Table	I640			485,490	CZ2V_ICC_MSG_TAB			
38	ICC_Buffer_Pointer	I430		480 485,490	480,485 750	CZ2V_ICC_BUF_POINT			
39	ICC_Buffer_Freeze_Flag	I420		361 490	480 485	CZ2V_ICC_BUF_FRZ			
40	Common_Set_Mask	I010.12		See APP.	E	CZ2B_CS	See Appendix E		

DATA TABLE 3.1.3.3-1 (cont'd)

NAME ICC_Message_Router (DME_ICC_ROUT)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
41	ICC_Address_Name	I650.01		485	485,490	CZ2V_ICC_NM			
42	GPC_Id	#001		300	See App. E	TFCMID			

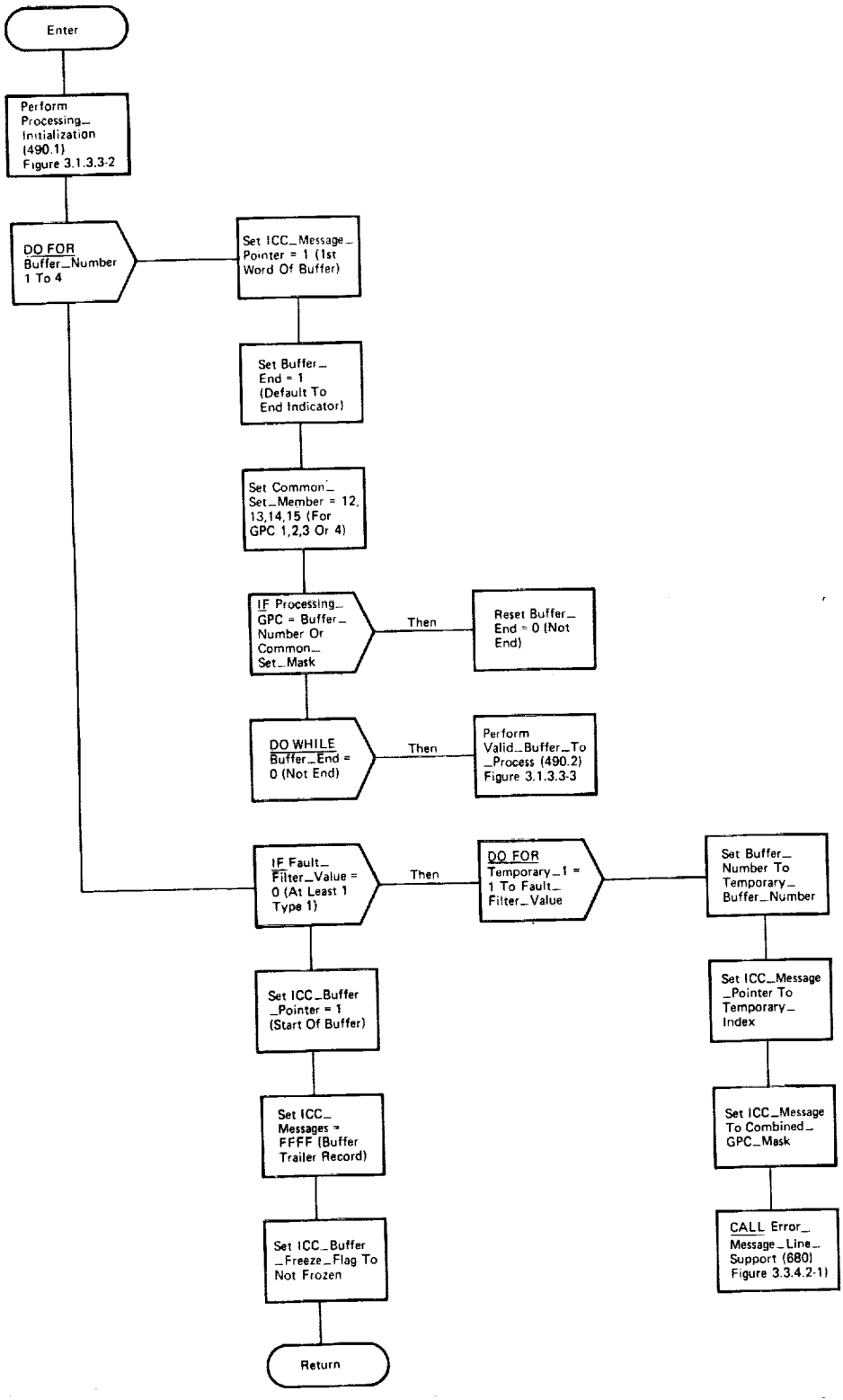


Figure 3.1.3.3-1. ICC_Message_Router (DME_ICC_ROUT)

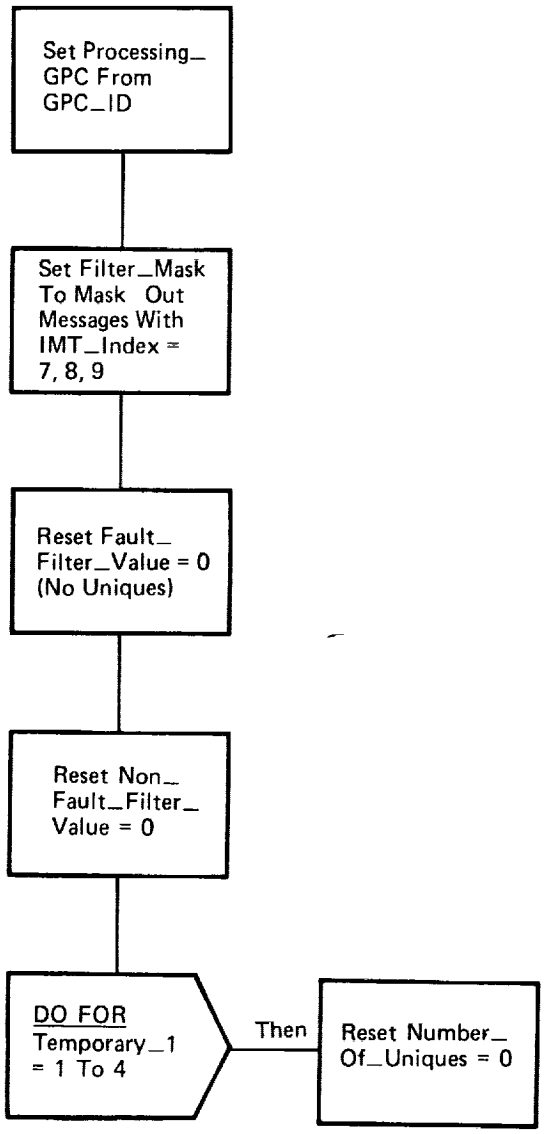


Figure 3.1.3.3-2. ICC_Message_Router Processing Initialization (490.1)

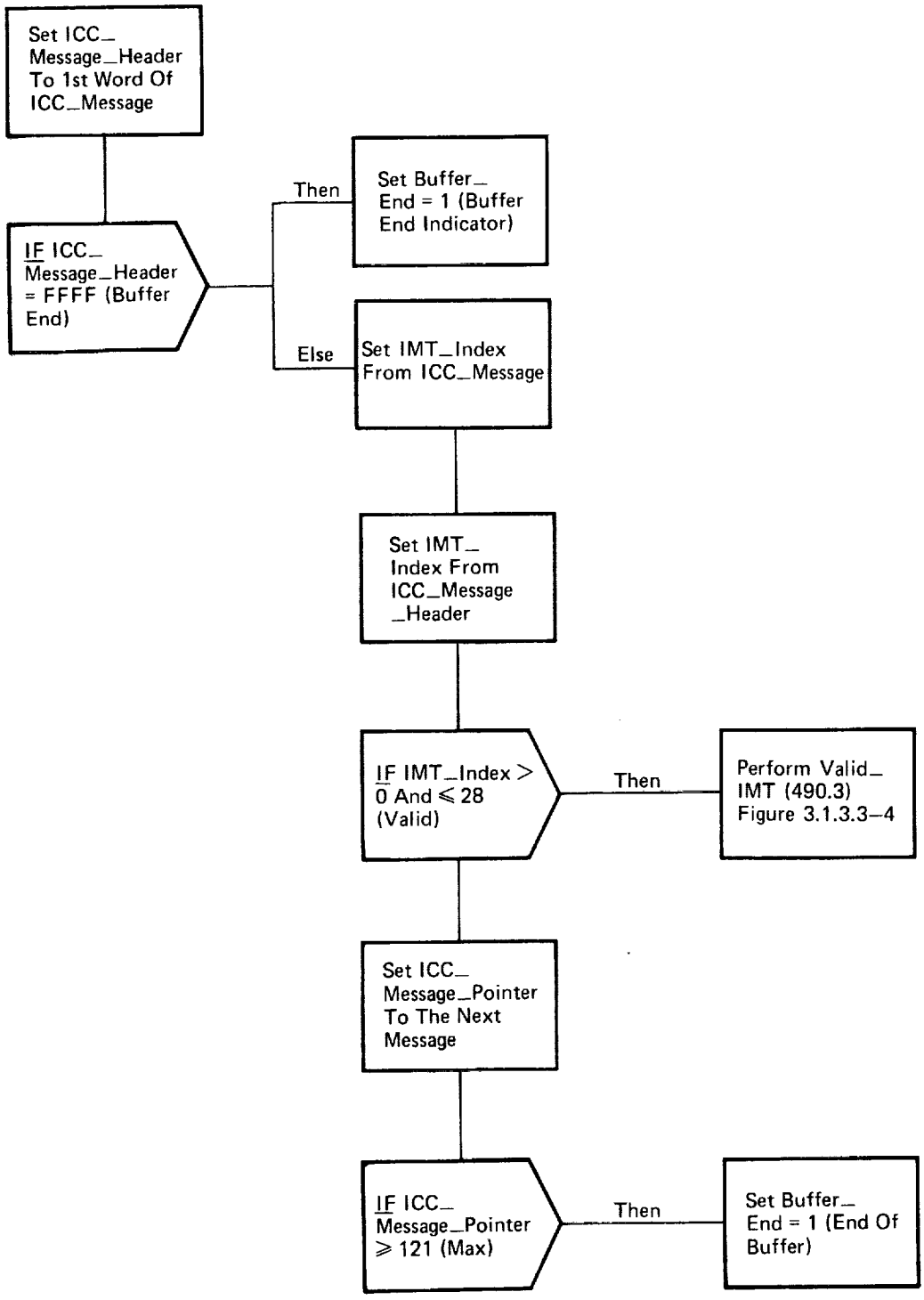


Figure 3.1.3.3-3. ICC_Message_Router Valid_Buffer_To_Process (490.2)

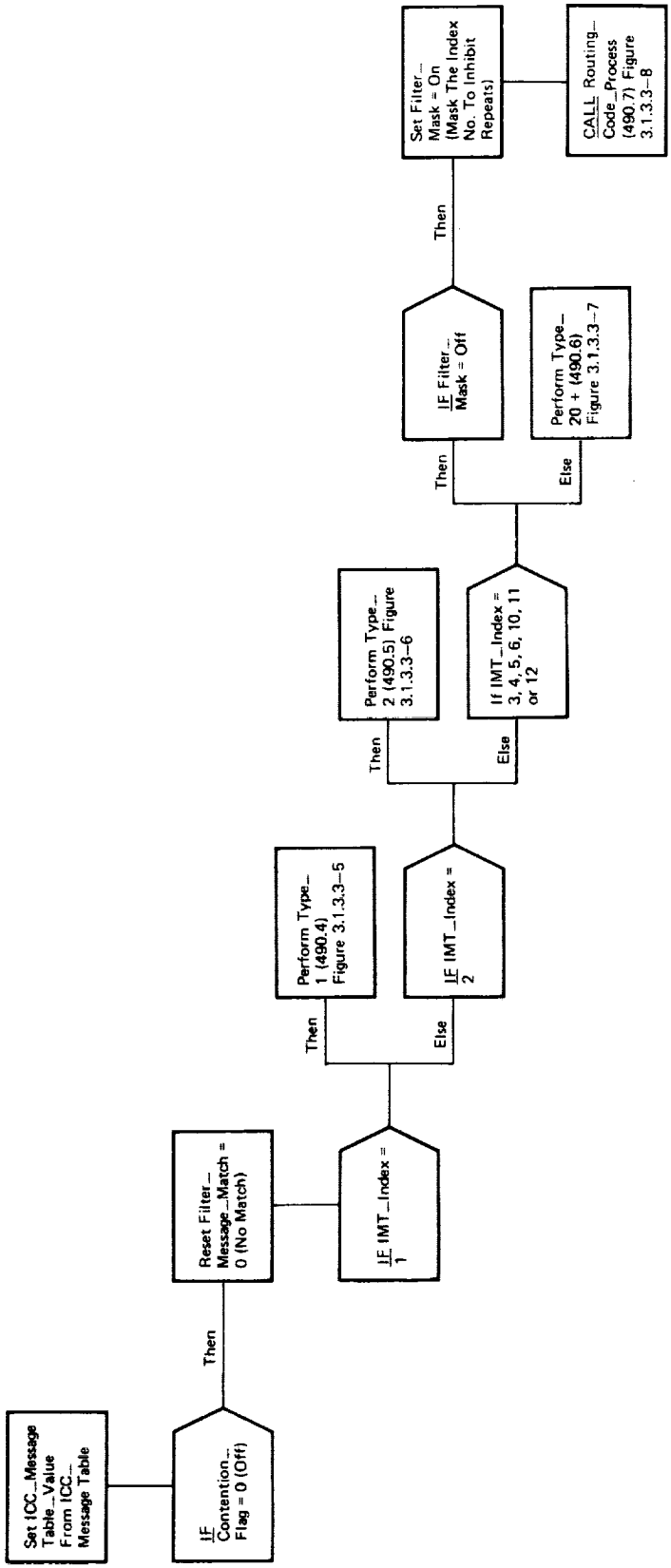


Figure 3.1.3.3-4. ICC_Message_Router Valid_IMT (490.3)

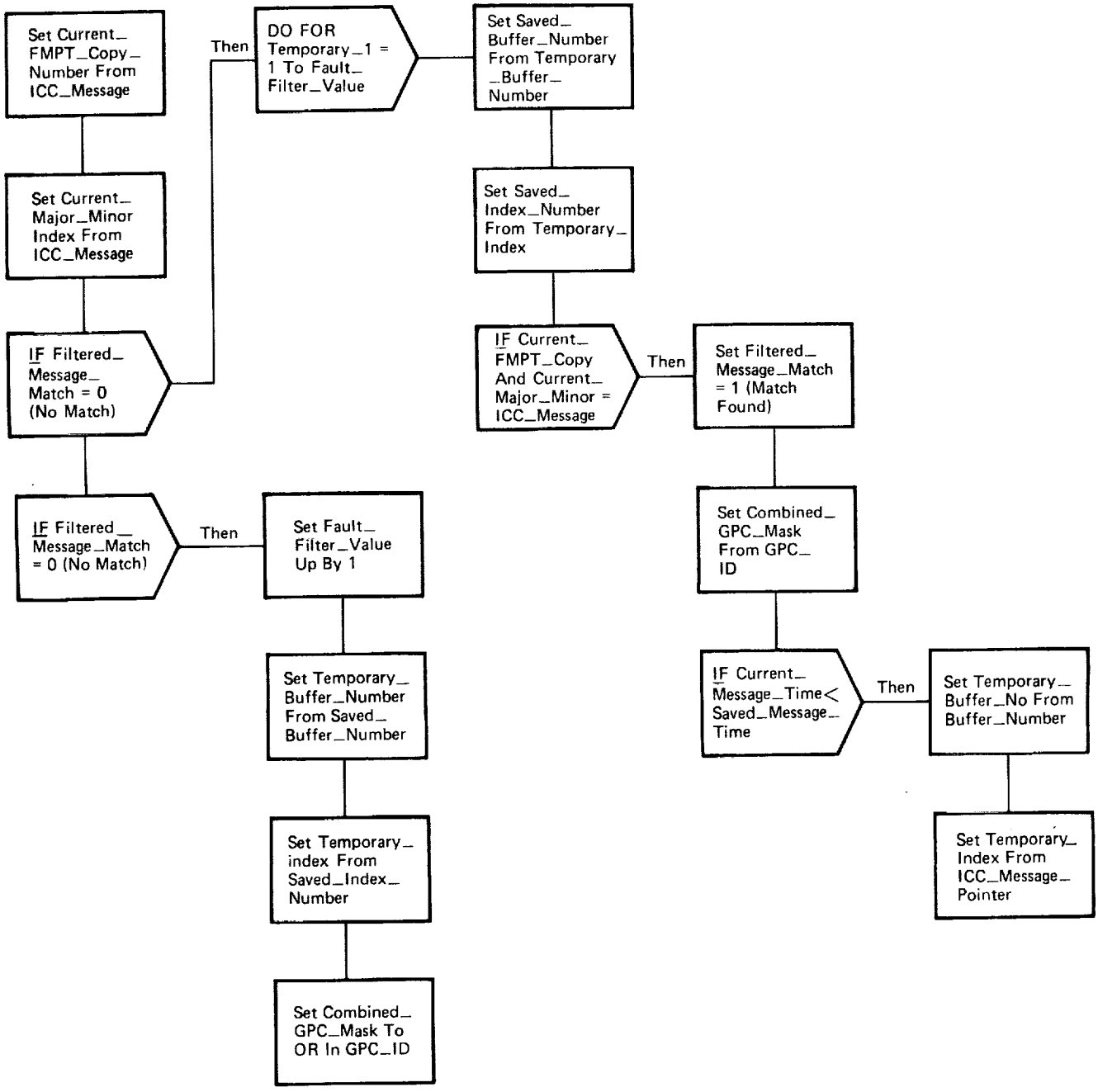


Figure 3.1.3.3-5. ICC_Message_Router Type_1 (490.4)

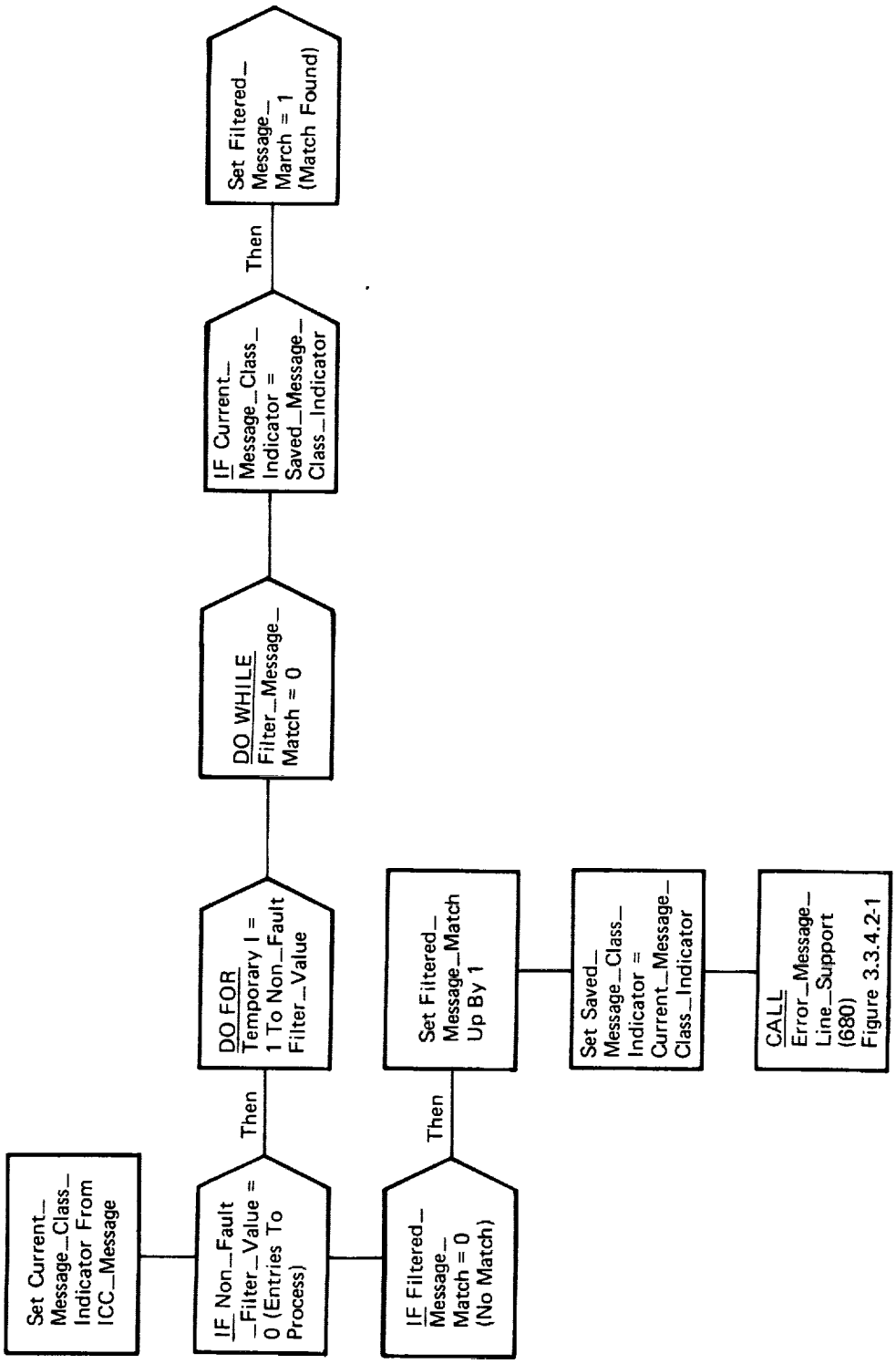


Figure 3.1.3.3-6. ICC_Message_Router Type_2 (490.5)

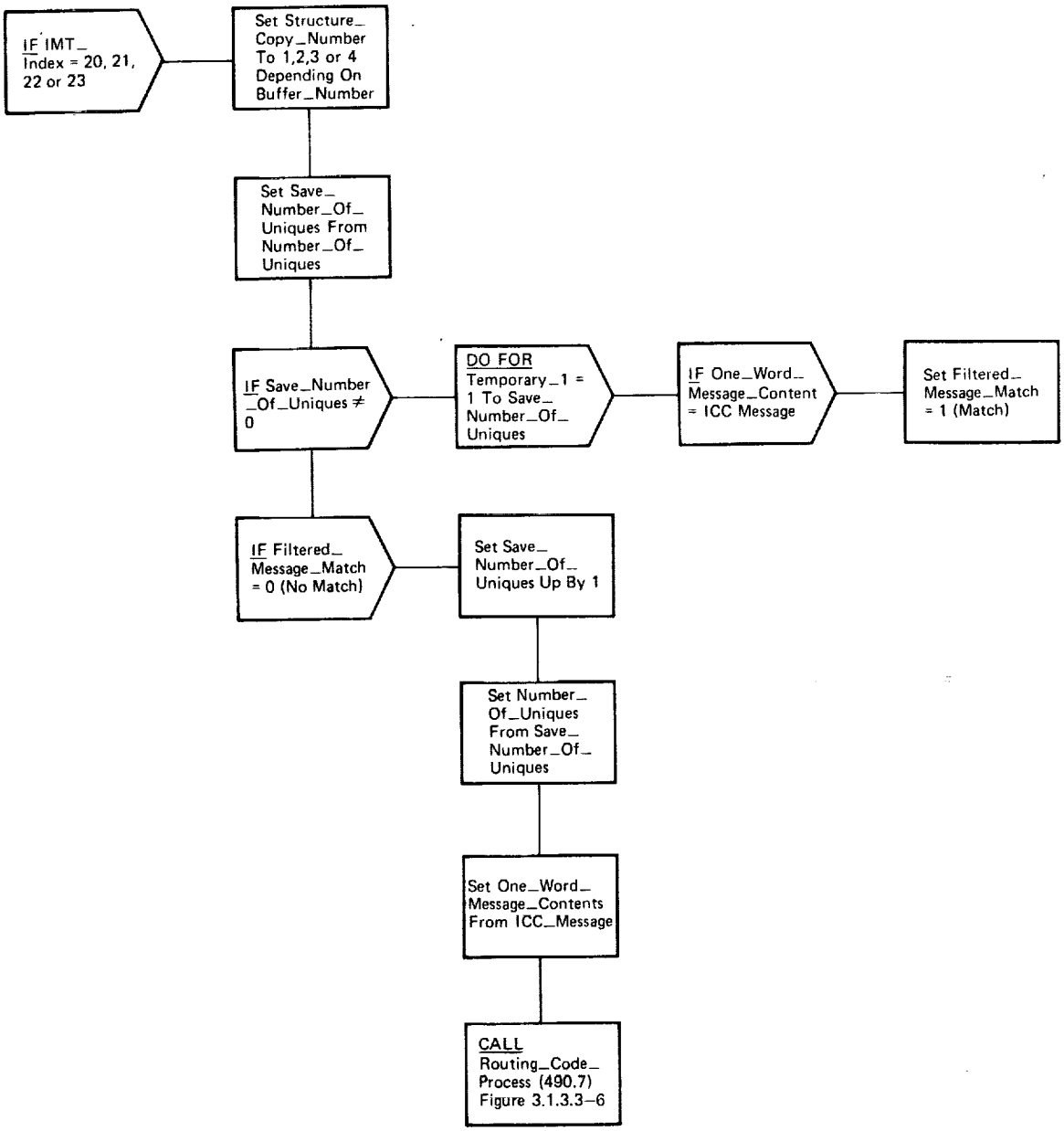


Figure 3.1.3.3-7. ICC_Message_Router Type_20+ (490.6)

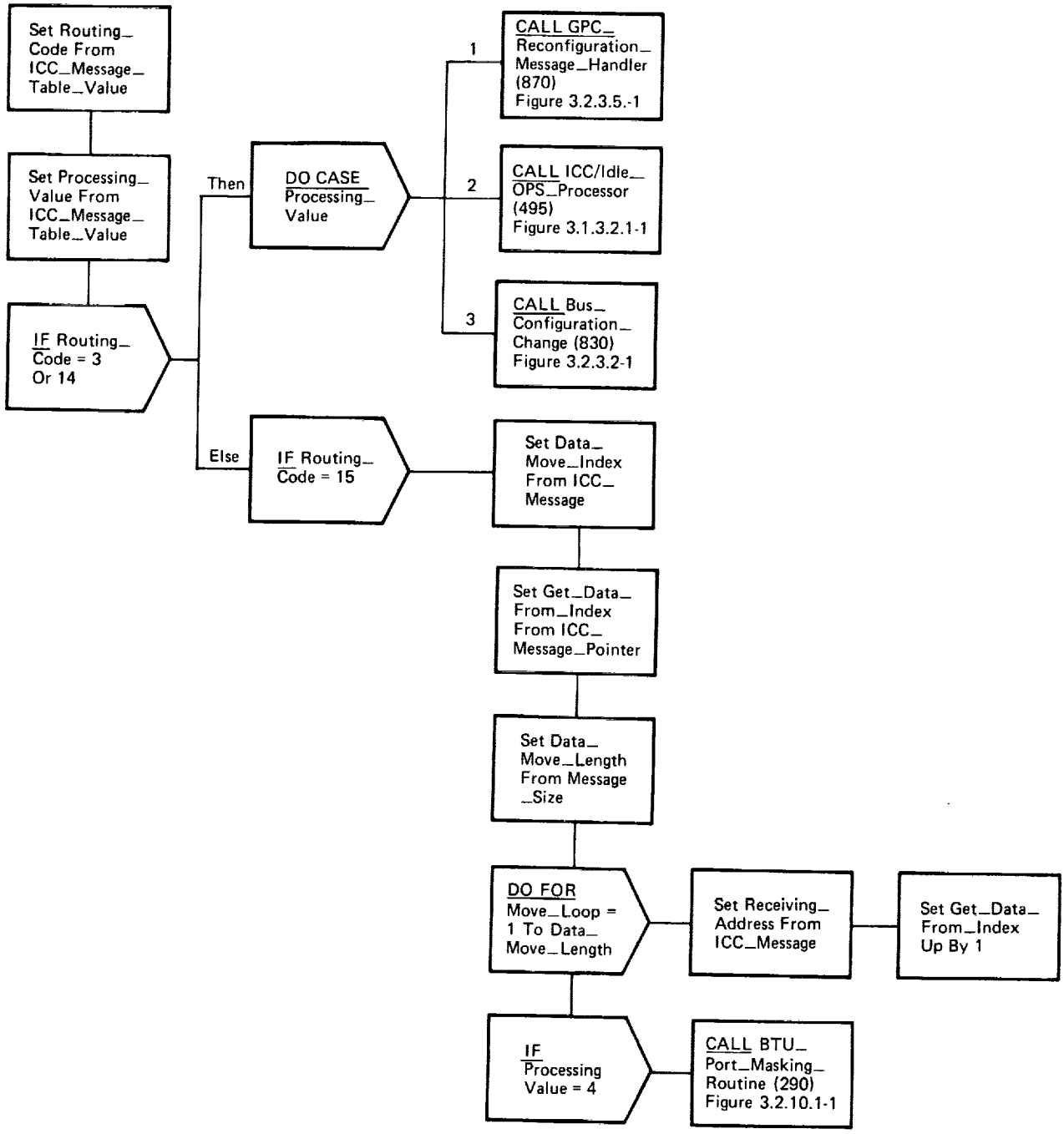


Figure 3.1.3.3-8. ICC_Message_Router Routing_Code_Process (490.7)

3.1.3.3.1 ICC_Idle_OPS_Processor (DID_IDLE_OPS_ICC) (495)

The purpose of this module is to service DPS Configuration Monitor page inputs in dissimilar GPC's.

a. Control Interface -

1. CALL DID_IDLE_OPS_ICC
2. CALLed by (490) ICC_Message_Router (DME_ICC_ROUT)

b. Input -

See Table 3.1.3.3.1-1.

c. Process Description - ICC/Idle_OPS_Processor first determines if the ICC Message issued is a Voter/Timer State Message. If it is a Voter On Message the Voter_Latch is set and the Timer_Latch is reset. If not a Voter On Message, the Timer_Latch is set and the Voter_Latch is reset. If the ICC Message is a LDB Start Command Message and this GPC is the prime or a member of the redendant set then the LDB_I/O_Processor is scheduled. If this GPC is not the prime and not in the redundant set DGI_LDB_IO is CANCELed. The control flow for this module is shown in Figure 3.1.3.3.1-1.

d. Output -

See Table 3.1.3.3.1-1.

e. Module References -

(440) LDB_I/O_Processor (DGI_LDB_IO) is SCHEDULed/CANCELed
(340) Miscellaneous_CM_Request Processor (FCMSVC)
(102) Processor_Scheduler (FPMSVC)

f. Module Attributes -

Procedure

g. Template References -

1. UI_FCOS_Shared_Compool (CZ2_COMMON)
2. LDB_I/O_Processor (DGI_LDB_IO)

h. Error Handling -

None

i. Constraints and Assumptions -

None

j. Detailed Implementation -

1. The following two parameters are input:

- ICC_Buffer_Number (DID_ICC_BUFNO)
- ICC_Message_Word_Index (DID_ICC_INDEX)

DATA TABLE 3.1.3.3.1-1

NAME ICC_Idle_OPS_Processor (DID_IDLE_OPS_ICC)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	Internal_ICC_Data_Word	L254.1		495	495	DID_ICC_DATA_WORD			
2	ICC_Message	I610.18		480 485,490	See APP. E	CZ2B_ICC_MSG_BUF			
3	Internal_ICC_Buffer_Number	L254.2		495	495	DID_ICC_BUFNO			
4	Internal_ICC_Index_1	L254.3			495	DID_ICC_INDEX			
5	Internal_ICC_Index_2	L254.4		495	495	DID_ICC_INDEX_2			
6	Timer_Latch	&011.2		495					
7	Voter_Latch	&011.1		495					
8	GPC_ID	#001		See APP.	E	TFCMID			
9	GPC_Prime_Id	I050		351,700 820,880	See APP. E	CZ2V_GPC_P			
10	Redundant_Set_Mask	I010.13		See APP.	E	CZ2B_RS	See Appendix E	X	
11	GSE_Poll_State	I300.02		495,820 830,870	830				

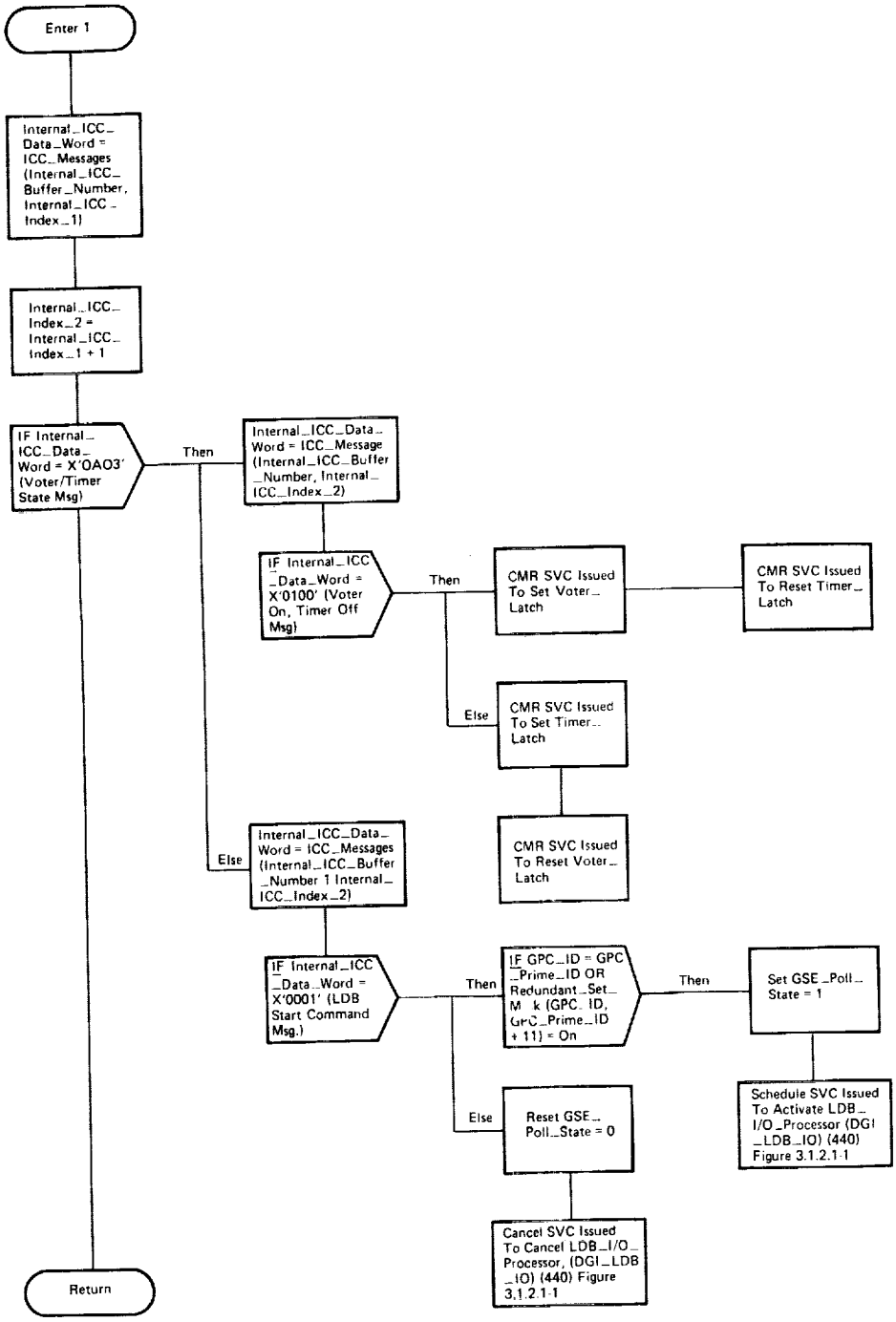


Figure 3.1.3.3.1-1. ICC_Idle_OPS_Processor (DIO_IDLE_OPS_ICC)

3.2 OPERATIONS CONTROL

The User Interface software provides a method whereby the user may communicate with the GPC for vehicle monitoring and control functions defined in the CPDS document. Operations Control decodes user inputs and coordinates the user to GPC processing (via Command Input Processing).

Operations Control is independant of the mission phase. The flight software is defined in terms of major functions (MF). The currently defined major functions are Guidance Navigation and Control (GN&C), Systems Management (SM), and Payload Management (PL). Each of the MF's is capable of executing simultaneously and independently of each other. A Maximum of 2 is allowed at once in a given GPC. There are three levels of functions which establish the hierarchy of MF support. These are the operational, specialist, and display functions, in that priority.

The user communications are via the MCDS/Ground interface. There is a single user/computer communications language. To provide for the support of the major functions and the mission phases and to provide control of the two-way communications, Operations Control has three major subdivisions: User Interface Control, Application Control, and Memory Overlay Coordiantion. (see Figure 3.2-1.) User Interface Control fulfills the two-way communications required by the MCDS/Ground interface language. The applications program (operational sequences and specialist functions) are defined in terms of control segments which allow a GPC-to-user communication via the CRT. Application Control provides the necessary control for fulfillment of CPDS requirements for the application program. The flight software is defined in terms of memory overlays. Each memory overlay contains sufficient application software to support a mission phase. Memory Overlay Coordination allows the establishment of a new memory load of the application software within a GPC at operational sequence (OPS) transition.

The data flow from user to GPC and GPC to user is via COMPOOL tables. A more complete definition of the tables defined here may be found in Appendix A. The DEU_Input_Table (DIT) is used to pass MCDS (and MCDS ground equivalent) keyboard messages from the Command Input Processor to Operations Control. The format of these messages is given in the GPC/DEU ICD. The Miscellaneous_Application_Control_Table (MACT) is used to maintain control of operational sequences and specialist functions by major functions. Data communications on a DEU basis is maintained in the MACT and updated to reflect the status of current processing. The MCDS_Allocation_Table (MAT) is used to associate the respective Major Functions to the DEU based upon the major function switch setting and provides the display format buffer necessary for a CRT cyclic update. The Display_Format_Buffer (DFB) is a UI COMPOOL area containing the Display Format Tables (DFT) necessary for the processing of the cyclic updates to the MCDS CRT. The display formats residing within the DFB are of two types: (1) critical formats with skeleton format control words (FCW's) in the DEU memory and foreground formatting data permanently resident in the GPC memory, and (2) noncritical formats residing in the GPC memory or on the Mass Memory and read in as needed. The Display_Format_

Mass_Memory_Directory (DMMD) is used by the User_Interface_Control_Supervisor (Section 3.2.1.1) to obtain the Mass Memory address of a display format table The Applications_Moding_Table (AMT) provides data for checking the SPEC's validity during this current OPS, and address data necessary for the scheduling requested OPS or SPEC. Additional data to identify the valid memory overlay for a requested OPS/major function is provided via the GPC_Reconfiguration_Table (GRT). The GRT is established prior to a mission to define all overlay definitions, while the AMT is a function of the current overlay. Appendix F represents a typical time line sequence of processing which shows how this section fits into the overall design of User Interface.

The Keyboard_Verification_Table (KVT) is defined as that part of the Display_Format_Table that specifies (1) the maximum number of items for the Display, and (2) the application keys valid for a particular display (ITEM key with data, ITEM key with EXEC ending, PRO,EXEC). It specifies if a renewal of the display is required after an ITEM update. It also specifies if limit checking is necessary, gives the input format of each of the items, and gives their upper and lower limits as required.

The applications control segments (OPS and SPEC) are defined in terms of a grammar designed to maintain user control of the application through MCDS type inputs. A grammar consisting of HAL macros allows the control segments to communicate with the user. This grammar is defined in Appendix H.

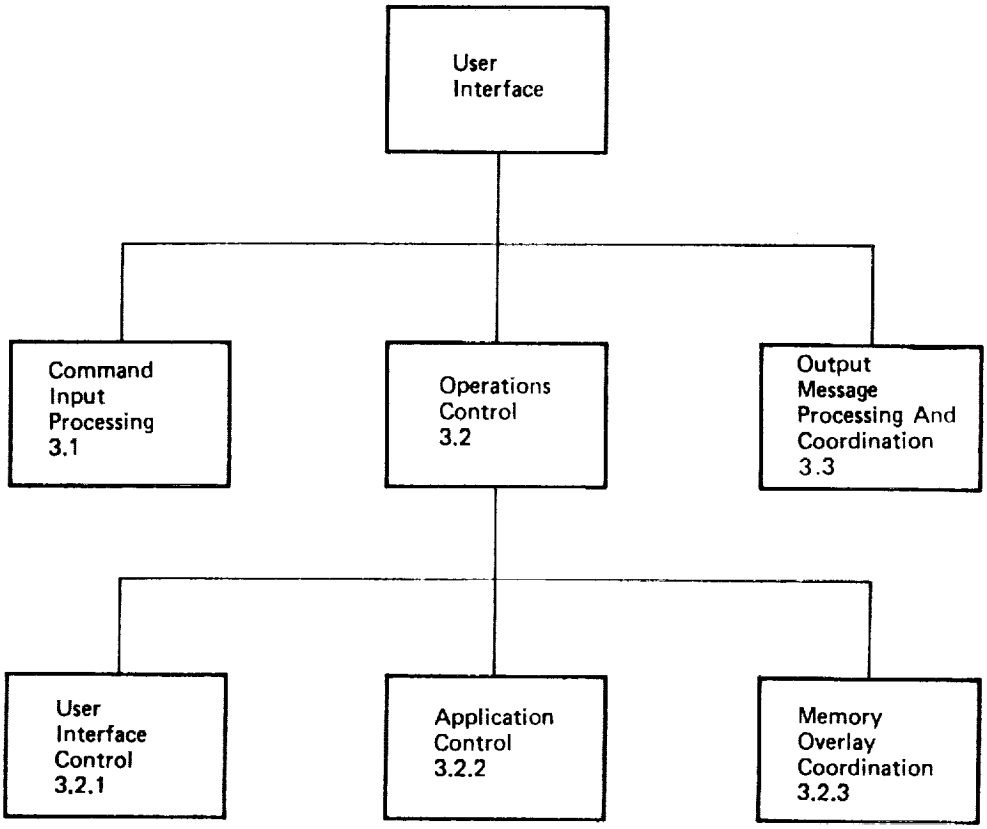


Figure 3.2-1 Operations Control Hierarchy Diagram

3.2.1 User Interface Control

The Shuttle Flight software is designed to be interactive with and in the control of the user. For this purpose, the software is structured to support a user/computer language defined in terms of a keyboard message. The inputs are via an MCDS or ground to MCDS interface. The input destinations are to the MCDS major function selected, and specifically to the control segment which supports the current MCDS display. The functions supported establish a hierarchy of operational, specialist, and display functions. The inputs are decoded according to the function supported for the MCDS originating the input. No more than one operational sequence per MF and no more than one specialist function per MF per DEU will execute simultaneously. Only one display function per DEU is supported at a time. The OPS for an MF is the highest level of control and will be under the control of any MCDS having its major function switch set to that MF. The SPEC is the second level of control and operates in parallel with the OPS. The SPEC is controlled by a single DEU. The lowest priority function is the DISPLAY.

The flight software may be partitioned to support a simplex, i.e., a computer may support one or two major functions or a redundant set of one or two major functions. The GPC configuration is established as a function of OPS transition. For a single GPC load, the software is divided into memory overlays accessible at the OPS transition. Memory overlays are defined in terms of the major function base containing programs common to all the OPS of a MF, and program overlays with those programs designed for the individual OPS. These overlays include programs and data. No memory overlays are allowed between OPS transitions with the exception of display formats, as required.

User Interface Control (Figure 3.2.1-1) receives user commands in the form of MCDS defined messages from Command Input processing and application service requests from Application Control. It coordinates the execution of OPS, SPEC, and DISPLAY functions by MF and DEU. To accomplish its function, User Interface Control consists of the User Interface Control Supervisor and its subfunctions which are:

a. (505) MCDS_Functions_Processor
b. (510) MCDS_Item_Processor
c. (520) MCDS_Display_Coordinator

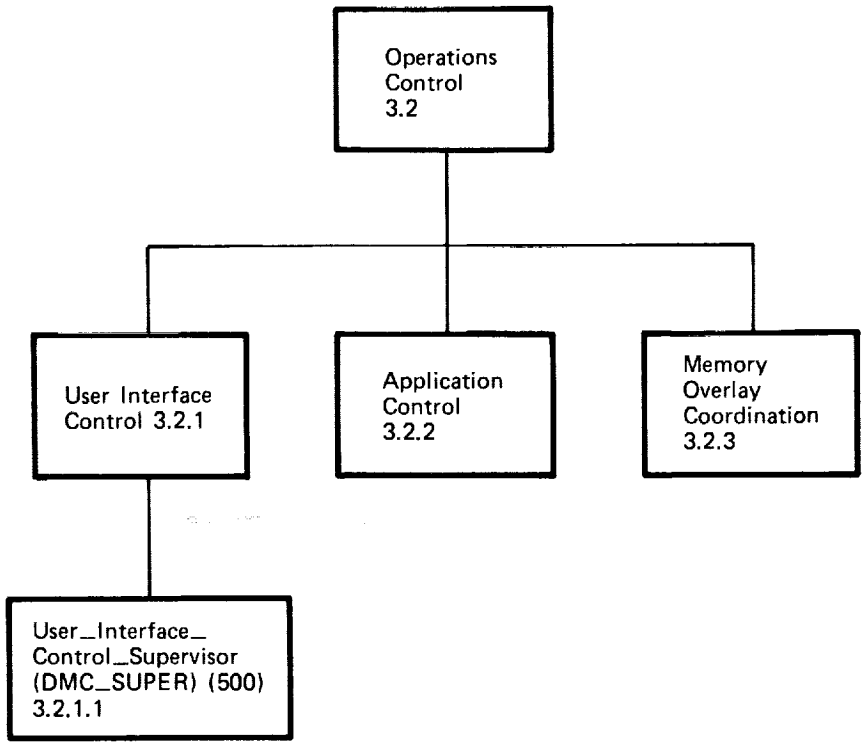


Figure 3.2.1-1 User Interface Control Hierarchy Diagram

3.2.1.1 User_Interface_Control_Supervisor (DMC_SUPER)(500)

This processor must supervise the User Interface inputs from (1) the MCDS keyboard (and LDB equivalent MCDS inputs),(2) the Application Control Segment service requests and (3) the I/O completion of Mass Memory reads. Based upon the inputs, it initiates the required processing.

a. Control Interface -

1. SCHEDULE DMC_SUPER PRIORITY(PRIO_DMC);

2. (a) Scheduled by (710)GPC_Startup (AIR_GPC_STARTUP).
   (b) Event Keyboard_Message_Ready_Event(CZ1E_D_MCDS_EVENT)set by (700) GPC_Startup (AIR_GPC_Startup)
   (c) Event Keyboard_Message_Ready_Event (CZ1E_D_MCDS_EVENT) set by (405) MCDS_Message_Processor (DMM_MCDS_PROCESS).
   (d) Event Keyboard_Message_Ready_Event(CZ1E_D_MCDS_EVENT) set by (800) GPC_Switch_Monitor(ARA_GPC_SWITCH).
   (e) Event Keyboard_Message_Ready_Event(Z1E_D_MCDS_Event) set by (830) Bus_Configuration_Change(ARD_BUS_CHG).
   (f) Event Application_Service_Request_Event(CDME_APP_SERVICE) set by (540)Display_Presentation_and_Control(DIS_PLAY)
   (g) Event Application_Service_Request_Event(CDME_APP_SERVICE) set by (550)Application_Moding_and Sequensor(DNX_BMS).
   (h) Event Application_Service_Request_Event(CDME_APP_SERVICE) set by (870) GPC_Reconfiguration_Message_Handler(ARG_RECONFIG_MSG)
   (i) Event Application_Service_Request_Event(CDME_APP_SERVICE) set by (880) Secondary_GPC_Reconfiguration(ARH_SEC_GPC_RECONFIG)
   (j) Event Application_Service_Request_Event(CDME_APP_SERVICE) set by (820)GPC_Reconfiguration(ARC_GPC_RECONFIG).
   (k) Event MM_Read_Complete_Event (CDME_ID_MM_EVENT)set by FCOS.

b. Inputs - See Table 3.2.1.1 - 1

c. Process Description - This process is scheduled at the GPC initialization based upon Initial Program Load (IPL) or Status change from HALT to RUN or STANDBY. Initialization data is set to allow for GPC initialization processing by sub processes. The Control_Segment_Request_Control_Block is set to the Initialization_Request_Control_Block. The DEU_Request_MACT_Index is set to zero to indicate that no request by DEU has been made. The OPS_0_00_Initialization_Flag is set to zero. The Reconfiguration_In_Progress,Reconfiguration_In_Inialization,and the MM/DFT_IO_Completion_Flag are zeroed. The Blank_Scratch_Pad_Line_Flag is zeroed to force scratch pad line clearing on new display updates.

User Interface Control Supervisor continues in execution or awaits inputs via events. The WAIT is for Keyboard_Message_Ready_Event or Application_Service_Request_Event or MM_Read_Complete_Event. This wait and associated processing after the wait is contained within a infinite loop until the process is halted (i.e. GPC Halt).

When the wait is terminated by an Event change, each event is tested in tern and processing of the respective request is done.

For the Keyboard Message_Ready_Event, the event is reset (disabled) and the DEU_Message_Ready_Flags are tested in a loop from 1 to the number of DEU's with DEU_Number as the loop counter. If a keyboard message is found, the MCDS_FUNCTION is set as the first keystroke of the entered keyboard message. The MCDS_Major_Function is taken as the DEU_MF_Switch_Setting +1. The data in the input message is tested for validity with the DEU_Message_Length_Table (for this DEU) is turned off, and the DEU_Poll_Flags for the DEU is turned on. Also a Keyboard error (DU00Q) is output to the DEU. For valid inputs, the MCDS_Functions_Processor is called.

For the Application_Service_Request_Event, the event is reset (disabled), the MCDS_Function is set to 11, and the MCDS_Functions_Processor is called.

For the MM_Read_Complete_Event, the event is reset (disabled), the MCDS_Function is set to 10, and the MCDS_Function_Processor is called.

At this point, the infinite loop returns the User_Interface_Control_Supervisor to the WAIT, awaiting further inputs via the above described events.

d. Outputs - See Table 3.2.1.1 - 1.

e. Module References -

1. (104) Wait_Processor(FPMWAIT) is called.
2. (171) Reset_Event_Processor(FPMRESET) is called.
3. (675) Annunciation_Macro_Interface(DMA_MAC) is called.
4. (505) MCDS_Functions_Processor(DMC_FUNCTIONS) is called.

f. Module Attributes - Program.

g. Template References -

1. FCOS_Compool(FCMCOM)
2. UI_Section_of_Common_Compool(CZ1_Common)
3. GPC_Reconfiguration(ARC_GPC_RECONFIG)
4. UI_FCOS_Shared_Compool(CZ2-Common)
5. UI_General_Compool(CDM_UI_COMPOOL)
6. Annunciation_Compool(CDL_ANNUN)
7. Annunciation_Macro_Interface(DMA_MAC)
8. Idle_Operational_Sequence (ARB_IDLE_ORS)
9. ICC_Message_Collector(DIM_ICC_COLLECTOR)

10. Time_Management_Specialist_Function(ASC_TIME_MGMT)
11. Read/Write_Specialist_Function(ASB_RD_WRT).
12. Display_Format_Buffer 1(CDB_DFB1_UI)
13. Display_Format_Buffer_2(DD-02_2DFB)
14. Display_Format_Buffer_3(CDF_02_3DFB)
15. Display_Format_Buffer_4(CDG_02_4DFB)
16. Display_Format_Buffer_5(CDH-02_5DFB)
17. DFT_Mass_Memory_Directory(CDR_02_DMMD)
18. Applications_Moding_Table(CDA_P02)

h. Error Handling - Messages via the DEU_Input_Table are tested for validity as to MCDS_Function,DEU_MF_Switch_Setting, and the DEU_Number_Of_Keystrokes. When they are found to be invalid, GPC Detected_Input_Message_Syntax_Error(DU009) is output to the DEU via the Annunciation_Macro_Interface.

i. Constraints and assumptions - none.

j. Detailed Implementation - none.

DATA TABLE 3.2.1.1-1

NAME User_Interface_Control_Supervisor (DMC_SUPER)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	Control_Segment_Request_Block	L118	LI	500,560	500,560	DMCV_REQBLK			
2	Initialization_Request_Control_Block	L280	LI	500	560	DMCV_IREQBLK			
3	DEU_Request_MACT_Index	L128	LI	560	500,505, 560	DMCV_DEU_MACT			
4	Reconfiguration_In_Progress	L120	LI	560	500,510, 520,560	DMC_REC_IN_PROGRESS			
5	Reconfiguration_In_Initialization	L193	LI	500,505, 560	505,560	DMC_REC_IP2			
6	MM/DFT_IO_Completion	L281	LO	500,505 520	505,520	DMC_IO			
7	Blank_Scratch_Pad_Line_Flag	L282	LO	500,505	600	DMC_CLEAR_SPL			
8	Keyboard_Message_Ready_Event	J020	ZIO	405,710, 800,830	500,815	CZIE_D_MCDS_EVENT			
9	Application_Service_Request_Event	B050	ZIO	540,820 870,880	660,810 820	CDME_APP_SERVICE			
10	MM_Read_Complete_Event	B060	ZIO	500	505	CDME_IO_MM_EVENT			
11	DEU_Message_Ready_Flags	J050	ZIO	405,505, 815,830	500,505	CZIB_D_DIT_MSG_READY			
12	DEU_Number	L127	LIO	500	See App. E	DMC_DIT_INDEX			
13	MCDS_Function	L219	LO	500,505	500,505, 510	DMC_CASE			
14	MCDS_Major_Function	L192	LO	500,505, 560	See App. E.	DMC_MF			
15	DEU_MF_Switch_Setting	J060.11	ZI		500				
16	DEU_Message-Length_Table	L283	L	500	500	DMC_DML			
17	DEU_Poll_Flag	J062	ZIO	See App. E	See App. E	CZIB_D_DEU_POLL_FLAG			
18	DEU_Number_Of_Keystrokes	J060.19	ZI	See App. A	See App. E	CZIV_D_NUMOFKEYS	V93M2002P, V93M2102P,V93M2202P		
19	DEU_Input_Table	J060	ZI	See App. E	See App. E	CZIV_D_DIT			
20	Initialization_Flag	L196	LO	500,560	500,560	DMC_INITIAL			

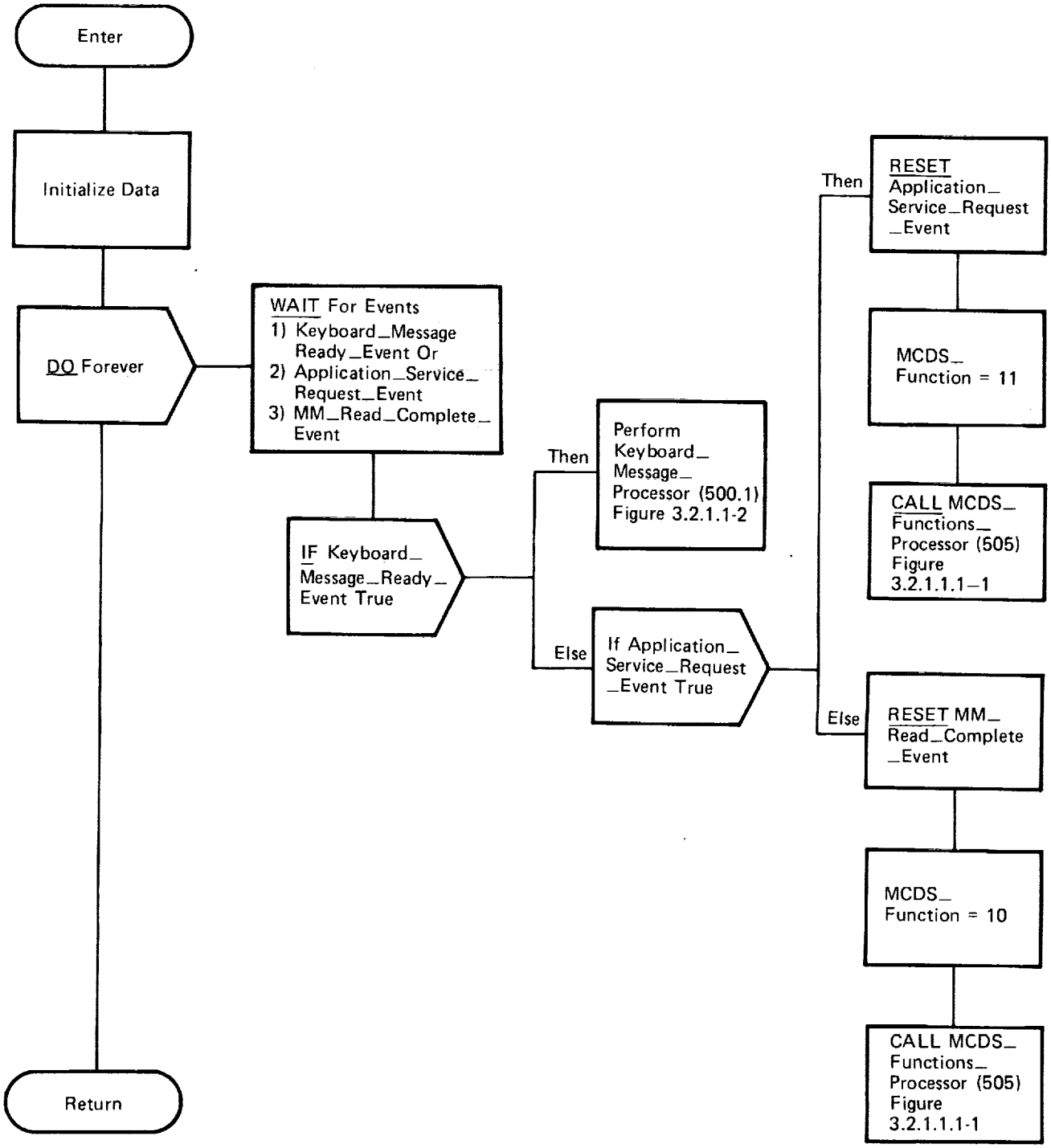


Figure 3.2.1.1-1. User_Interface_Control_Supervisor (DMC_SUPER)

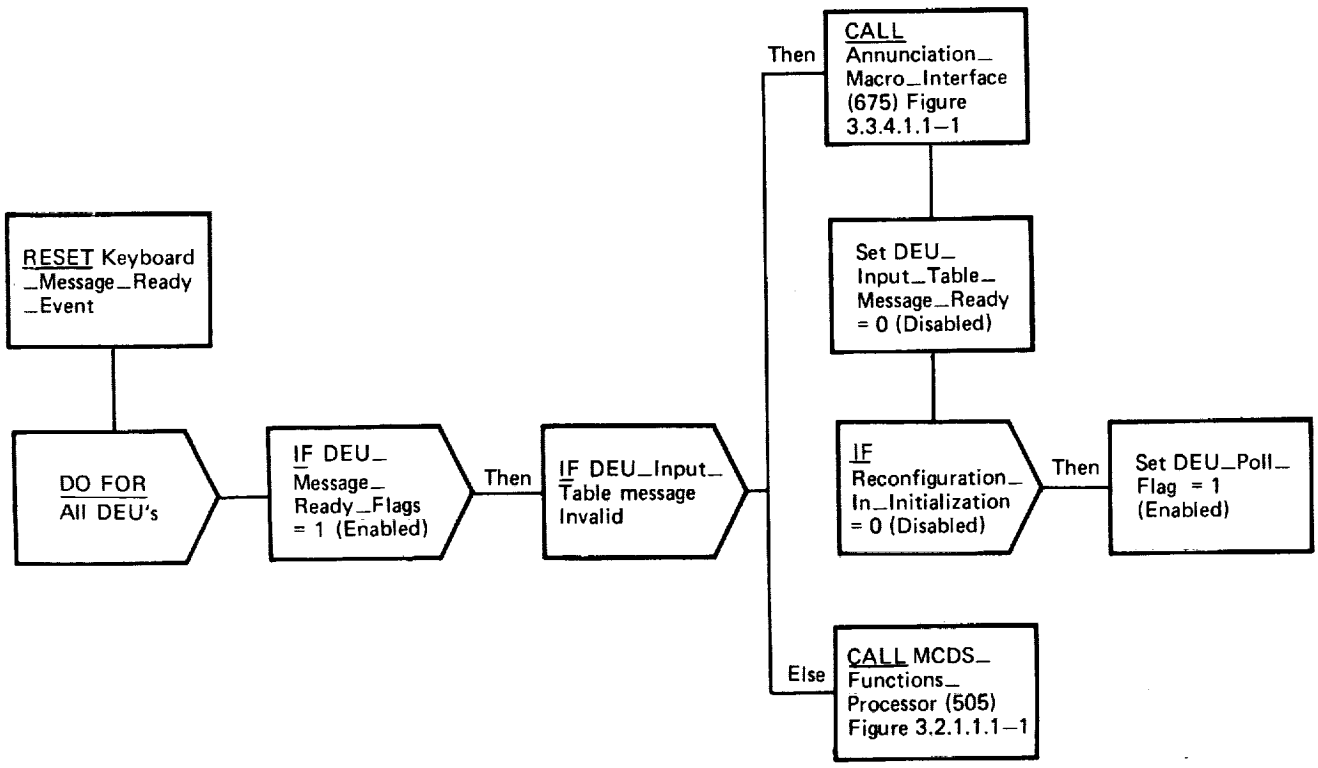


Figure 3.2.1.1-2. User_Interface_Control_Supervisor Keyboard_Message_Processor (500.1)

3.2.1.1.1 MCDS_Functions Processor (DMC_FUNCTIONS) (505)

This processor provides the functional processing necessary to support the MCDS keyboard. It consists of a series of cases each of which supports a specific keyboard function key either directly or by a call to the appropriate subroutine.

a. Control Interface -

1. CALL DMC_FUNCTION;

2. CALLed by (500) User_Interface_Control_Supervisor(DMC_SUPER)

b. Inputs - See Table 3.2.1.1.1-1.

c. Process Description - MCDS_Functions_Processor receives calls to service three types of inputs: 1) MCDS or MCDS equivalent (i.e. LDB) keyboard messages. 2) application service requests, and 3) mass memory IO completion. Each input is processed as a member of a group of cases whose case number is specified by the MDCS_Function less the Function_Case_Offset. The MCDS_Function is based upon the Level A description of the MCDS keyboard and the processing required for each function. The Mass Memory IO Completion (MCDS_Function = 10) comes as a result of the request (by keyboard or control segment input) of a display whose Display Format Table (DFT) resides on the MM and requires MM I/O. The application service requests (MCDS_Function = 11) are fielded through the Miscellaneous_Application_Control_Table (MACT). The remainder of the inputs are through the DEU_Input_Table (DIT).

A WAIT for Cyclic_Display_Processor to complete processing is done to prevent a conflict in the use of MCDS_Allocation_Table (MAT) data. Initialization data is set as follows: 1) GPC_ID is moved from FCOS Compoool to local data. 2) the Annunciation_Error_ID is set to -1 (any error discovered results in this parameter being set to the appropriate ID which is in turn annunciated through Annunciation_Macro_Interface. A core saving results in there being only one call at the end of this process instead of many scattered throughout the process); 3) the Display_Request_Level is set to 3 (display) and 4) Bits 9 to 16 of the ICC_Status_Word is set to zero to flag a valid ICC return. For keyboard inputs only, the MCDS_MACT_Index is set for the given DEU from the Active_Control_Segment_MACT_Index (for OPS or SPEC) and the Keyboard_Next_Input is incremented by one.

The DO_CASE which follows consists of 22 cases to support all currently defined keyboard functions. Some of the cases are left undefined. The cases are defined as follows with each case number being determined by MCDS_Function-Function_Case_Offset.

1. Case 1 is the Display MM IO Completion processing. The MM/DFT_IO_ Completion is set to 1 and the MCDS_Display_Coordinator is called.

2. Case 2 is the application interface processing. A loop of seven is set up to check each MACT for an application request through (a) the MACT_Service_Request_Ready_Events and (b) the MACT_Service_Request_ Flags. The former indicates which application (MACT) is making the request and the latter flags the specific request. If a MACT request is found, the MACT_Service_Request_Ready_Events is RESET. The Major_ Function_ID, the DEU_Number, and the MCDS_MACT_Index are set respectively from the Major_Function_ID (in the MACT), the Keyboard_Input_ DEU_Number and the Loop_Counter. As each MACT_Service_Request_Flags is found to be Set = 1 (Enabled), it is Reset = 0 (Disabled) and the service performed. The following outlines each service request:

   a. For the New_Display_Request_Flag enabled - if the DEU_Number is zero (i.e. an OPS), the Display_Request_Level is set to 1. The Display_Request_Number is set to the New_Display_Number and MCDS_ Display_Coordinator is called. The New Display_Number is then set to zero.

   b. For the ITEM_Processing_Complete_Flag - the MCDS_Function is set to zero and the DEU_Number is set to the Keyboard_Input_DEU_ Number. If the Loop_Counter is zero (OPS 0-00) the Major_Function__ID (local) is set to the MAT_Major_Function_Setting for this DEU. MCDS_ITEM_Processor is called. The MCDS_Functions is Reset to 11.

   c. For the Control_Segment_Termination__Flag_Sequence_Request_ Processor is called with a calling parameter (action type) of zero.

   d. For the GPC_Reconfiguration_Completion (tested only whom Loop_ Counter is 1) Sequence_Request_Processor is called with a calling parameter (action type) of one (1).

   e. For the GPC_Secondary_Reconfiguration_Request (tested only when Loop_Counter is 1) - Sequence_Request_Processor is called with a calling parameter (action type) of two (2).

3. Case 3 is the major function switch change processing. The Major Function switch change is a pseudo keyboard message generated when the DEU is switch from one GPC to another or when the MF switch for that DEU is changed. The DEU_Display_Support_Level, the Display_ Level_Page_Number (1 to 3) and Reconfiguration_In_Initialization are set to zero. The MAT_Major_Function_Setting is set to the Major_ Function_ID (local). The CRT_Status_Flag 1 (freeze) is Reset. If the second element of the Keyboard_Message is not zero, another GPC

will take control of the DEU and no further processing is done on this DEU. If zero, the DEU is to be switched to the new Major Function (MF) with the SPEC or OPS display which may be supported in that MF at that DEU. If a SPEC is being supported (that is the MCDS_MACT_Index is not equal to the Active_Control_Segment_MACT_Index for this MF's OPS or the Active_Control_Segment_MACT_Index for this DEU's SPEC and for this MF then MCDS_Display_Coordination is called to drive the SPEC display. The MCDS_MACT_Index is reset to the OPS index from the Active_Control_Segment_MACT_Index and the Display_Request_Number is reloaded with the Current_Display_Numbers (the OPS Display). The Display_Request_Level is set to 1 (OPS) and MCDS_Display_Coordination is called to drive the OPS display.

4. Case 4 is the fault summary key processing. The Display_Request_Level is set to 3. The Display_Request_Number is set to 51 (the page number for the Fualt Summary page). MCDS_Display_Coordination is called to request the display.

5. Case 5 is undefined as no MCDS function has been defined for this number.

6. Case 6 is undefined as no MCDS function has been defined for this number.

7. Case 7 is the acknowledge key processing. The Annunciation_Error_ID is set to 1 to be passed to Annunciation_Macro_Interface which will as a result initiate the acknowlege key processing.

8. Case 8 is the OPS key processing Sequence _Request_Processing is called with a calling parameter (action type) of 17 to process the OPS request.

9. Case 9 is the SPEC key processing. Sequence_Request_Processing is called with a calling parameter (action type) of 18 to process the SPEC request.

10. Case 10 is the display key processing. The display request may be an actual keyboard request with 6 keystrokes or a pseudo request with 2 keystrokes to process the freeze (the second keystroke in Keyboard_Message_ is a 1) or the unfreeze (the second keystroke is a 0. The three types of processing follow:

    a. Freeze - the CRT_Status_Flag 1 (one time update) and the CRT_Status_Flag 2 (Freeze) are each set.

    b. Unfreeze - If the Display_Level_Page_Number for the display (DEU_Display_Support_Level = 3) is zero, the Display_Request_Level will be reset to 1. If the SPEC display (Display_Level_Page_Number) for the SPEC is not zero, the DEU_Display_Support_Level = is set to 2. The Display_Request_Number is set to the Display_Level_Page_Number indexed by the Display_Request_Number. The CRT_Status_Flag 1 is Reset to clear freeze condition.

The Blank_Scratch_Pad_Line_Flag is set to 1 to inhibit the SPL clear. MCDS_Display_Coordinator is called to redrive the frozen display and the Blank_Scratch_Pad_Line_Flag is reset to zero.

c. Display request - the Keyboard_Message inputs 1 to 3 are combined to form a display number and placed in Display_Request_Number. If this number is 51 (the Fault Summary page) the Annunciation_Error_ID is set to zero to be passed to Annunciation_Macro_Interface to cause a clearing of the Fualt Summary page errors. The MCDS_Display_Coordination is called to drive the requested display to the DEU.

11. Case 11 is the item key processing. The MCDS_ITEM_Processor is called to process these inputs.

12. Case 12 is undefined as no MCDS function has been defined for this number.

13. Case 13 is the message reset key processing. The Annunciation_Error_ID is set to 2 to be passed to Annunciation_Macro_Interface which will as a result initiate the message reset key processing.

14. Case 14 is undefined as no MCDS function has been defined for this number.

15. Case 15 is undefined as no MCDS function has been defined for this number.

16. Case 16 is the CMPTR/CRT key processing. The keyboard message consist of a GPC number A (A is the second keystroke of Keyboard_Message) and a CRT number B (B is the third keystroke of Keyboard Message). The inputs are valid if A is less than 5 and B is less than 4 and B is not equal to zero.

    a. For valid inputs, the second element of the ICC_Message_For_CMPTR/CRT_Key is packed as follows: bits 1 to 4 are the DEU_Number, bits 5 to 8 are A and bits 9 to 16 are B. The ICC_Message_Collector is called to pass this message to Bus_Configuration_Change in each of the G.P.C.

    b. For invalid inputs, Annunciation_Error_ID is set to 42 (SC009) to be passed to Annunciation_Macro_Interface.

17. Case 17 is undefined as no MCDS function has been defined for this number.

18. Case 18 is the resume key processing. This input is made to terminate a display or SPEC support at the DEU.

    a. If the third element of Display_Level_Page_Number is not zero then terminate display support as follows: (1) DEU_Display_Support_Level is reset to zero. (2) If the second element of Display_Level_Page_Number is zero then the Display_Request_Level is set to 1 and MCDS_Display_Coordination is called.

b. Otherwise, if the second element of Display_Level_Number is not zero, the SPEC is terminated as follows: (1) If the MCDS_MACT_Index is 1, the Sequence_Request_Processor is called with a calling parameter (action type) of zero to terminate SPEC 00, (2) otherwise the Keyboard_Message_ID is set to the MCDS_Function, bits 1 to 3 of MACT_Application_Control_Flags are set, and the Name_of_Keyboard_Message_Event is SET to signal the SPEC control segment to terminate normally.

c. Otherwise, the Annunciation_Error_ID is set to 43 (See Appendix D for error DU011).

19. Case 19 is undefined as no MCDS function has been defined for this number.

20. Case 20 is the CMPTR/BUS key processing. The keyboard message consists of a Bus string number A (A is the second keystroke of Keyboard_Message) and a bus ID B (B is the third keystroke of Keyboard_Message). The inputs are valid if (A is 6 and B is 6 or 7) or both A and B are zero.

    a. For valid inputs, the second element of the ICC_Message_For_CMPTR/BUS_Key is packed as follows: bits 1 to 4 are the DEU_Number, bits 5 to 8 are A and bits 9 to 16 are B. The ICC_Message_Collector is called to pass this message to Bus_Configuration_Change in each of the GPCs.

21. Case 21 is the EXEC key processing. The input is checked for validity against the current Display_Format_Table. An invalid input is one for which (1) the ITEM_PRO_EXEC_Allow is set or (2) the DEU_Display_Support_Level is 3 or (3) the Keyboard_Verification_Table_Data bit for this key is reset. The Annunciation_Error_ID is set to K4 (See Appendix D for errors DU014, DU015, DU028). For valid inputs, the Keyboard_Input_DEU_Number is set to the DEU_NUMBER, the Keyboard_Message_ID is set to the MCDS_Function, and the Name_of_Keyboard_Message_Event is SET.

22. Case 22 is the PRO key processing. This processing is the same as for the EXEC key. (See case 21 above.)

Case 22 ends the DO_CASE. If the MCDS_Function is greater than eleven then the process waits (i.e. does a loop) until the MCDS_Input_Busy is off. The DEU_Message_Ready_Flags for this DEU is reset. If the Reconfiguration_In_Progress is zero, the DEU_Pole_Flags for this DEU is set.

If the Annunciation_Error_ID is not -1, it and the DEU_Number are passed in a call to Annunciation_Macro_Interface.

d. Outputs - See Table 3.2.1.1.1-1.

e. Module References -

1. (104) Wait_Processor (FPMWAIT) is called.
2. (520) MCDS_Display_Coordination (DMC_Display) is called.
3. (510) MCDS_ITEM_Processor (DMC_ITEM_PROC) is called.
4. (560) Sequence_Request_Processor (DMC_SEQ_REQ_PROC) is called.
5. (171) Reset_Event_Processor (FPMRESET) is called.
6. (170) Set_Event_Processor (FPMSET) is called.
7. (480) ICC_Message_Collector (DIM_ICC_COLLECTOR) is called.
8. (675) Annunciation_Macro_Interface (DMA_MAC) is called.

f. Module Attributes - Internal Procedure

g. Template References - none

h. Error Handling -

1. Invalid CMPTR/CRT key input are rejected with an SC009 error message.
2. Resume key input with OPS supported is rejected with a DU011 error message.
3. Invalid CMPTR/BUS key inputs are rejected with a SC003 error message.
4. EXEC key inputs are rejected if incompatible with current format or current format not supported, or display supported with DU014, DU028, or DU014 respectively.
5. PRO key inputs are rejected if incompatible with current format, or current format not supported, or display supported with DU015, DU028, DU015 respectively.

i. Constraints and Assumptions - none.

j. Detailed Implementation - none.

DATA TABLE 3.2.1.1.1-1

NAME MCDS_Functions_Processor (DMC_FUNCTIONS)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	Function_Case_Offset	L284	L	505	505	DMC_CASE_OFFSET			
2	MF_Change_Function_Number	L285	L	505	505	DMCKL_MF_CHANGE			
3	Cyclic_In_Process_Event	B090	ZI	620	500,505, 520,560	CDME_CYCLIC_EVENT			
4	MCDS_FUNCTION	L219	LIO	500 505	500,505 510	DMC_CASE			
5	ID_GPC	L190	L	505	505,520 560	DMC_GPCID			
6	GPC_ID	#001	ZI	300	See Appendix E	TFCMID			
7	Annunication_Error_ID	L286	LO	505	505	DMC_FMPT_ERR			
8	Active_Control_Segment_MACT_Index	L118.70	LI	560	505 560	DMCV_RQMACT_IN			
9	MCDS_MAJOR_FUNCTION	L192	LIO	500,505 560	See Appendix E	DMC_MF			
10	DEU_Number	L127	LO	500	See Appendix E	DMC_DIT_INDEX			
11	MCDS_Allocation_Table	B010	ZIO			CDMV_MAT_TABLE			
12	MCDS_MACT_Index	L126	LO	505	505,510 520,560	DMC_MACT_INDEX			
13	Miscellaneous_Application_Control_Table	J040	ZIO	See Appendix E	See Appendix E	CZIV_D_MACT			
14	DEU_Number	L287	L	505	505	DMC_DEU_NUMBER			
15	Next_Input	L223	LIO	505,510	505 510	DMCV_NXT_INP			
16	Display_Request_Level	L194	LO	505,510 520,560	505,510 520,560	DMCVL_D_LVLN			
17	ICC_Status_Word	L288	LI	480,505 505	480 505	DMC_ICC_STATUS			
18	MM/DFT_ID_Completion	L281	LO	500,505 520	505 520	DMC_IO			
19	Loop_Counter	L289	L	505	505	DMC_I			
20	MACT_Copies	J006	ZI		505 560	CZ1K_D_MACT_COPIES			

DATA TABLE 3.2.1.1.1-1 (cont'd)

NAME MCDS_Functions_Processor (DMC_FUNCTIONS)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
21	MACT_Service_Request_Ready_Events	J160	Z	See App. E	505,810	CZ1E_D_MACT_EVTS			
22	Major_Function_ID	J040.35	Z	560	505,560	CZ1V_D_MF_ID			
23	DEU_Request_MACT_Index	L128	LI	560	500,505, 560	DMCV_DEU_MACT			
24	MACT_Service_Request_Flags	J040.01	ZIO	See Appendix E	505,540, 810,820	CZ1B_D_FLAGS			
25	New_Display_Request_Flag	J040.03	Z	505,540	505				
26	New_Display_Number	J040.40	ZIO	505 540	505	CZIV_D_NEW_DISP			
27	Display_Request_Number	L195	L	505,510 520,560	505,510 520,560	DMCVL_D_DISP			
28	ITEM_Processing_Complete_Flag	J040.07	ZJO	505,540	505				
29	Keyboard_Input_DEU_Number	J040.51	ZI	505,510 560	505,810, 900,910	CZ1V_D_DEU_NUMBER			
30	MAT_Major _Function_Setting	B010.10	ZIO	505,520	505,560 600,620	CDMV_MAT_MF			
31	Control_Segment_Termination_Flag	J040.09	Z	505,550	505				
32	GPC_Reconfiguration_Completion	J040.13	ZIO	505,820, 870,880	505,560				
33	GPC_Secondary_Reconfiguration_Request	J040.14	ZIO	505,820 870	505				
34	DEU_Display_Support_Level	B010.15	ZIO	505,520, 620	505,560 600	CDMV_MAT_LEVEL			
35	Display_Level_Page_Number	B010.35	ZIO	505,560, 560	505,620	CDMV_MAT_DISP_NUMBER			
36	CRT_Status_Flags	B010.40	ZIO	See Appendix E		CDMB_MAT_CRT_STAT			
37	Keyboard_Message	J060.20	ZI	See Appendix E		CZIV_D_DIT_KYBD_MSG	See Appendix E		
38	Reconfiguration_In_Initialization	L193	LIO	500,505 560	505,560	DMC_REC_IP2			
39	Blank_Scratch_Pad_Line_Flag	L282	LIO	500,505	600	DMC_CLEAR_SPL			
40	ICC_Message_For_CMPTR/CRT_Key	L290	LO	505	480,830	DMC_CMPTR_CRT_MSG			

DATA TABLE 3.2.1.1.1-1 (Cont'd)

NAME MCDS_Functions_Processor (DMC_FUNCTIONS)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
41	Keyboard_Message_ID	J040.49	ZO	505,510 540	540,810 900,910	CZ1V_D_KEY_ID			
42	MACT_Application_Control_Flags	J040.18	ZO	505,550 560	550	CZ1B_D_FLAG2			
43	Name_of_Keyboard_Message_Event	J040.45	ZO	See App. E	540	CZ1E_D_KYBD_MSG_EVT			
44	ICC_Message_For_CMPTR/BUS_Key	L291	LO	505	480,830	DMC_CMPTR_BUS_MSG			
45	DEU_Message_Ready_Flags	J050	ZO	405,505 815,830	500,505	CZ1B_D_DIT_MSG_READY			
46	DEU_Poll_Flag	J062	ZIO	See Appendix E		CZ1B_D_DEU_POLL_FLAG			
47	MCDS_Input_Busy	I680	ZI	400	See App. E	CZ2V_MCDS_IN			
48	DEU_Input_Table	I060	ZI	See Appendix E		CZ1V_D_DIT			
49	Current_Display_Number	J040.36	ZIO	505,540 550,560	505,560	CZ1V_D_DISP			
50	CRT_Status_Flag1	B010.41	ZIO	400,505 600	505,620 625				
51	CRT_Status_Flag2	B010.42	ZIO	400,505	505,620				
52	Keyboard_Verification_Table_Data	L220.40	LI	520	505,510 520	DMCB_KVT_DATA			
53	Item_PRO_EXEC_Allow	L292	LI	505	505	DMCVC_ITEM_PRO_EXEC_ALLOW			
54	DEU_Number_of_Keystrokes	J060.19	ZI	See Appendix E		CZ1V_D_DIT_NUMOFKEYS	V93M2002P V93M2102P V93M2202P		

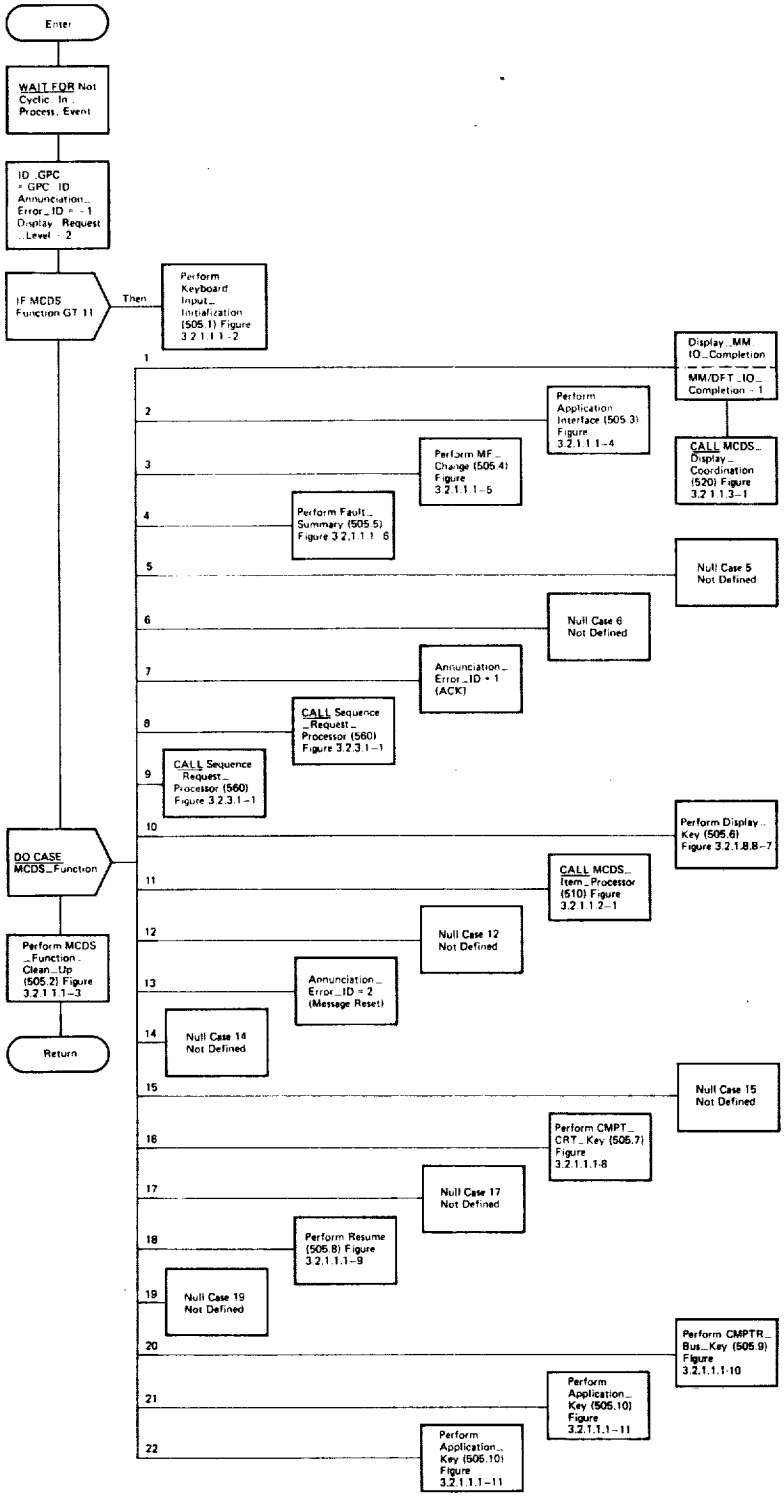


Figure 3.2.1.1.1-1. MCDS_Functions_Processor (DMC_Functions)

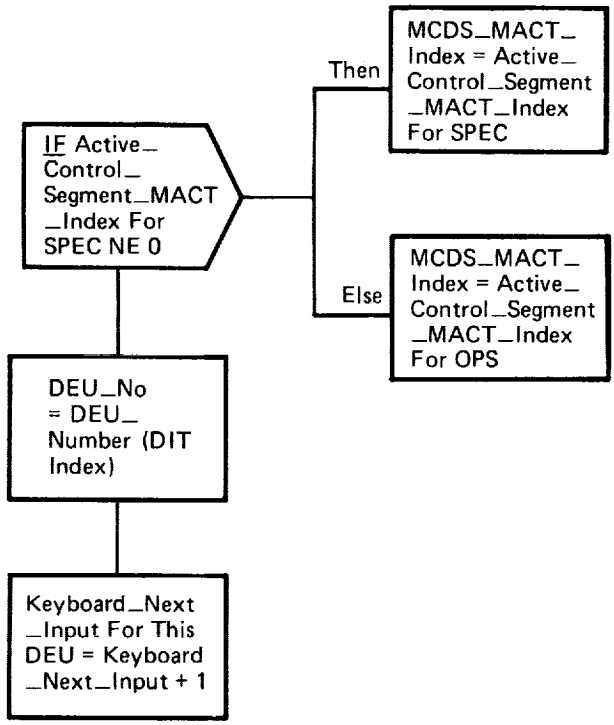


Figure 3.2.1.1.1-2. MCDS_Functions_Processor Keyboard_Input_Initialization (505.1)

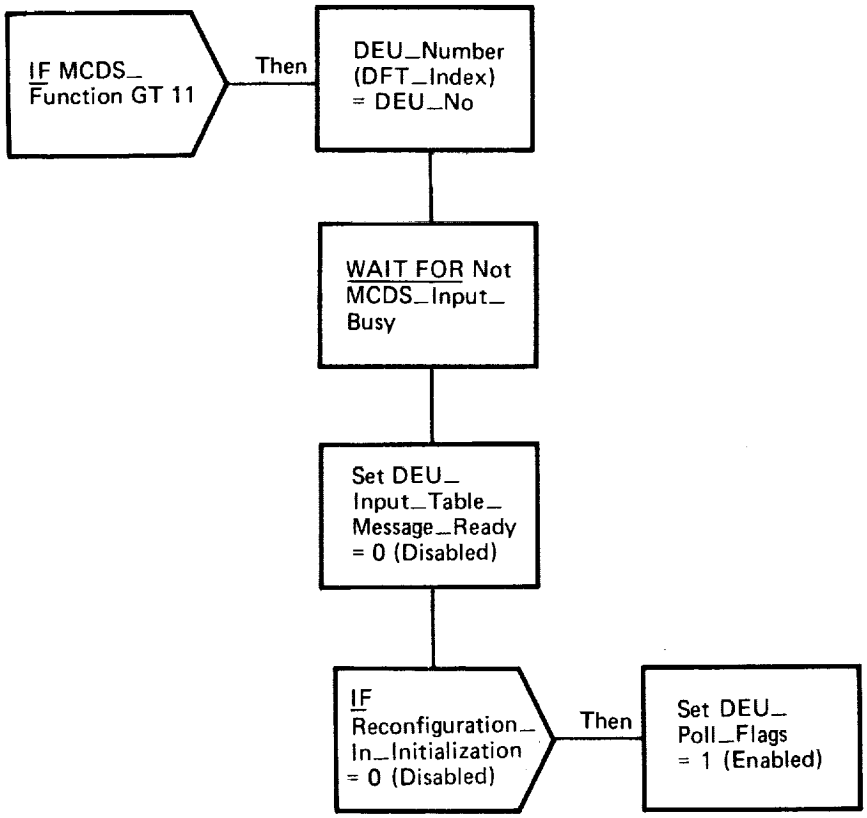


Figure 3.2.1.1.1-3. MCDS_Functions_Processor MCDS_Functions_Clean_Up (505.2)

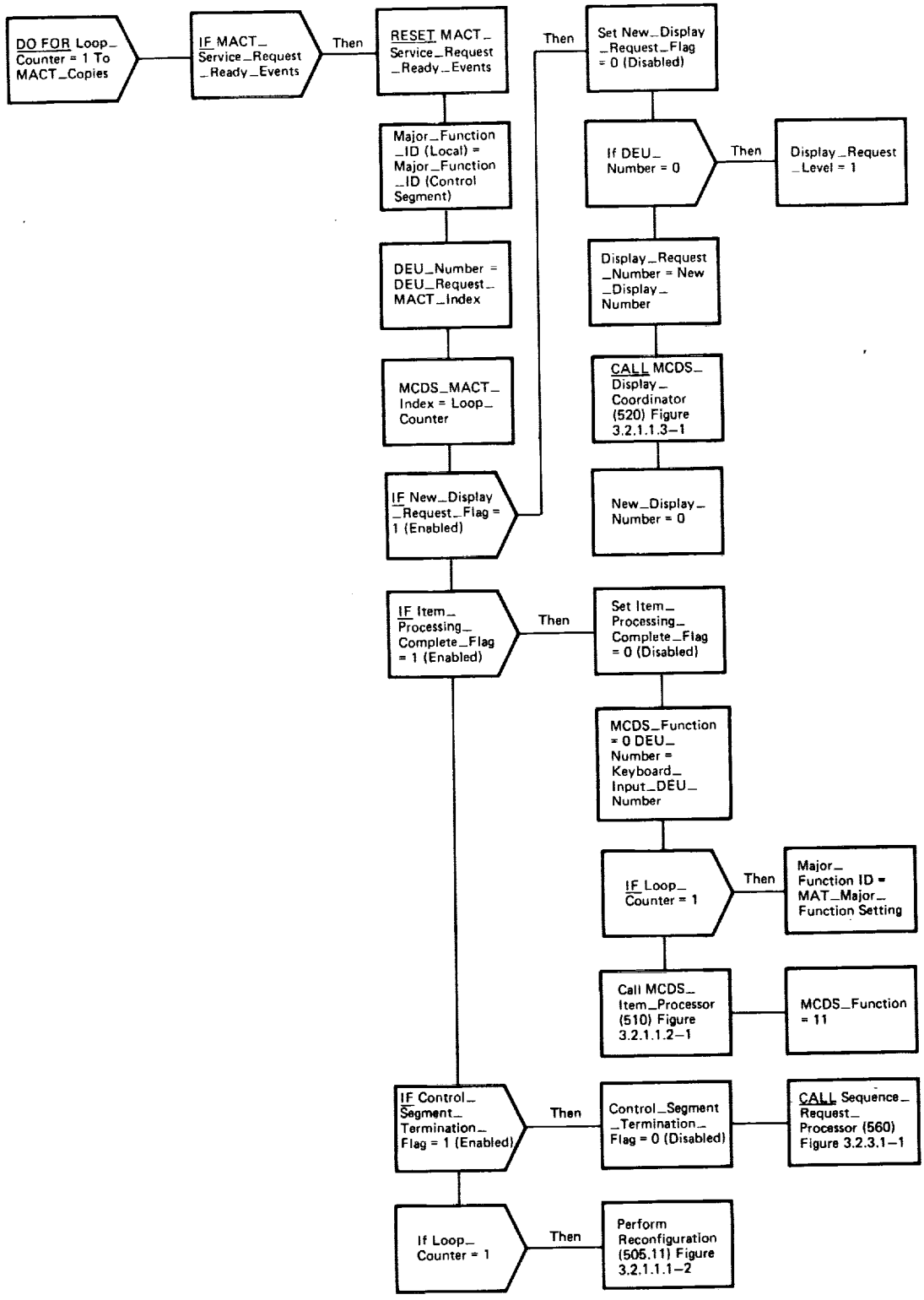


Figure 3.2.1.1.1-4. MCDS_Functions_Processor Application_Interface (505.3)

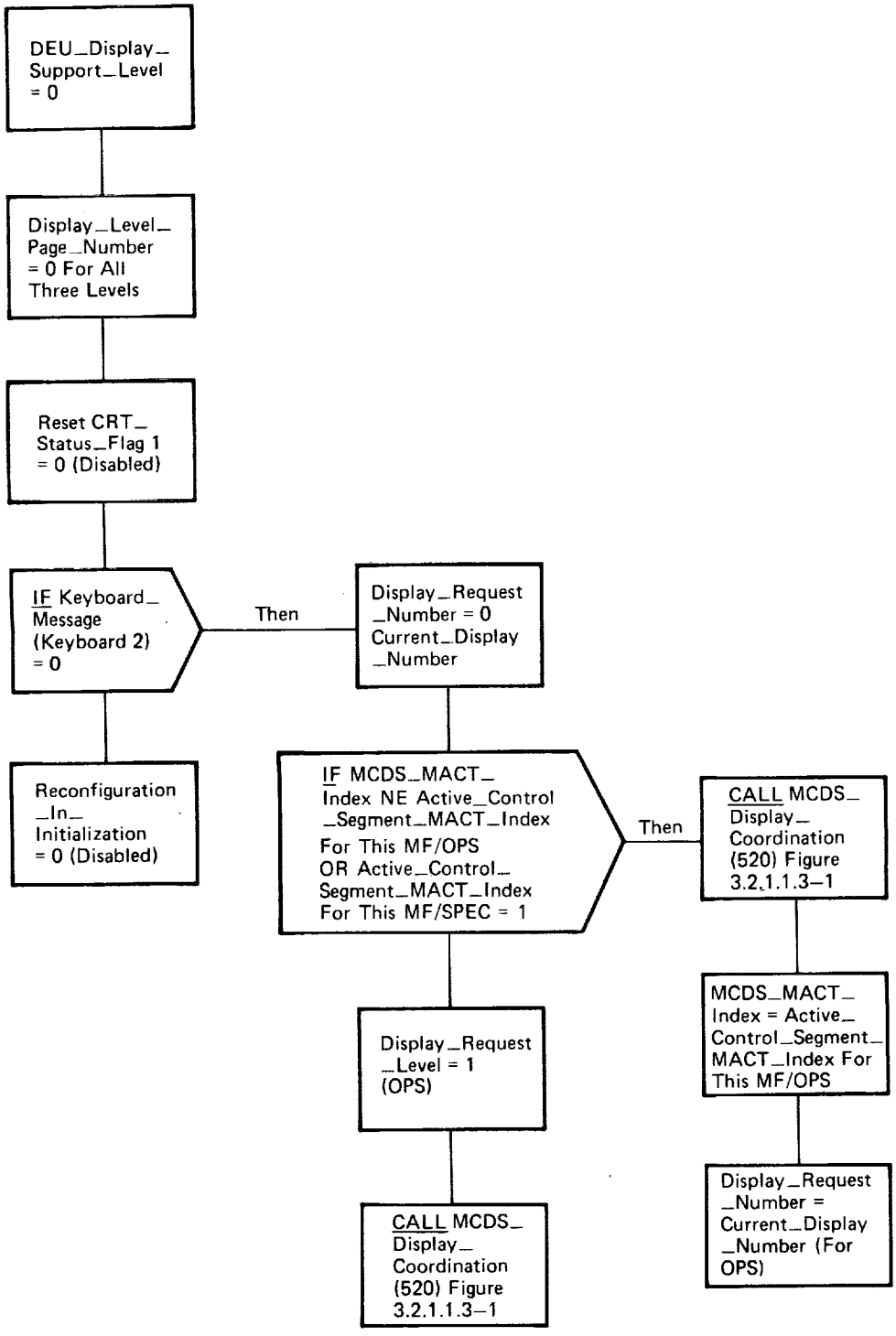


Figure 3.2.1.1.1-5. MCDS_Functions_Processor MF_Change (505.4)

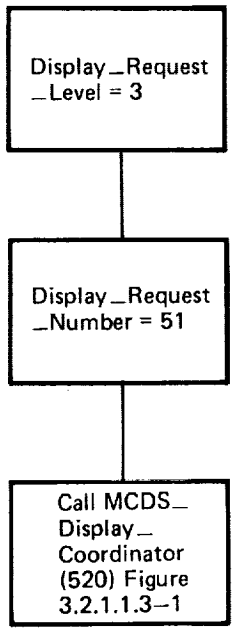


Figure 3.2.1.1.1-6. MCDS_Functions_Processor Fault_Summary (505.5)

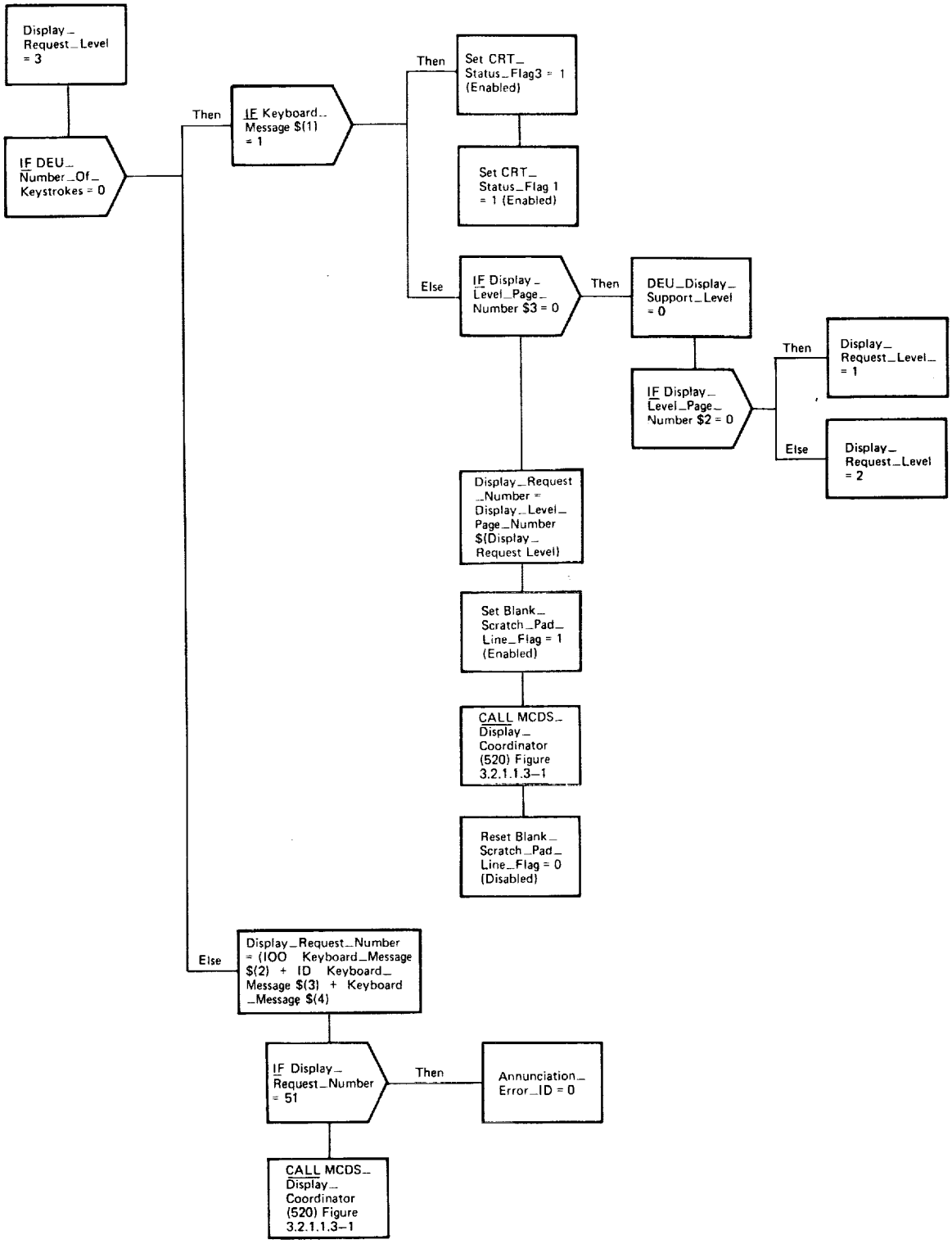


Figure 3.2.1.1.1-7. MCDS_Functions_Processor Display_Key (505.6)

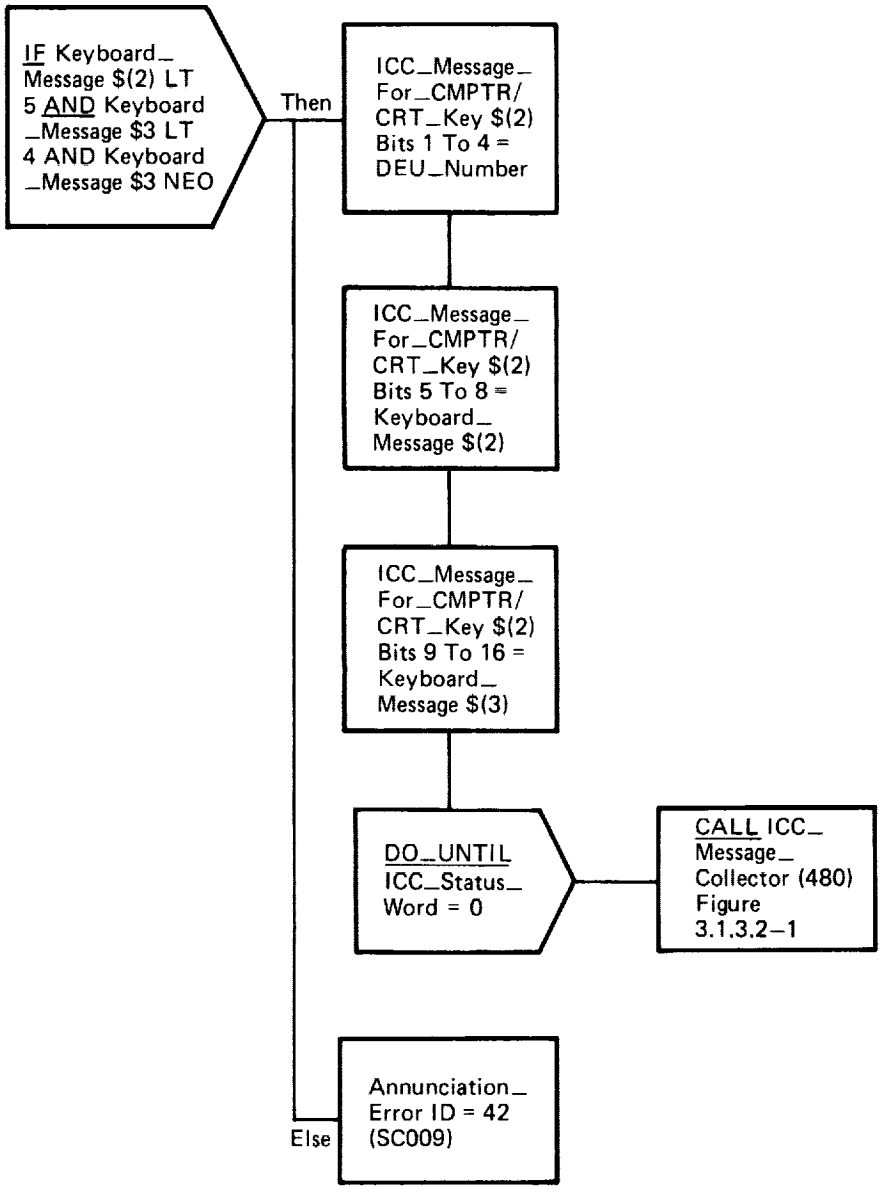


Figure 3.2.1.1.1-8. MCDS_Functions_Processor CMPT_CRT_Key (505.7)

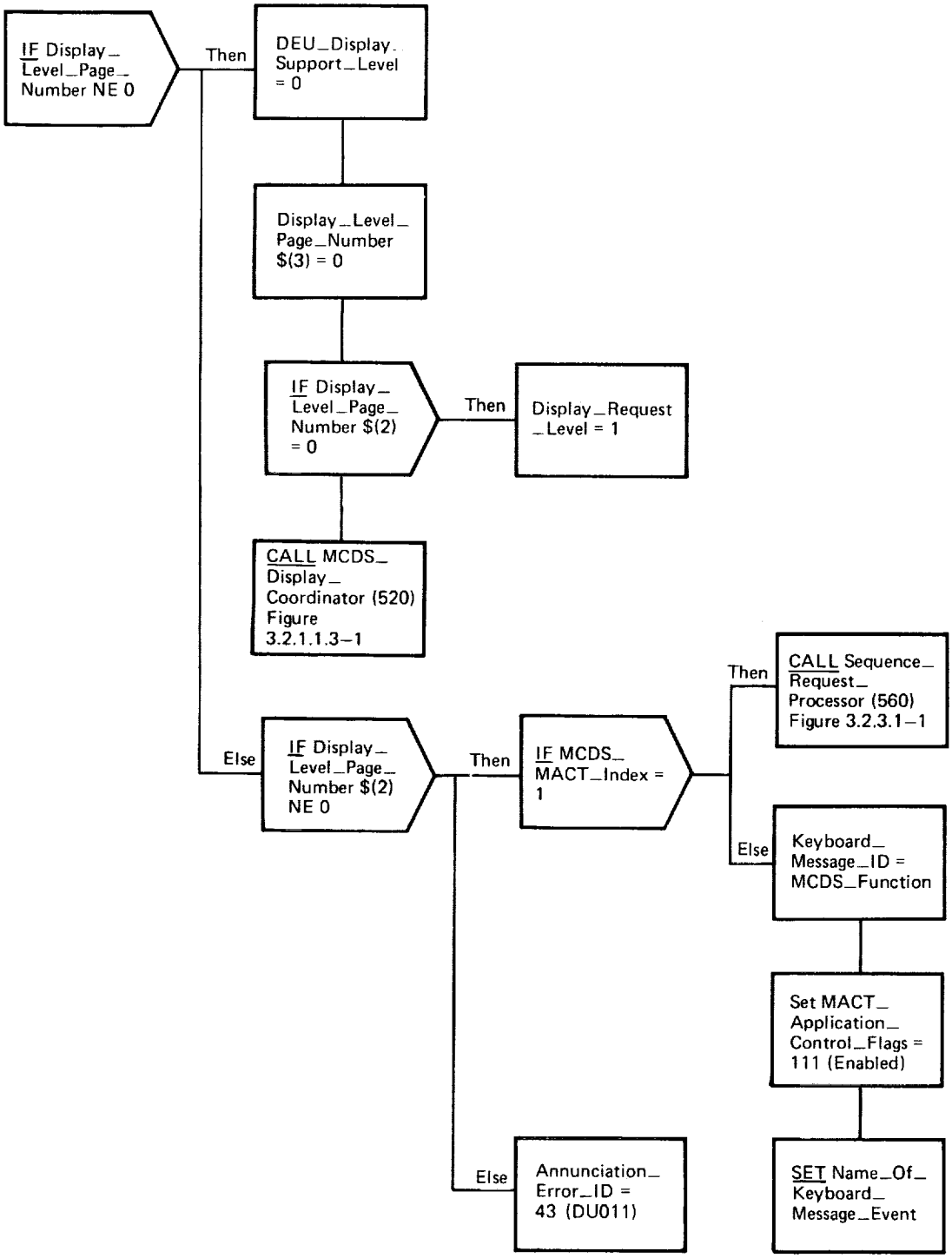


Figure 3.2.1.1.1-9. MCDS_Functions_Processor Resume (505.8)

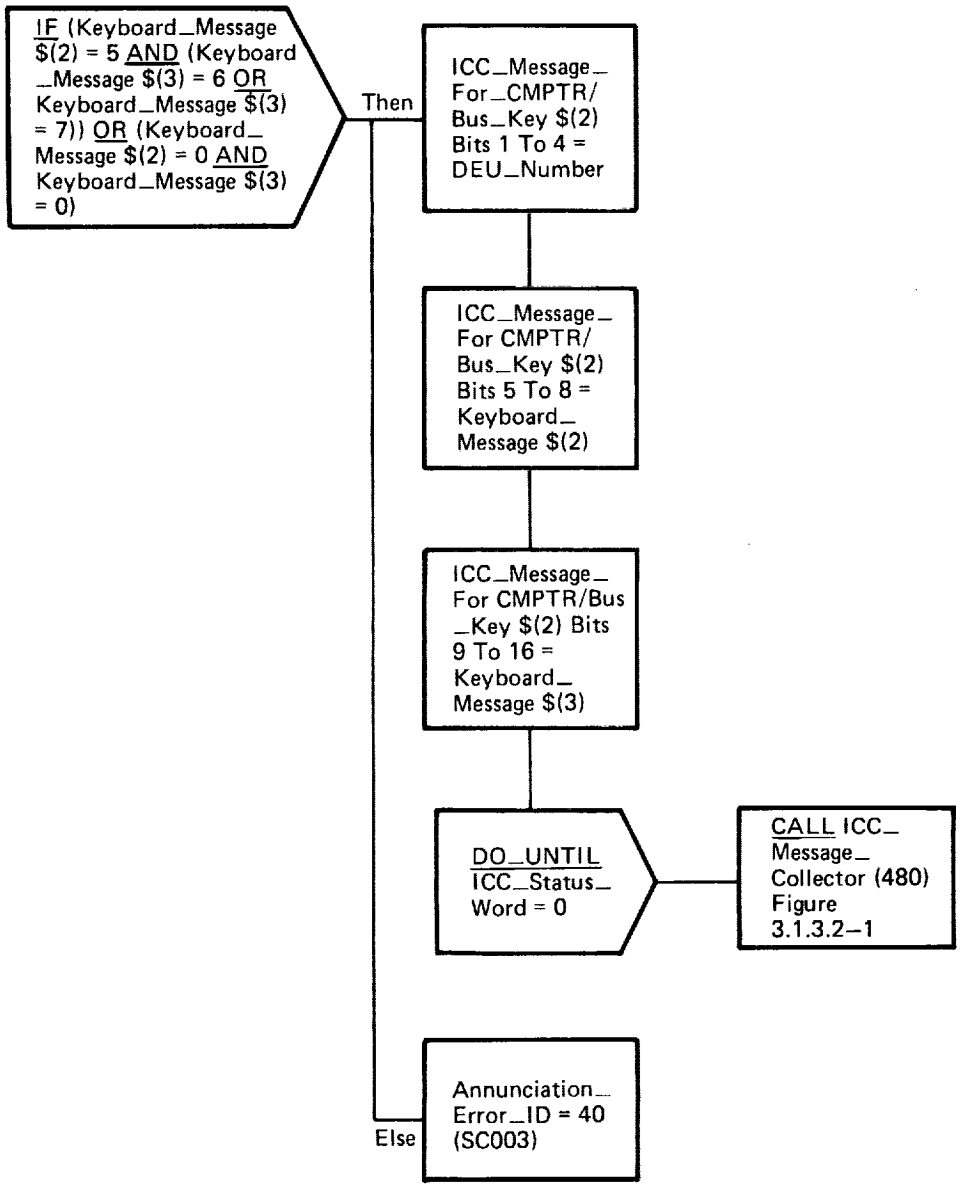


Figure 3.2.1.1.1-10. MCDS_Functions_Processor CMPTR_Bus_Key (505.9)

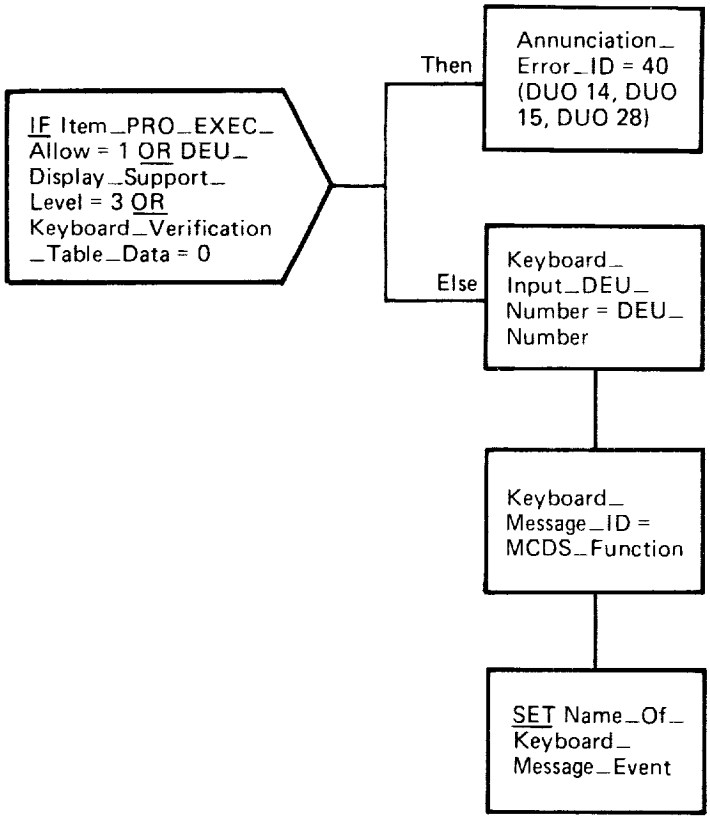


Figure 3.2.1.1.1-11. MCDS_Functions_Processor Application_Key (505.10)

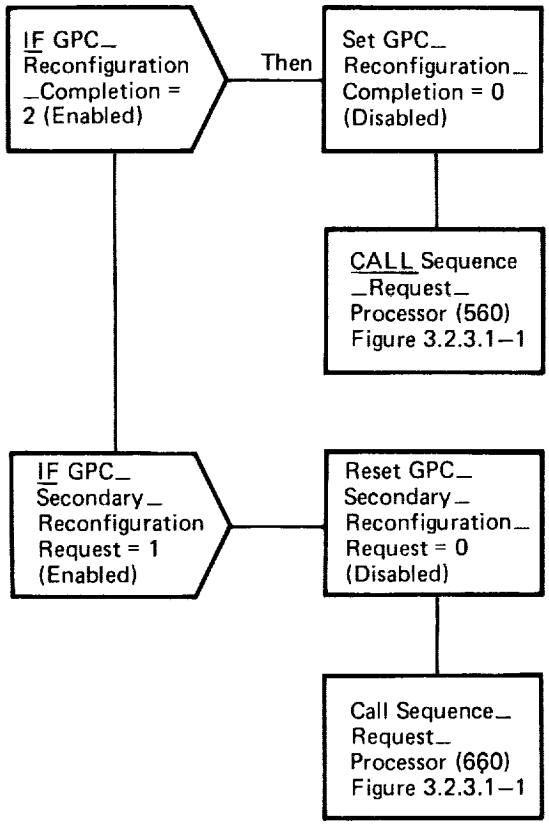


Figure 3.2.1.1.1-12. MCDS_Functions_Processor Reconfigurator (505.11)

3.2.1.1.2 MCDS_ITEM_Processor (DMC_ITEM_PROC) (510)

To be provided.

3.2.1.1.3 MCDS_Display_Coordinator (DMC_DISPLAY) (520)

MCDS_Display_Coordinator searches all Display Format Buffers (DFB's) to determine the starting location of the Display Format Table (DFT) containing the requested display number. If found, it updates the MCDS_Allocation Table (MAT) and performs the logic necessary to control output of the background FCW's to the DEU. If not found, it initiates a Mass Memory read to load the proper DFT into the buffer.

a. Control Interface -

1. CALL DMC_DISPLAY
2. a.CALLed by (505) MCDS_Functions_Processor (DMC_FUNCTIONS)
   b.CALLed by (560) Sequence_Request_Processor (DMC_SEQ-REQ_PRQC)

b. Inputs -

See Table 3.2.1.1.3-1

c. Process Description - MCDS_Display Coordinator receives requests for new displays based upon 1) keyboard inputs, 2) MF switch changes at a DEU 3) status change of a DEU (FREEZE), and 4) requests via initiated control segments. There are five Display Format Buffers (DFB's) which contain the Display Format Tables (DFT's) necessary to support a given display request. DFB 1 contains permanently resident displays. DFB 2 contains displays resident for a given MF overlay. DFB 3 contains displays resident for a given program overlay. DFB 4 and DFB 5 are buffers set aside for displays to be read from the Mass Memory (MM) as needed. The MM addresses to these displays are contained in the DFT MM Directory (DMMD). Each of the five DFB's are searched for a display. If found, the MCDS_Allocation_Table (MAT) is updated and the background is output via New_Display_Processor. If not found in one of the five DFB's, the DMMD is searched to determine if the required display is available on the MM. If found, the necessary I/O is initiated to obtain the DFT for the requested display (provided DFB 4 and DFB 5 are not both busy with a previous display request) into DFB 4 or DFB 5. A subsequent request for the display will be made by MCDS_Functions_Processor when the I/O is complete should any error be encountered (i.e., IO error to MM or DEU, requested display not found, DFB 4 and 5 busy, etc.) an appropriate error will be annunciated via the Annunciation_Macro_Interface.

d. Outputs -

See Table 3.2.1.1.3-1

e. Module References -

1. (600) New_Display_Processor (DMC_NEW_DISPLAY) is called
2. (675) Annunication Macro_Interface (DMA-MAC) is called
3. (104) Wait_Processor (FPMWAIT) is called.

f. Module Attributes -

Internal Procedure

g. Template References -

None

h. Error Handling -

1. Display not found in DFB 1 requested during GPC reconfiguration rejected with a DU028 error.

2. Invalid display request rejected with a DU010 error.

3. Buffer not available for MM DFT error indicated with a DU016 error.

i. Constraints and Assumptions -

1. All displays will be formatted into DFT's via the offline Display_Format_Generator (DFG)

2. Each memory configuration will contain valid DFB's 2,3,4, and 5 and a valid DMMD.

j. Detailed Implementation -

See Figure 3.2.1.1.3-1

DATA TABLE Table 3.2.1.1.3-1.

NAME MCDS_Display_Coordinator (DMC_Display)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	FMPT_ERROR_ID	L293	L	520	675	DMC_FMPTE			
2	Display_Request_Status	L294	L	520	520	DMCVL_D_STATUS			
3	MM/DFT_IO_Completion	L281	LIO	500,505 520	505 520	DMC_IO			
4	I_Index	L295	L	520	520	DMC_I			
5	DFB_Pointer	L296	L	520	520	DMC_BUFP			
6	DFB_Store_Index	L297	L	520	520	DMC_B_STORI			
7	DFB_Read_Status	L298	L	520	520	DFB_STAT_I			
8	J_Index	L299	L	520	520	DMC_J			
9	Display_Buffer_Availability_Indicator	I600.01	LIO	520, 560 995	520, 560 995	CZ2V_RBUF_DISP			
10	MCDS_MACT_Index	L126	LI	505	505,510, 520,560	DMC_MACT_INDEX			
11	Save_MACT_Index	L300	L	520	520	DMC_SAVMACT			
12	DFB_Buffer_Number	L301	L	520	520	DMC_BUFFI			
13	Display_Request_Number	L195	LI	505,510, 520,560	505,510, 520,560	DMCVL_D_DISP			
14	Display_Buffer_ Allocation_Indicator	I600.02	Z	520,560, 995	520,560, 995	CZ2V_RBUF_P			
15	Display_Request_Level	L194	LIO	505,510 520,560	505,510 520,560	DMCVL_D_LVLN			
16	IO_Display_Level	L302.20	L	520	520	DMC_RBUF_LVLN			
17	MCDS_Major_Function	L192	LIO	500,505 560	See App.E	DMC_MF			
18	IO_Display_MF	L302.40		520	520	DMC_RBUF_MF			
19	DEU_Number	L127	LI	500	See Ap E	DMC_DIT_INDEX			
20	Keyboard_Verification_Table_Data	L220.40		520	505,510 520	DMCB_KVT_DATA			

DATA TABLE Table 3.2.1.1.3-1. (Cont'd)

NAME MCDS_Display_Coordinator (DMC_Display)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
21	KVT_Data_Save	L303	LI	520	520	DMCB_SAV_KVT			
22	IO_Display_DEU	L302.30	L	520	520	DMC_RBUF_DEU			
23	Display_Found_ID	L304	L	520	520	DMC_FOUND			
24	DFB_Number	L305	L	520	520	DMCVC_D_BUFFNO			
25	Reconfiguration_In_Progress	L120	LI	560	500,510, 520,560	DMC_REC_IN_PROGRESS			
26	DFB_Index	L306	L	520	520	DMC_BUFFI			
27	AMT_Table_Valid	L177	LI	560	520 560	DMC_AMTVALID			
28	IO_Display_Buffer_Count	L302.10	L	520	520	DMC_RBUF_COUNT			
29	DFB_Number_For_Processing_Display	B010.30	ZIO	520,540, 560	540,600, 620	CDMV_MAT_DFB_NUMBER			
30	F_Index	L307	L	520	520	DMC_F			
31	DFB_Offset_Index	L308	L	520	520	DMCVL_D_OFFSET			
32	Number_of_Displays	L309	L	520	520	DMC_NBRDISP			
33	DFT_DMF	L310	L	520	520	DMC_DMF			
34	Display_Buffer_Name	L178.10	LI	560	510,520, 560,600	DMC_BUF			
35	Block_String	L311	L	520	520	DMCB_BLK_STRING			
36	Valid_DEU	L312	L	520	520	DMC_VDEU			
37	Number_of_DFTs	L313	L	520	520	DMC_NDFT			
38	DFB_Max_Size	L314	L	520	520	DMC_DFBSZE			
39	DFT_Length	L315	L	520	520	DMC_INCR			
40	Bit_Value	L316	L	520	520	DMC_TVALB			

DATA TABLE Table 3.2.1.1.3-1. (Cont'd)

NAME MCDS_Display_Coordinator (DMC_Display)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
41	DFT_Display	L317	L	520	520	DMC_DDISP			
42	Current_Display_Number	J040.36	ZO	505,540 550,560	505,560	CZ1V_D_DISP			
43									
44	K_Index	L319	L	520	520	DMC_K			
45	Start_DEU	L320	L	520	520	DMC_SDEU			
46	End_DEU	L321	L	520	520	DMC_EDEU			
47	MAT_Major_Function_Setting	B010.10	ZO	505,520	505,560, 600,620	CDMV_MAT_MF			
48	Cyclic_In_Process_Event	B009	ZI	620	See Ap E	CDME_CYCLIC_EVENT			
49	Display_Level_Page_Number	B010.35	ZIO	505,520, 560	505,620	CDMV_MAT_DISP_NUMBER			
50	Processing_Display_DFB_Index	B010.25	ZO	520,540, 560	540,600, 620	CDMV_MAT_DFB_INDEX			
51	CRT_Status_Flags	B010.40	ZO	See Ap E	See Ap E	CDMB_MAT_CRT_STAT			
52	DEU_Display_Support_Level	B010.15	Z	505,520, 560	505,560, 600	CDMV_MAT_LEVEL			
53	Display_Number_With_Major_Function_Setting	B020.20	ZO	520	660	CDMV_UI_DOWNLIST_FORMAT_ID			
54	ICC_Status_Flags	I320	ZO	See Ap E	See Ap E	CZ2B_ICC_FLAG			
55	Scalar_Limits_Offset	L220.20	LO	520	510	DMCV_SL_OST			
56	General_Limits_Offset	L220.30	LO	520	510	DMCV_GL_OST			
57	Item_Start_Offset	L220.10	LO	520	510	DMCV_ITEM_START			
58	Read_Pending_Indicator	L322	L	520	520	DMC_RPEND			
59	New_Display_Status	L088	LIO	505	505,600	DMC_Status			
60	DFB_4_Read_Table	L323	LO	520	520	DMC_MMD_PLIS4			

DATA TABLE Table 3.2.1.1.3-1. (Cont'd)

NAME MCDS_Display_Coordinator (DMC_Display)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
61	DFB_5_Read_Table	L324	LO	520	520	DMC_MMRD_PLIS5			
62	DFB_4_Read_MM_Address	L323.10	LO	520	520	DMC_MMRD_PLIS4.DBLOCK			
63	DFB_5_Read_MM_Address	L324.10	LO	520	520	DMC_MMRD_PLIS5.DBLOCK			

Table 3.2.1.1.3-2

DEVICE	DEVICE ID	OP CODE	SVC NO.	SYNC TYPE	COMFAULT STATUS	ICC TYPE	TIME TAG	WAIT	PROT.	STATUS	INPUT OUT-PUT	MAJ. FUNC. ID	I/O PRIOR. ITY	WORD COUNT	BUFFER	EVENT	BUS NOMINAL
MMU	10	3	24	RS	NO	No	No	No	Yes	Yes	I	1,2,3	N/A	706	DFB4 or DFBS	TIME_ID_MM_ EVENT	18,19

DATA TABLE 3.2.1.1.1-1 (cont'd)

NAME MCDS_Functions_Processor (DMC_FUNCTIONS)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
21	MACT_Service_Request_Ready_Events	J160	Z	See App. E	505,810	CZ1E_D_MACT_EVTS			
22	Major_Function_ID	J040.35	Z	560	505,560	CZ1V_D_MF_ID			
23	DEU_Request_MACT_Index	L128	LI	560	500,505, 560	DMCV_DEU_MACT			
24	MACT_Service_Request_Flags	J040.01	ZIO	See Appendix E	505,540, 810,820	CZ1B_D_FLAGS			
25	New_Display_Request_Flag	J040.03	Z	505,540	505				
26	New_Display_Number	J040.40	ZIO	505 540	505	CZIV_D_NEW_DISP			
27	Display_Request_Number	L195	L	505,510 520,560	505,510 520,560	DMCVL_D_DISP			
28	ITEM_Processing_Complete_Flag	J040.07	ZJO	505,540	505				
29	Keyboard_Input_DEU_Number	J040.51	ZI	505,510 560	505,810, 900,910	CZ1V_D_DEU_NUMBER			
30	MAT_Major _Function_Setting	B010.10	ZIO	505,520	505,560 600,620	CDMV_MAT_MF			
31	Control_Segment_Termination_Flag	J040.09	Z	505,550	505				
32	GPC_Reconfiguration_Completion	J040.13	ZIO	505,820, 870,880	505,560				
33	GPC_Secondary_Reconfiguration_Request	J040.14	ZIO	505,820 870	505				
34	DEU_Display_Support_Level	B010.15	ZIO	505,520, 620	505,560 600	CDMV_MAT_LEVEL			
35	Display_Level_Page_Number	B010.35	ZIO	505,560, 560	505,620	CDMV_MAT_DISP_NUMBER			
36	CRT_Status_Flags	B010.40	ZIO	See Appendix E		CDMB_MAT_CRT_STAT			
37	Keyboard_Message	J060.20	ZI	See Appendix E		CZIV_D_DIT_KYBD_MSG	See Appendix E		
38	Reconfiguration_In_Initialization	L193	LIO	500,505 560	505,560	DMC_REC_IP2			
39	Blank_Scratch_Pad_Line_Flag	L282	LIO	500,505	600	DMC_CLEAR_SPL			
40	ICC_Message_For_CMPTR/CRT_Key	L290	LO	505	480,830	DMC_CMPTR_CRT_MSG			

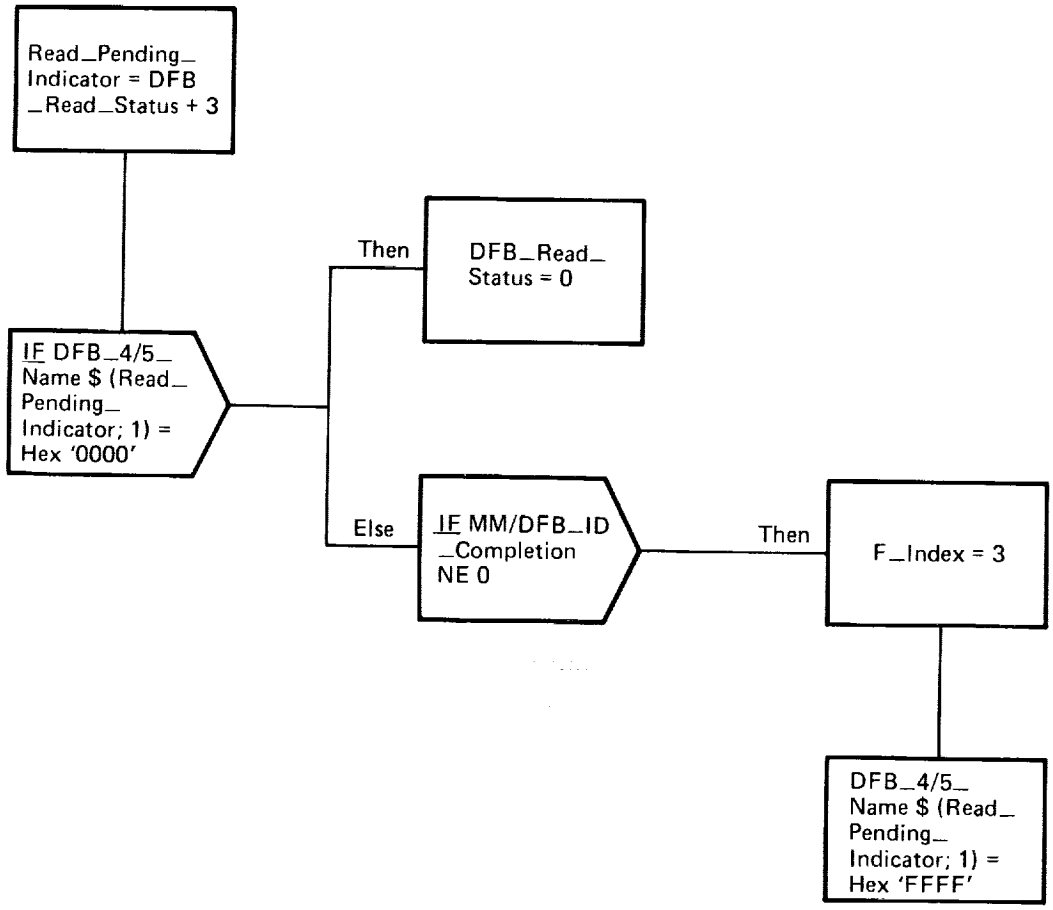


Figure 3.2.1.1.3-2. MCDS_Display_Coordinator DFB_Buffer_Status (520.1)

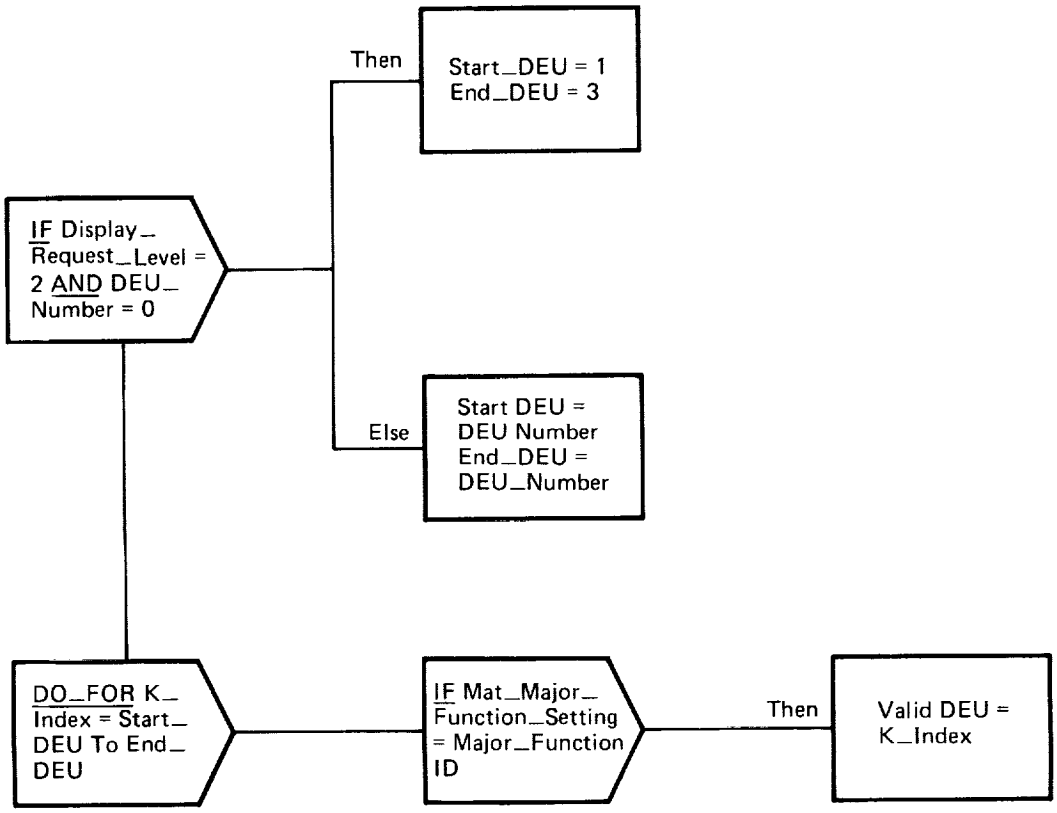


Figure 3.2.1.1.3-3. MCDS_Display_Coordinator DEU_Find (520.2)

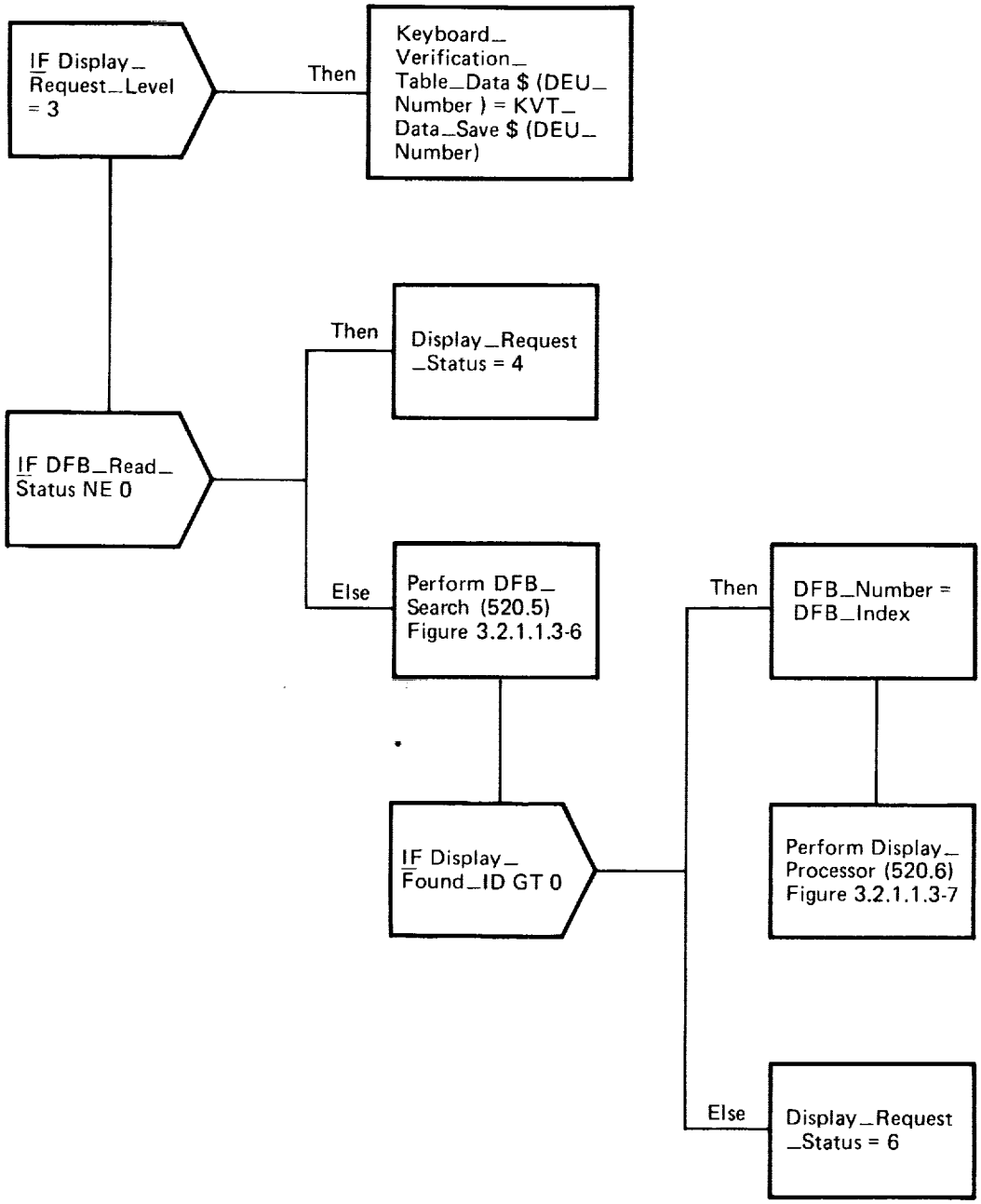


Figure 3.2.1.1.3-4. MCDS_Display_Coordinator IO_Display_Complete (620.3)

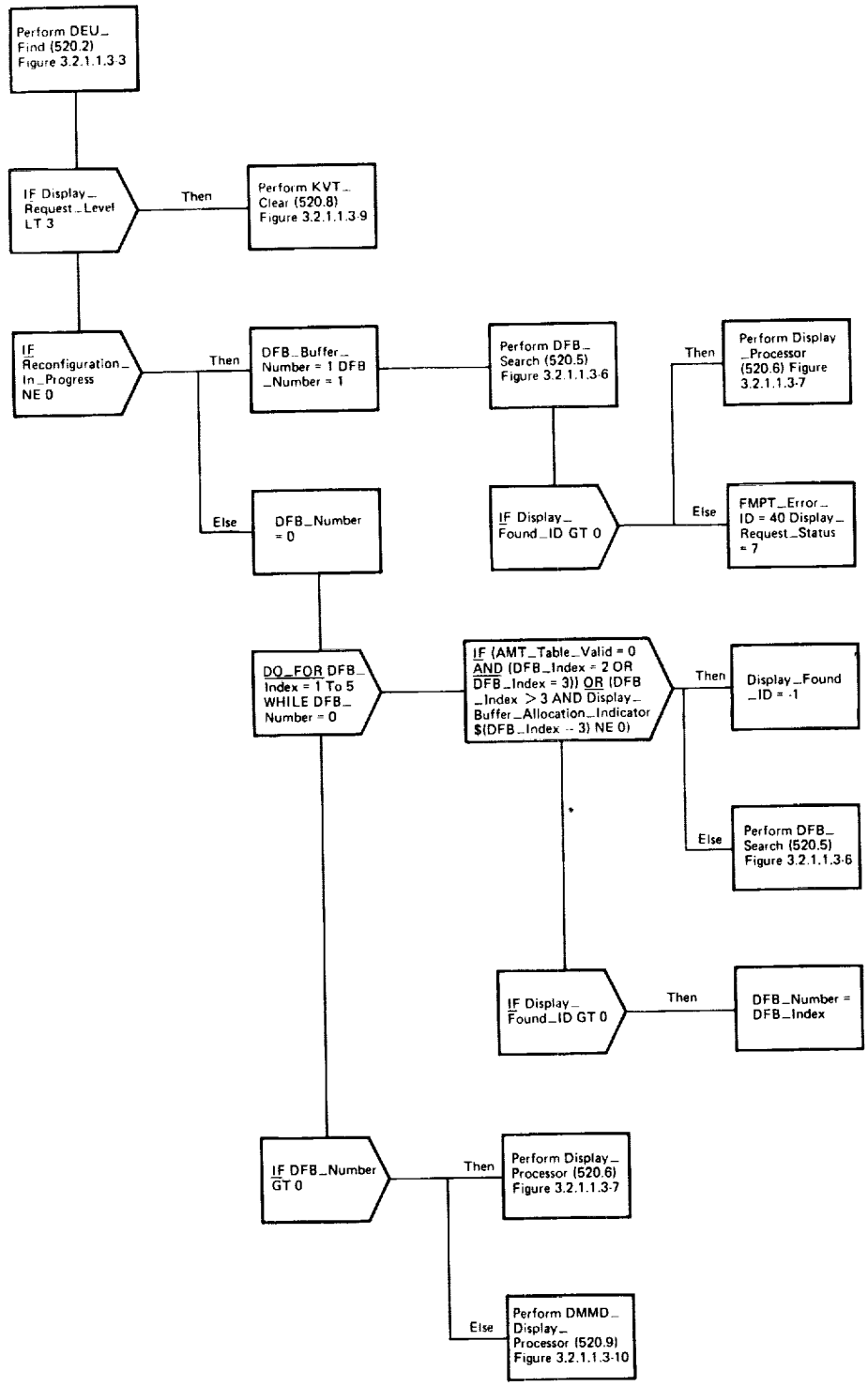


Figure 3.2.1.1.3-5. MCDS_Display_Coordinator DMMD_Display_Processor (520.9)

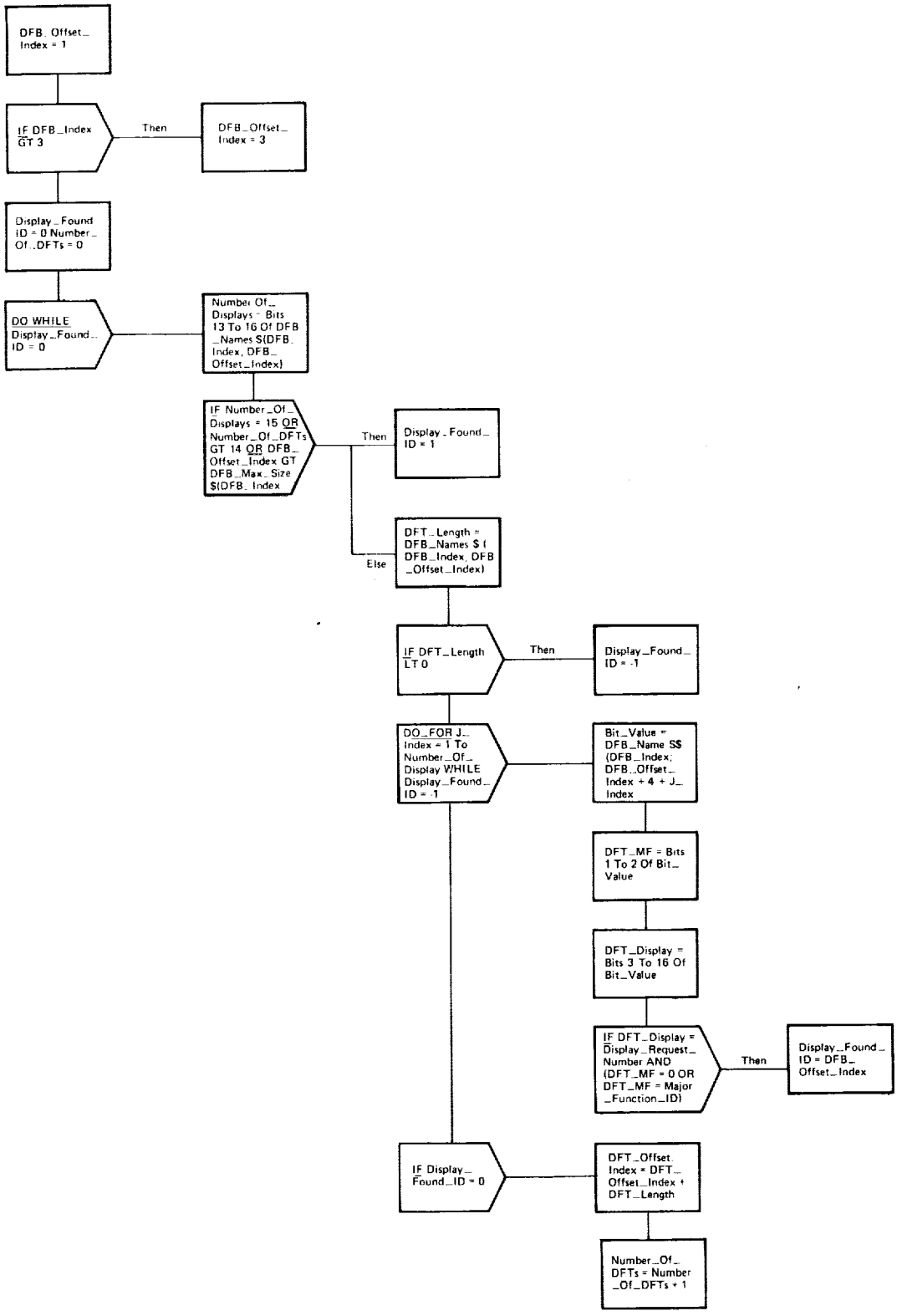


Figure 3.2.1.1.3-6. MCDS_Display_Coordinator DFB_Search (520.5)

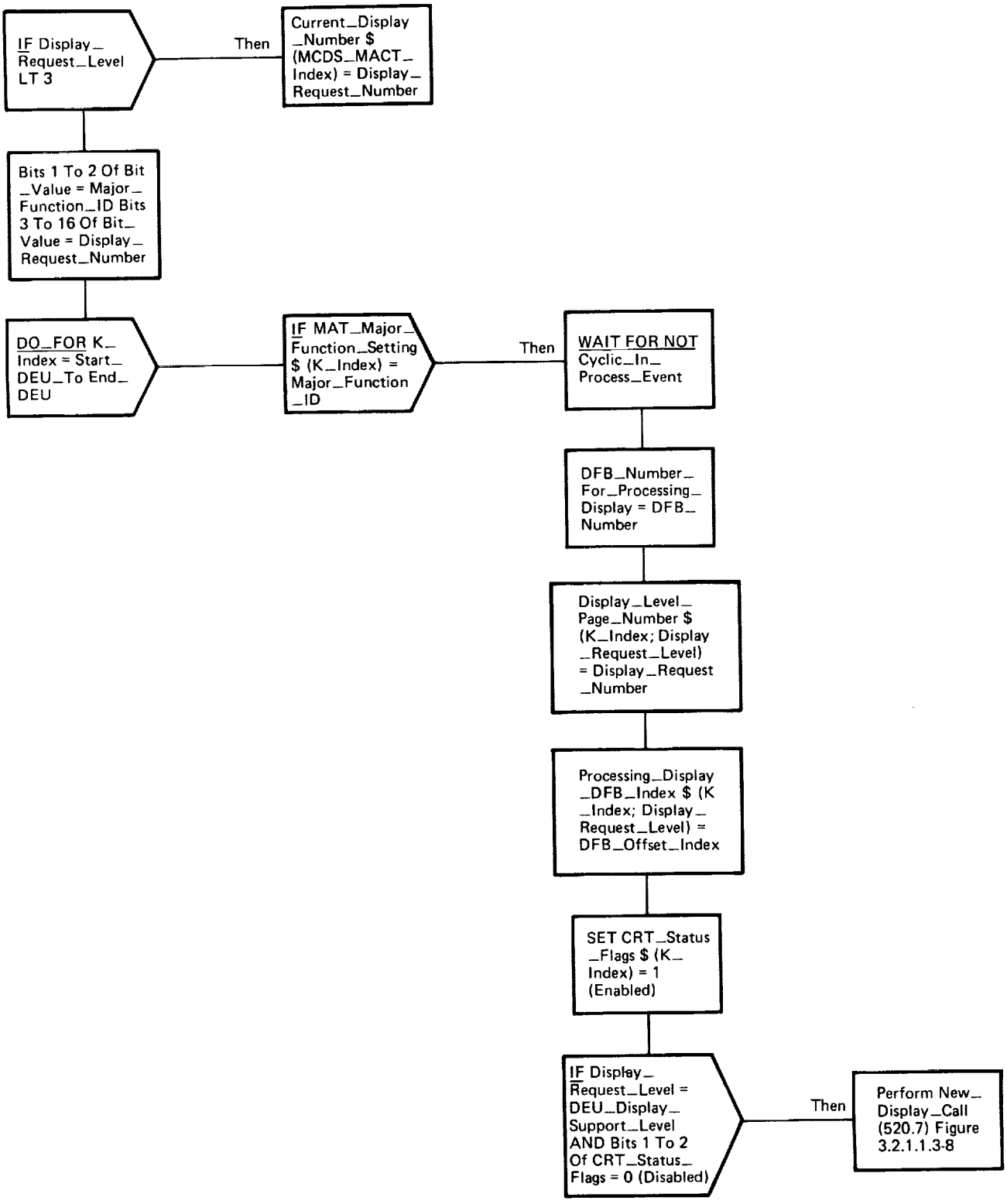


Figure 3.2.1.1.3-7. MCDS_Display_Coordinator Display_Processor (520.6)

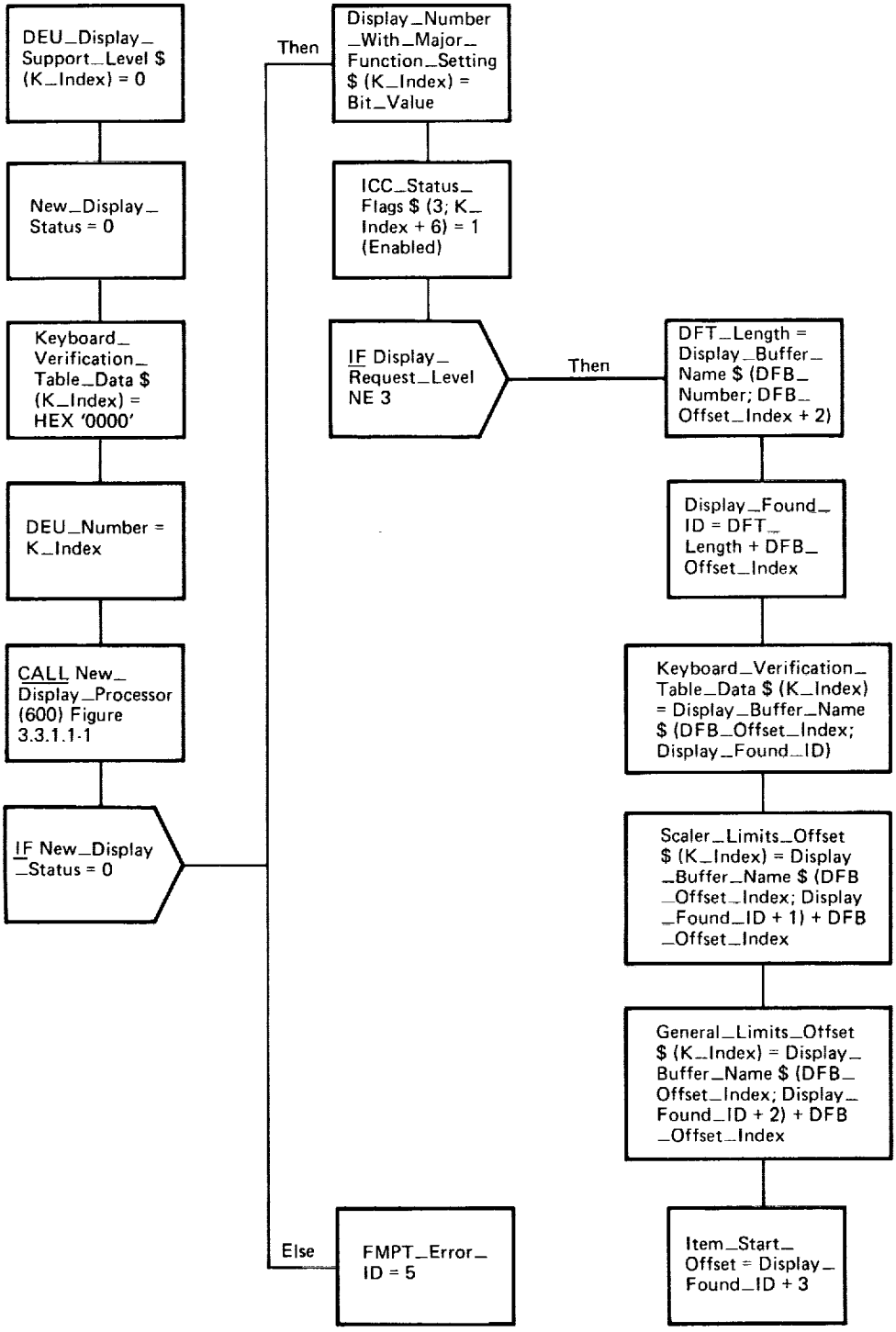


Figure 3.2.1.1.3-8. MCDS_Display_Coordinator New_Display_Call (520.7)

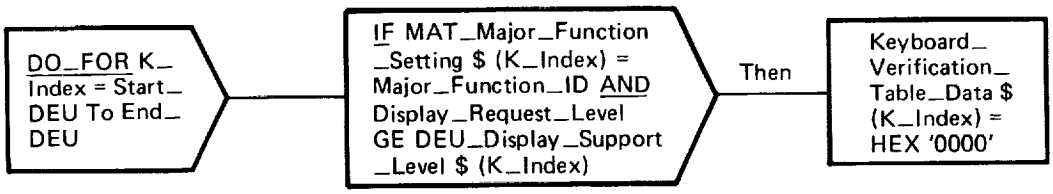


Figure 3.2.1.1.3-9. MCDS_Display_Coordinator KVT_Clear (520.8)

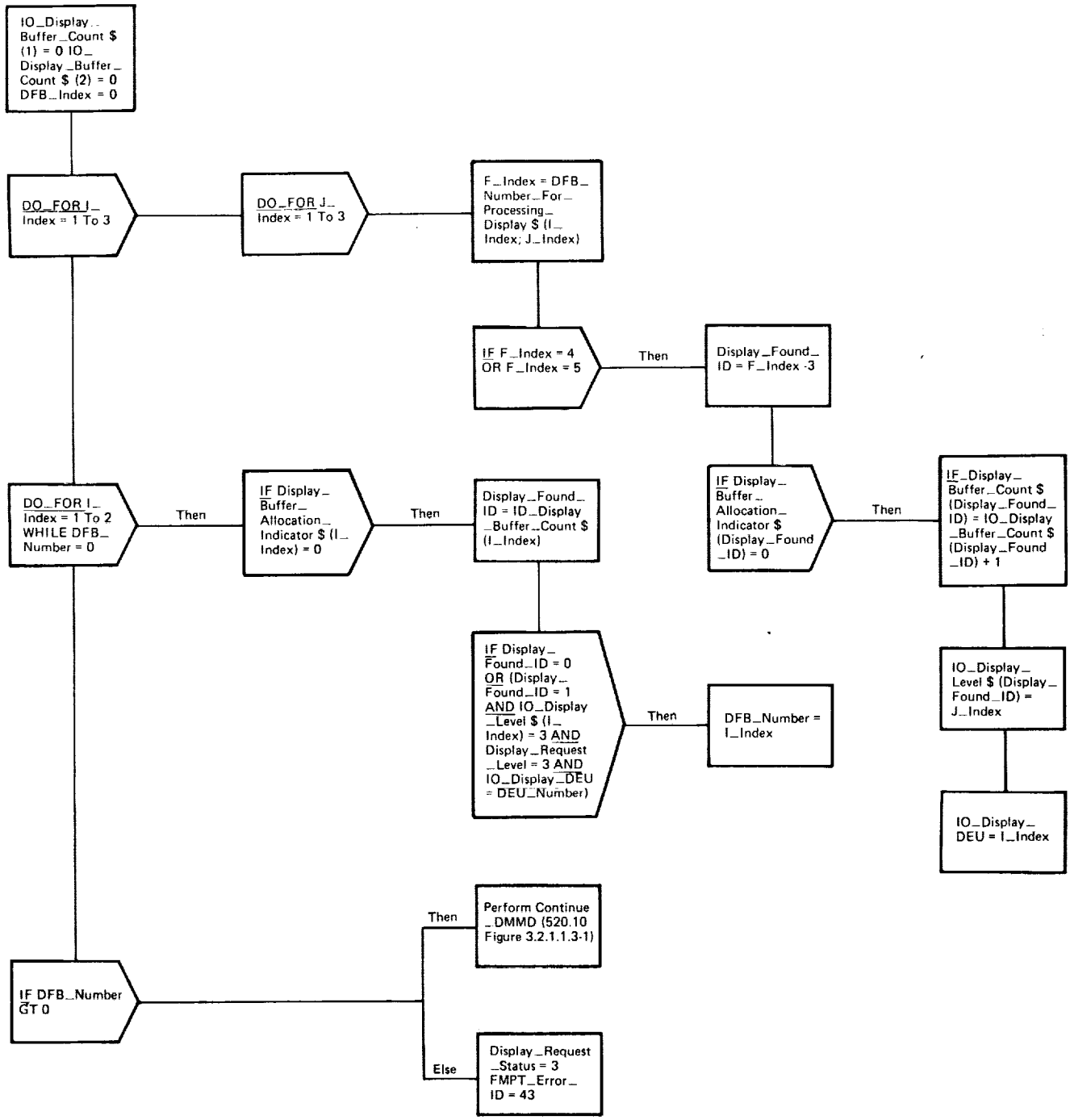


Figure 3.2.1.1.3-10. MCDS_Display_Coordinator DMMD_Display_Processor (520.9)

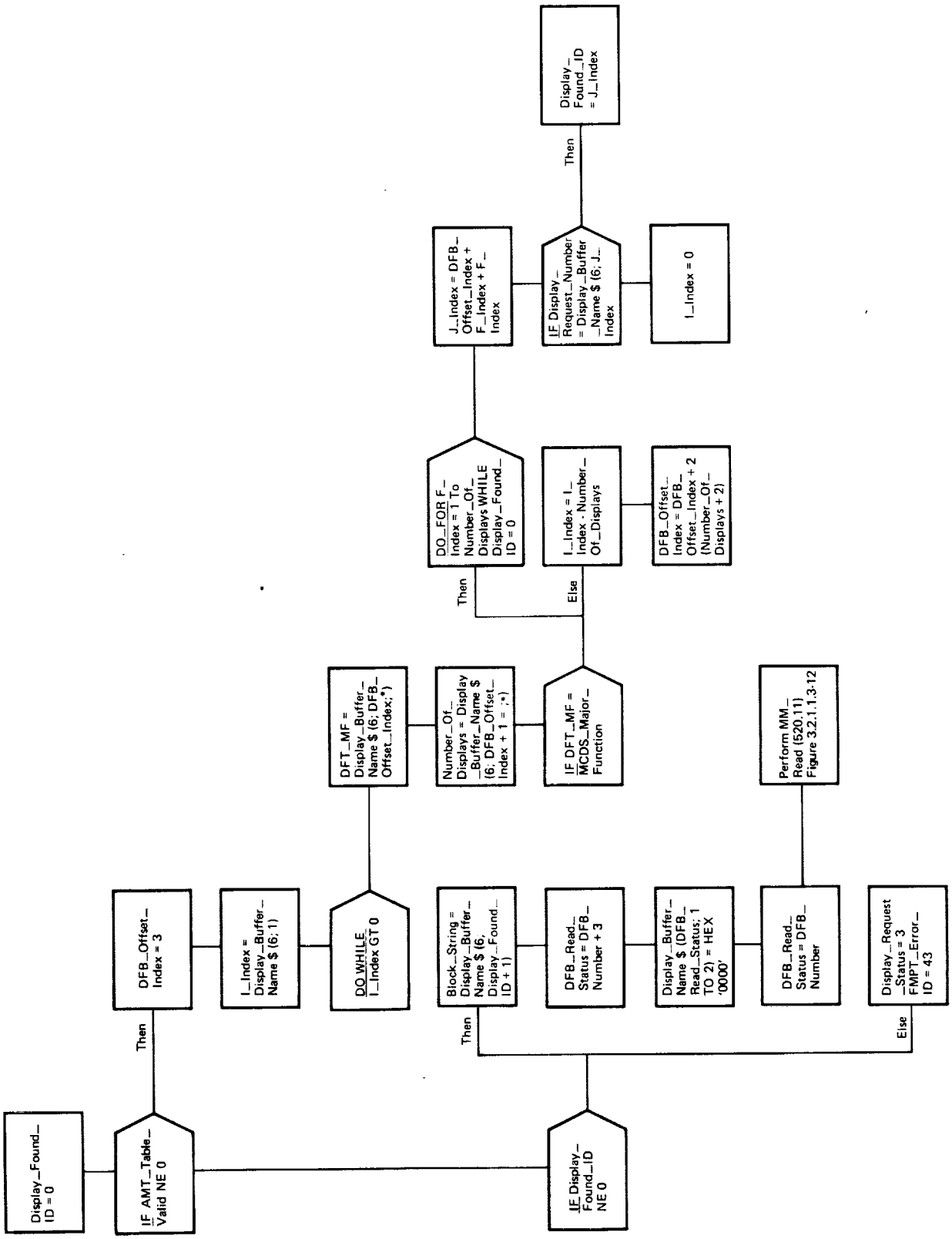


Figure 3.2.1.1.3-11. MCDS_Display_Coordinator Continue_DMMD (520.10)

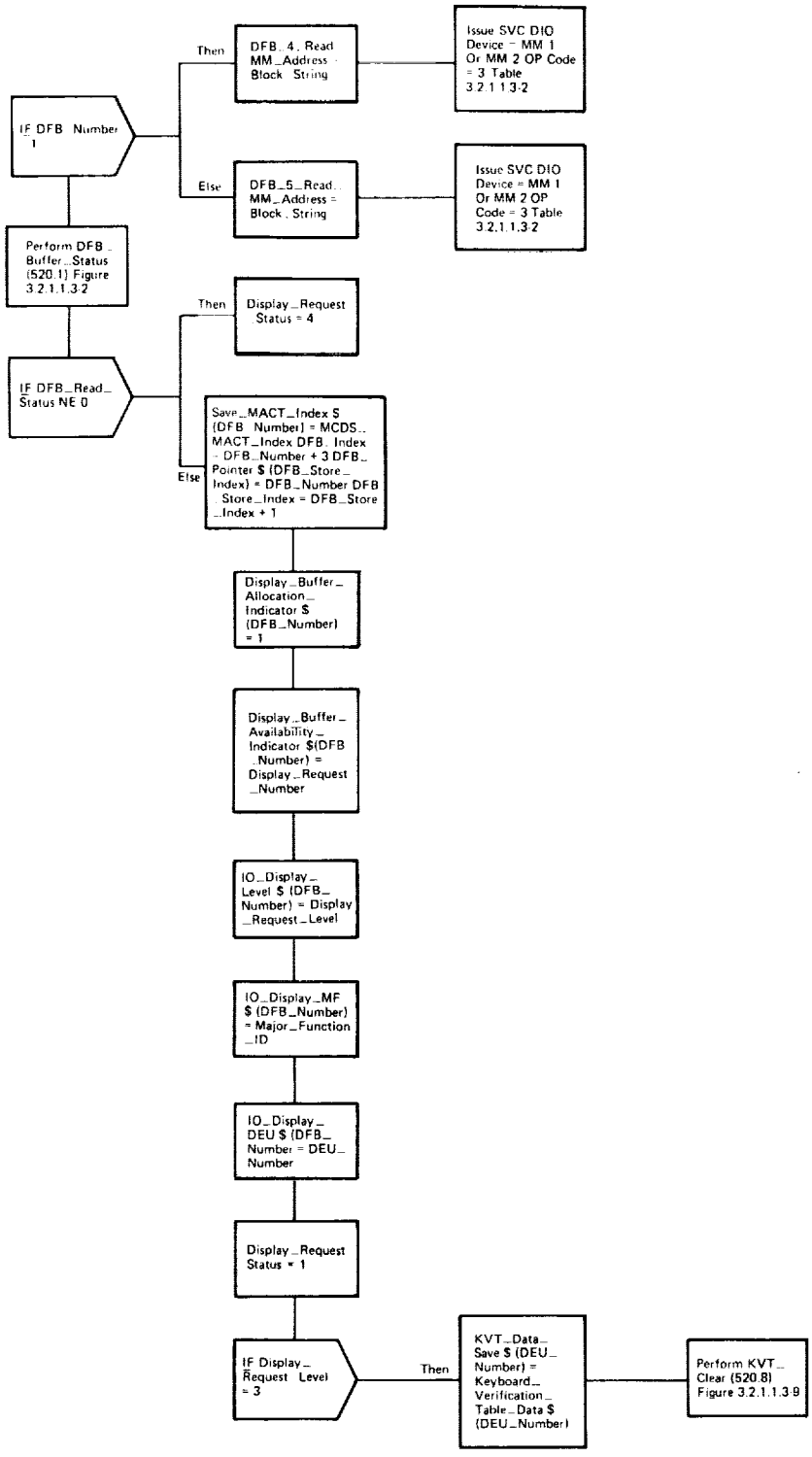


Figure 3.2.1.1.3-12. MCDS_Display_Coordinator MM_Read (520.11)

3.2.2 Application Control

Control Segments are functional components of the software system. All operational sequences and specialist functions shall conform to the constraints of control segments. A grammar (See Appendix H) consisting of HAL macros defines the limits and constraints of control segments.

Control segments (i.e., OPS and SPEC) shall conform to the following operational requirements:

a. Control segments and their inputs via MCDS are bound to the current display.

b. Sequencing may be a function of MCDS input or event.

c. OPS may be entered at multiple entry points (MODEs); SPEC amy be entered at only the topmost point of their logic.

d. The path through an OPS or SPEC is defined by the application.

e. An OPS may not initiate a SPEC.

The structure of the control segment and the service required from User Interface is provided through the grammar macros provided by User Interface and jointly supported by Application Control and the User_Interface_Control_Supervisor. The control segment defines three levels of processing: (1) OPS or SPEC, (2) Mode (OPS only), and (3) Block. Each level is provided with two levels of application processing: (1) Initiation and (2) Cleanup. The block has an additional level required to satisfy the requirements of the display defined for the block level. It is at the block level that the display is requested. The blocks are associated with the display pages in an orderly sequence on a one-for-one basis. The numbering scheme defines the structured format of the blocks.

Application Control provides the control necessary to maintain user control of the application programs. It also provides display and control services to the application through the control segment. The control and services are provided through the grammar macros by (1) Display_Presentation_And_Control, and (2) Application_Moding_And_Sequensor. (See Application Control Hierarchy, Figure 3.2.2-1).

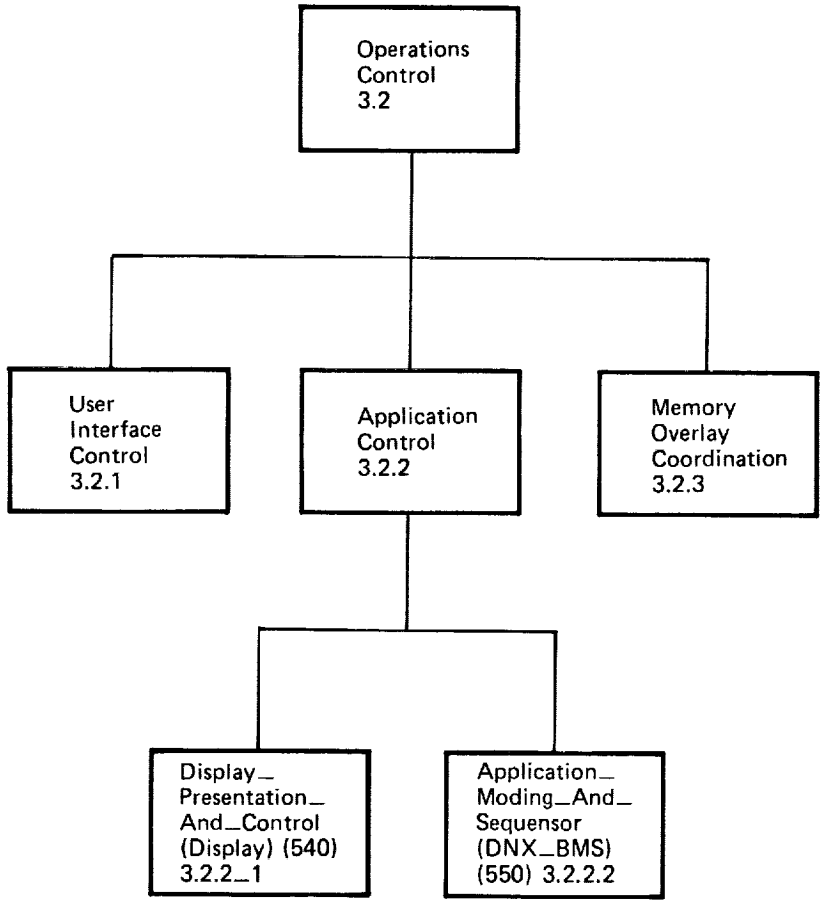


Figure 3.2.2-1. Application Control Hierarchy Diagram

3.2.2.1 Display_Presentation_and_Control (DIS_PLAY) (540)

This process provides the User Interface logic required for an application control segment to initiate a display and wait for a response.

a. Control Interface -

1. CALL DIS_PLAY (DIS_PLAY_NUM, DIS_MACT_INDEX);

2. CALLed by (a) (810) Idle_(Operational_Sequence (ARB_IDLE_OPS)
(b) (910) Read/Write Specialist Function (ASB RD WRT)
(c) (950) TIME_MANAGEMENT_Specialist_Function (ASC_TIME_MGMT)

b. Inputs - See Table 3.2.2.1-1

c. Process Description - Display Presentation_And_Control is entered by the control segments under two conditions: (1) new Display Request, and (2) to continue processing after servicing a user input. The data via the calling sequence are: (1) the Required Display Number and (2) the MACT-Index.

The Application_Service_Event_Flag is reset = 0 (Disabled). If the Control_Segment_First_Display_Entry_Flag is set =2 (Enabled), it is reset =0 (Disabled) and the INIT Block flag is set =1 (Enabled). Further, if the Required_Display_Number is not equal to the Current Display Number, the New_Display_Request Flag is set =1 (Enabled), the New_Display_Number is set equal to the Required_Display_Number and the Application_Service_Event_Flag is set =1 (Enabled). The following processing describes the ELSE processing to the above described THEN processing. If the NEW_Block_Number is greater than or equal to zero, the Control_Segment_Block_Termination_Request is set =1 (Enabled). Further, if the New_Block_Number is greater than zero and the Current_Mode_Number is greater than or equal to zero the New_Mode_Number is computed as the remainder of the Current_Block_Number divided by 1000 which is then divided by 10. (Note: the Current_Block_Number is set by the Grammar macro - CHANGE (See Appendix H - when a new mode or ops request is made). If the New_Mode_Number is not equal to the Current Mode Number, a mode switch is being requested the Control_Segment_Mode_Termination_Request is set =1 (Enables). In either case the New_Block_Number is set to the remainder of the New_Block_Number divided by 10. A return to the calling program is made to finish block change processing.

If the Keyboard_Message_ID is ITEM (See Grammar Macros Appendix H) or ITEM_EXEC then the Item_Processing_Complete_Flag is set =1 (Enabled) and the Application_Service_Request_Event is set =1 (Enabled).

The Keyboard_Message_ID is set to zero. If the Application_Service_Request_Event_Flag is set = 1 (Enabled), then the MACT_Service_Request_Ready_Events is set and the Application_Service_Request_Event is SET. This process then WAIT(s) for Name_Of_Keyboard_Message_Event or OPS_Advance_Event or Mode_Advance_Event or Block_Advance_Event. If the Name_Of_Keyboard_Message_Event is SET, then it is RESET otherwise the following processing is done. The Control_Segment_Block_Termination_Request is set = 1 (Enabled). If the OPS_Advance_Event was SET then the Control_Segment_ Mode_Termination_Request and the Control_Segment_Termination_Request are each set = 1 (Enabled). If the Mode_Advance_Event was SET then the Control_Segment_Mode_Termination_Request is set = 1 (Enabled).

d. Outputs - See Table 3.2.2.1-1

e. Module References -

1. (170) Set_Event_Processor (FPMSET) is called.

2. (171) Reset_Event_Processor (FPMRESET) is called.

f. Module Attributes - External Reentrant Procedure

g. Template References -

1. UI_Section_Of_Common_Compool (CZ1_COMMON)

2. UI_General_Compool (CDM_UI_COMPOLL)

h. Error Handling - NONE

i. Constraints -Calls must be made by Grammar Macros (See Appendix H)

j. Assumptions - NONE

DATA TABLE 3.2.2.1-1

NAME Display_Presentation_And_Control (DIS_PLAY)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	Required_Display_Number	L076	LI	810,910 915,950	540	DIS_DISPLAY_NUM			
2	MACT_Index	L077	LI	810,910 915,950	540	DIS_MACT_INDEX			
3	Application_Service_Event_Flag	L078	L	540	540	DIS_FLAG_APP_SERVICE_EVT			
4	Control_Segment_First_Display_Entry_Flag	J040.10	OZ	540	540				
5	INIT_Block_Flag	J040.12	OZ	540,550	540				
6	New_Display_Request_Flag	J040.03	OZ	505,540	505				
7	New_Mode_Number	J040.42	IOZ	540,550 560	550	CZ1V_D_NEW_MOD			
8	Current_Mode_Number	J040.38	IZ	540,550	550,810, 910,950	CZ1V_D_MOD			
9	New_Block_Number	J040.43	IZ	540,550, 560	550	CZ1V_D_NEW_BLK			
10	Keyboard_Message_ID	J040.49	IZ	505,510 540	540,810, 900,910	CZ1V_D_KEY_ID			
11	Item_Processing_Complete_Flag	J040.07	OZ	505,540	505				
12	New_Display_Number	J040.40	IOZ	540,505	505	CZ1V_D_NEW_DISP			
13	MACT_Service_Request_Ready_Events	J160	OZ	See App. E	505,810	CZ1E_D_MACT_EVTS			
14	Application_Service_Request_Event	B050	OZ	540,820, 870,880	660,810, 820	CDME_APP_SERVICE			
15	Name_Of_Keyboard_Message_Event	J040.45	IOZ	See App. E	540	CZ_E_D_KYBD_MSG_EVT			
16	OPS_Advance_Event	J040.46	IZ	810,900 910,950	540	CZ_E_D_OPS_ADV_EVT			
17	Mode_Advance_Event	J040.47	IZ	810	540	CZ_E_D_MOD_ADV_EVT			
18	Block_Advance_Event	J040.48	IZ	810,900 910,950	540	CZ_E_D_BLK_ADV_EVT			

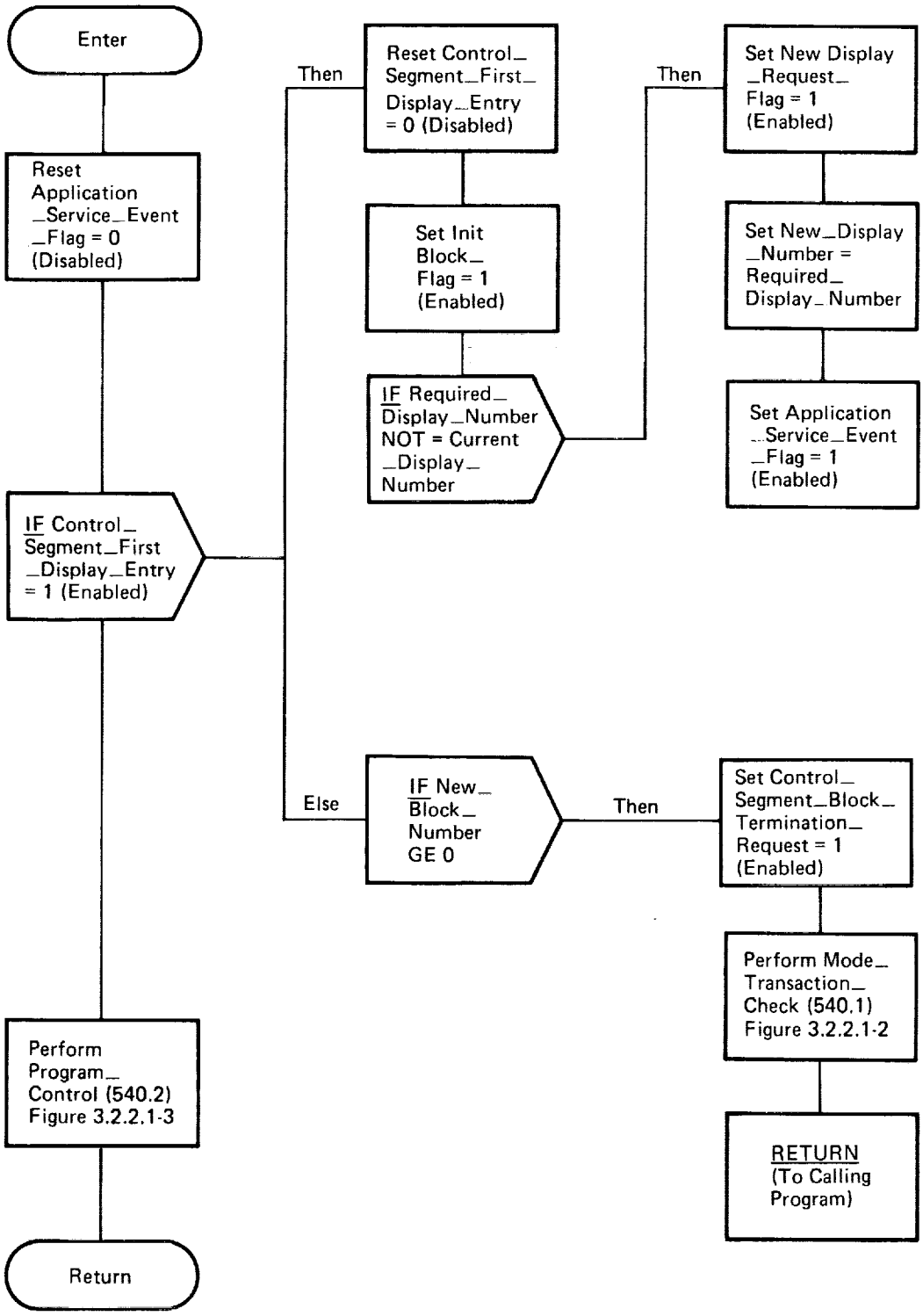


Figure 3.2.2.1–1. Display_Presentation_And_Control (DIS_PLAY)

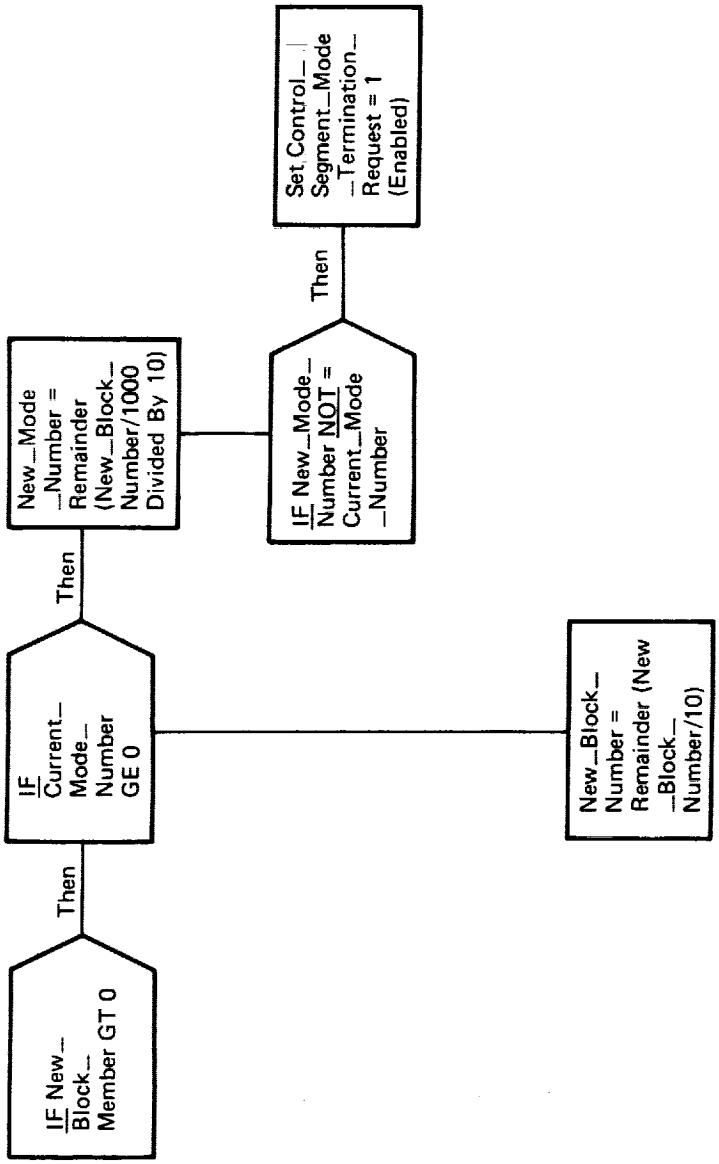


Figure 3.2.2.1–2. Display_Presentation_Control Mode_Transition_Check (540.1)

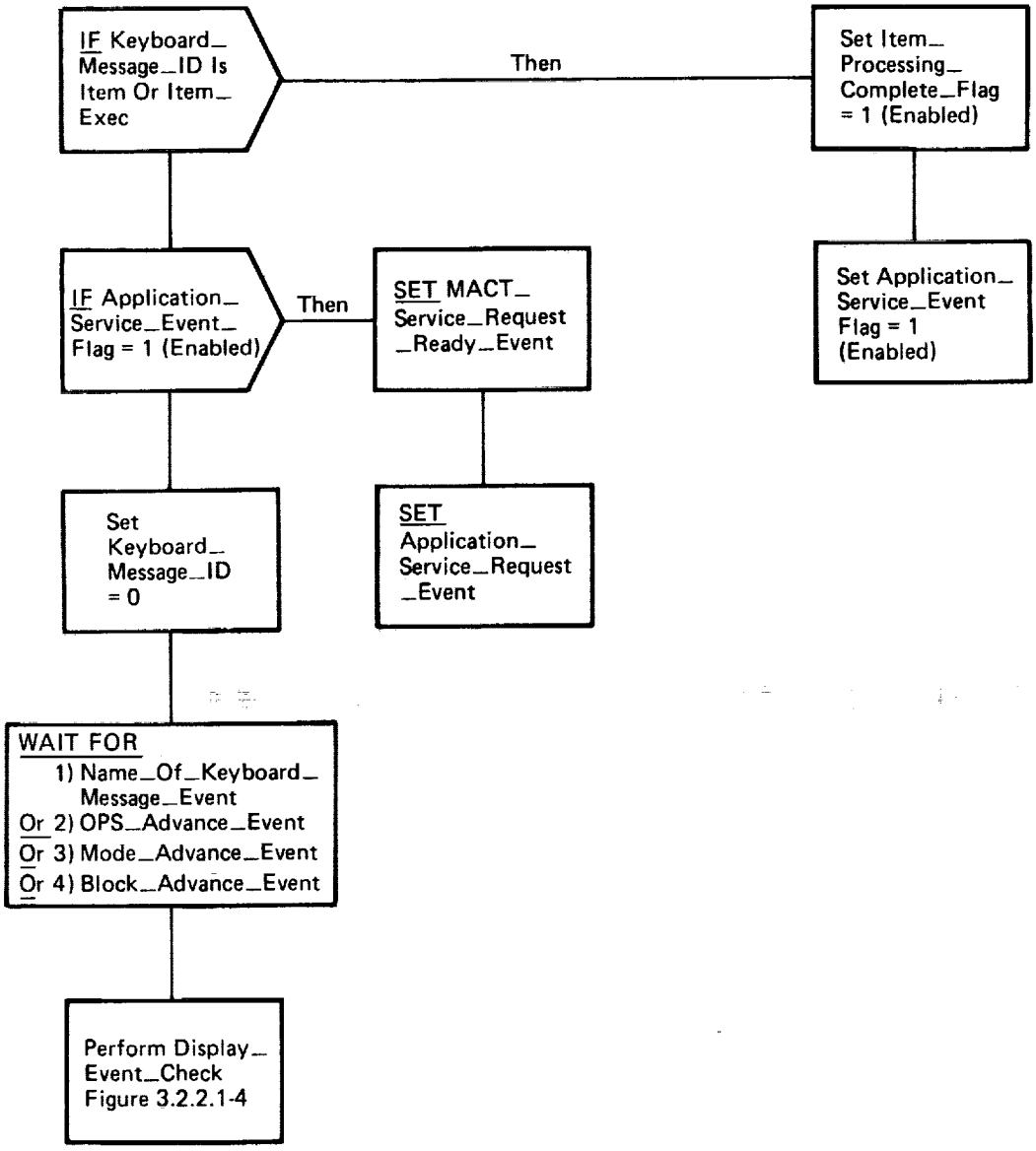


Figure 3.2.2.1–3. Display_Presentation_And_Control Program_Control (540.2)

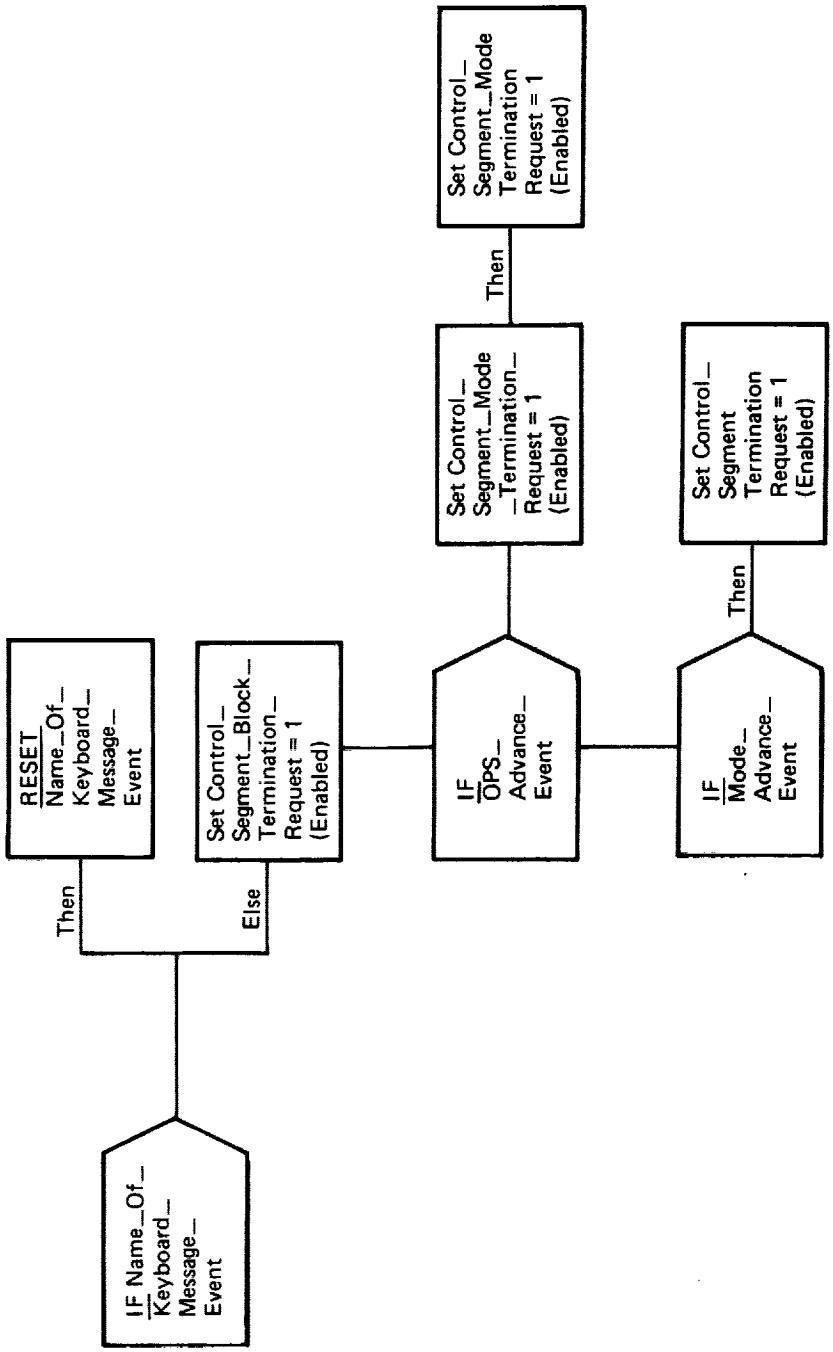


Figure 3.2.2.1–4. Display_Presentation_And_Control Display_Event_Check (540.3)

3.2.2.2 Application Moding and Sequencor (DNX_BMS) (550)

This processor establishes communication between a control segment and the User_Interface_Control_Supervisor and provides the logic to control the flow through the modes and/or blocks of the control segment.

a. Control Interface -

1. CALL DNX_BMS(DNX_LEVEL)
ASSIGN(DNX_MACT_INDEX);

2. CALLed by a) (810) Idle_Operational_Sequence (ARB_IDLE_OPS)
b) (910) Read/Write_Specialist_Function (ASB_RD_WRT)
c) (950) Time_Management_Specialist_Function (ASC_TIME_MGMT)

b. Inputs - See TAble 3.2.2.2-1

c. Process Description - This process is called from control segments by the GRAMMAR macros (See Appendix H). On first entry, the CS_MACT_Index is zero.This indicates an initial SCHEDULE of the CS requiring inialization of the Miscellaneous_Application_Control_Table copy pointed to by the CS_Index_To_MACT. Any further entries will be for the purpose of a CS, Mode, or Block level processing.

For the initial entry, the Current_OPS_Number is set to the New_OPS_Number, the Current_Mode_Number is set to the New_Mode_Number -1 and the Name_Of_Keyboard_Message_Event is set to point to the corresponding element (CS_MACT_INDEX) of the Keyboard_Message_Ready_Event_Array. The following data items are initialized to zero: 1) Current_Display_Number, 2) Current_Block_Number, 3) New_OPS_Number, 4) New_Mode_Number, 5) New_Block_Number, 6) MACT_ Service_Request_Flags, and 7) MACT_Application_Control_Flags. The MACT_Service_Request_Ready_Events is RESET. The MACT_Index_Event is SET to signal Sequence_Request_Processing that the required initialization has been completed.

All further processing is based upon the CS_Process_Level and is processed by case with case 1 (CS_Process_Level = 1) for the OPS/SPEC level, case 2 for the mode level and case 3 for the block level.

The OPS/SPEC (Segment) level is reserved for control segment termination. The Control_Segment_Termination_Flag is SET, the Control_Segment_Termination_Request is reset, the MACT_Service_ Request_Ready_Events is set, and the Application_Service_Request_Event is SET to signal the User_Interface_Control_Supervisor of the control segment termination request.

The Mode level is reserved for Mode_to_Mode transitions which may include OPS_CLEAN_UP (structured to be just another mode - see grammer macros).

The Control_Segment_Mode_Termination_Request is reset, the Current_Block_Number is set to zero, and the INIT_Block_Flag is reset. If the Control_Segment_Termination_Request is set, the Current_Mode_Number is set to 1+ Number_Modes. Otherwise, if the New_Mode_Number is zero, the Current_Mode_Number is incremented by 1. For non zero the Current_Mode_Number is incremented by 1. For non zero New_Mode_Number, the Current_Mode_ Number is set to the New_Mode_Number and the New_Mode_Number is set to zero.

The block level is reserved for block transitions, for mode or SPEC terminations, or for block to block transitions. The Control_Segment_Block_Termination_Request is reset, the Control_Segment_First Display_Entry_Flag is set, and the Name_Of_Keyboard_Message_Event is reset. If the Control_Segment_Mode_Termination_Request is set or the Control_Segment_Termination_Request is set, then the processing required is as follows:

If the Current_Mode_Number is less than zero, this is a SPEC termination with the SPEC_Clean_Up block given by the Termination_Sequence_Index +1. Otherwise it is a mode termination with the pointer (to the mode clean up block) into the OPS_Moding_Data given by Mode_Data_Array_Pointer indexed by the Termination_Sequence_Index plus the Current_Mode_Number minus 1. The Current_Block_Number is given by the integer value of the last 4 bits of that 32 bit array.

For a block change, if the New_Block_Number is less than or equal to zero the Current_Block_Number is incremented by 1; otherwise the Current_Block_Number is set equal to the New_Block_Number. In either case, the New_Block_Number is set to -1.

d. Outputs - See TAble 3.2.2.2-1

e. Module References -

1. (171) Reset_Event_Processor (FPMRESET) is called.

2. (170) Set_Event_Processor (FPMSET) is called.

f. Module Attributes - External Reentrant Procedure.

g. Template References -

1. UI_Section_Of_Common_Compool (CZ1_COMMON)

2. UI_General_Compool (CDM_UI_COMPOOL)

3. Applications_Moding_Table (CDA_P02)

h. Error Handling - NONE

i. Constraints - CALLs must be made by the Grammar Macros (See Appendix H)

j. Assumptions - NONE

h. Detailed Implementation - NONE

DATA TABLE 3.2.2.2-1

NAME Application_Moding_And_Sequencor (DN _ BMS)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	CS_Process_Level	L079	IL	810,910, 950	550	DNX_LEVEL			
2	CS_MACT_Index	L080	IOL	550,810, 910,950	See App. E	DNX_MACT_INDEX			
3	CONTROL_SEGMENT_Index_To_MACT	B030	Z,I	560	550,560	CDMV_MACT_INDEX			
4	Miscellaneous_Application_Control_Table	J040	Z,I,O	505, 510 540, 550	See APP E.	CZ1V_D_MACT			
5	Current_Display_Number	J040.36	Z,O	505,540 550, 560	505,560	CZ1V_D_DISP			
6	Current_OPS_Number	J040.37	Z,O	550,560	440,550	CZ1V_D_OPS			
7	Current_Mode_Number	J040.38	Z,O	540,550	550,810, 910,950	CZ1V_D_MOD			
8	Current_Block_Number	J040.39	Z,O	550	810, 910 950	CZ1V_D_BLK			
9	New_OPS_Number	J040.41	Z,I,O	550,560	550	CZ1V_D_NEW_OPS			
10	New_Mode_Number	J040.42	Z,I,O	540,550, 560	550	CZ1V_D_NEW_MOD			
11	New_Block_Number	J040.43	Z,I.O	540,550 560	550	CZ1V_D_NEW_BLK			
12	Name_Of_Keyboard_Message_Event	J040.45	Z,O	See App. E	540	CZ1V_D_KYBD_MSG_EVT			
13	MACT_Service_Request_Ready_Event	J160	Z,O	See App. E	505,810	CZ1V_D_MACT_EVTS			
14	MACT_Index_Event	B040	Z,O	550,560	560	CDME_MACT_INDEX			
15	Control_Segment_Termination_Flag	J040.09	Z,O	505,550	505				
16	Control_Segment_Termination_Request	J040.21	Z,O	505,550, 560	550				
17	Control_Segment_Mode_Termination_Request	J040.20	Z,O	505,550, 560	550				
18	Control_Segment_Block_Termination_Request	J040.19	Z,O	505,550, 560	550				
19	MACT_Service_Request_Flags	J040.01	Z,O	See App. E	505,540	CZ1B_D_FLAGS			
20	MACT_Application_Control_Flags	J040.18	Z,O	505,550, 560	550	CZ1B_D_FLAG2			

DATA TABLE 3.2.2.2-1 (cont'd)

NAME Application_Moding_And_Sequencor (DNX_BMS)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
21	INIT_Block_Flag	J040.12	OZ	540,550	540				
22	Termination_Sequence_Index	J040.56	ZI	560	540,550, 560	CZ1V_D_SEQUENCE_INFO			
23	Mode_Data_Array_Pointer	A080.30	ZI	DFG	550	CDAV_OP_MODXFR			
24	OPS_Moding_Data	A120	ZI	DFG	550	CDAV_MOD_VAL_ARR			
25	Keyboard_Message_Ready_Event_Array	J064	ZIO	550	540,550	CZ1E_D_KYBD_MSG_ARRAY			
26	Number_Modes	A080.20	Z,I	DFG	550,560	CDAV_OP_NMODES			
27	Application_Service_Request_Event	B050	Z,O	See App. E	660,810, 810	CDME_APP_SERVICE			
28	Control_Segment_Termination_Flag	J040.09	ZIO	505,550	505				
29	Control_Segment_First_Display_Entry_Flag	J040.10	ZIO	540	540				
30	Block_Index	L081	L	550	550	DNX_MVA_I			

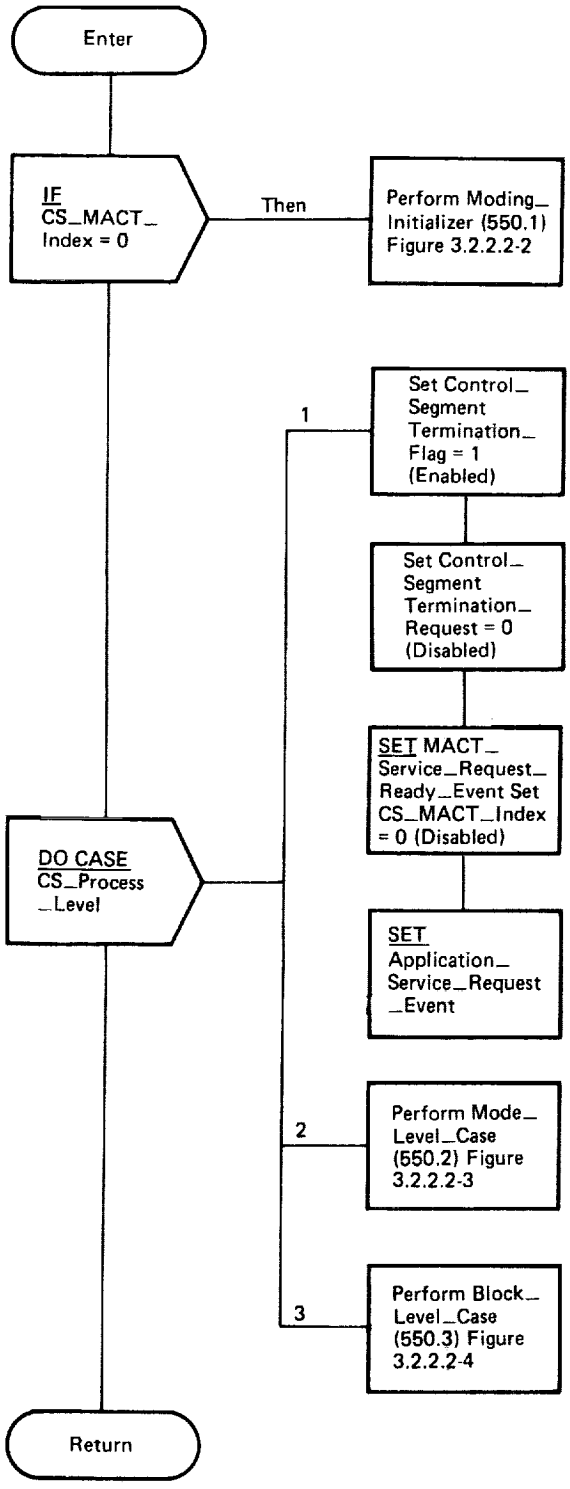


Figure 3.2.2.2-1. Application_Moding_And_Sequencor (DNX_BMS)

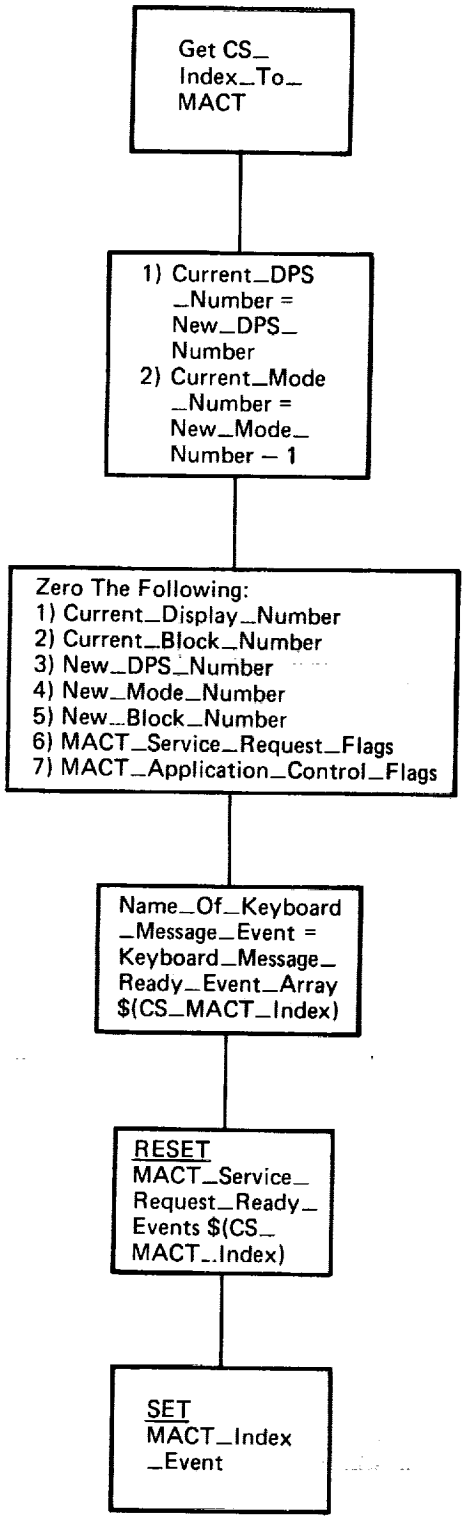


Figure 3.2.2.2-2. Application_Moding_And_Sequencor Moding_Initializer (550.1)

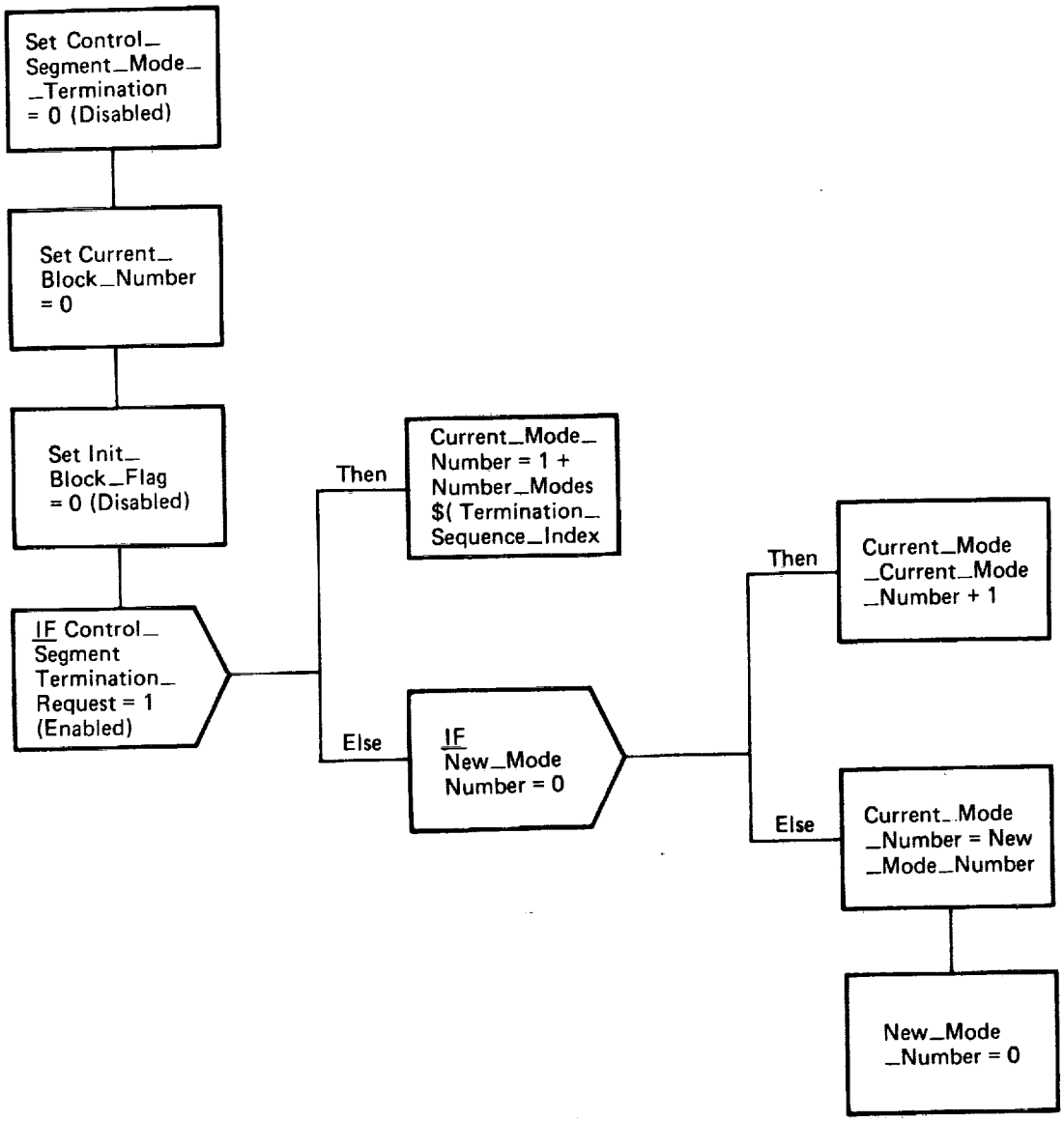


Figure 3.2.2.2-3. Application_Moding_Sequencor Mode_Level_Case (550.2)

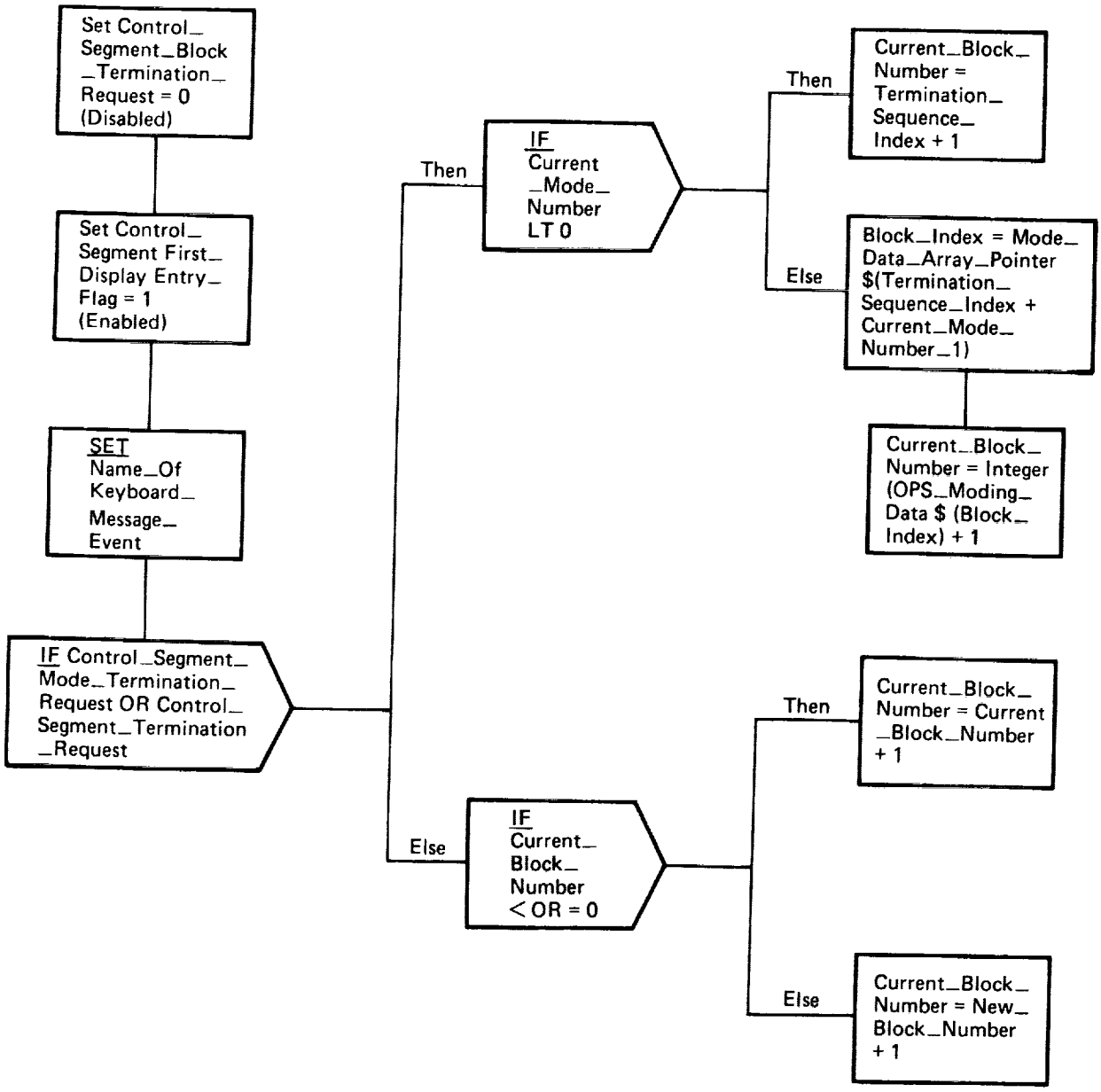


Figure 3.2.2.2-4. Application_Moding_Sequencor Block_Level_Case (550.3)

3.2.3 Memory Overlay Coordination

The FCOS has the capability to perform overlays from mass memory. Program overlays are limited to points of operational sequence transition. OPS transitions involve the orderly termination of current OPS processes, memory overlay, and OPS initiation. Memory overlay may involve one or more GPC's in simplex or redundant sets requiring system reconfiguration of the GPC's. The transitions are initiated by user input which must be verified with current functions and with current and requested GPC status. Memory Overlay Coordination must verify and coorindate all processing associated with the initiation of the requested memory ocerlay. Memory Overlay requirements are satisfied by Sequence_Request_Processing and GPC_Reconfiguration.

Sequence_Request_Processing must verify the OPS transition request, provide for the overlay termination of current OPS processes and the initiation of the requested OPS process.

GPC_Reconfiguration must form the GPC reconfiguration and invoke the FCOS Program Overlay service to perform the program overlay.

3.2.3.1 Sequence_Request_Processor (DMC_SEQ_REQ_PROC) (560)

Sequence_Request_Processor provides the logic necessary to validate OPS requests, perform OPS to OPS transitions, perform mode to mode transitions, validate SPEC requests, perform SPEC scheduling, and provide reconfiguration processing pertaining to the formation of redundant GPC operation.

a. Control Interface -

1. CALL DMC_SEQ_REQ_PROC (Action_Type)
2. CALLed by (505) MCDS_Functions_Processor (DMC_FUNCTIONS)

b. Inputs - See Table 3.2.3.1

c. Process Description - This procedure will respond to four (4) types of requests as specified by the caller. The caller identifies the type of logic and processing required via the passed parameter, Action_Type.

The four types of requests which will be honored and the value of Action_Type associated with each are:

1. Processing required as the result of the termination of an OPS or SPEC control segment. Action_Type = 0.

2. Processing required to bring all secondary GPCs together prior to the formation of a redundant set. Action_Type = 2.

3. Processing required after reconfiguration has been completed. Action_Type = 1.

4. Processing required as the result of an OPS or SPEC keyboard input request, Action_Type = 17 for an OPS request and Action_Type = 18 for a SPEC request.

Sequence_Request_Processor will perform the requested processing through the use of in-line logic and seven (7) procedures internal to Sequence_Request_Processor.

Since this procedure performs a variety of tests and executes a complex set of logic paths a functional view of the processing performed will be described here and a detailed description of all Perform Blocks and internal procedurer is provided later in this section.

When Action_Type = 0, the Termination_Processor logic is envoked to update internal parameters associated with the major function of the terminating control segment and provided any control segment scheduling required as the result of the control segment termination.

When Action_Type = 2, the Secondary Reconfiguration Processor logic is envoked to provide the processing necessary to cancel all System Specialist functions active in all major functions.

When Action_Type = 1, the Reconfiguration_Complete_Processor logic is envoked to determine status of the requested reconfiguration and to develop a request for the requested OPS requireing reconfiguration when no errors occured during reconfiguration,or to develop a request for SPEC 00.

When Action_Tupe = 17 or 18, the Keyboard_Processor logic is envoked to determine the type of request (OPS or SPEC) and calculate the requested OPS and Mode number or the requested SPEC number.

The following logic will be executed when Action_Type = 1 or 17, or 18.

When the request is an OPS request (as the result of Reconfiguration_Complete_Processor or Keyboard_Processor) the OPS number being requested is tested. When the requested OPS number = 0, the OPS_Zero_Processor logic is envoked to provide the processing for all OPS0-00 requests, otherwise the OPS_Non_Zero_Processor logic is envoked to determine OPS and mode legality, provide the processing necessary to determine whether the request is a mode_to_mode transition, or an OPS transition with or without reconfiguration.

When the request is a SPEC request (as the result of Reconfiguration_Complete_Processor or Keyboard_Processor), the SPEC_Validity_Processor logic is envoked to determine if the requested SPEC is a valid SPEC (legal for current OPS, not active elsewhere, and two SPECS are not already active for the major function associated with the SPEC request). When the SPEC is legal, the SPEC_Processor logic is envoked to provide the processing necessary to schedule the requested SPEC or to terminate an active SPEC, and upon receiving its termination indication thru a request with Action_Type = 0, to schedule the requested SPEC.

The control flow for Sequence_Request_Processor, its "Perform block", and its internal procedures is described in Figures 3.2.3.1-1 thru 3.2.3.1-22.

All indexing described in the control flow is specified within brackets [] and comments are placed with parenthesis ()

The following are descriptions of all Perform block logic and all procedures internal to Sequest_Request_Processor.

Termination_Processor

When the passed parameter Action_Type equals zero (0), the caller is requesting processing as the result of an OPS or SPEC control segment terminating.

This processing consists of locating and removing the terminated control segments indication in the Control_Segments_Request_Block, and reducing the count of active control segments by one (1).

When Process_Type = 1 (OPS transition) and Wait_Count = 0 (no active control segments), the Control_Segment_Scheduler is called to schedule the new OPS and if Reconfiguration_In_Progress is not equal zero (Reconfiguration required), the Sequence_Reconfiguration_Processor is called to determine if reconfiguration can begin.

When Process_Type is not = 1 (not OPS Transition) and the control segment that terminated was an OPS then this major function will go to OPS0-00. This is done by calling the Control_Segment_Terminator to terminate any active SPECS and load the Control_Segment_Request_Block with a request for OPS0 mode 1. When all SPECS have terminated, the Control_Segment_Scheduler will be called to schedule OPS0-00.

When the Process_Type is not = 1 (not OPS transition) and the control segment that terminated was a SPEC, then either a SPEC will be scheduled or the OPS page will be placed on the DEU associated with the terminated SPEC. The Next_SPEC_Number is loaded and if it is not = 500 then the SPEC terminated as the result of a request for new SPEC to be scheduled in its place. The Control_Segment_Scheduler is called to schedule the waiting SPEC after loading the Sequence_MACT_Index from the Next_SPEC_MACT and the Sequence_AMT_Index from the Next_SPEC_AMT tables and setting Sequence_Request_Mode to minus 1 (to indicate a SPEC schedule).

When the Next_SPEC_Number is 500, the SPEC did not terminate as the result of another SPEC waiting to be scheduled. The MCDS_Display_Coordinator is called to process a display request for the active OPS display to be placed on the DEU previously controlling the terminated SPEC.

The control flow for Termination_Processor is described in Figure 3.2.3.1-2.

Secondary_Reconfiguration_Processor

When the passed parameter Action_Type equals two (2), the caller is requesting a secondary GPC to perform preparitory functions related to formation of a redundant set and to specify through the ICC when these functions have been completed.

System Specialist 02 is removed from the list of legal System Specialists so that if requested it can not be scheduled.

The OPS_Zero_Page is set to zero (0) if the Reconfiguration OPS value is 2, otherwise it is set to one (1). (The OPS_Zero_Page = 0 indicates that the normal OPS0-00 display is no longer valid and it will not be used).

Reconfiguration_In_Progress is set to minus one hundred (-100) to indicate secondary reconfiguration and Reconfiguration_IN-Initialization is set to two (2) to indicate reconfiguration has begun in this secondary GPC.

The Sequence_Reconfiguration_Pre-Processor internal procedure is called to switch the OPS zero page from Display Format Buffer (DFB) three to a blank page in DFB one since DFB three will be part of the overlay. This procedure will also cause all System Specialists that were active to be cancelled.

The Sequence_Reconfiguration_Processor internal procedure is then called to determine if all System Specialists in all major functions have been cancelled and if so to send an ICC message indicating this condition.

The control flow for Secondary_Reconfiguration_Processor is described in Figure 3.2.3.1-3.

Reconfiguration_Complete_Processor

When the passed parameter Action_Type equals one (1), the caller is requesting processing as the result of the completion of reconfiguration.

Reconfiguration_In_Process and Reconfiguration_In_Initialization are set to zero (0) to specify reconfiguration completion.

The status of the reconfiguration is tested and, if overlay was performed, the status of the Applications_Moding_Table (AMT) is determined (if an error occured with overlay the AMT is no longer valid, otherwise it is valid).

The Buffer_Address_Mover internal procedure is called to determine the proper addresses to use for Display Format Buffer (DFB) 1 thru 5 and the Mass_Memory_Directory address.

The status of the reconfiguration is tested and if an error exists (either overlay performed or not), a request for SPEC00 is developed and Reconfiguration_DEU is restored to DEU_Number and Reconfiguration_Major_Function is restored to MCDS_Major_Function.

When no error exists, the real OPS0-00 page is restored if it is valid, and Reconfiguration_OPS_Request is restored to Sequence_Request_OPS/SPEC, Reconfiguration_Mode_Request is restored to Sequence_Request_Mode, Reconfiguration_Major_Function is restored to MCDS_Major_Function and an OPS request is developed.

The control flow for Reconfiguration_Complete_Processor is described in Figure 3.2.3.1-4.

OPS_Zero_Processor

Processing required for an OPS0-00 request will be to determine if its a GPC OPS Zero request or an OPS0-00 request for a single major function only.

If GPC_OPS_Zero_Keystrokes is greater than five (5), the request is for GPC OPS zero processing. If Initialization_Flag = 0 then initialization is required, and the internal procedure Buffer_Address_Mover is called to establish the Display_Format_Buffer (DFB) and Mass_Memory_Directory addresses, followed by a call to the Control_Segment_Scheduler to schedule the Idle_Operational_Sequence control segment (810). When Initialization is not required then all major function are requested to transition to OPS0-00.

When the request is not for GPC OPS Zero processing, the Mode_to_Mode-Processor is called if the current active OPS is currently OPS0-00, otherwise a call to the Control_Segment_Terminator is made to request a transition to OPS0-00.

The control flow for OPS_Zero_Processor is described in Figure 3.2.3.1-6.

OPS_Non_Zero_Processor

The OPS_Non_Zero_Processor will determine if the request is a mode to mode transition, an OPS transition with reconfiguration required, or an OPS transition without reconfiguration required.

When Sequence_Current_OPS equals Sequence_Request_OPS/SPEC the request is for a mode to mode transition. When Sequence_Request_Mode (requested mode) is legal the Mode_To_Mode_Processor is called to provide the processing required to transition to the requested mode within the current OPS. The determination of mode legality is made by extracting the Number_modes value for the current active OPS from the Applications_Moding_Table (AMT) and that value must be greater than or equal to the requested mode.

When the request is not for a mode to mode transition the AMT/GRT_Search_Processor is called to determine legality of the request, and to determine if reconfiguration is required. AMT/GRT_Search_Processor will return status of the request in the variable Sequence_Process_Type.

When Sequence_Process_Type = 1, the request is legal and it is an OPS transition without reconfiguration. This is accomplished by calling the Control_Segemnt_Terminator to request all active control segments to terminate and if Wait_Count = 0, the requested OPS is scheduled via a call to the Control_Segment_Scheduler. Note that Wait_Count will be zero when the active control segments were SPEC 00 and/or OPS0-00 only.

When Sequence_Process_Type = 7, the request is legal and its an OPS transition requiring reconfiguration. This processing includes saving the requested OPS parameters, calling the Reconfiguration_Pre-Processor internal procedure to provide access to the "blank" OPS-00 page and request transition to OPS0-00 for all major functions, and calling the Reconfiguration_Processor to determine if reconfiguration can begin.

The control flow for OPS-Non_Zero_Processor is described in Figure 3.2.3.1-7.

SPEC_Validity_Processor

The SPEC_Validity_Processor will determine if two-SPECS are currently active on DEU's other than the one requesting this SPEC, determine if the requested SPEC is legal for the OPS that is active, determine if the requested SPEC is currently active, and when all checks have been completed, the variable Sequence_Process_Type will reflect the status of the request.

The two SPEC active validity check is performed in two parts.

1. Determining if a SPEC is currently active on the same DEU as the SPEC being requested. If true, then no further test need be made and the current active SPEC on that DEU will be made to terminate prior to scheduling the requested SPEC provided the requested SPEC is not active.

2. The DEU_Request_MACT_Index array is searched for active SPECS within the requested major function. (This array is updated upon scheduling a SPEC and reflects the current status of all active SPECS). When the two entries associated with this major function are non zero it indicates that two SPECS are currently active and neither are active on the DEU of the requested SPEC, therefore an error condition exists and error message DU026 is annunciated. When at least one of the entries tested in the DEU_Request_MACT_Index array associated with requesting major function is zero then the requested SPEC may be scheduled immediately provided the requested SPEC is not active.

   (The array entries checked are 3, 4 for major function GN&C and 6, 7 for major function SM).

The legality of the SPEC request is determined by matching the requested SPEC with all System_SPEC_IDS and if its not a System SPEC envoking the SPEC_Search_OPS_Type logic to determine if the requested SPEC is legal for the current OPS.

When it has been determined that the requested SPEC is a System SPEC or a SPEC allowed in the current OPS the SPEC_Active_Search logic is envoked to determine if the requested SPEC is currently active.

The control flow for SPEC_Validity_Processor is described in Figure 3.2.3.1-8.

SPEC_Search_OPS_Type

The SPEC_Search_OPS_Type processing consists of locating the SPECs associated with the active OPS and matching these against the Requested SPEC to determine if the requested SPEC is legal for the current OPS.

The value Active_OPS_AMT_Index is an index to all data in the Applications_Moding_Table (AMT) associted with the active OPS. This value is used to calculate the beginning and ending loop variables to use in matching the requested SPEC against all SPECS legal for the active OPS.

The control flow for SPEC_Search_OPS_Type is described in Figure 3.2.3.1-9.

SPEC_Active_Search

The SPEC_Active_Search processor will determine if the requested SPEC is active within the requested major function when the requested SPEC is not a system SPEC and will determine if the requested SPEC is active on any major function when the requested SPEC is a system SPEC.

The search for SPEC activity is made by looping thru the Active_Control_Segment_MACT_Index array values associated with SPECs (2 thru 4). When a non zero value is found, the value is used as an index into the Miscellaneous_Application_Control_Table (MACT) and the requested SPEC is matched against the value Current_OPS (which will contain the SPEC number). When a match occurs Sequence_AMT_Index is set to minus one to indicate the SPEC was active, otherwise, Sequence_AMT_Index is not updated to indicate the SPEC was not found to be active.

The control flow for SPEC_Active_Search is described in Figure 3.2.3.1-10.

SPEC_Processor

The SPEC_Processor will cause the requested SPEC to be scheduled immediately or will cause the active SPEC on the DEU thats requesting a new SPEC to terminate prior to scheduling the requested SPEC.

The active SPEC, except SPEC 00, will be required to terminate prior to scheduling the requested SPEC when Sequence_MACT_Index is less than 100. The requested SPEC id and the calculated values, Sequence_MACT_Index, and Sequence_AMT_Index are saved to be used as scheduling information when the active SPEC terminates. The MACT_Application_Control_Flags are set to reflect a SPEC termination and the Name_Of_KYBD_MSG_Event is set to request the termination to begin.

When the requested SPEC can be scheduled immediately, the Control_Segment_Scheduler is called to provide the processing associated with the Schedule.

The control flow for SPEC_Processor is described in Figure 3.2.3.1-11

Control_Segment_Scheduler

The Control_Segment_Scheduler internal procedure is called to perform all processing associated with the scheduling of OPS and SPEC control segments.

This processing includes updating the Control_Segment_Request_Block structure to reflect the new control segment being scheduled, loading the scheduling parameters (program name and priority), setting the DEU_Display_Support_Level to zero (0) so that the control segment display

request will be honored, updating the Miscellaneous_Application_Control_Table to provide information needed by the control segment and UI, issuing the schedule statement to cause the control segment to become active and issuing a Wait statement to determine when the control segment has gained control.

The Idle_Operational_Sequence (810) control segment (OPS0-00) will cause the issuance of a schedule statement only if Initialization_Flag = 0. All other requests for OPS0-00 will cause a display request for display 1 (Idle_Operational_Sequence display page). All requests for OPS0-00 are controlled from Miscellaneous_Application_Control_Table copy one (1).

The Idle_Specialist_Function (900) control segment (SPEC 00) will never result in the issuance of a schedule statement since it is the same control segment as Idle_Operational_Sequence. All requests for SPEC 00 result in a display request for display 1, level 2 (SPEC level display request) and are controlled from Miscellaneous_Application_Control_Table copy one (1).

The Active_Control_Segment_MACT_Index array which is part of the Control_Segment_Request_Block structure is the mechanism used to determine what control segments are active for specific major function and is updated as follows:

Active_Control_Segment_MACT_Index [1] will contain the MACT copy controlling the OPS control segment, [2], [3], and [4] will contain the MACT copy controlling a SPEC function active on DEU 1, 2, and 3 respectively a zero (0) when no SPEC is active on a DEU.

When Sequence_Request_Mode is minus one it indicates that it is a SPEC that is to be scheduled and the SPEC_Control_Processor logic is envoked to perform processing unique to scheduling a SPEC, otherwise, the OPS_Control_Processor logic is envoked to perform processing unique to scheduling an OPS.

The DEU_Support_Level and the SPEC and DISP Display_Level_Page_Numbers are set to zero for the DEU(s) supporting the same major function as the OPS or SPEC being scheduled (one DEU is tested in the case of a SPEC being scheduled and all DEU(s) are tested in the case of an OPS being scheduled). This processing is required to allow the OPS or SPEC displays request to cause a DEU background update.

If Display_Request_Level is non zero a call to MCDS_Display_Processor is made to request the OPS0-00, or SPEC00 page.

When the Display_Request_Level is zero the proper Miscellaneous_Application_Table will be loaded with parameters required by UI and the control segment. These parameters are major function, DEU number, Termination information, the OPS/SPEC number, and the mode number (-1 in case of a SPEC). The MACT_Index_Event is reset, SCHEDULE of the OPS/SPEC is issued and a wait is issued to determine when the control segment has gained control.

The control flow for Control_Segment_Scheduler is described in Figure 3.2.3.1-12.

SPEC_Control Processor

The SPEC_Control_Processor will update the DEU_Request_MACT_Index array to reflect the DEU controlling the SPEC to be scheduled, determine which SPEC to schedule, collect the schedule parameters, and update the Active_Control_Segment_MACT_Index array to reflect the miscellaneous_Application_Control_Table (MACT) copy controlling the SPEC to be scheduled.

The DEU_Request_MACT_Index array is a seven entry array which is parallel to the seven copies of the MACT. This array will contain zero for those MACTs controlling OPS control segments (i.e., copy 1, 2, & 5) and will contain the DEU number for those MACTS controlling SPECs or zero when no SPEC is controlled by the MACT (i.e., 3, 4 are copies reserved for GN&C SPECS and copy 5, 6 are copies reserved for SM SPEC).

Sequence_AMT_Index is tested and if its greater than 100 the SPEC to be scheduled is not a System SPEC and its name and priority are extracted from the Application_Moding_Table (AMT). When the SPEC to be scheduled is a System SPEC its name and priority are extracted from a table internal to Sequence_Request_Processor

The Active_Control_Segment_MACT_Index array which contains the MACT copies of all active control segments for a major function. Entry one contains the MACT copy controlling the OPS control segment, Entries 2, 3, 4 contain the MACT copies controlling SPEC control segments active on DEU 1, 2, 3 respectively or contain zero when a SPEC is not active on its appropriate DEU.

The control flow for SPEC_Control_Processor is described in figure 3.2.3.1-13.

OPS_Control_Processor

The OPS_Control_Processor will update Control_Segment_Request_Block to reflect the scheduling of this OPS, collect the scheduling parameters, and perform all processing unique to scheduling OPS0-00 if appropriate.

The Control_Segment_Request_Block entry, Process_Type, is set to 3 (Schedule), and the Active_Control_Segment_MACT_Index array entry 1 is loaded with the MACT copy controlling the OPS being scheduled.

When the OPS being scheduled is not OPS0-00 the scheduling parameter (Name and priority) are loaded from the Application_Moding_Table (AMT) and the OPS_Transition_Parameters are loaded with the Update_OPS_ Transition_Parameters.

When the OPS being scheduled is OPS0-00 and Initialization_Flag = 0, the name and priority are loaded with the name of ARB_IDLE_OPS (810) and its associated priority and Initialization_Flag is set non zero.

When the OPS being scheduled is OPS0-00 and Initialization_Flag is non zero the OPS0-00 control segment will not be scheduled, as it has never been terminated. Parameters are loaded to cause the OPS0-00 page to be placed on all DEU's which are supporting the requested major function.

When Old_OPS (OPS cancelled prior to this schedule) is not equal zero then the OPS_Cancel_Processor SVC is issued (110) to cancel all processes active for this major function, and if all major functions are in OPS0-00 then the redundant set of GPCS is "broken" by setting Redundant_Set_Mask to Hex '0000' and issueing the Sync_Mask_Build_Routine SVC (390).

The Old_OPS test is required to prevent cancelling of active processes and "breaking" the redundant set when the transition to OPS0-00 is as a result of reconfiguration.

The control flow for OPS_Control_processor is described in Figure 3.2.3.1-14.

Control_Segment_Terminator

The Control_Segment_Terminator internal procedure is called to perform all processing required to terminate all control segments active for the major function as specified in Sequence_Major_Functions.

This processing includes saving the OPS number of the OPS control segment being terminated in Old_OPS, updating the Control_Segment_Request_ Block to reflect an OPS transition and the new OPS parameters, locating and requesting active control segments to terminate, and calculating the number of control segments that were requested to terminate.

Locating and requesting the active control segments to terminate is accomplished by searching the Active_Control_Segment_MACT_Index array. A non zero entry in this array indicates a control segment active and controlled by the MACT copy value found in this array. This value is used as an index to the Miscellaneous_Application_Control_Table. The New_OPS_Number, New_Mode_Number are updated with Sequence_Request_OPS/SPEC and Sequence_Request_Mode respectively to provide the active control segment being terminated with the new OPS/mode being requested. The MACT_Application_Control_Flags are set to indicate a block, mode, and control segment termination request and the Name_of_KYBD_MSG_Event is set to signal the request for termination to begin.

OPS0-00 and SPEC 00 are not terminated and do not contribute to the count of active control segments which were requested to terminate. In the case of SPEC 00 its entry in the Active_Control_Segment_MACT_Index is removed.

The control flow for Control_Segment_Terminator is described in Figure 3.2.3.1-15.

Sequence_Reconfiguration_Pre-Processor

The Sequence_Reconfiguration_Pre-Processor internal procedure is called to perform all processing associated with preparing a GPC for reconfiguration (Prime GPC(s) and/or Secondary GPC(s)).

The processing includes providing access to the "blank" OPS0-00 page, and requesting all major functions to transition to OPS0-00.

Access to the "blank" OPS0-00 page is provided by loading Display Format Buffer (DFB) 1 with the display number 1, and since MCDS_Display_Coordinator (520) searches the DFBs in the order 1 to 5, it will find display 1 (OPS0-00 display page) in DFB1. This is necessary since the "real" OPS0-00 page is located in DFB3 which will become part of the overlay.

All control segments for all major functions are then requested to transition to OPS0-00 via a call to the Control_Segment_Terminator. For all major functions not equal to Reconfiguration_In_Progress or when Memory_Source is zero (0) (overlay) the active control segment is "told" that he is transitioning to OPS0-00 so that the control segment can take appropriate action of terminating processes that would normally run across the transition, otherwise the control segment is told the specific OPS/MODE. The Requested_OPS_Number and Requested_Mode_Number entries in the Control_Segment_Request_Block are loaded to specify a transition to OPS mode 1 (OPS0-00).

The Wait_Count is tested and if zero (0) then the Control_Segment_Scheduler is called to schedule OPS0-00.

The control flow for Sequence_Reconfiguration_Pre-Processor is described in Figure 3.2.3.1-16.

Sequence_Reconfiguration_Processor

The Sequence_Reconfiguration_Processor internal procedure is called to determine when all processing prepatory to a reconfiguration has been completed (all major functions are in OPS0-00) and when true, provide processing essential to reconfiguration.

To determine the status of a major function, the Active_Control_Segment_MACT_Index is compared against the Initialization_MACT_Index_Block. When these two arrays are equal, the tested major function is in OPS0-00 only.

When all major functions are in OPS0-00 only, the following processing occurs:

1. A wait is issued to insure that the Cyclic_Display_Processor (620) is not active

2. The DEU(s) are set to I/O Suppressed condition to stop DEU updating

3. The Downlist_Formatter_Enabled_Flag is set to prevent execution of the Downlist Formatter (660)

4. The DEU_Poll_Flags are set to disallow polling on any DEU.

5. If Reconfiguration_In_Progress is less than zero (secondary reconfiguration), the ICC_Message_Collector (480) is called to indicate that this secondary GPC has completed all processing requested, otherwise GPC_Reconfiguration (820) is scheduled (primary reconfiguration).

The control flow for Sequence_Reconfiguration_Processor is described in Figure 3.1.2.1-17.

AMT/GRT_Search_Processor

The AMT/GRT_Search_Processor internal procedure is called to determine the legality of a non-zero OPS request and to determine if reconfiguration is required.

The In_Core_Tester logic is envoked to search the Applications_Moding_Table (AMT) for the requested OPS. The status of this search is returned in Sequence_AMT_Index. (0 not in core, non zero in AMT).

The Overlay_Allowed_Tester logic is envoked to search the GPC_Reconfiguration_Table (GRT) to determine if the request OPS is an overlay initiator. The status of this search is returned in GRT_Index. (0 is not an overlay initiator, non zero is the GRT table index defining this OPS).

When GRT_Index = 0, Sequence_AMT_Index is tested and if non zero an OPS transition without overlay is specified in Sequence_Process_Type. Otherwise, an error condition exists and error message OT001 is annunciated.

When GRT_Index is non zero and this GPC is in "STANDBY" mode, reconfiguration is required if Sequence_AMT_Index = 0 (not in core) or Action_Type is greater than 1 (keyboard request) and Memory_Source = 0 (overlay required).

When GRT_Index is non zero and this GPC is not in "STANDBY" mode, the GPC_Set (the GPC(s) required for this OPS execution as defined in the GRT) is used to determine if the required GPCs are in "RUN" mode.

When none of the required GPC(s) is in "RUN" mode, error message OT005 is annunciated, otherwise reconfiguration is required if the requested OPS is not in core or Sequence_Run_Bits (GPC(s) required that are also in "RUN" mode) is not equal the Redundant_Set_Mask (current redundant set of GPC(s)) or Memory_Source = 0 (overlay required) and Action_Type greater than 1 (keyboard request). A warning message (OT004) will be issued if at least one but not all the required GPC(s) are in a mode other than "RUN".

The control flow AMT/GRT_Search_Processor is described in Figure 3.2.3.1-18.

IN_Core_Tester

The IN_Core_Tester will search the Application_Moding_Table (AMT) for the requested OPS and determine its mode validity if found.

The loop variables that will be used to search the AMT are determined by the first copy of OPS data in the AMT for this major function (First_OPS_Copy_Number) and the number of OPS control segments in this major function (Number_Legal_OPS_).

When the major function is supproted in the AMT (loop variables are both non zero) and the AMT table is considered valid (AMT_Table_Valid non zero) the AMT is searched for the requested GPC. When found, Sequence_AMT_Index is set to proper copy value and then requested mode is matched against Number_Modes. (Maximum mode allowed this OPS) when the mode is in error, Found_Value is set to zero.

The control flow for IN_Core_Tester is described in Figure 3.2.3.1-19.

Overlay_Allowed_Tester

Overlay_Allowed_Tester will search the GPC_Reconfiguration_Table (GRT) for the requested OPS and when found, determine the mode validity.

Sequence_Major_Function and Sequence_Request_OPS/SPEC are matched against Major_Function and OPS_Number for all memory configuration defined in the GRT. When a match is found, GRT_Index is set to the GRT copy number where the match occured and requested mode is verified. When Maximum_Number_Modes as defined in the GRT is less than the requested mode, Found_Value is set to zero to indicate a mode error.

The control flow for Overlay_Allowed_Tester is defined in Figure 3.2.3.1-20.

Mode_To_Mode_Processor

The Mode_to_Mode_Processor internal procedure will be called to perform the processing required when the requested OPS is equal to the current OPS.

This processing includes updating the New_Mode_Number with Sequence_Request Mode, setting the MACT_Application_Control_Flags to request block and mode termination, and setting the Name_Of_KYBD_MSB_Event to signal the request for block and mode termination to begin.

The control flow for Mode_To_Mode_Processor is described in Figure 3.2.3.1-21.

Buffer_Address_Mover

The Buffer_Address_Mover internal procedure is called to move the addresses of Display_Format_Buffers (DFB) 1, 2, 3, 4, 5, and the Mass_Memory_Directory address from the Application_Moding_Table (AMT) to the Display_Buffer_Name_Table (local storage table within User_Interface_Control_Supervisor) and to the DFB_Name_Structure (for use by Cyclic_Display_Processor (620)).

The data will be moved only if AMT_Table_Valid is non zero. It is set to zero whenever an overlay error occurs and is set non-zero upon the completion of a successful overlay.

The control flow for Buffer_Address_Move- is described in Figure 3.2.3.1-22.

d. Outputs - See Table 3.2.3.1

e. Module Reference -

1. (480) ICC_Message_Collector (DIM_ICC_COLLECTOR) is CALLed
2. (520) MCDS_Display_Coordination (DMC_DISPLAY) is CALLed
3. (675) Annunciation_Macro_Interface (DMA_MAC) is CALLed
4. (810) Idle_Operational_Sequence (ARB_IDLE_OPS) is SCHEDULed
5. (820) GPC_Reconfiguration (ARC_GPC_RECONFIG) is SCHEULed
6. (910) Read/Write_Specialist_Function (ASB_RD_WRT) is SCHULed
7. (950) Time_Management_Specialist_Function (ASC_TIME_MGMT) is SCHEDULed
8. All OPS control segments as defined in any Application_Control_Table (AMT) are SCHEDULed as required
9. All Specialist (SPEC) control segments are defined in any Application_Control_Table (AMT) are SCHEDULed as required
10. (101) Process_Scheduler (FPMSCHED) FCOS SVC envoked
11. (104) Wait_Processor (FPMWAIT) FCOS SVC envoked
12. (107) Close_Processsor (FPMCLOSE) FCOS SVC envoked
13. (110) OPS_Cancel_Processor (FPMCANCL) FCOS SVC envoked
14. (170) Set_Event_Processor (FPMSET) FCOS SVC envoked
15. (171) Reset_Event_Processor (FPMRESET) FCOS SVC envoked
16. (390) Sync_Mask_Build_Routine (FCMSMASK) FCOS SVC envoked

f. Module Attributes - Internal Procedure

g. Template References - See User_Interface _Control_Supervisor (500) Section 3.2.1.1.

h. Error Handling -

When the requested OPS is not found in the Application_Moding_Table and it is not found in the GPC_Reconfiguration_Table (GRT), error message OT001 is annunciated and processing is terminated.

When the requested OPS is found in the GRT but there are no target GPCs associated with the requested OPS, error message OT003 is annunciated and processing is terminated.

When the requested OPS is found in the GRT but at least one target GPC is found in "RUN" mode but not all target GPC are found in "RUN" mode

the warning error message OT004 is annunciated and processing continues.

When the requested OPS is found in the GRT but none of the target GPC are found in the "RUN" mode error message OT005 is annunciated and processing is terminated.

When the requested OPS has a mode requested that is greater than the maximum number of modes for that OPS as defined in the AMT and/or GRT, error message OT010 is annunciated and processing is terminated.

When a SPEC is requested and it is found to be currently active (in any major function if its a System SPEC or the requested major function if its not a System SPEC), error message DU007 is annunciated and processing is terminated.

When a SPEC is requested and it is found not to be a System SPEC and it is not a SPEC defined in the AMT for the current OPS, error message DU012 is annunciated and processing is terminated.

When a SPEC is requested and two other SPECs are found active on DEUs other than the one requesting this SPEC, error message DU026 is annunciated and processing is terminated.

When an OPS or SPEC keyboard request is received during the process of OPS transition for the requested major function or during the time reconfiguration is being performed, error message DU028 is annunciated and processing is terminated.

i. Constraints and Assumptions - The caller must provide a proper value in the passed parameter Action_Type. This value must be 0, 1, 2, 17 or 18.

j. Detailed Implementation -

None

DATA TABLE 3.2.3.1-1

NAME Sequence_Request_Processor (DMC_SEQ_REQ_PROC)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	Action_Type	L119	L	505	505 560	DMC_TYPE			
2	Reconfiguration_In_Progress	L120	L	560	See APP.E	DMC_REC_IN_PROGRESS			
3	Process_Type	L118.20	L	560	500,510 520,560	DMCV_RQPROC_TY			
4	Sequence_Test_1	L121	L	560	560	DMC_TVAL1			
5	Sequence_Major_Function	L122	L	560	560	DMC_MFI			
6	Sequence_Process_Type	L123	L	560	560	DMC_PROCTYPE			
7	Terminator_Index	L124	L	560	560	DMC_TERMI			
8	Test_Index	L125	L	560	560	DMC_TINDX			
9	Active_Control_Segment_MACT_Index	L118.70	L	560	505 560	DMCV_RQMACT_IN			
10	MCDS_MACT_Index	L126	L	505	505, 510 520, 560	DMC_MACT_INDEX			
11	Wait_Count	L118.10	L	560	560	DMCV_RQWTC			
12	DEU_Number	L127	L	500	See APP.E	DMC_DIT_INDEX			
13	DEU_Request_MACT_Index	L128	L	560	500,505 560	DMCV_DEU_MACT			
14	SPEC_Zero_MACT	L129	L	560	560	DMC_SZFMACT			
15	Sequence_AMT_Index	L130	L	560	560	DMC_AMT_I			
16	MCDS_Major_Function	L192	L	500 505,560	See App.E	DMC_MF			
17	Active_OPS_AMT_Index	L118.30	L	560	560	DMCV_RQAMT_IN			
18	Sequence_MACT_Index	L167	L	560	560	DMC_MACT_I			
19	Sequence_Request_OPS/SPEC	L175	L	560	560	DMC_REQOSNBR			
20	Next_SPEC_Number	L168.10	L	560	560	DMC_SPECID			

DATA TABLE 3.2.3.1-1 (Cont'd.)

NAME Sequence_Request_Processor (DMC_SEQ_REQ_PROC)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
20	Next_SPEC_AMT	L169.20	L	560	560	DMC_AINXTS			
21	Next_SPEC_MACT	L169.10	L	560	560	DMC_MINXTS			
22	Display_Request_Level	L194	L	505, 510 520, 560	505,510 520,560	DMCVL_D_LVLN			
23	Display_Request_Number	L195	L	505,510, 520,560	505,510 520,560	DMCVL_D_DISP			
24	Current_Display_Number	J040.36	I,O	505, 540 550, 560	505 560	CZ1V_D_DISP			
25	System_SPEC_ID	L170.10	L	560	560	DMCV_SYS_ID			
26	Reconfiguration_OPS_Request	I520	I,O	560 820	560 820,880	CZ2V_REC_OPS			
27	OPS_Zero_Page	L176	L	560	560	DMC_OZPAGE_VAL			
28	Reconfiguration_In_Initialization	L193	L	500,505, 560	505 560	DMC_REC_IP2			
29	AMT_Table_Valid	L177	L	560	520 560	DMC_AMTVALID			
30	Display_Buffer_Name	L178.10	L	560	510,520 560,600	DMC_BUF			
31	Sequence_Request_Mode	L179	L	560	560	DMC_REQMODE			
32	Reconfiguration_Mode_Request	I530	I,O	560 820	560 820	CZ2V_REC_MODE			
33	Reconfiguration_DEU_Source	I500	I,O	560	560 820	CZ2V_REC_DEU			
34	Reconfiguration_Major_Function	I510	I,O	560	560 820,880	CZ2V_REC_MF			
35	Reconfiguration_Status	I540	I,O	560,810 820,880	560,810 820,880	CZ2V_REC_XERR			
36	GPC_OPS_Zero_Keystrokes	L180	L	560	560	DMC_GPCZ			
37	DEU_Number_OF_Keystrokes	J060.03	I	560		CZ1V_D_DIT_NUMOFKEYS			

DATA TABLE 3.2.3.1-1 (Cont'd)

NAME Sequence_Request_Processor (DMC_SEQ_REQ_PROC)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
38	Keyboard_Message	J060.20	I	See APP. E	See APP. E	CZ1V_DIT_KYBD_MSG	SEE APPENDIX E		
39	Sequence_Update_Index	L198	L	560	560	DMC_UNDX			
40	Sequence_Current_OPS	L183	L	560	560	DMC_CUR_OP			
41	Initialization_Flag	L196	L	500 560	500 560	DMC_INITIAL			
42	Number_Modes	A080.20	I	DFG	550 560	CDAV_OP_NMODES			
43	Assign_MACT_Copy	L197	L	560	560	DMCV_MACT_MF			
44	Reconfiguration_Table_Index	I490	L	560 870	560, 810 820, 880	CZ2V_REC_GRT_INDEX			
45	Sequence_Test 4	L202	L	560	560	DMC_TVAL4			
46	Sequence_Test 3	L201	L	560	560	DMC_TVAL3			
47	Sequence_Tester	L199	L	560	560	DMC_TVAL			
48	Sequence_Test 2	L200	L	560	560	DMC_TVAL2			
49	First_SPEC_Copy	A080.50	I	DFG	560	CDAV_OP_SPEC_ST			
50	Number_SPECs	A080.40	I	DFG	560	CDAV_OP_NSPEC			
51	SPEC_Number	A090.10	I	DFG	560	CDAV_SP_ID			
52	MACT_Application_Control_Flags	J040.18	O	505 550,560	550	CZ1B_D_FLAG2			
53	Name_Of_Keyboard_Message_Event	J040.45	O	SEE APP.E	540	CZ1E_D_KYBD_MSG_EVT			
54	Sequence_Value	L181	L	560	560	DMC_SEQVAL			
55	Sequence_Wait_Count	L182	L	560	560	DMC_WTCT			
56	MAT_Major_Function_Setting	B010.10	I,O	505 520	505,560 600,620	CDMV_MAT_MF			
57	DEU_Display_Support_Level	B010.15	I,O	505,520 620	505,560 600	CDMV_MAT_LEVEL			

DATA TABLE 3.2.3.1-1 (Cont'd)

NAME Sequence_Request_Processor (DMC_SEQ_REQ_PROC)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
58	Display_Level_Page_Number	B010.35	O	505,520 560	505 620	CDMV_MAT_DISP_NUMBER			
59	Control_Segment_Index_To_MACT	B030	O	560	550 560	CDMV_MACT_INDEX			
60	Application_MACT_Index	J066	O	560	810, 900 910, 950	CZ1V_D_MACT_INDEX			
61	Name_MACT_Copy	L172	L	560	560	DMC_NMACT			
62	Miscellaneous_Application_Control_Table	J040	I,O	505,510 540,550	Appendix E	CZ1V_D_MACT			
63	Major_Function_ID	J040.35	O	560	505 560	CZ1V_D_MF_ID			
64	Keyboard_Input_DEU_Number	J040.51	O	505 510,560	505,810 900,910	CZ1V_D_DEU_NUMBER			
65	Termination_Sequence_Index	J040.56	O	560	540 550, 560	CZ1V_D_SEQUENCE_INFO			
66	New_OPS_Number	J040.41	O	550 560	550	CZ1V_D_NEW_OPS			
67	New_Mode_Number	J040.42	O	540 550, 560	550	CZ1V_D_NEW_MODE			
68	MACT_Index_Event	B040	I,O	550 560	560	CDME_MACT_INDEX			
69	SPEC_Blocks	A090.20	I	DFG	560	CDAV_SP_BLOCKS			
70	Sequence_Program_Name	L173	L	560	560	DMC_PROG_NAME			
71	SPEC_Program_Name	A110.10	I	DFG	560	CDAV_SP_SPECNM			
72	Sequence_Program_Priority	L174	L	560	560	DMC_PROG_PRIOR			
73	SPEC_Priority	A090.30	I	DFG	560	CDAV_SP_PRIOR			
74	System_SPEC_Blocks	L170.20	L		560	DMCV_SYS_BLOCKS			
75	System_SPEC_Program_Name	L171.10	L	560	560	DMCV_SYS_SPNAME			
76	System_SPEC_Priority	L170.30	L		560	DMCV_SYS_PRIOR			
77	Requested_OPS_Number	L118.50	L	560	560	DMCV_RQOSNBR			

DATA TABLE 3.2.3.1-1 (Cont'd)

NAME Sequence_Request_Processor (DMC_SEQ_REQ_PROC)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
78	Old_OPS	L184	L	560	560	DMC_SAVE			
79	ICC_Current_Load	I320.06	O	820 880	560,810	CZ2B_ICC_FLAG			
80	Redundant_Set_Mask	I010.13	I,O	Appendix E	Appendix E	CZ2B_RS	See Appendix E		
81	ID_GPC	L190	L	505	505 520, 560	DMC_GPCID			
82	Last_Major_Function	I450	O	Appendix E	405	CZ2V_MF_OLD			
83	ICC_CR/RS	I320.08	O	820 880		CZ2B_ICC_FLAG			
84	Current_OPS	I010.02	O	560 820, 880	Appendix E	CZ2V_CURRENT_OPS			
85	OPS_Transition_Parameters	I580	O	560		CZ2V_RM_OTP			
86	Update_OPS_Transition_Parameters	I590	I	560		CZ2V_UPDATE_RM_OTP			
87	OPS_Program_Name	A100.10	I	DFG	560	CDAV_OP_PROGNM			
88	OPS_Priority	A080.70	I	DFG	560	CDAV_OP_PRIOR			
89	Name_Request_Block	L166	L	560	560	DMC_NRQBLK			
90	Requested_Mode_Number	L118.60	L	560	560	DMCV_RQMODE			
91	Memory_Source	I040.02	I	Appendix E	Appendix E	CZ2B_DPS_STATUS			
92	Initialization_MACT_Index_Block	L165.70	L		500 560	DMCV_RQMACT_IN			
93	CYCLIC_In_Process_Event	B090	I	620	500,505 560,520	CDME_CYCLIC_EVENT			
94	CRT_Status_Flag 2	B010.42	O	560	560	CDMB_MAT_CRT_STAT			
95	Downlist_Formatter_Enabled_Flag	I010.16	O	560,700 820,880	710 750	CZ1B_STATUS			
96	ICC_GST_Status	I320.09	O	Appendix E	142	CZ2B_ICC_FLAG			
97	DEU_Poll_Flags	J062	O	SEE APP. E	SEE APP. E	CZ1B_D_DEU_POLL_FLAG			

DATA TABLE 3.2.3.1-1 (Cont'd)

NAME Sequence_Request_Processor (DMC_SEQ_REQ_PROC)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
98	Sequence_Secondary_ICC_Message	L188	L	560	485 560	DMC_REC_IMSG			
99	Sequence_Assign_Value	L189	L	485 560	485 560	DMC_ASSIGN			
100	Sequence_Assign_Status	L189.10	L	485	560				
101	Found_Value	L191	L	560	560	DMC_FOUND			
102	GRT_Index	L185	L	560	560	DMC_GRTIX			
103	Discrete_Bit_Standby	&005.02	I						
104	GPC_Set	I210	I	810 860	560,810 820,860	CZ2B_GRT_GPC_SET			
105	Discrete_Bit_Run_Mode	&005.03	I						
106	Sequence_Run_Bits	L186	L	560	560	DMCB_RUNBITS			
107	First_OPS_Copy_Number	A070.20	I	DFG	560	CDAV_MF_FOPS			
108	Number_Legal_OPS	A070.10	I	DFG	560	CDAV_MF_NOPS			
109	OPS_Number	A080.10	I	DFG	560	CDAV_OP_OPSID			
110	Number_Modes	A080.20	I	DFG	550 560	CDAV_OP_NMODES			
111	Major_Function	I620.04	I		560 800,860	CZ2V_GRT_MF_VALUE			
112	OPS_Number	I620.05	I		560 850	CZ2V_GRT_OPS_NBR			
113	Maximum_Number_Modes	I620.06	I		560	CZ2V_GRT_MAX_MODE			
114	Display_Buffer_Name_Table	L178	L	560	560	DMC_BUFNAMES			
115	AMT_Buffer_Name_Table	A010	I,O	DFG	560	CDAV_BNAME			
116	DFB_Name_Structure	B100	I,O	560	560 620	CDMV_B_NAMES			
117	I	L203	T	560	560	I			

DATA TABLE 3.2.3.1-1 (Cont'd)

NAME Sequence_Request_Processor (DMC_SEQ_REQ_PROC)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
118	J	L0204	T	560	560	I			

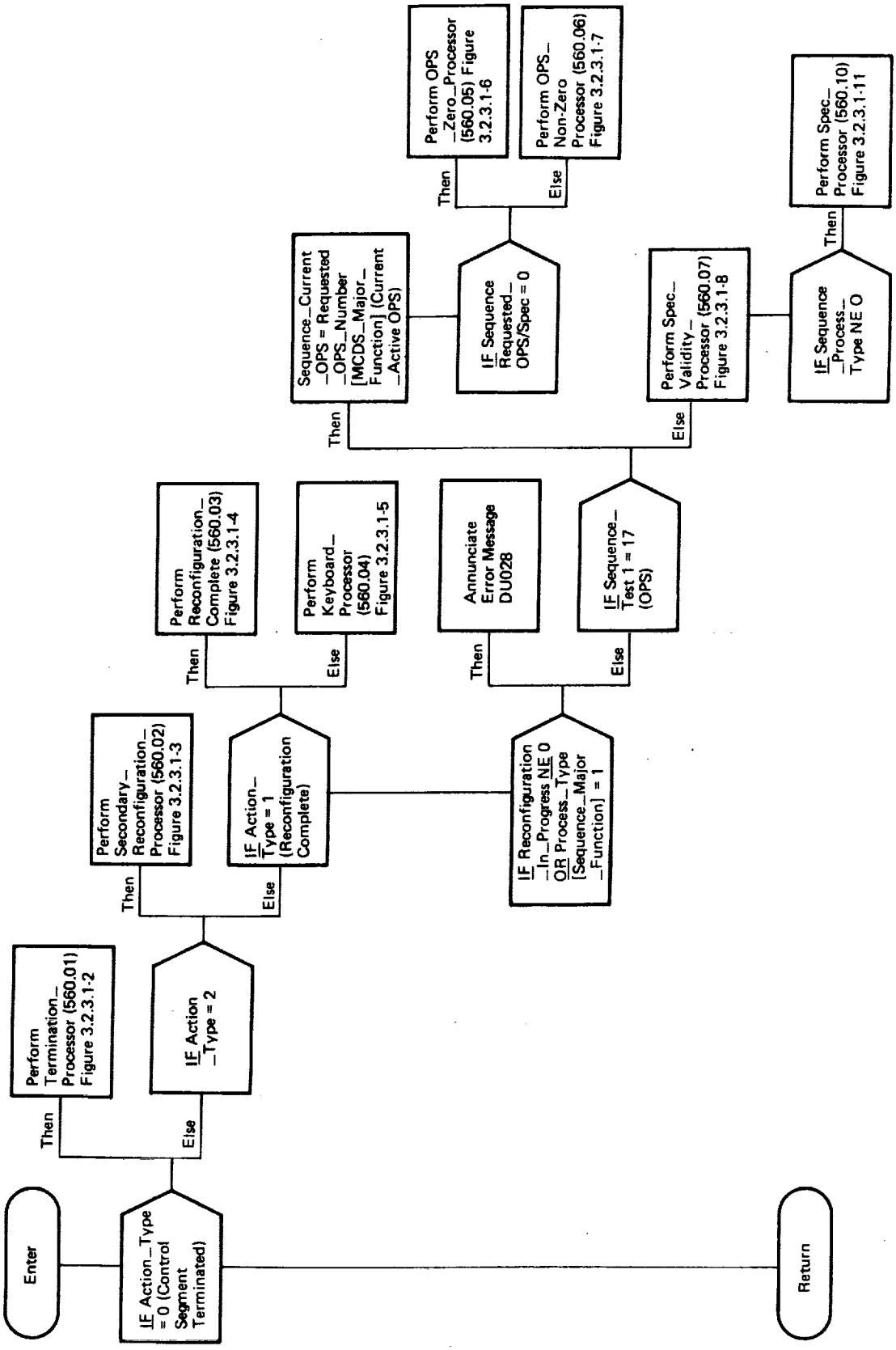


Figure 3.2.3.1-1. Sequence_Request_Processor (DMC_SEQ_REQ_PROC)

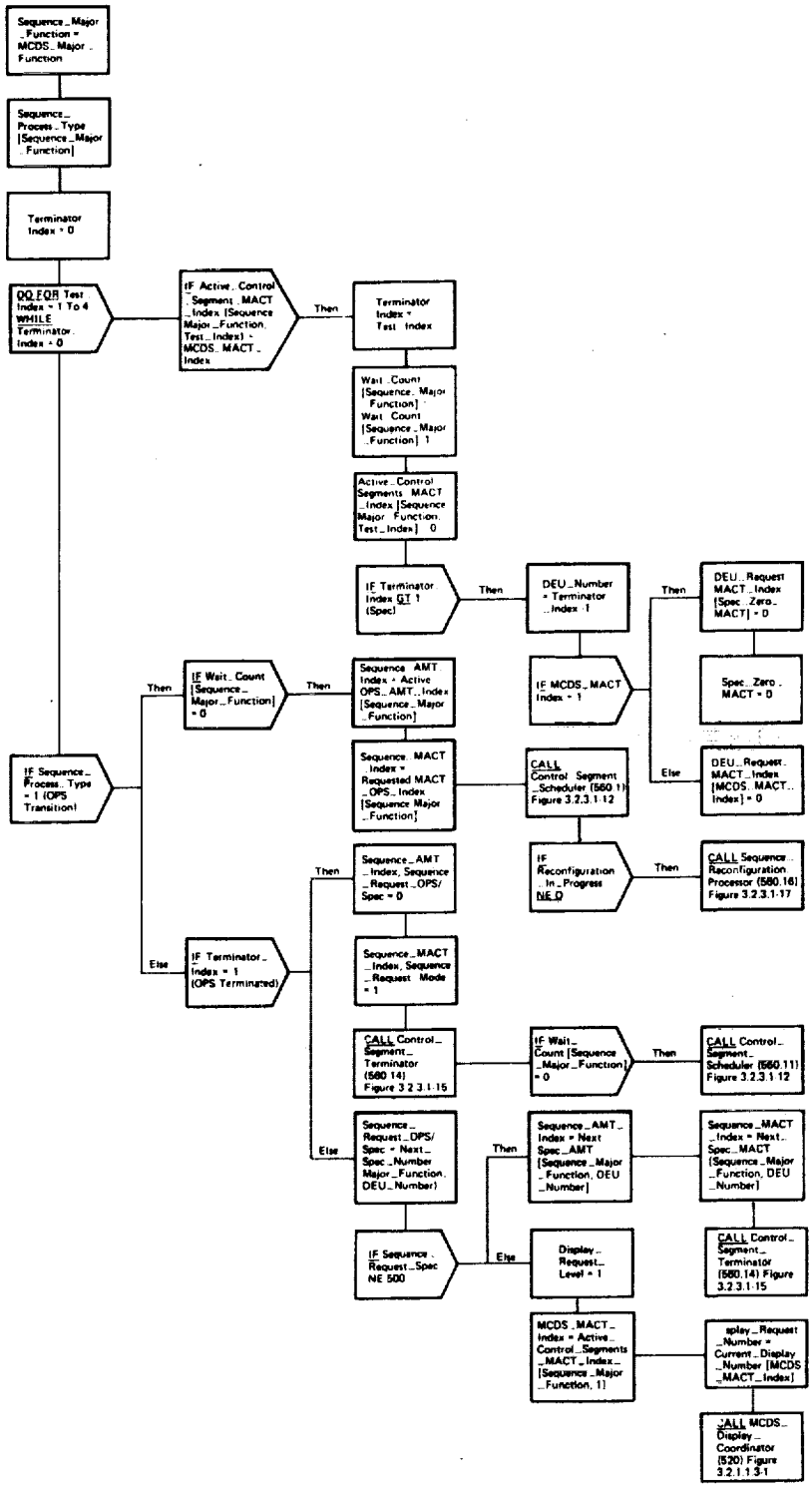


Figure 3.2.3.1-2. Sequence_Request_Processor Termination_Processor (560.01)

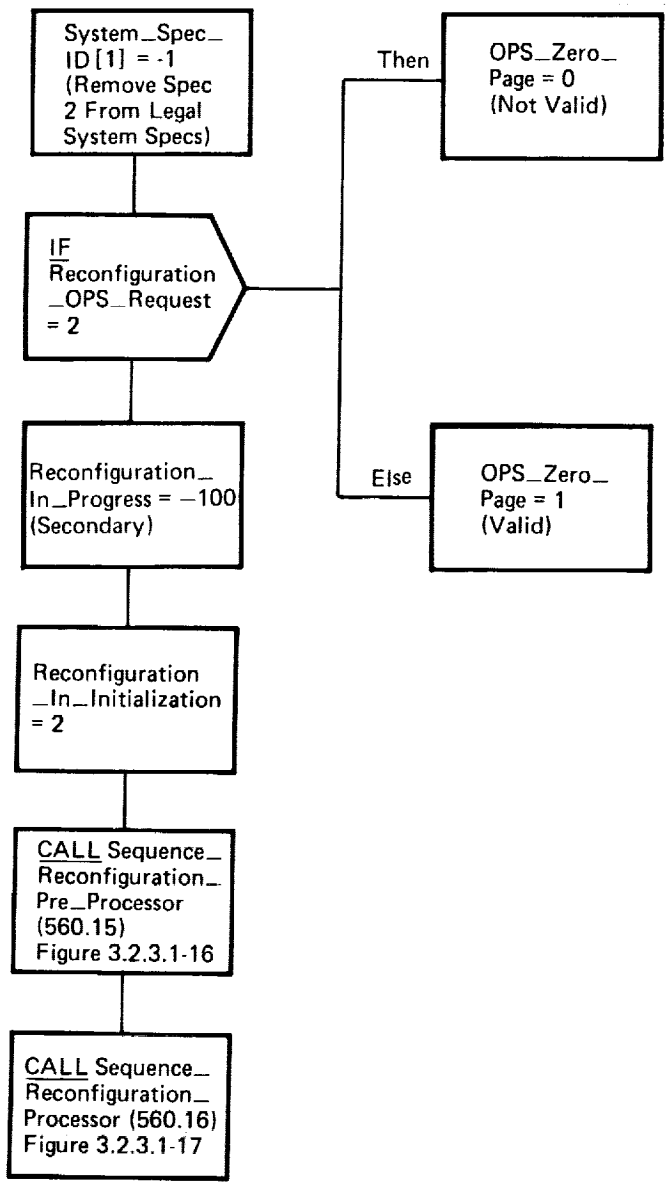


Figure 3.2.3.1-3. Sequence_Request_Processor Secondary_Reconfiguration_Processor (560.02)

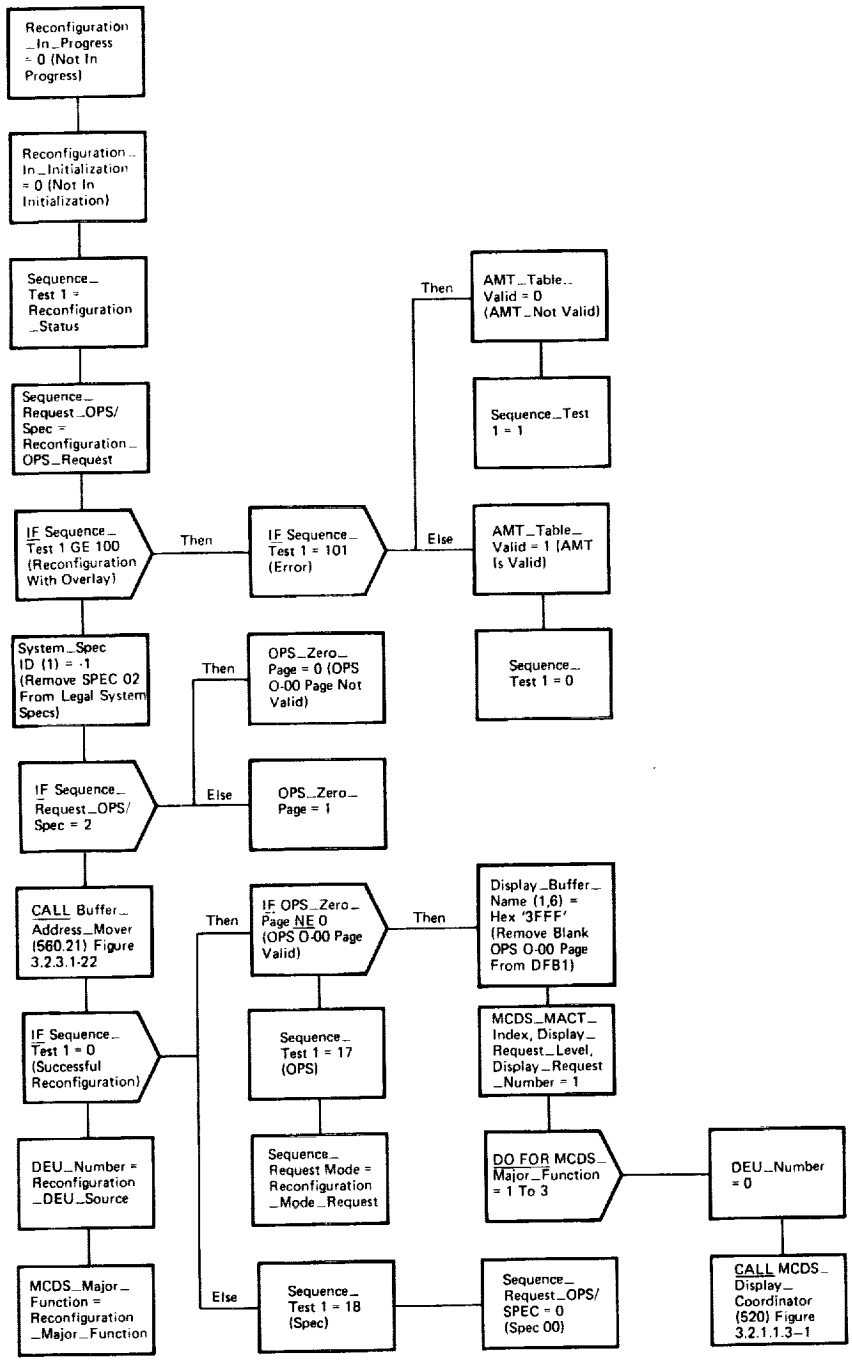


Figure 3.2.3.1-4. Sequence_Request_Processor Reconfiguration_Complete (560.03)

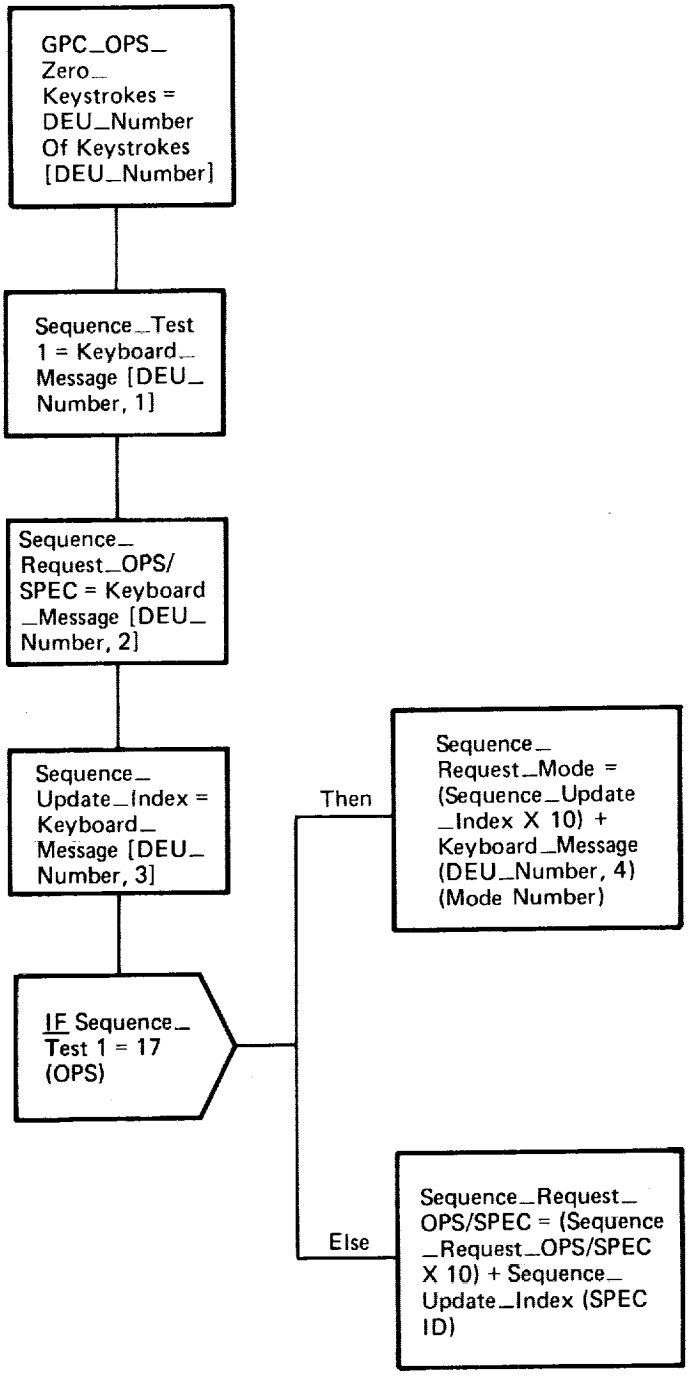


Figure 3.2.3.1-5. Sequence_Request_Processor Keyboard_Processor (560.04)

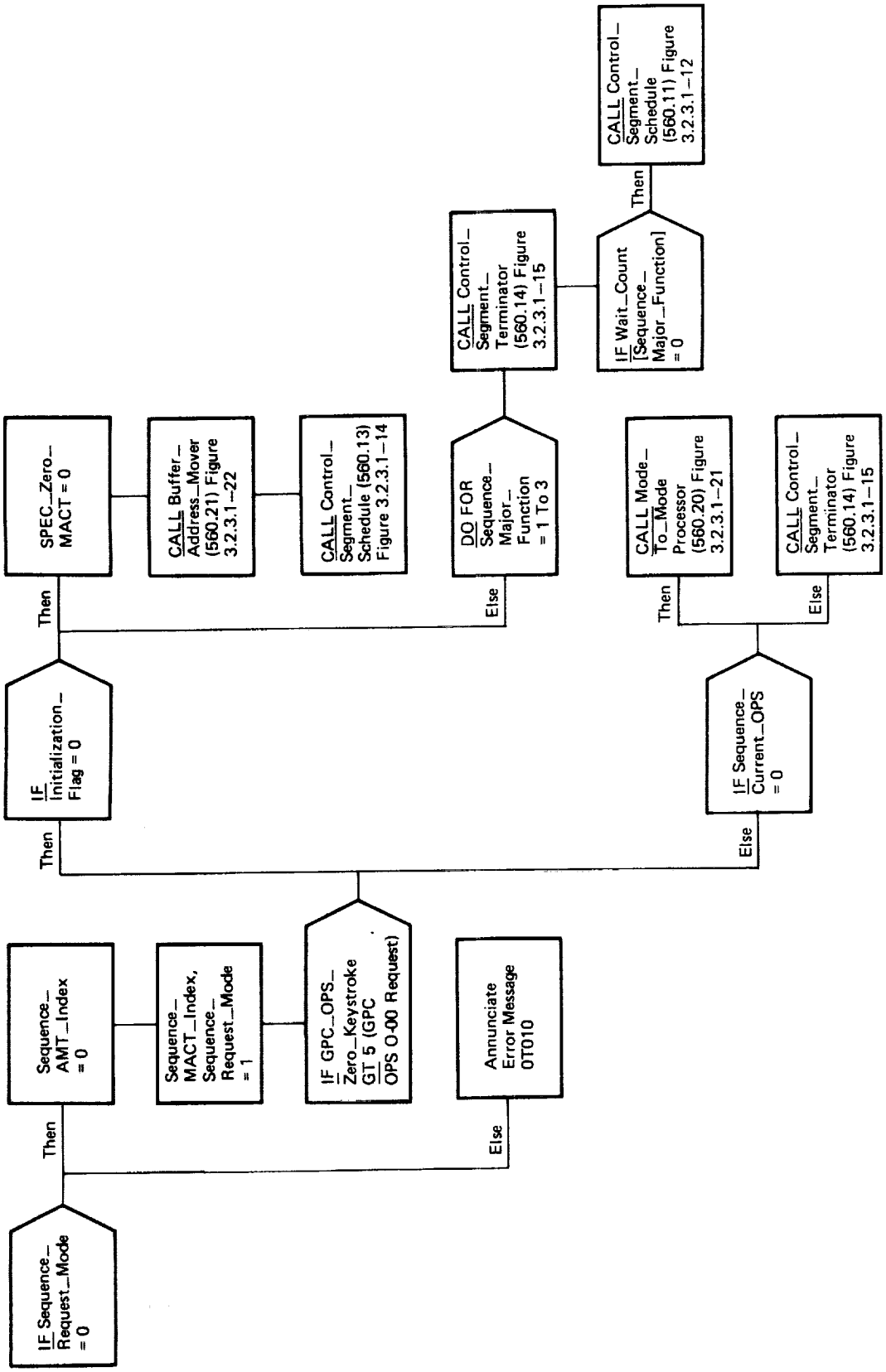


Figure 3.2.3.1-6. Sequence_Request_Processor OPS_Zero_Processor (560.05)

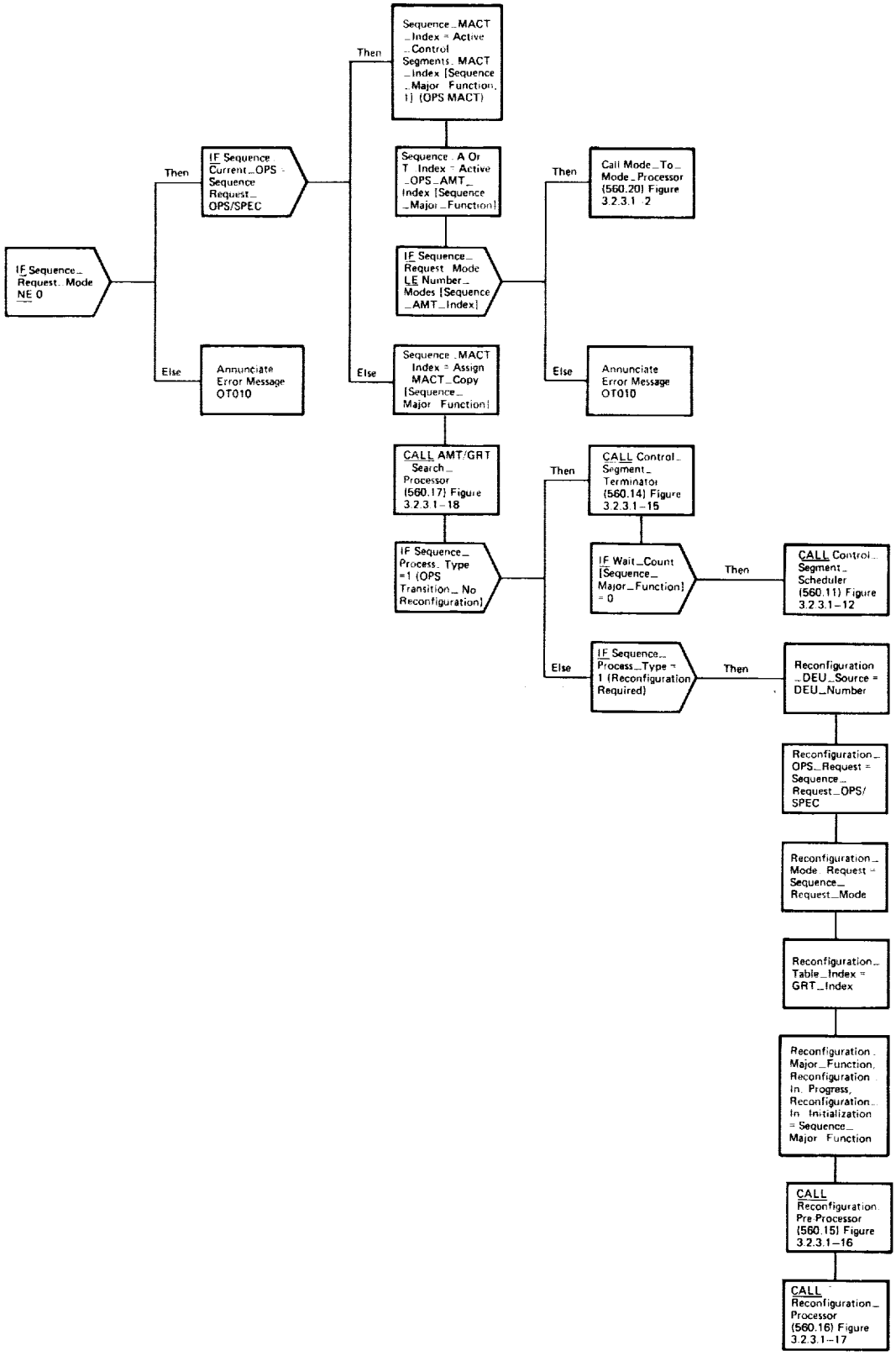


Figure 3.2.3.1-7. Sequence_Request_Processor OPS_Non_Zero_Processor (560.06)

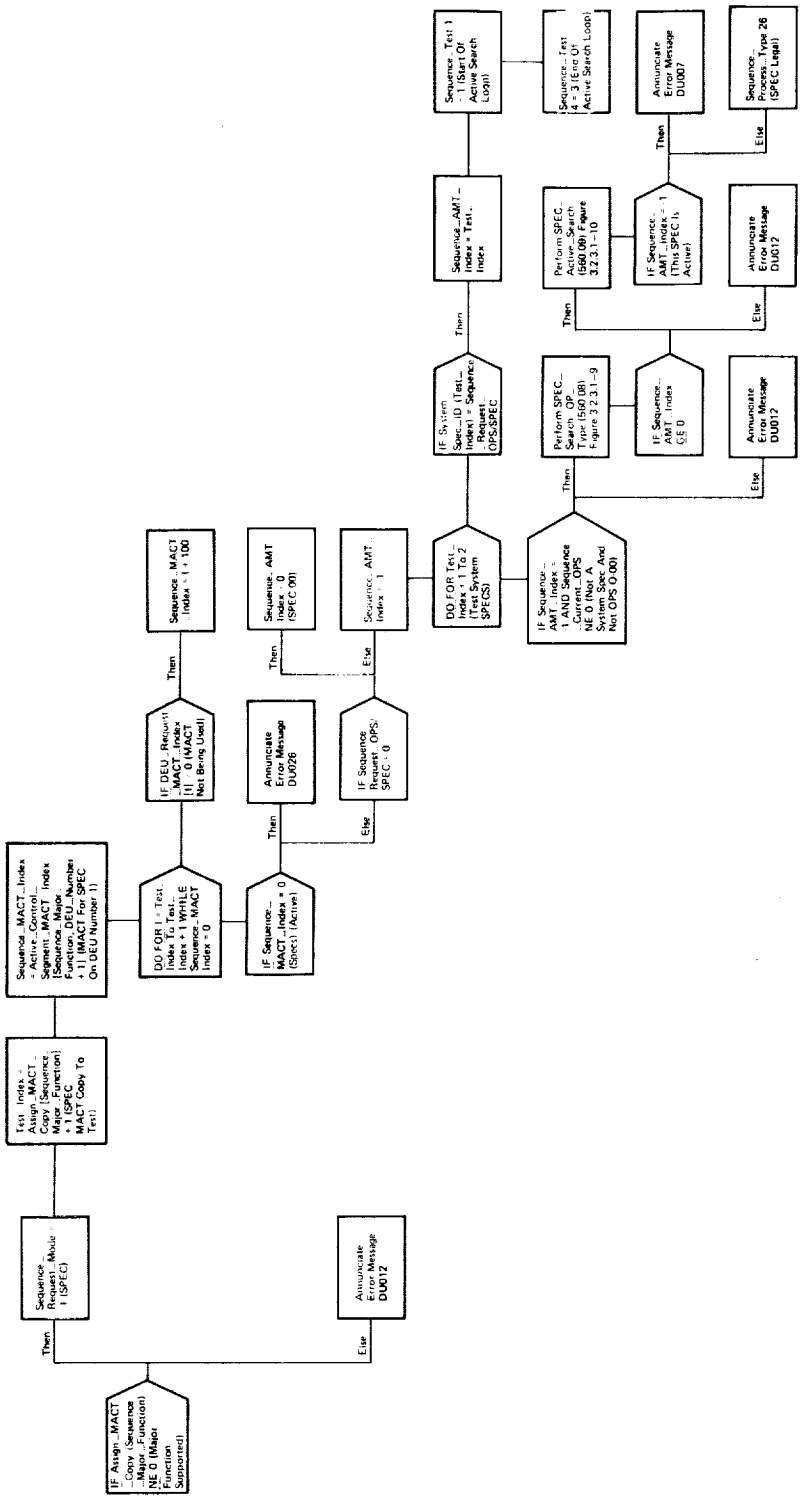


Figure 3.2.3.1-8. Sequence_Request_Processor SPEC_Validity_Processor (560.07)

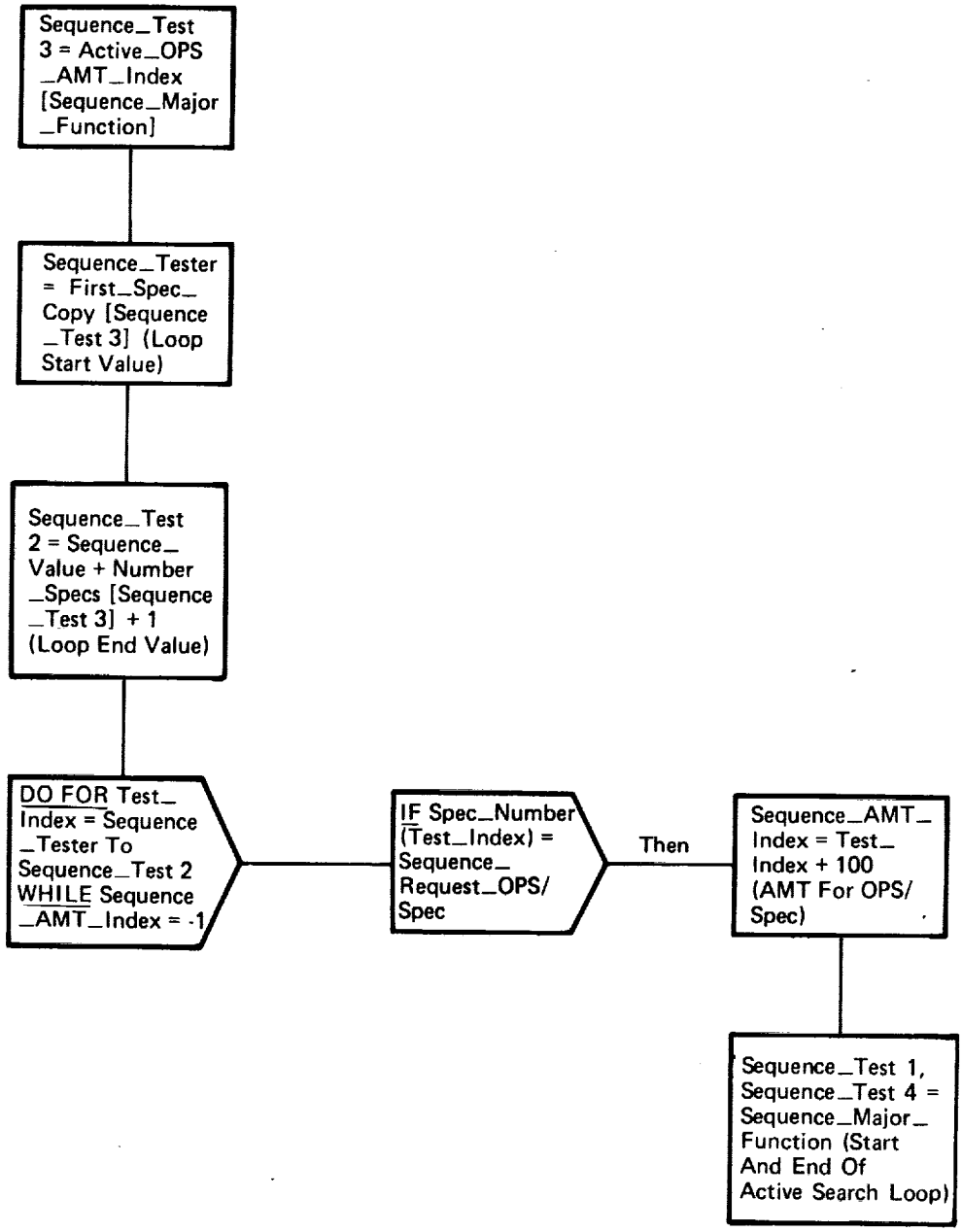


Figure 3.2.3.1-9. Sequence_Request_Processor SPEC_Search_OPS_Type (560.08)

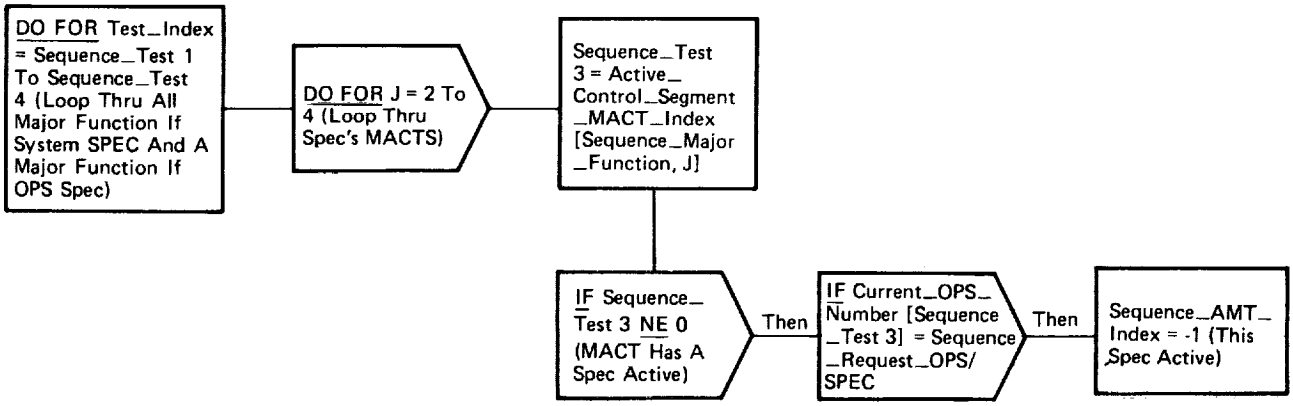


Figure 3.2.3.1-10. Sequence_Request_Processor SPEC_Active_Search (560.09)

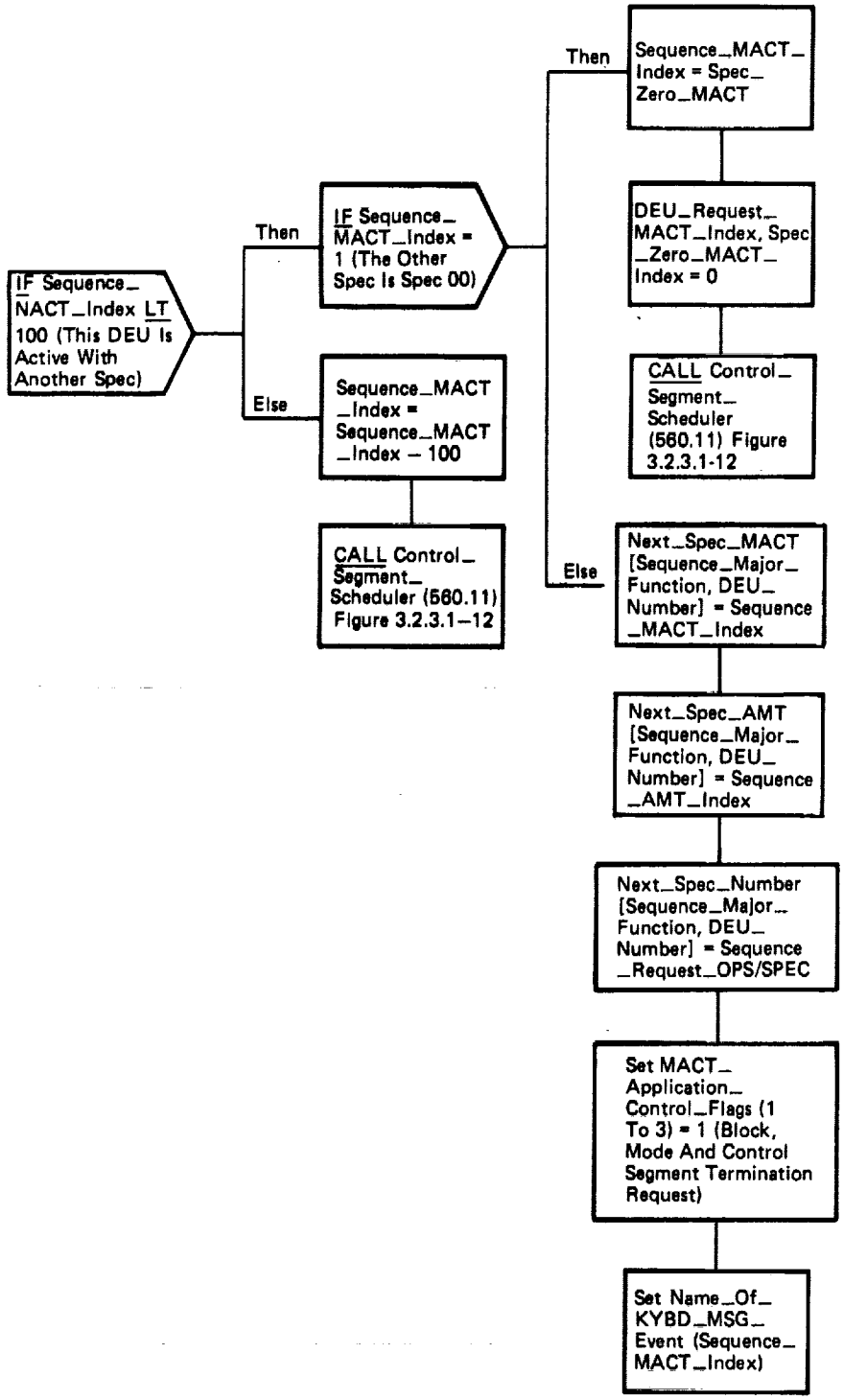


Figure 3.2.3.1-11. Sequence_Request_Processor SPEC_Processor (560.10)

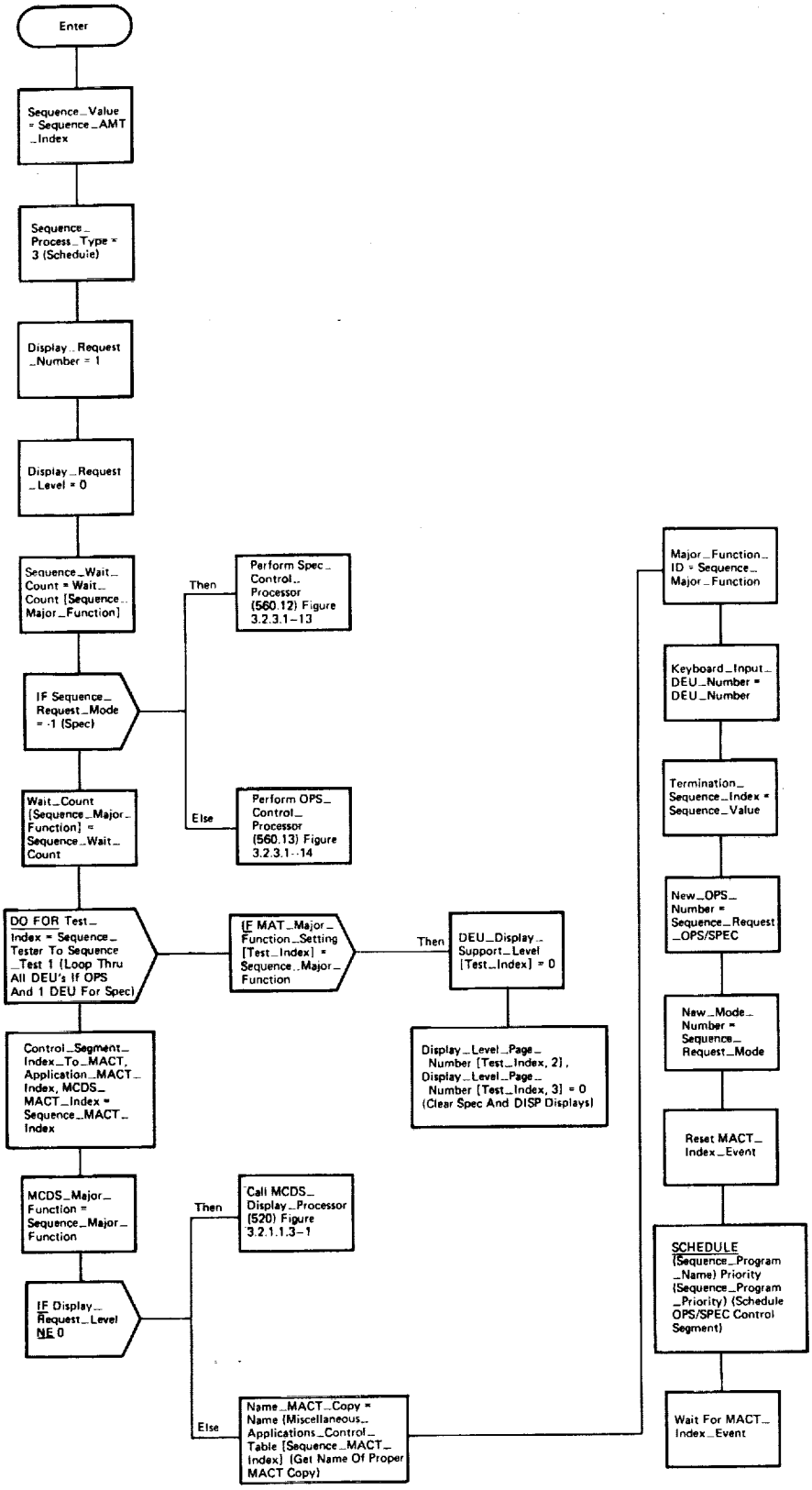


Figure 3.2.3.1-12. Sequence_Request_Processor Control_Segmet_Scheduler (560.11)

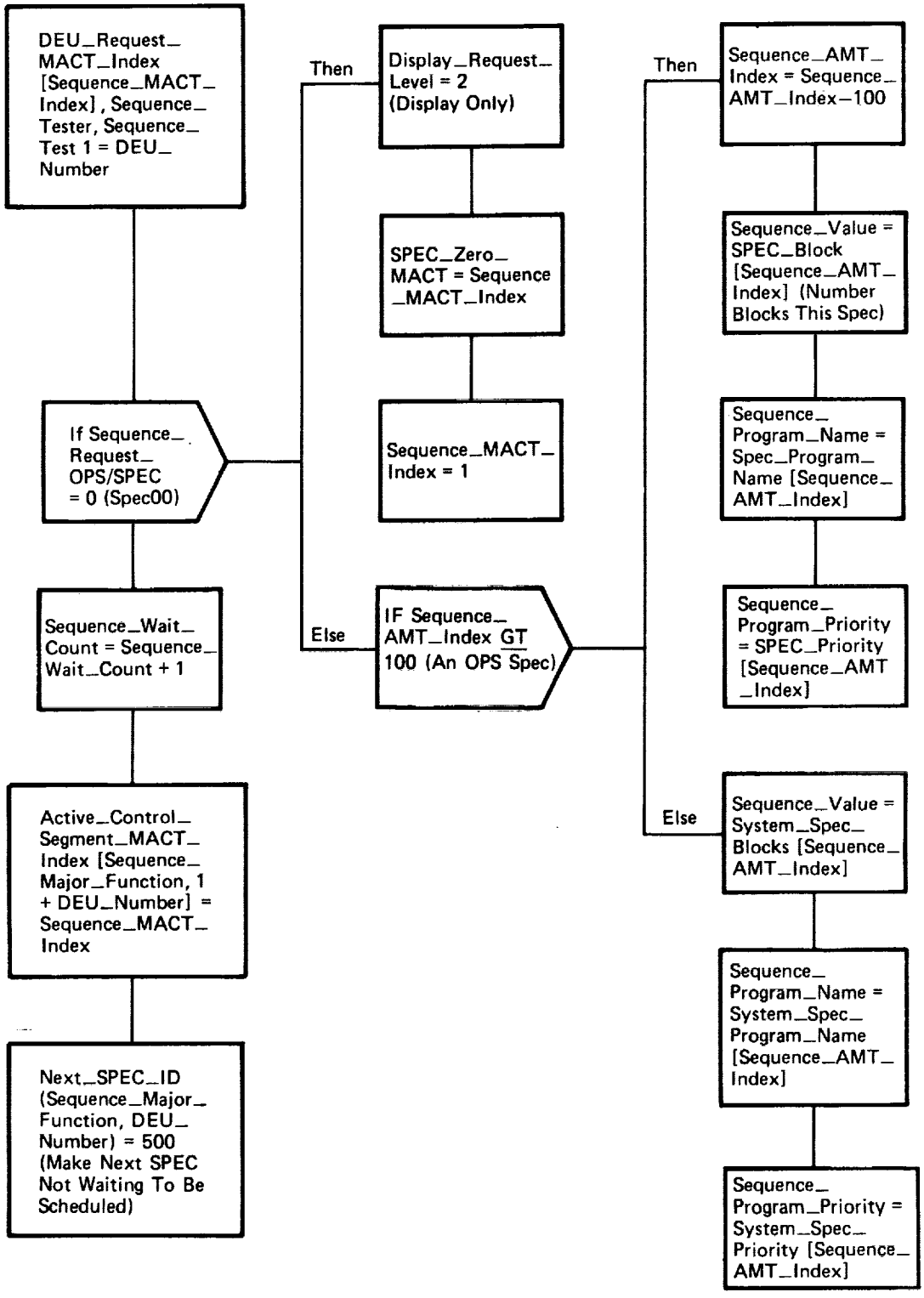


Figure 3.2.3.1-13 Sequence_Request_Processor SPEC_Control_Processor (560.12)

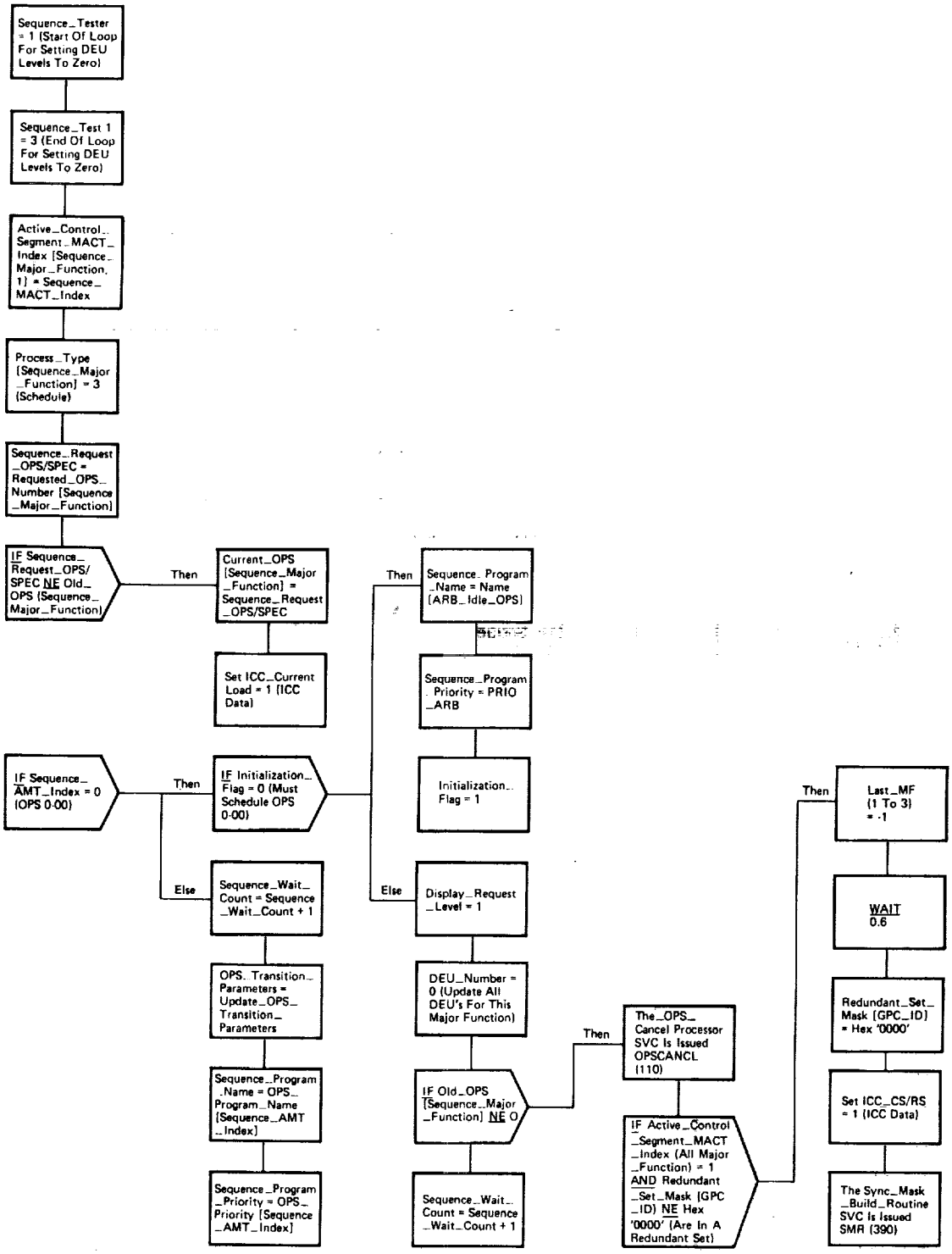


Figure 3.2.3.1-14. Sequence_Request_Processor OPS_Control_Processor (560.13)

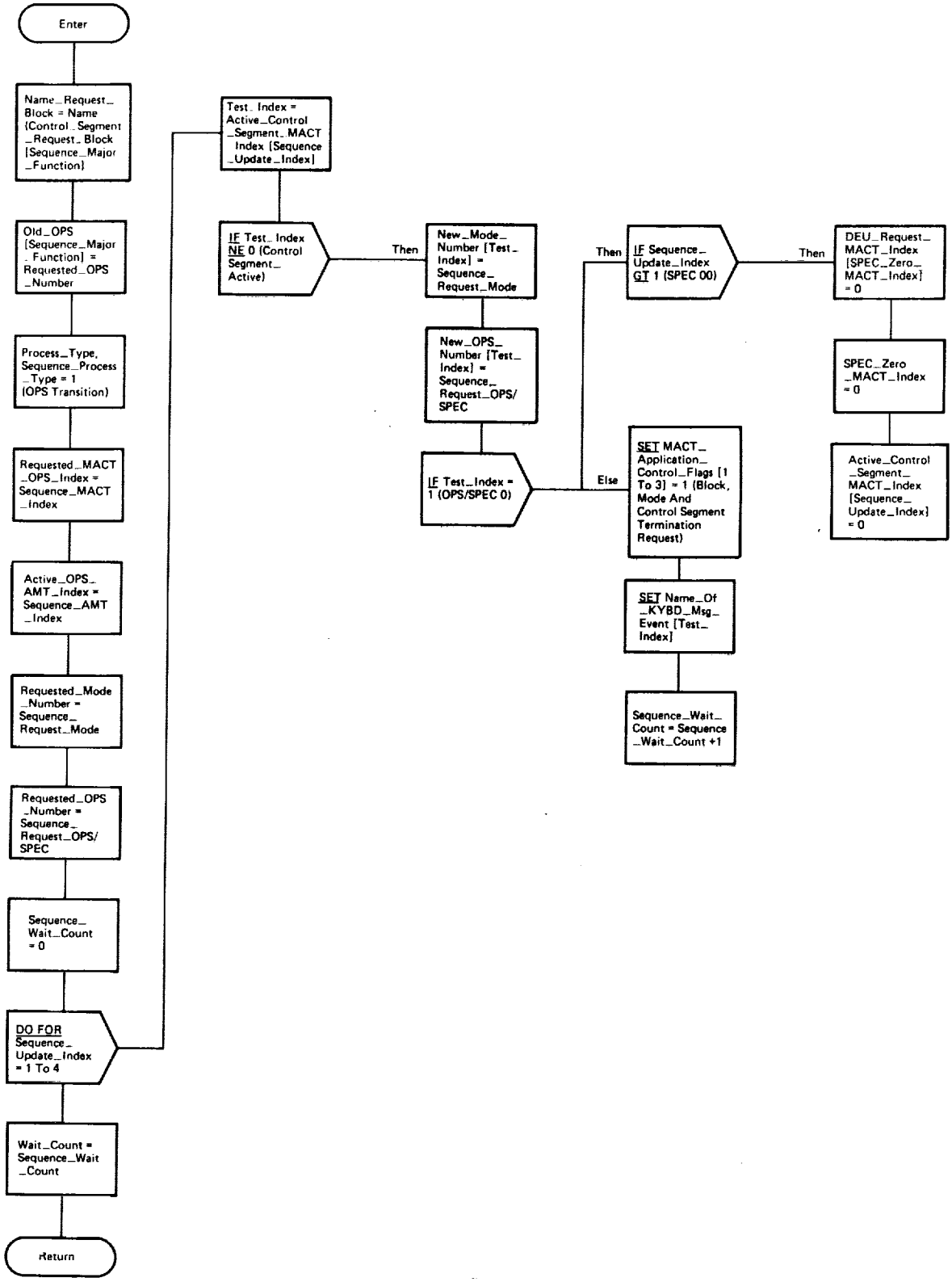


Figure 3.2.3.1-15. Sequence_Request_Processor Control_Segment_Terminator (560.14)

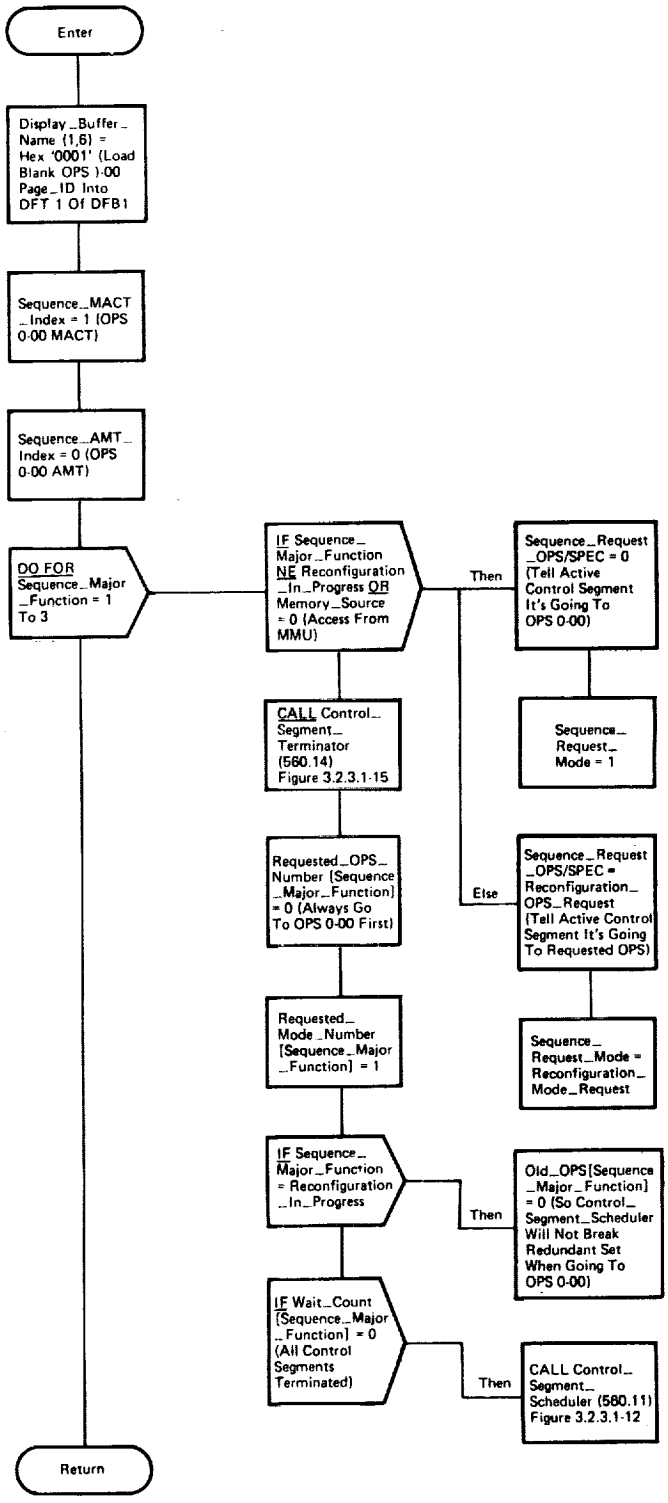


Figure 3.2.3.1-16. Sequence_Request_Processor
Sequence_Reconfiguration_Pre-Processor (560.15)

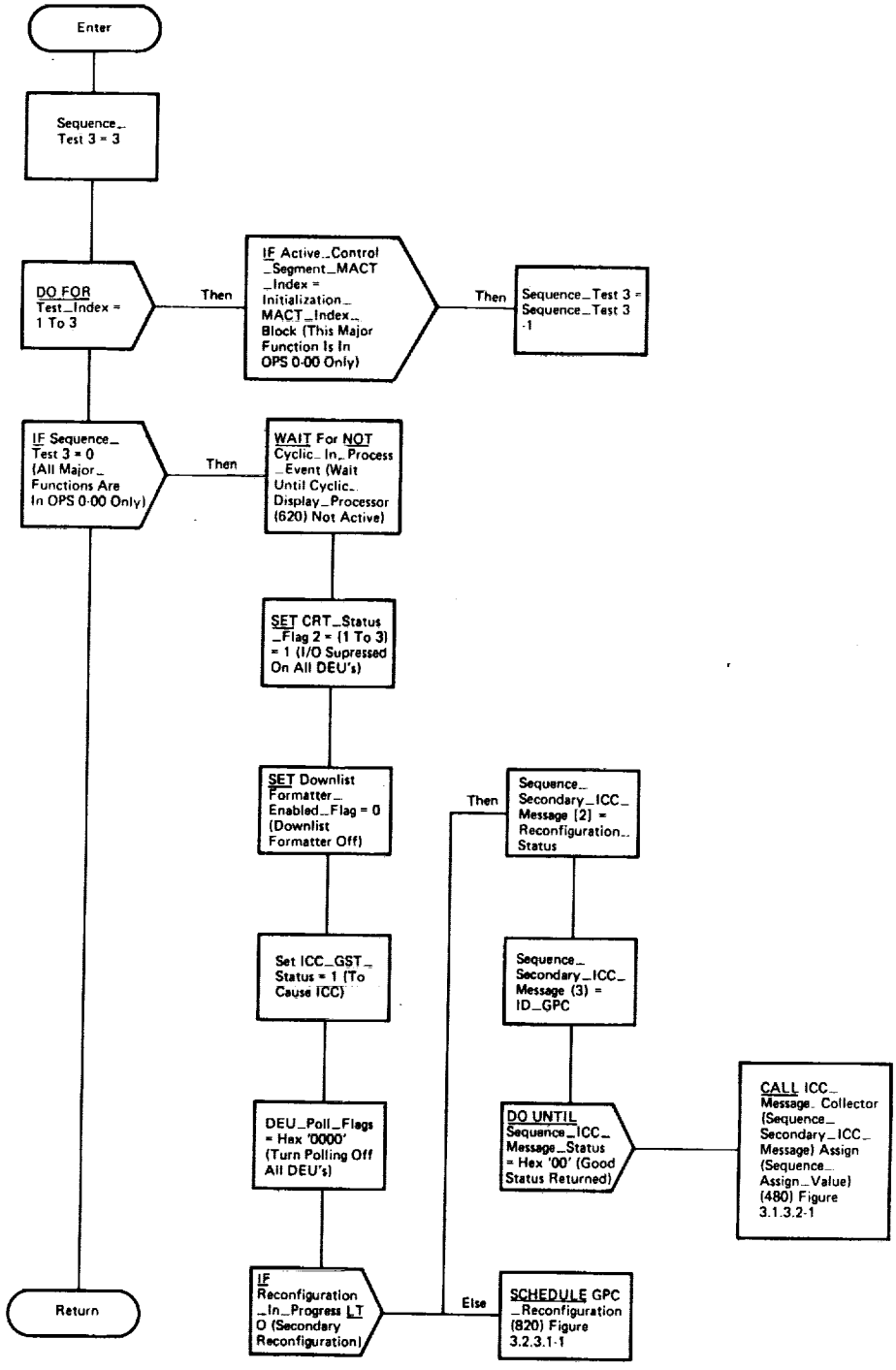


Figure 3.2.3.1-17. Sequence_Request_Processor Sequence_Reconfiguration_Processor (560.16)

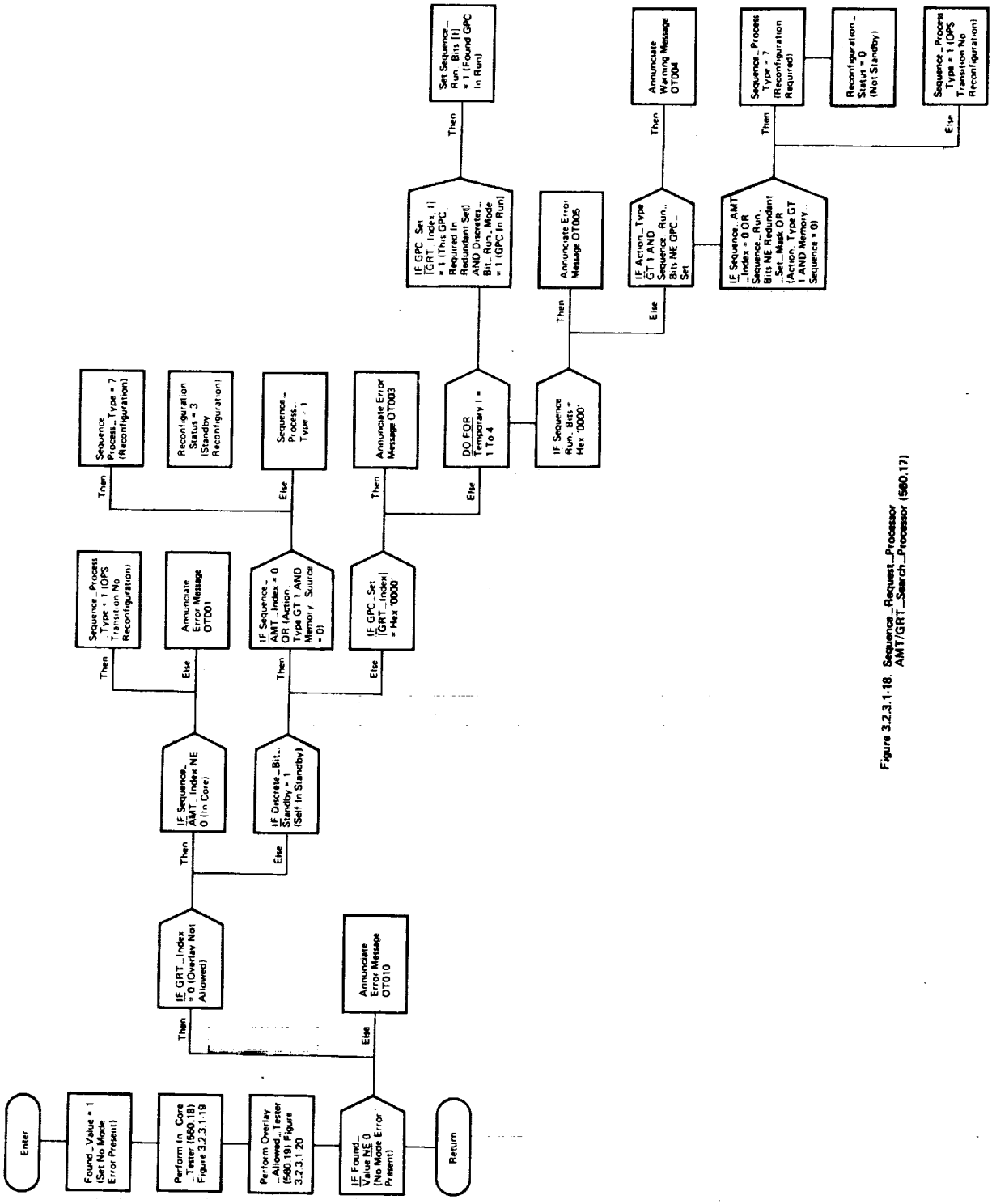


Figure 3.2.3.1-18. Sequence_Request_Processor AMT/GRT_Search_Processor (560.17)

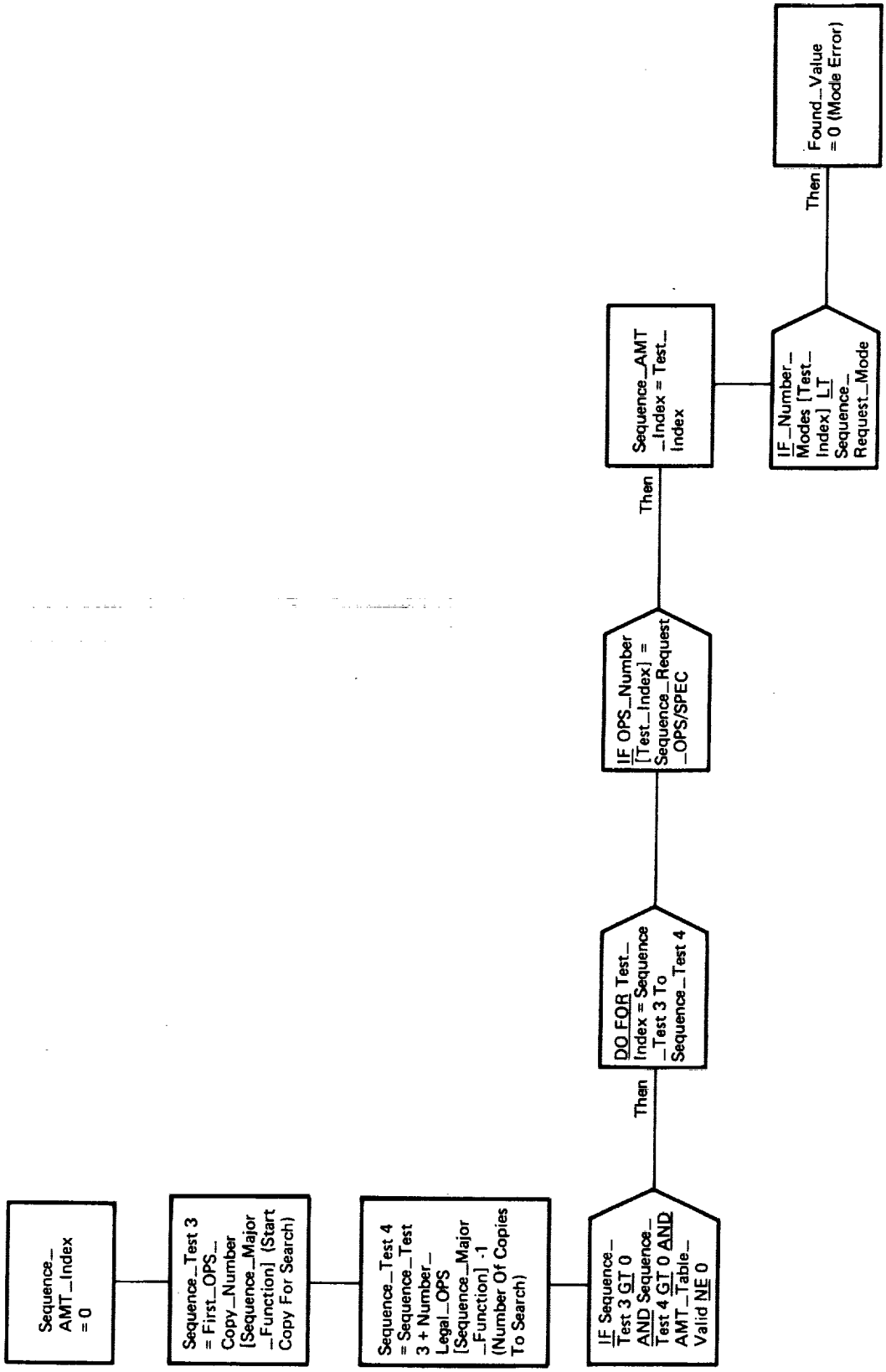


Figure 3.2.3.1-19. Sequence_Request_Processor In_Core_Tester (560.18)

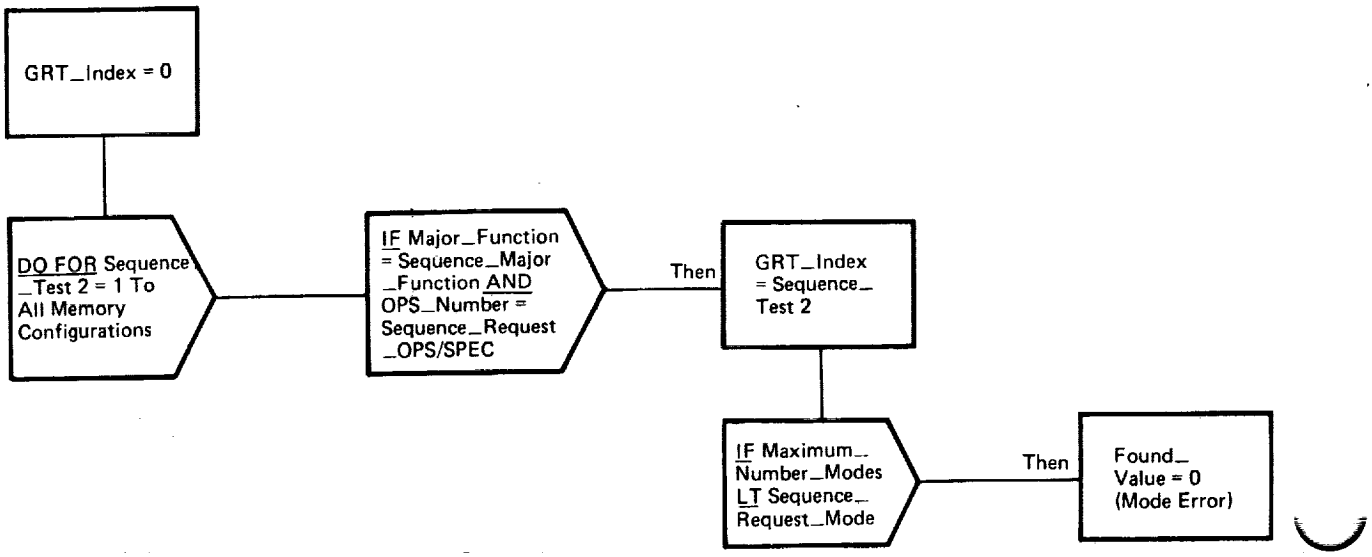


Figure 3.2.3.1-20. Sequence_Request_Processor Overlay_Allowed_Tester (560.19)

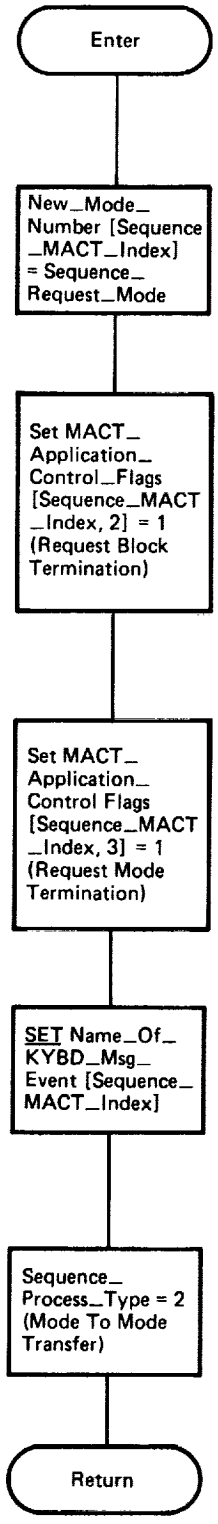


Figure 3.2.3.1-21. Sequence_Request_Processor Mode_To_Mode_Processor (560.20)

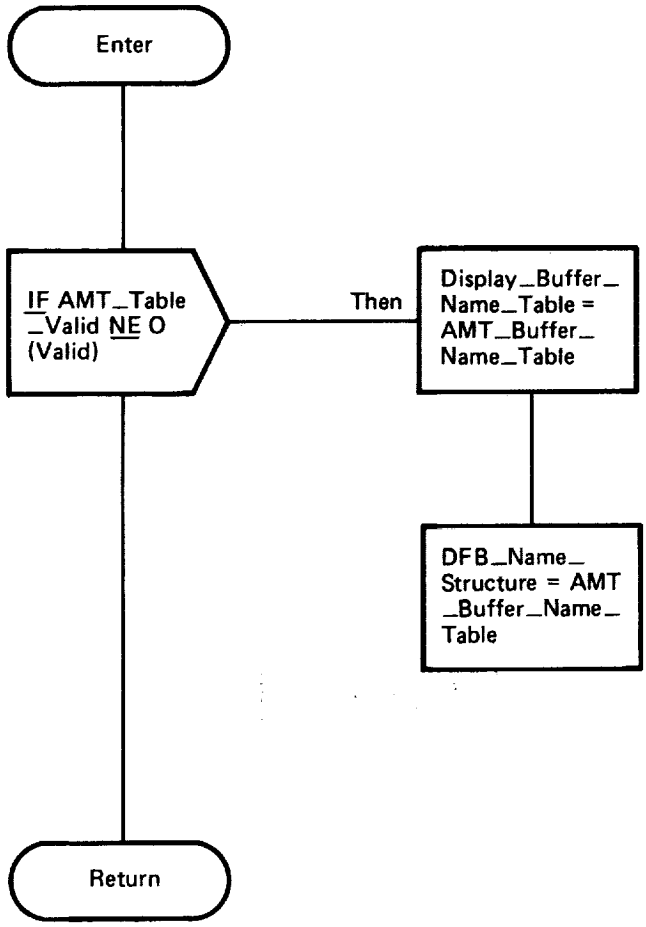


Figure 3.2.3.1-22. Sequence_Request_Processor Buffer_Address_Mover (560.21)

3.3 OUTPUT MESSAGE PROCESSING AND COORDINATION

The Output Message Processing and Coordination software provides flight software with the capability of presenting displays, messages, lights and alarms to the crewman through the MCDS and of communicating with the ground by way of the downlist and launch data bus. It has four major subdivisions. See Figure 3.3-1 for a hierarchial diagram of Output Message Processing and Coordination.

- CRT Interface generates the Format Control Words (FCWs) which produce the CRT displays. The FCWs are regenerated on a cyclic basis to provide updated display parameters. The CRT Interface also supports the error message line and the tutorial message line on the displays.

- The Downlist Interface is the means through which the various applications and system services provide data to the ground via the PCM Master Unit (PCMMU). The Downlist software collects, buffers, formats and transfers the data to the PCMMU.

- The LDB Interface has the responsibility of coordinating messages sent to the ground via the launch data bus. It provides the data formatting and buffering required to accomplish this communication.

- The Annunciation Support software provides the interface for flight software to drive the Caution and Warning indicator alarm, the alert light and the alert tone. It also supports the Fault Summary Page Display on the MCDS which contains a history of software-detected fault messages.

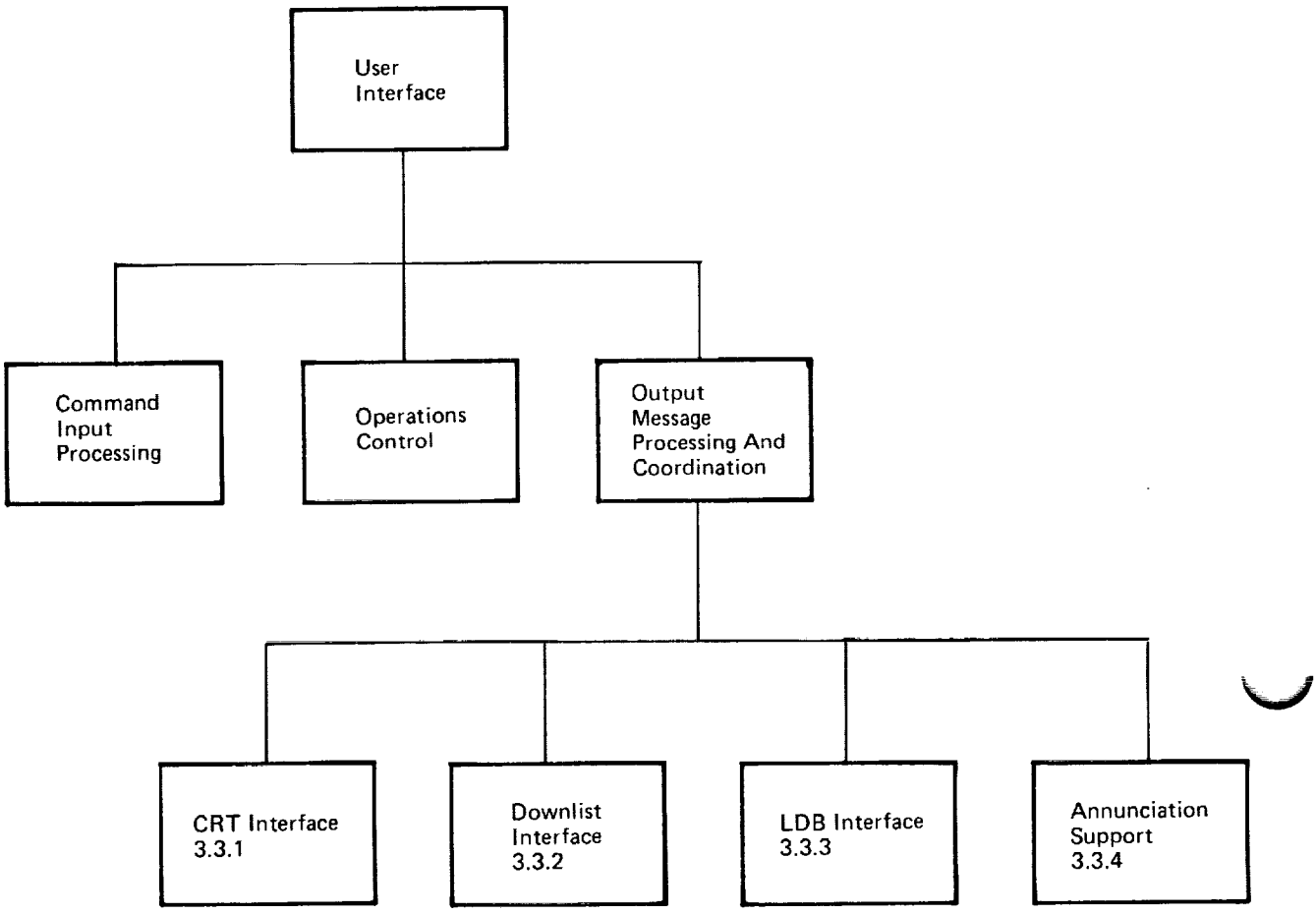


Figure 3.3-1. Output Message Processing And Coordination Hierarchial Diagram

3.3.1 CRT Interface

CRT Interface provides for the support of the MCDS CRT by generating and updating displays. A display consists of background CRT presentations which remain constant throughout the display's update and variable CRT presentations which may change during update. Both background and variable data updates are supported, as well as message line presentations. Figure 3.3.1-1 presents a hierarchial diagram of the CRT Interface.

a. New_Display_Processor provides for the generation of the background presentation. It is an internal procedure in the User-Interface_Control_Supervisor.

b. Variable Data Processing provides for the generation of the dynamic portions of the display. It is comprised of the following:

   - o Cyclic_Display_Processor - Provides for updating the message and tutorial lines.
   - o Data_Formatter - Formats and interprets user inputs for display on CRT.
   - o Data_Conversion - Converts and formats time and other numerical data into displayable form for the CRT.

   In addition, DEU_I/O_Management is used to output data to the CRT and Header_Builder builds the header information for display on the CRT. All programs mentioned are internal procedures to Cyclic_Display_Processor.

c. Message_Line_Support_Functions formats tutorial messages for the tutorial message line of the CRT. FCW_Builder is an internal procedure.

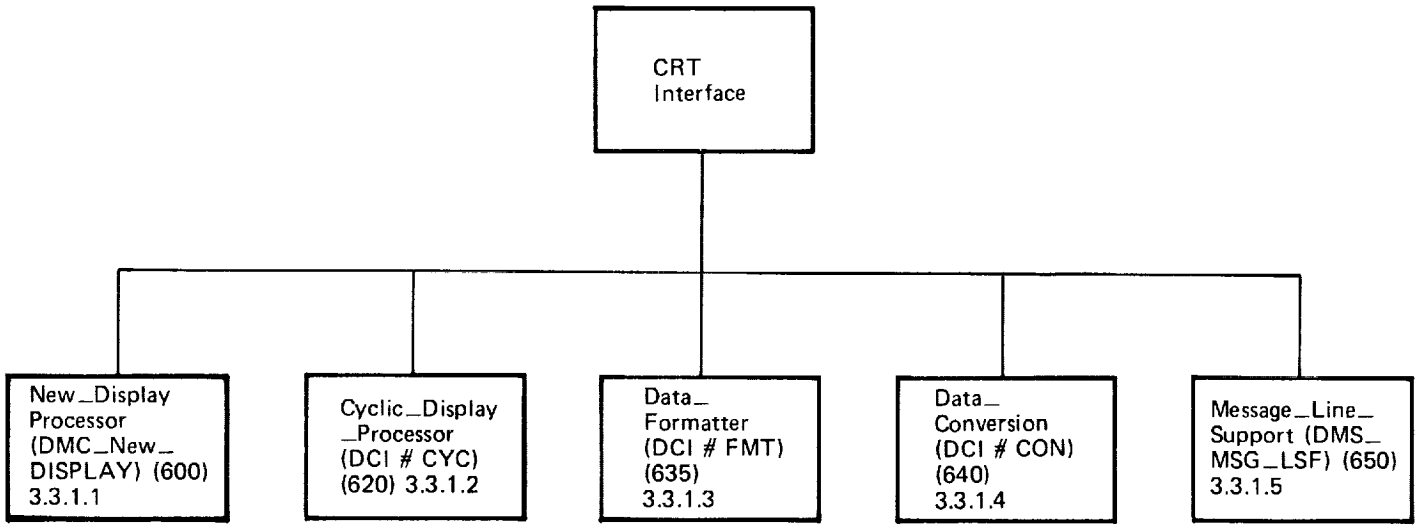


Figure 3.3.1-1. CRT Interface Hierarchy

3.3.1.1 New_Display_Processor (DMC_NEW_DISPLAY) (600)

This module does I/O to the DEUs with background FCWs and resets the scratch pad line.

a. Control Interface

1. CALL DMC_NEW_DISPLAY
2. CALLed by (520) (MCDS_Display_Coordination) (DMC_DISPLAY)

b. Input - See Table 3.3.1.1-1.

c. Process Description - This module when called will issue I/O requests to stop updating the variable portion of the current display, background FCWs for the new display and resetting of the scratch pad line.

The control flow for this module is presented in Figure 3.3.1.1-1.

d. Outputs - See Table 3.3.1.1-2.

e. Module References - None

f. Module Attributes - Internal Procedure

g. Template References - None

h. Error Handling - If a bad return_status is received from FCOS concerning an I/O transaction, the return status indicator is set for MCDS_Display_Coordination.

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.3.1.1.-1

NAME New Display Processor (DMC_NEW_DISPLAY)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	CRT_Status_Flags	B010.44	0	400 600.650	600 620.680	CRT_STATUS_FLAG_4			
2	Return Status Flag	L088		405 500	405 600	DMC_STATUS			

TABLE 3.3.1.1-2

DEVICE	DEVICE ID	OP CODE	SVC NO.	SYNC TYPE	COMFAULT STATUS	ICC TYPE	TIME TAG	WAIT	PROT.	STATUS	INPUT OUT-PUT	MAJ. FUNC. ID	I/O PRIOR-ITY	WORD COUNT	BUFFER	EVENT	BUS NOMINAL
DEU #1	5	1	24	RS	YES	NO	NO	YES	NO	YES	0	NO	160	Variable	DMC_DEU_BUF		6
	5	6	24	RS	NO	NO	NO	NO	NO	NO	0	NO	160	3	DMC_DEU_BUF		6
DEU #2	6	1	24	RS	YES	NO	NO	YES	NO	YES	0	NO	160	Variable	DMC_DEU_BUF		7
	6	6	24	RS	NO	NO	NO	NO	NO	NO	0	NO	160	3	DMC_DEU_BUF		7
DEU #3	7	1	24	RS	YES	NO	NO	YES	NO	YES	0	NO	160	Variable	DMC_DEU_BUF		8
	7	6	24	RS	NO	NO	NO	NO	NO	NO	0	NO	160	3	DMC_DEU_BUF		8

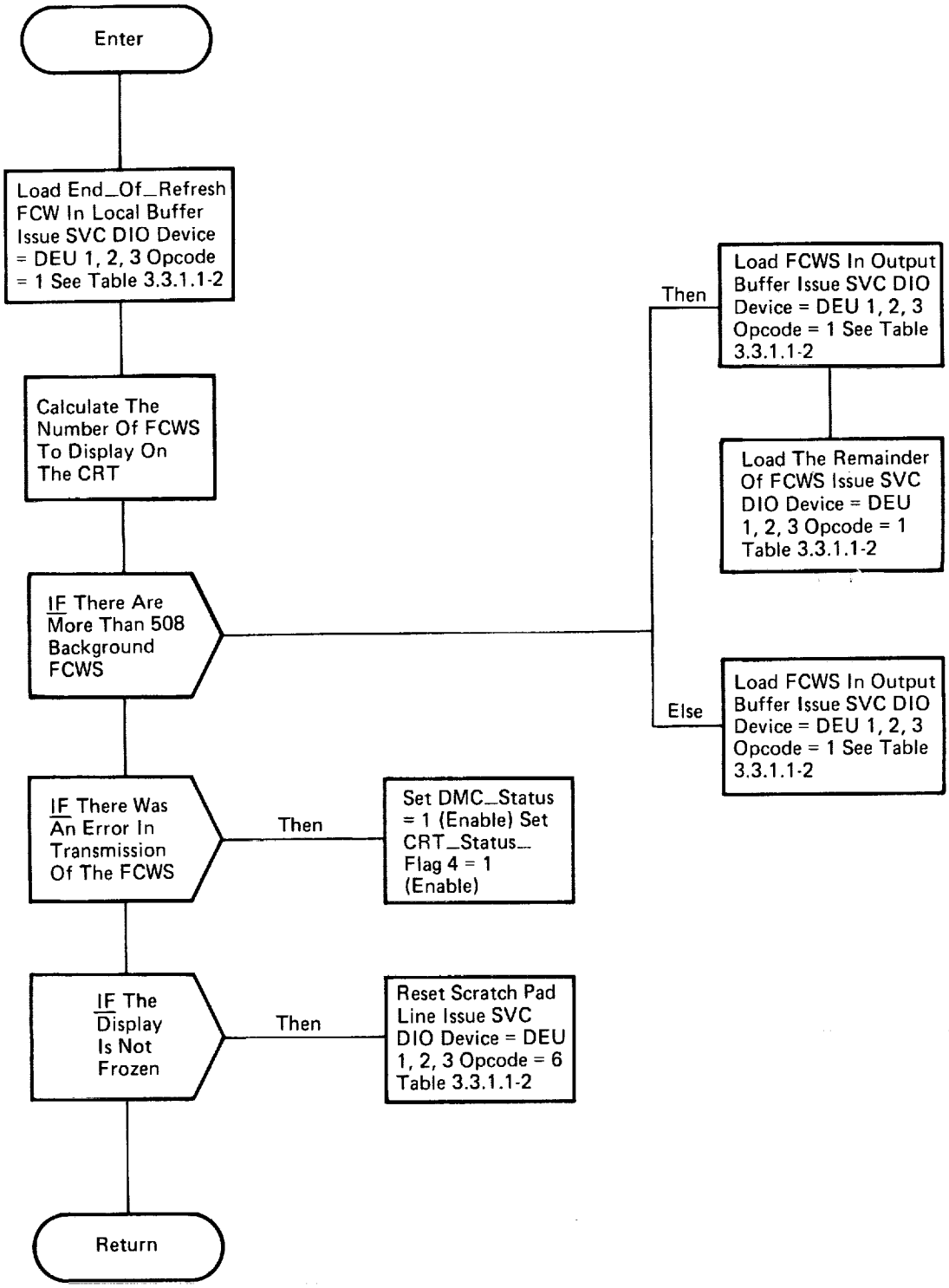


Figure 3.3.1.1–1. New_Display_Processor (DMC_NEW_DISPLAY)

3.3.1.2 Cyclic_Display_Processor (DCI#CYC) (620)

Provides the logic necessary to control the update process. It is responsible for the variable portion of all displays on CRTs, does the I/O and time conversion for Annunciation, and displays MET on each CRT.

a. Control Interface -

1. SCHEDULE CYCLIC_DISPLAY_PROCESSOR priority (PRIO_DCI) repeat every TIME_DCI.

2. SCHEDULEd by (710) GPC_Startup (AIR_GPC_STARTUP)

b. Input - See Table 3.3.1.2-1.

c. Process Description - This module checks the Cyclic_Display_Initialize_Flag to see if normal reinitialization has occurred. If so, several internal data items are reset. The data items are as follows:

- CRT_Next_Dynamic_Fill
- CYCLIC_LOGIC_CONTROL_FLAG
- CRT_Feature_CONTROL_WD1
- CRT_Feature_CONTROL_WD2
- CRT_Feature_CONTROL_WD3
- CYCLIC_OUTPUT_BUFFER

The module then goes into a loop from 1 to Cyclic_Number_DEUS. In this loop DEU_Update_Rate is used to determine the current DEU_Update_Rate, it is calculated by decrementing DEU_Update_Rate until equal to 0 at which time it is reset to 16 and CRT_Status_Flag 9 is disabled indicating display has not been processed. If CRT_Status_Flag6 enabled, MAT_Major_Function positive, CRT_Status_Flag2 disabled, CRT_Status_Flag 7 disabled, and CRT_Status_Flag10 disabled then, if CRT_Status_Flag1 enabled pick up fault_Message_Time and place in Cyclic_Time_Conversion_DDT. Call Data_Conversion to convert Fault_Message_Line_Time into displayable format for output to DEU. After returning from Data-Conversion, format time in Fault_Message_Line_Time in the following format: hh:mm:ss. Reset CRT_Status_Flag11 disabled.

Reset CRT_Status_Flag6 disabled and issue SVC DIO. If I/O_Error_Return_Buffer not zero (disabled) set CRT_Status_Flag6 enabled, increment to next copy of MCDS_Allocation_Table and end loop.

If DEU number one DEU_Update_Rate indicates that this is a once per second entry and CRT_Status_Flag2 enabled the Met_SVC is issued to get current met. The met is stored in Cyclic_Time_Conversion_DDT along

with the address of the met save area. Data_Conversion is then called to format met time.

A loop is initiated to perform processing of display format tables, for each active DEU, scanning from 1 to Cylic_Number_Deus. If Mat_Major_Function_Setting greater than zero, CRT_Status_Flag2 disabled, CRT_Status_Flag9 disabled and DEU_Display_Level greater than zero calculate DFT address and store in Cyclic_DFT_Save_Area. Store DEU number in Cyclic_DEU_Save-Area. If CRT_Status_Flag1 disabled or if CRT_Status_Flag3 enabled initialize Cyclic_Dynamic_Branch_Address. If DEU_Update_Rate setting indicate once per second entry or if CRT_Status_Flag3 enabled call Header_Builder else change CRT_Next_Dynamic_fill to X'0433' for fill with no header data.

Call Data-Formatter to process each dynamic data table entry associated with the DFT for this DEU. Set CRT_Status_Flag3 disabled and call DEU_I/O_Management to output dynamic data to DEU. Increment to next copy of the MCDS_Allocation_Table and end loop. The control flow for this module is presented in Figure 3.3.1.2-1.

d. Outputs - See Table 3.3.1.2-1.

e. Module References -

1. (630) Header_Builder (DCIBHDR) is CALLed
2. (635) Data_Formatter (DCI#FMT) is CALLed
3. (625) DEU_I/O_Management (DCIBIO) is CALLed
4. (640) Data_Conversion (DCI#CON) is CALLed

f. Module Attributes - Program

g. Template References - None

h. Error Handling - If an error is detected during I/O for message line processing CRT_Status_Flag6 is re-enabled so that the message is re-issued on the next cycle of Cyclic_Display_Processor.

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.3.1.2-1

NAME CYCLIC_DISPLAY_PROCESSOR (DCI#CYC)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	CYCLIC_DISPLAY_INITIALIZE_FLAG	I300.03	IOZ	620 710	620				
2	CRT_INIT_DYNAMIC_FILL	L062	L	620	620,675	DCIBLOCH			
3	CYCLIC_LOGIC_CONTROL_FLAG	L063	L	620	620,625 640,675	CLOCFLGS			
4	CRT_FEATURE_CONTROL_WD1	L065	L	620	620	CLOCFS01			
5	CRT_FEATURE_CONTROL_WD2	L066	L	620	620	CLOCFS02			
6	CRT_FEATURE_CONTROL_WD3	L067	L	620	620	CLOCFS03			
7	CYCLIC_OUTPUT_BUFFER	L068	L	620	620	CLOCOUT			
8	CYCLIC_NUMBER_DEUS	L0069	L	620	620	DCIS#DEU			
9	CRT_NEXT_DYNAMIC_FILL	1070	L	620	620,625	CLOCNXTA			
10	CRT_STATUS_FLAG9	B010.47	IOZ	620	620	CDMB_MAT_CRT_STATUS/CMATFLAG			
11	DEU_UPDATE_RATE	B010.20	IOZ	500,620 820,880	620,625	CDMV_MAT_DEU_RATE/CMATRATE			
12	CRT_STATUS_FLAG6	B010.45	IOZ	620	620	CDMB_MAT_CRT_STATUS/CMATFLAG			
13	MAT_MAJOR_FUNCTION_SETTING	B010.10	IZ	505,520	505,560 600,620	CDMV_MAT_MF/CMATMF			
14	CRT_STATUS_FLAG2	B010.42	IZ	400 505	505 620	CDMB_MAT_CRT_STATUS/CMATFLAG			
15	CRT_STATUS_FLAG7	B010.46	IZ	650	620	CDMB_MAT_CRT_STATUS/CMATFLAG			
16	CRT_STATUS_FLAG10	B010.48	IZ	650	620	CDMB_MAT_CRT_STATUS/CMATFLAG			
17	CRT_STATUS_FLAG11	B010.49	IOZ	650	620	CDMB_MAT_CRT_STATUS/CMATFLAG			
18	FAULT_MESSAGE_LINE_TIME	B010.75	IZ	680	620	CDMV_MAT_TIME/CMATIME			
19	CYCLIC_TIME_CONVERSION_DDT	L071	L	620	620	CLOCDDTE			
20	I/O_ERROR_RETURN_BUFFER	B010.50	IOZ	244	620	CDMV_MAT_ERROR_RETURN/CMATERR			

DATA TABLE 3.3.1.2-1 (cont'd)

NAME CYCLIC_DISPLAY_PROCESSOR (DCI#CYC)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
21	CRT_STATUS_FLAG2	B010.42	IOZ	400,505	505,620	CDMB MAT_CRT_STAT/CMATFLAG			
22	MET_BUFFER	L072	L	620,630	620,630	CLOCTMES			
23	CYCLIC_DFT_SAVE_AREA	L073	L	620	620	CLOCDFT			
24	CYCLIC_DEU_SAVE_AREA	L074	L	620	620	CLOCDEU			
25	CYCLIC_DYNAMIC_BRANCH_ADDRESS	L075	L	620	620,625	CLOCBRAN			

TABLE 3.3.1.2-2 CYCLIC_DISPLAY_PROCESSOR (DCI#CYC) (6.0)

DEVICE	DEVICE ID	OP CODE	SVC NO.	SYNC TYPE	COMFAULT STATUS	ICC TYPE	TIME TAG	WAIT	PROT.	STATUS	INPUT OUT-PUT	MAJ. FUNC. ID	I/O PRIOR-ITY	WORD COUNT	BUFFER	EVENT	BUS NOMINAL
DEU1	5	1	24	RS	NO	NO	NO	YES	PROT.	YES	0	NO	160	67	TUTORIAL_ MESSAGE_ LINE_BUFFER		6
DEU2	6	1	24	RS	NO	NO	NO	YES	PROT.	YES	0	NO	160	67	TUTORIAL_ MESSAGE_ LINE_BUFFER		7
DEU3	7	1	24	RS	NO	NO	NO	YES	PROT.	YES	0	NO	160	67	TUTORIAL_ MESSAGE_ LINE_BUFFER		8

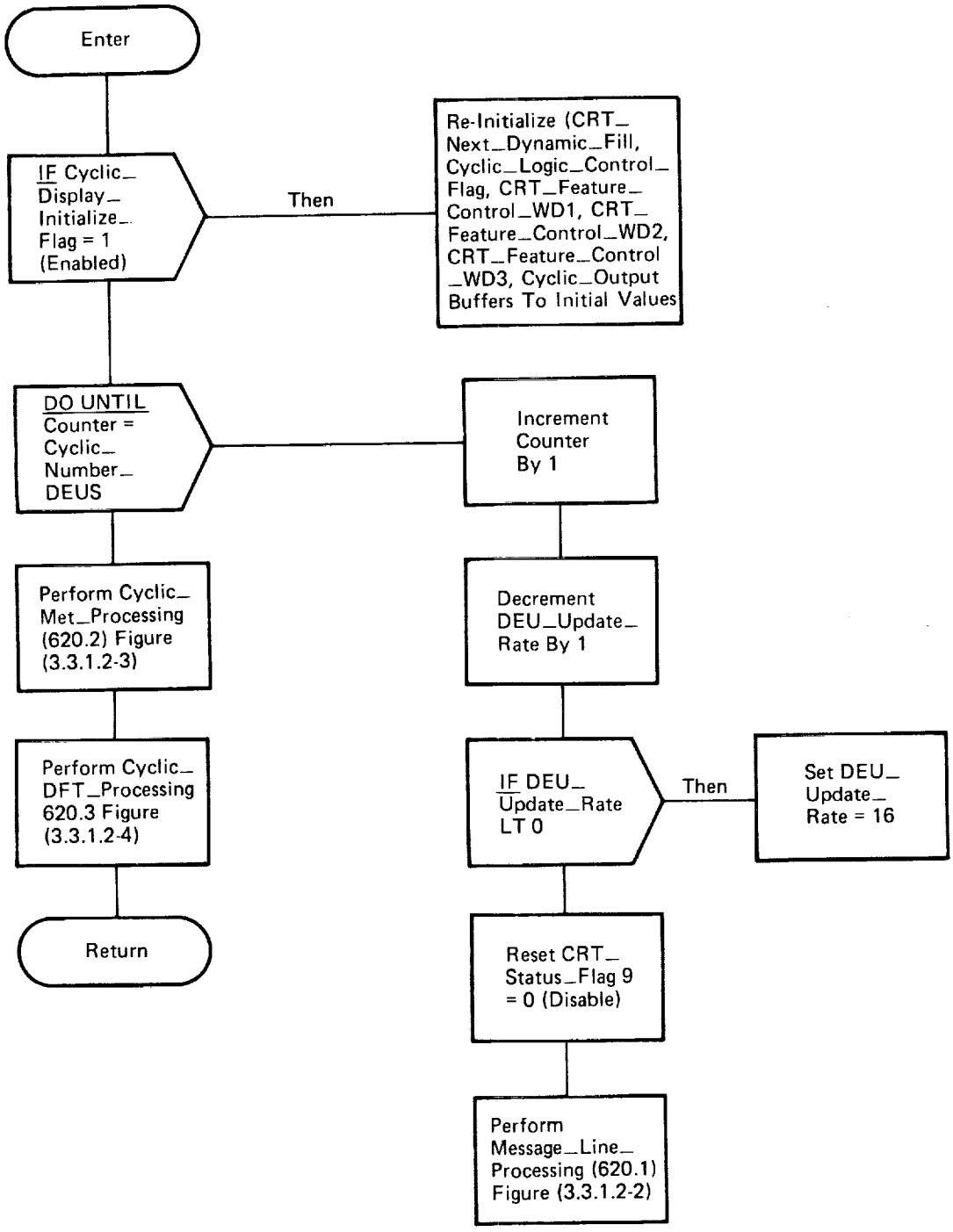


Figure 3.3.1.2-1. Cyclic_Display_Processor (DCI#CYC)

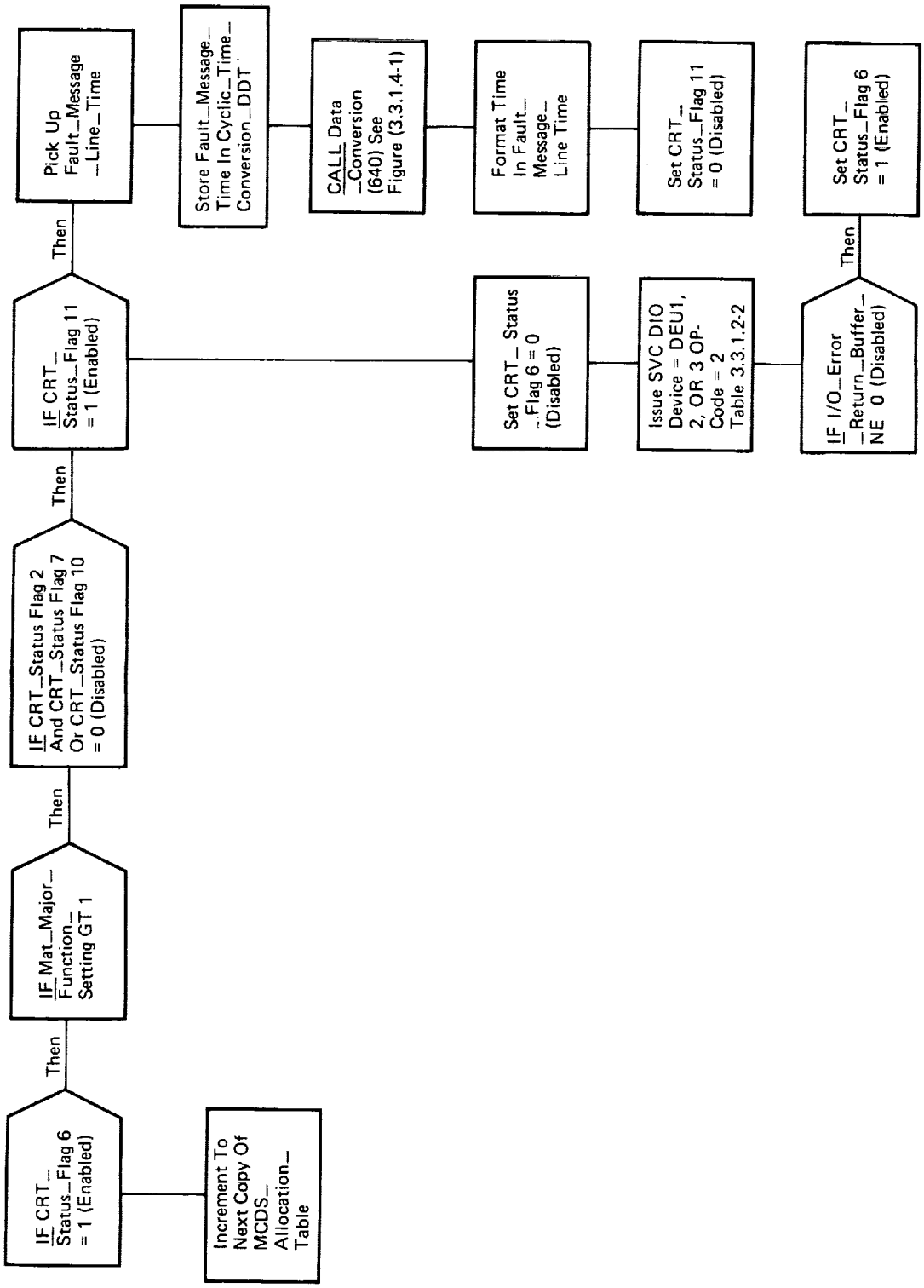


Figure 3.3.1.2-2. Cyclic_Display_Processor Message_Line_Processing (620.1)

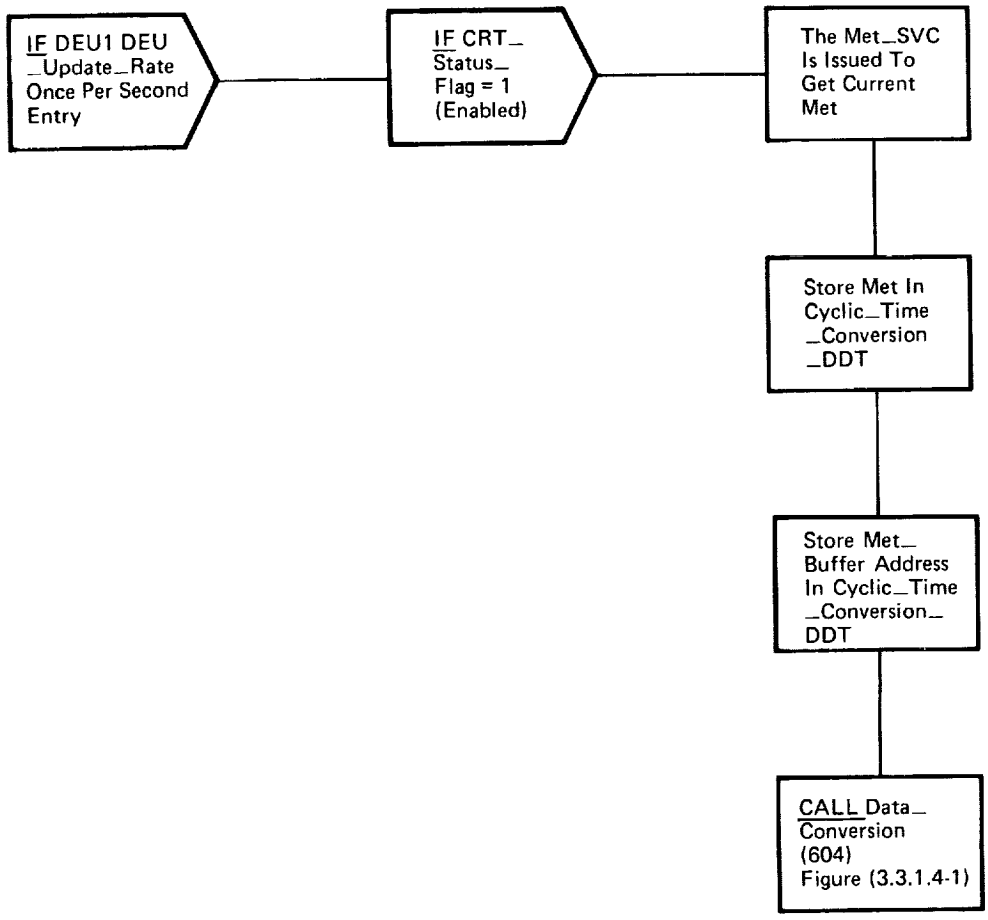


Figure 3.3.1.2-3. Cyclic_Display_Processor Cyclic_Met_Processing (620.2)

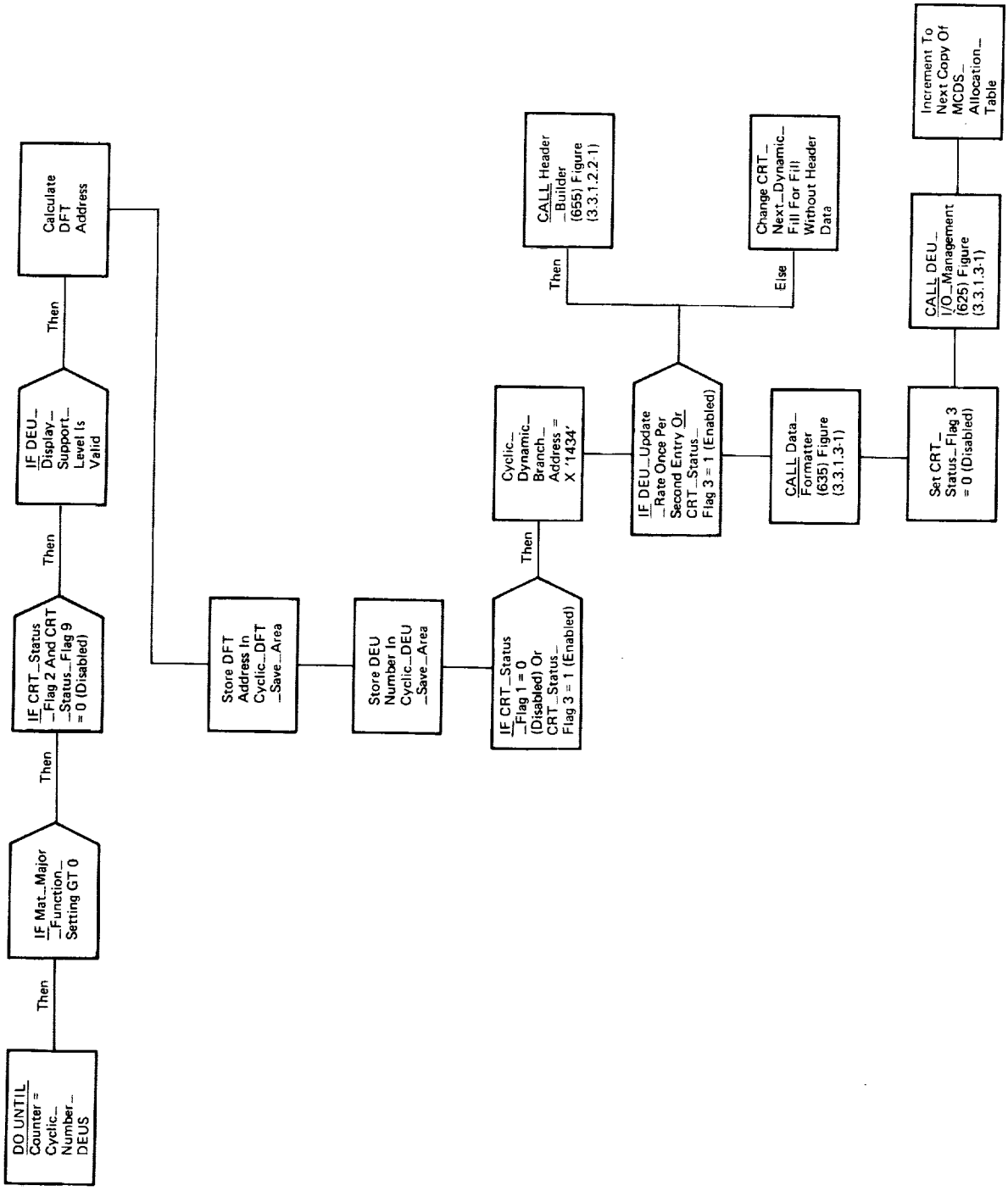


Figure 3.3.1.2-4. Cyclic_Display_Processor Cyclic_DFT_Processing (620.3)

3.3.1.2.1 DEU_I/O_Management (DCIBIO) (625)

Provides logic to output data to the DEU's.

a. Control Interface -

1. Call DEU_I/O_Management

2. Called by (620) Cylic_Display_Processor(DCI#CYC)

b. Inputs - See Table 3.3.1.2.1-1

c. Process Description - The Module picks up a pointer to the cyclic_Output_Buffer and stores an End of Refresh Feature control word (FCW) in the Buffer, if the cyclic_Logic_Control_flag indicates that this is the last fill for a display or if the CRT_status_ FLAG3 is enabled else the cyclic_Output_Buffer, pointer is decremented. An End Of Refresh FCW indicates to the DEU's symbol generator that all of the symbols have been processed for a refresh cycle. A calculation of the amount of data is made and stored in FCW's Parameter list along with the DEU address and address of Data to be sent to the screen. The data is then transmitted to the DEU via a DEU fill SVC (SVC DIO).

The Module checks the next highest DEU to see if same display is to be processed and CRT_Status_flags equal zero. If so, a check is made on CRT_Status_Flag1 and CRT_Status_Flag2 to see if they are disabled and change the DEU_Update_rate to that of the processed display if test is true. If once per second entry for DEU and Cyclic_Logic_Control_flag equal x '0008' disabled call Header_Builder to have header information placed in buffer for this DEU. The Device ID is stored and the displayed is shipped via the FCOS DEU fill SVC (SVC DIO) and CRT_Status_Flag9 is enabled.

If cyclic_DEU_Save_area high order bit is enabled (i.e. Two or more fill Buffers needed to update display) add number of FCWS outputed to DEU Fill address and if CRT_Status_ Flag3 enabled decrement sum by one. Else, point to start of Dynamic area in DEU and re-initialize cyclic_dynamic_Branch_Address. The results obtained by the IF processing is stored in CRT_Next_Dynamic_fill.

d. Outputs - See Table 3.3.1.2.1-1

e. Module References -

1. (630) Header_Builder (DCIBHDR) is CALLed.

f. Module Attributes - Internal Procedure

g. Template References - None

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.3.1.2.1-1

NAME DEU_I/O_MANAGEMENT (DCIBIO)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	CYCLIC_OUTPUT_BUFFER	L068	L	620	620 625	CLOCOUT			
2	CYCLIC_LOGIC_CONTROL_FLAG	L063	L	620	620, 625 640, 675	CLOCFLGS			
3	CRT_STATUS_FLAG3	B010.43	IOZ	400	625	CDMB_MAT_CRT_ STAT/CMATFLAG			
4	WORD_COUNT_TRANSFERRED	L261	O	625	625	CLOCCNT			
5	DEU_FILL_ADDRESS	L262	O	625	625				
6	DATA_ADDRESS	L263	O	625	625	CLOCADDR			
7	DEU_DEVICE_ID	L264	O	625	625	CLOCDEU			
8	DEU_UPDATE_RATE	B010.20	IOZ	500,620 820,880	620 625	CDMV_MAT_DEU_RATE /CMATRATE			
10	CYCLIC_DEU_SAVE_AREA	L074	L	620	620	CLOCDEU			
11	CRT_NEXT_DYNAMIC_FILL	L070	L	620	620 625	CLOCNXTA			
12	CYCLIC_DYNAMIC_BRANCH_ADDRESS	L075	L	620	620 625	CLOCBRAN			
13	CRT_STATUS_ FLAG1	B010.41	IZ	400,505 600	505,620 625	CDMB_MAT_CRT STAT/CMATFLAG			
14	CRT_STATUS_ FLAG2	B010.42	IZ	400,505	505,620	CMDB_MAT_CRT STAT/CMATFLAG			

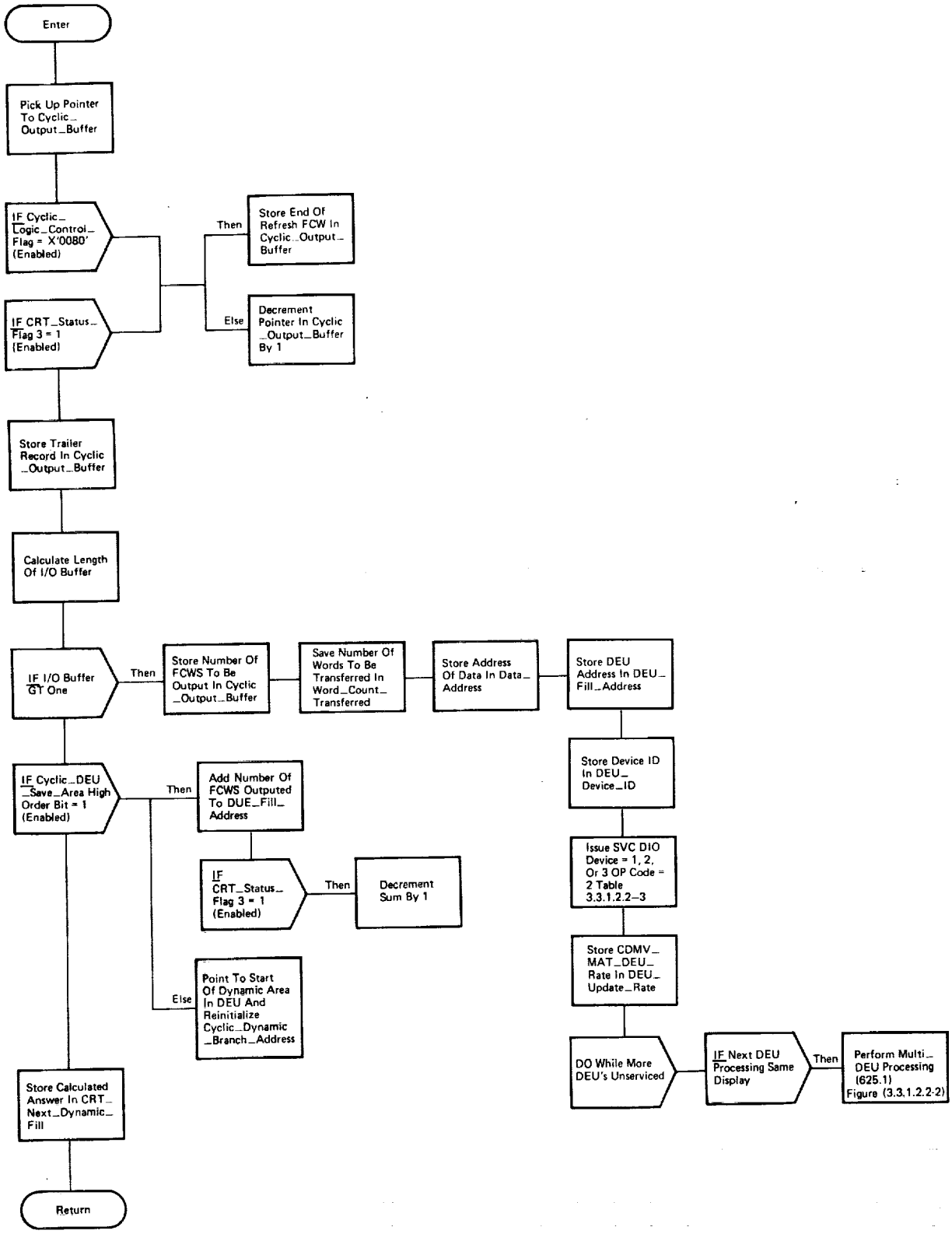


Figure 3.3.1.2.1-1. DEU_I/O_Management (DCIBIO)

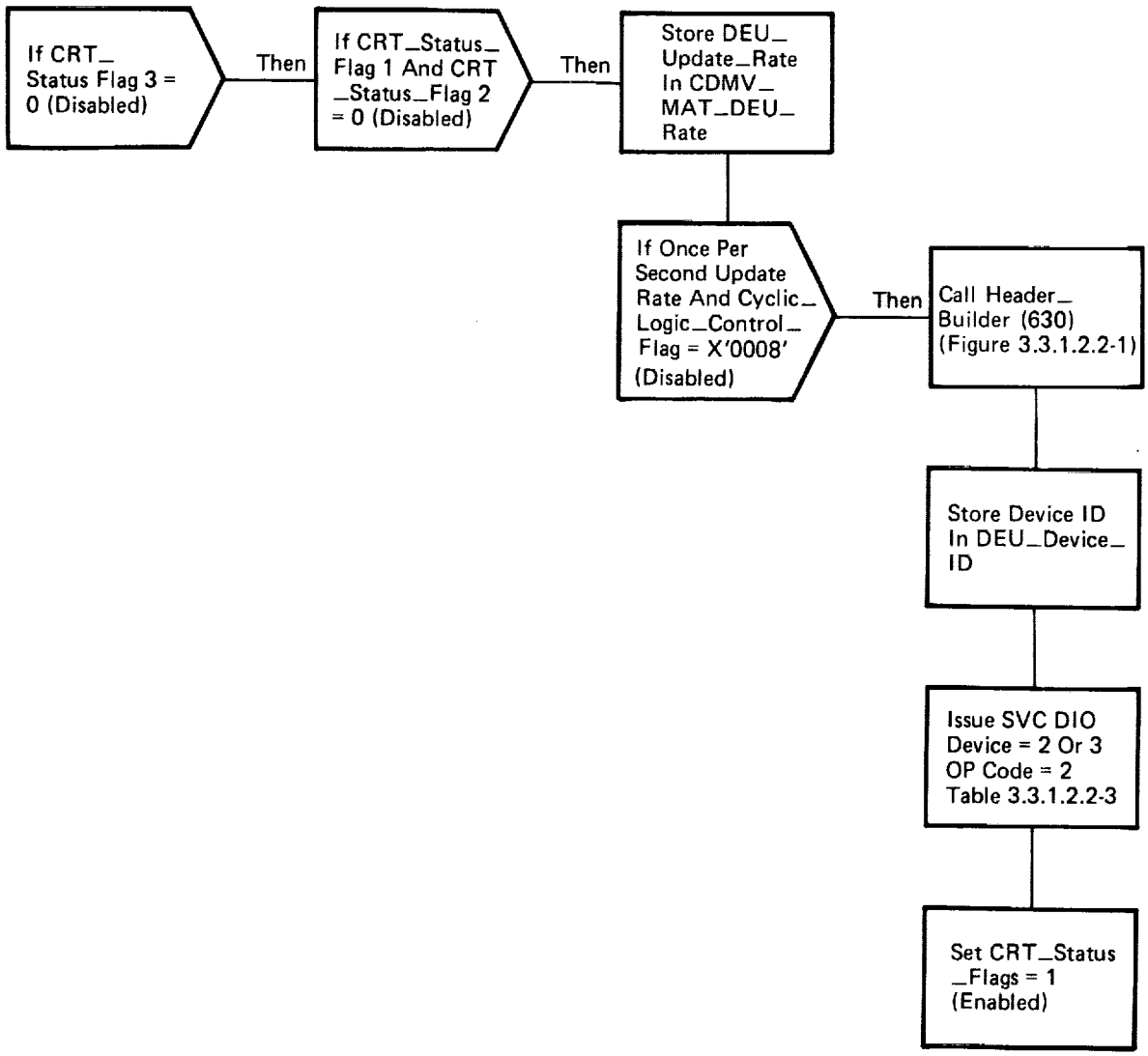


Figure 3.3.1.2.1-2. DEU_I/O_Management Multi_DEU_Processing (625.1)

3.3.1.2.2 Header_Builder (DCIBHDR)(630)

This subroutine provides logic to build the display header containing the OPS_Page_Number, the SPEC_Page_Number, DISP_Page_Number, GPC_ID and the mission elapsed time in days, hours, minutes and seconds.

a. Control Interface -

1. CALL DCIBHDR
2. CALLed by (620) Cyclic_Display_Processor (DCI_CYC)
3. CALLed by (625) Deu_I/O_Management (DCIBIO)

b. Input - See Table 3.3.1.2.2-1.

c. Process Description - The dynamic header information is built in the following manner: First, the x and y position of the OPS_Page_Number to be displayed is obtained from the values stored in OPS_Page_X_Coordinate and OPS_Page_Y_Coordinate. These values are placed in the Output_Buffer to force the cursor on the CRT to move to this location. Next the address of the OPS_Page_Number is obtained and the OPS_Page_Number is sent to Data_Conversion where this OPS level will be converted to displayable format and placed in the Output_Buffer. So far the header has the OPS_Page_Number ready for display, and the Output_Buffer pointer is at the end of the OPS.

If there is a current SPEC display, SPEC_Page_Number greater than zero, the address of this SPEC level is placed in Pseudo_DDT_Entry to pass to Data_Conversion where the SPEC will be converted to displayable format and placed in the Output_Buffer. The pointer is still at the end of the OPS display, so, at this time, a slash fcw is placed between the OPS and SPEC. The converted number returned from Data Conversion includes a sign which, for this type of data, is a blank. The slash overlays the blank and is the only character showing at that location. If there is no SPEC_Page_Number, first a slash is placed after the OPS displayed and then NOOP_FCW is generated for each position that would have had the SPEC displayed. The Output_Buffer is updated to point to the end of the SPEC.

The same test is made to determine if there is a display level to display. If DISP_Page_Number is non-zero, then the address of that number is placed in Psuedo_DDT_Entry, Data_Conversion is called and the DISP level is placed in the Output_Buffer in converted format. The pointer to the Output_Buffer is at the end of the SPEC so a slash is placed in the Output_Buffer and the pointer moves to the end of the DISP. If there is no DISP_Page_Number, a slash is placed after the SPEC, and NOOP_FCW fills the vacant Display Locations. The OPS, SPEC, and DISP are all placed in the Output_Buffer for display.

The cursor on the CRT is moved to the location to display the GPC ID by placing the GPC_ID_X_Coordinate fcw in the Output_Buffer. Next, the GPC_ID fcw itself is placed in the Output_Buffer. The pointer to the Output_Buffer is updated to the x coordinate to display mission elapse time (MET). The MET, already in converted format, is stored in the MET_Buffer. This buffer is moved a half-word at a time into the Output_Buffer until the whole MET is moved. The finished header looks as follows (with numbers added for illustration).

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51

2011/0321/0051                          2  041/06:08:22
  ↑    ↑    ↑                             ↑
 OPS SPEC DISP                         GPCID    MET
                                               (days,hrs,mins,secs)
```

The control flow for this subroutine can be found in Figure 3.3.1.2.2-1.

d. Output - See Table 3.3.1.2.2-1.

e. Module References -

1. (640) Data_Conversion (DCI#CON) is CALLed.

f. Module Attributes - Internal Procedure

g. Template References - None

DATA TABLE 3.3.1.2.2-1

NAME HEADER_BUILDER (DCIBHDR)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	OPS_PAGE_X_COORDINATE	L082		630	630	DCIBXC1			
2	OPS_PAGE_Y_COORDINATE	L083		630	630	DCIBYC1			
3	OPS_PAGE_NUMBER	L084		630	630	CMATPAGE			
4	SPEC_PAGE_NUMBER	L085		630	630	CMATPAGE +1			
5	DISPLAY_PAGE_NUMBER	L086		630	630	CMATPAGE +2			
6	GPC_ID	#001		300	SEE APP. E	TFCMID			
7	OUTPUT BUFFER	L068		620 630	620,625 630	CLOCOUT			
8	NOOP_FCW	L101		630	630	DCIBNOOP			
9	MET_BUFFER	L072		620 630	620 630	CLOCTMES			
10	PSEUDO_DDT_ENTRY	L087		630	630	CLOCDDTP			
11	GPC_ID_X_COORDINATE	L117		630	630	DCIBXC3			

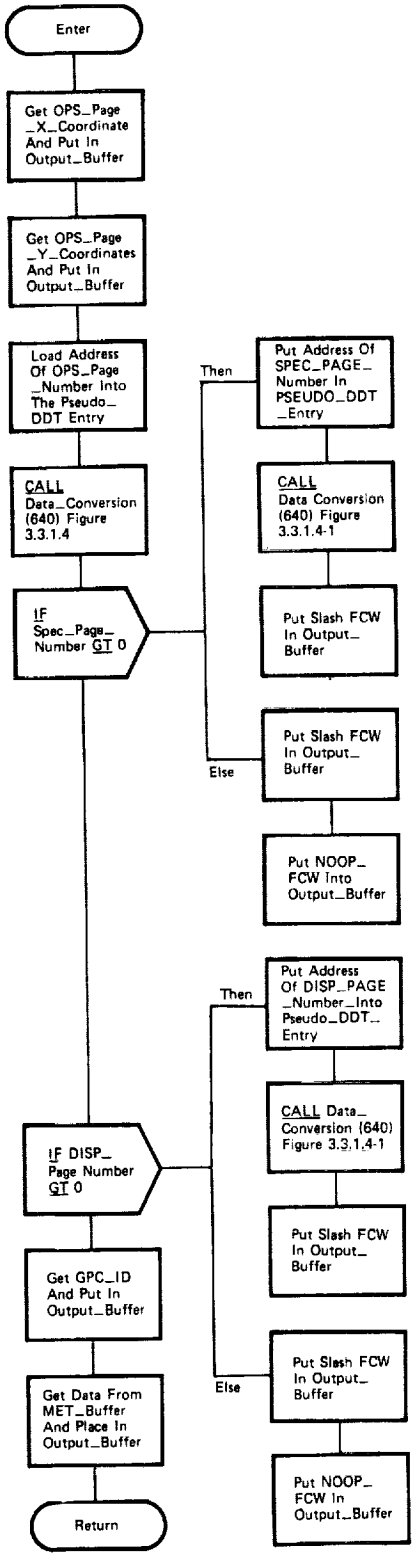


Figure 3.3.1.2.2-1. Header_Builder (DCIBHDR)

3.3.1.3 Data_Formatter (DCI#FMT) (635)

Provides logic to scan the Dynamic Data Table (DDT) and build Format Control Words based on information contained in the DDT entries.

a. Control Interface

1. CALL DATA_FORMATTER

2. CALLed by (620) Cyclic_Display_Processor (DCI#CYC)

b. Input - See Table 3.3.1.3-1

c. Process Description - This module picks up the DFT Address from the Cyclic_DFT_Save_Area and calculates the address of the end of the DFT using the DFT_Length field of the DFT. The DFT_DDT_Pointer is then obtained to provide the address of the first DDT to be processed. The module looks at the second DDT entry to determine if it is an On_Demand_CMD, (the first DDT entry is a Rate_CMD), it so, the On_Demand_Bit_Number is used on the On_Demand_Bit_Address to point to a bit that the application user sets to indicate that the dynamic portions of this display are to be refresh entirely with regards to the Rate_CMD's. If the test bit is enabled the CRT_Status_Flag3 is enabled and processing continues.

The module enters a DO Loop until the previously calculated End of DFT address is reached picking up DDT Op_Codes and performing a DO case on the OP_codes as described below.

1. Bi-Level_Test - this routine picks up the CRT_Feature_Control_WD3 initializes the high intensity bit and stores the CRT_Feature_Control_WD3 in the Cyclic_Output_Buffer If bit in BCT_Adrl_ptr enabled store character "m" in Cyclic_Output_Buffer else if bit in BLT_Adr2_ptr enable store DOWN Arrow else store blank in Cyclic_Output_Buffer.

   The routine stores a backspace FCW and stores the desired character as determined above again in the Cyclic_Output_Buffer. This procedure provides for the double Bright/Double Overwrite capability of this command. The normal intensity bit is set in the CRT_Feature_Control_WD3 and the control word is stored in the Cyclic_Output Buffer to resume normal intensity on the CRT screen. The DDT pointer is indexed to point to the next DDT entry.

2. FCW_Remote_CMD - if FCW_Type disabled pick up pointer to fault_Summary_Page_Buffer and use message number to index to the text associated with this entry. The text is moved into the Cyclic_Output_Buffer two characters at a time. If the Time_Conversion_Bit

has been enabled by the annunciation routine fixed point time is stored in the Cyclic_Time_Conversion_DDT and Data_Conversion is called to convert the data to displayable format. The fixed point time is passed at the end of the text for the displayed message. The routines concludes by incrementing to the next DDT Entry.

3. Remote_Text_CMD - The test bit is obtained and if enabled with the RTC_Control_Bit enabled to ones or if test bit disabled and the RTC_Control_Bit disabled to zeros store CRT_Feature_Control WD3 with high intensity bit enabled in Cyclic_Output_Buffer. The high intensity bit indicates double intensity is to be used on the CRT. If test are not true the Cyclic_Logic_Control_Flag is set equal to X '4000'.

   If test bit is disabled point to second set of text data else point to first set of text Data_Text data is pointed to by RTC_Adr2 with both sets of text residing adjacent to each other and of the same length. The text data is sotred in the Cyclic_Output_Buffer and the Cyclic_Logic_Control_Flag is tested to see if equal to X'4000'. If test is true the CRT_Feature_Control_WD3 is stored in the Cyclic_Output_Buffer with the normal intensity bit enabled. The routine finally encrements to the next DDT entry.

5. Variable_Parameter_CMD - The next DDT entry is checked to see if a status_byte_CMD is the next to be processed. If so, the high order bit is checked to see if initialized and blanks are stored in the Cyclic_Output_Buffer for the length provided by Vparm_FMT1 if test is true, else Data_Conversion is called. Data_Conversion will interpret the Variable_Parameter_CMD and format the output for display (see writeup for Data_Conversion 3.3.1.4). The routine new increments the DDT pointer to the next DDT entry.

10. Remote_Character - A DO LOOP is created until Remote_Character_Length equals zero picking up characters from Remote_Character_Addr. The characters are formatted into format control words (FCW) and stored in the Cyclic_Output_Buffer. Remote_Character_Length is Decremented and Remote_Character_Addr is incremented to the next characters. The loop is re-cycled from this point.

    At exit from the above loop the DDT pointer is incremented to the next DDT entry.

13. Status_Byte_CMD - The CRT_Feature_Control_WD3 is stored in the Cyclic_Output_Buffer with the high intensity bit enabled. The status bit is checked in Status_Byte_Addr and if disabled a blank character is stored in the Cyclic_Output_Buffer else the character is picked up out of the Status_Byte_Char_Table and stored in

the Cyclic_Output_Buffer. Next, the routine stores a backspace FCW in the Cyclic_Output_Buffer and the character is stored in the Cyclic_Output_Buffer again. This procedure is enacted to provide for double overbright/overwrite of status indicators.

The CRT_Feature_Control_WD3 is stored in the Cyclic_Output_Buffer with the normal intensity flag enabled and the DDT pointer is incremented to the next DDT entry.

17. Multiple_Discrete - A loop is invoked on the entries presented in the MDT_Xtable until a minus one is encountered. A discrete is tested and if enabled a work register has a bit set if not the bit is set off with the register than being shifted one position to the left. This procedure allows the program to eventually end up with an index to be used to index into the MDT_Ytable. The next MDT_Xtable_Addr is picked up a recycle of the loop is executed if necessary.

    The index this calculated is now used to point into the MDT_Ytable to obtain and index into the MDTXtble. The index into the MDTXtble is used to point to characters within this table for display purposes. The characters are formatted into FCWS and stored in the Cyclic_Output_Buffer. The routine increments the DDT pointer to the next DDT entry.

18. Test_CMD - The Test_Addr is picked up and a calculation of the test bit is performed by using the Test_Control_Bit. If the test bit is enabled add Test_DDTS_To_Skip to the current DDT pointer.

    The current DDT pointer is incremented to the DDT entry.

20. Immediate_Data - The first FCW to be moved is checked to see if it is one of the three CRT_Feature_Control_Words, if so, the appropriate CRT_Feature_Control_Word is replaced by the new FCW.

    A Do loop is enacted until Number_Of_FCWS equal zero storing the FCW in the Cyclic_Output_Buffer and incrementing the DDT pointer by one.

21. Rate_CMD - The Cyclic_Dynamic_Branch_Address is stored in the Cyclic_Output_Buffer. If Cyclic_Dynamic_Branch_Address plus Rate_FCW count exceeds Cyclic_Output_Buffer length the high order bit of DEU_Device_ID is enabled and DEU_I/O_Management is called to output the buffer to the CRT. Upon return, the high order bit of DEU_Device_ID is disabled and Cyclic_Logic_Control_Flag set equal X'0008'.

    If CRT_Status_Flag3 disabled the Rate_Value is checked to see if equal to six in which case the routine is exited. Else, the DEU_Update_Rate is picked up and calculations are performed to see if rate

should be executed on this cycle, if not the routine is exited. Else, the Cyclic_Dynamic_Branch_Address is added to Rate_FCW_Count and stored in Cyclic_Dynamic_Branch_Address.

The routine now increments the DDT pointer to the next DDT entry.

22. Branch_CMD - The Branch_DDT_Skip_Count is added to the DDT pointer to get to next DDT entry.

23. On_Demand_CMD - The DDT pointer is incremented to the next DDT entry.

The routine recycles through the loop if necessary, if not, the routine is complete.

d. Outputs - See Table 3.3.1.3-1.

e. Module References - (640) Data_Conversion (DCI#CON)

f. Module Attributes - Internal procedure.

g. Template References - None

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.3.1.3-1

NAME DATA_FORMATTER (DCI#FMT)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	CYCLIC_DFT_SAVE_AREA	L073	L	620	620	CLOCDFT			
2	DFT_DDT_POINTER	D050	IZ	DFG	600, 620 635				
3	ON_DEMAND_CMD	D222.00	IZ	DFG	635	DEMAND			
4	ON_DEMAND_BIT_NUMBER	D222.03	IZ	DFG	635				
5	ON_DEMAND_BIT_ADDR	D222.04	IZ	DFG	620 635				
6	CRT_STATUS_FLAG3	B010.43	IZ	400	625				
7	BILEVEL_TEST_CMD	D200	IZ	DFG	620 635	BLT			
8	FCW_REMOTE_CMD	D201.00	IZ	DFG	635	FCWSR			
9	REMOTE_TEXT_CMD	D202.00	IZ	DFG	620 635	RTC			
10	VARIABLE_PARAMETER_CMD	D204	IZ	DFG	620 635	VPARM			
11	REMOTE_CHARACTER	D211.00	IZ	DFG	620 635	RCHAR			
12	STATUS_BYTE_CMD	D213.00	IZ	620	620 635	SBC			
13	MULTIPLE_DISCRETE_TEST_CMD	D217.00	IZ		620 635	MDT			
14	RATE_COMMAND	D218.00	IZ		620 635	TEST			
15	BRANCH_COMMAND	D219.00	IZ		620 635	IMMEDIATE			
16	RATE_	D220.00	IZ		620 635	RATE			
17	BRANCH_	D221.00	IZ		620 635	BR			
18	CRT_FEATURE_CONTROL_WD3	L067	L	620	620	CLOCFS03			
19	CRT_FEATURE_CONTROL_WD2	L066	L	620	620	CLOCFS02			
20	CRT_FEATURE_CONTROL_WD1	L0068	L	620	635	CLOCFS03			

DATA TABLE 3.3.1.3-1 (cont'd)

NAME DATA_FORMATTER (DCI#FMT)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
21	CYCLIC_OUTPUT_BUFFER	L0068	LQ	620	635				
22	BLT_ADR1_PTR	D200.04	IZ	DFG	635				
23	BLT_ADR2_PTR	D200.05	IZ	DFG	635				
24	TIME_CONVERSION_BIT	T030.02	IZO	685	635,680 685	CDL_CONVERSION_BIT			
25	CYCLIC_TIME_CONVERSION_DDT	L071	L	620	620	CLOCDDTE			
26	RTC_ADR1_PTR	D202.05	IZ	DFG	620 635				
27	RTC_CONTROL_BIT	D202.02	IZ	DFG	620 635				
28	CYCLIC_LOGIC_CONTROL_FLAG	L063	L	620	620,625 640,675	CLOCFLGS			
29	VPARM_FMT1	D205.00	IZ	DFG	635 640				
30	REMOTE_CHARACTER_LENGTH	D211.03	IZ	DFG	635 640				
31	REMOTE_CHARACTER_ADDR	D211.04	IZ	DFG	635 640				
32	STATUS_BYTE_ADDR	D213.03	IZ						
33	STATUS_BYTE_CHAR_TABLE	L274	L	635	635	CLOCSB			
34	MDT_XTABLE_ADDR	D217.02	IZ	DFG	620 635				
35	MDT_YTABLE_ADDR	D217.03	IZ	DFG	620 635				
36	TEST_CONTROL_BIT	D218.02	IZ	DFG	620 635				
37	TEST_ADDR	D218.04	IZ	DFG	620 635				
38	TEST_DDTS_TO_SKIP	D218.03	IZ	DFG	620 635				
39	NUMBER_OF_FCWS	D219.02	IZ		620 635				
40	RATE_VALUE	D220.02	IZ	DFG					

DATA TABLE 3.3.1.3-1 (cont'd)

NAME DATA_FORMATTER (DCI#FMT)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
41	CYCLIC_DYNAMIC_BRANCH_ADDRESS	L075	L	620	620,625	CLOCBRAN			
42	RATE_FCW_COUNT	D220.03	IZ	DFG	620 635				
43	DEU_DEVICE_ID	L264	O	625	635	CLOCDEV			
44	BRANCH_DDT_SKIP_COUNT	D221.02	IZ	DFG	620 635				
45	DEU_UPDATE_RATE	B010.20	IOZ	500,620 820,880	620,625	CDMV_MAT_DEU_ RATE/CMATRATE			

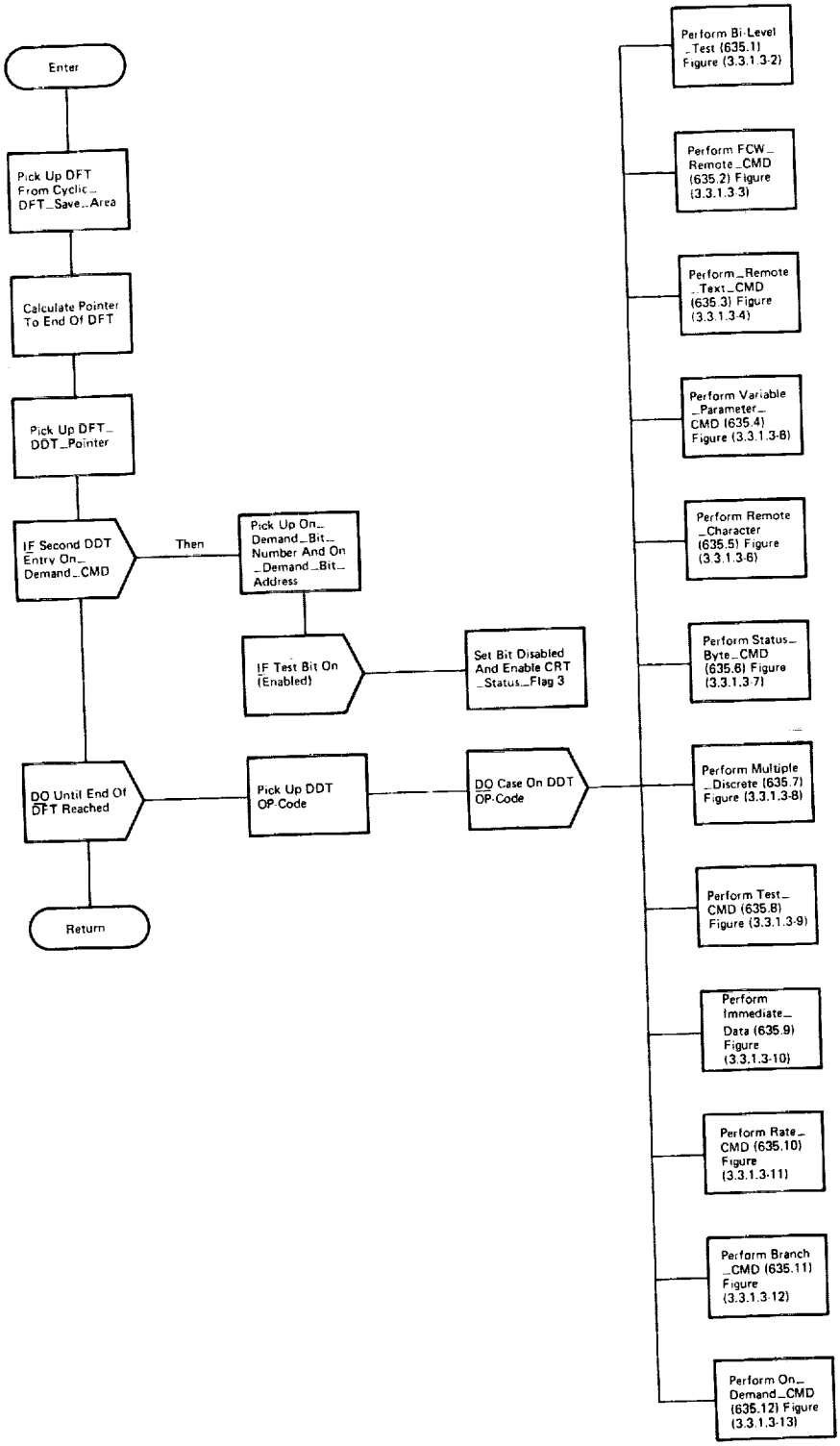


Figure 3.3.1.3-1. Data_Formatter (DCI#FMT)

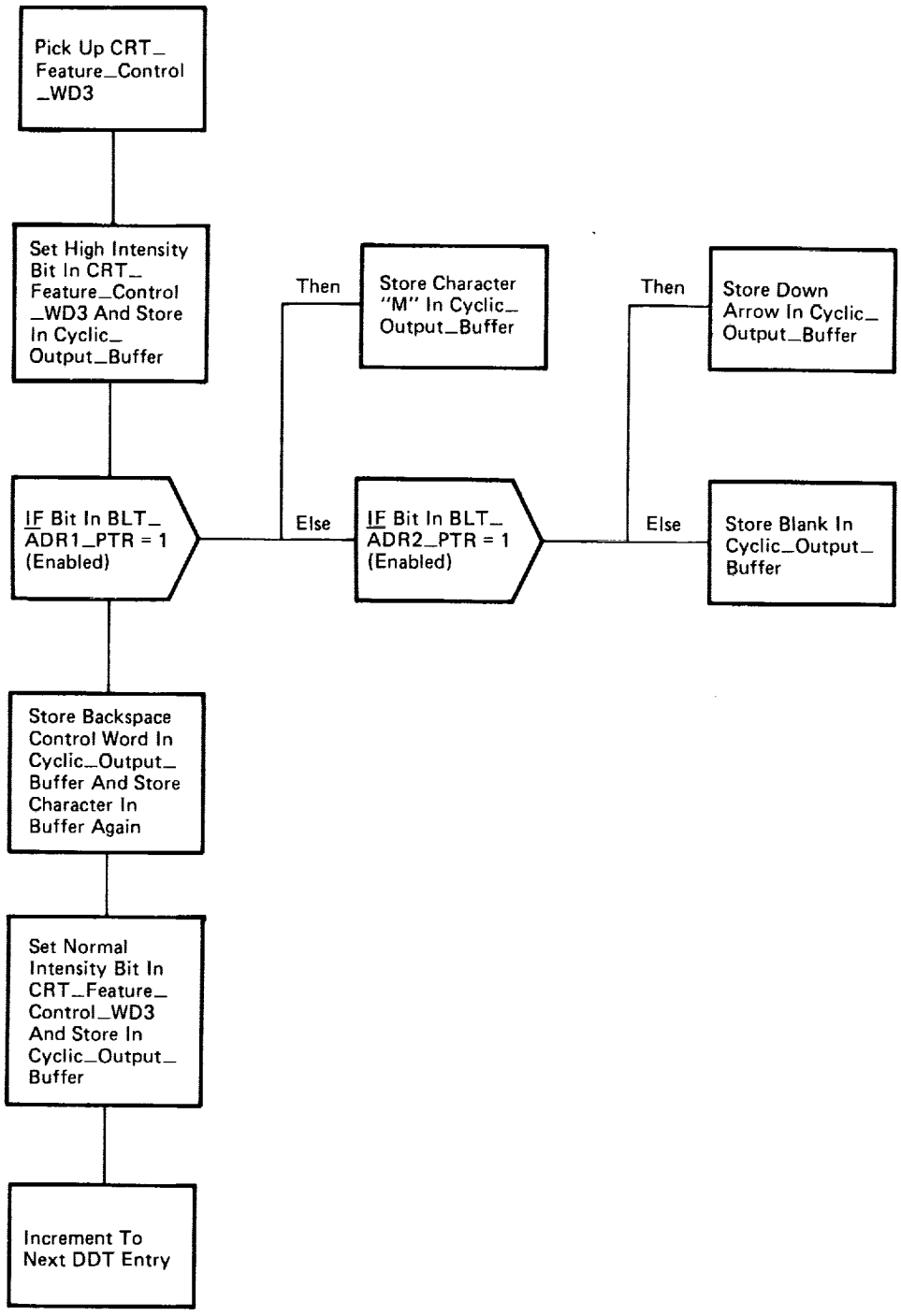


Figure 3.3.1.3-2. Data_Formatter Bi_Level_Test (635.1)

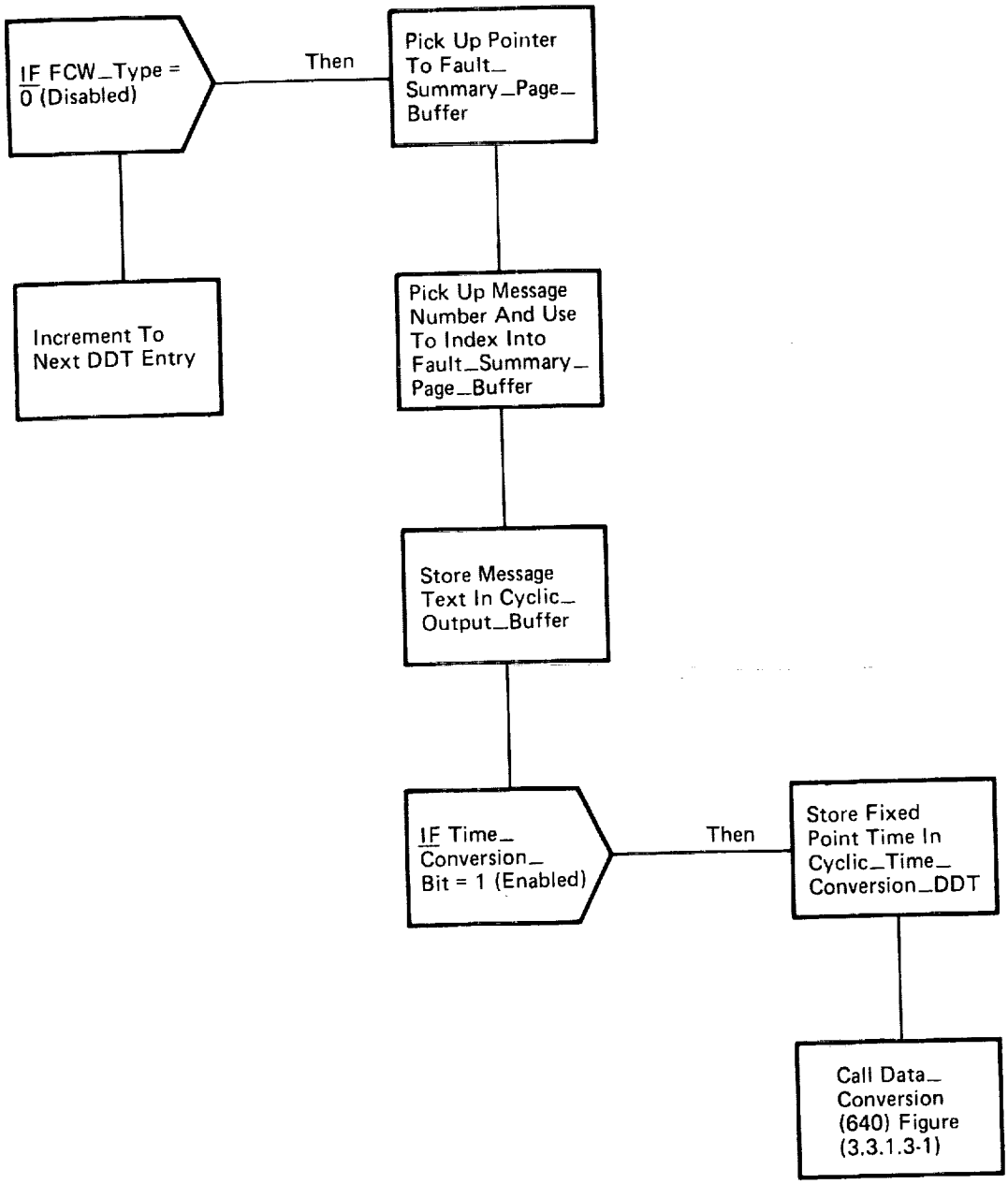


Figure 3.3.1.3-3. Data_Formatter FCW_Remote_CMD (635.2)

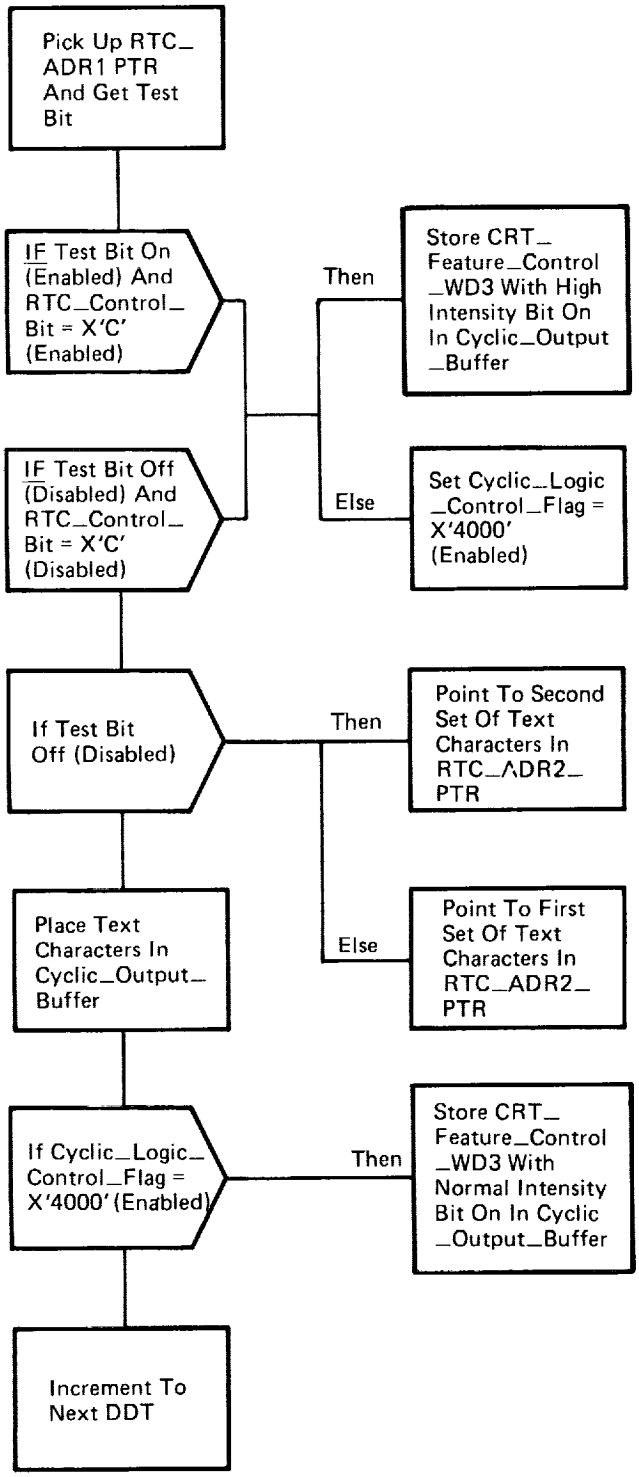


Figure 3.3.1.3-4. Data_Formatter Remote_Text_CMD (635.3)

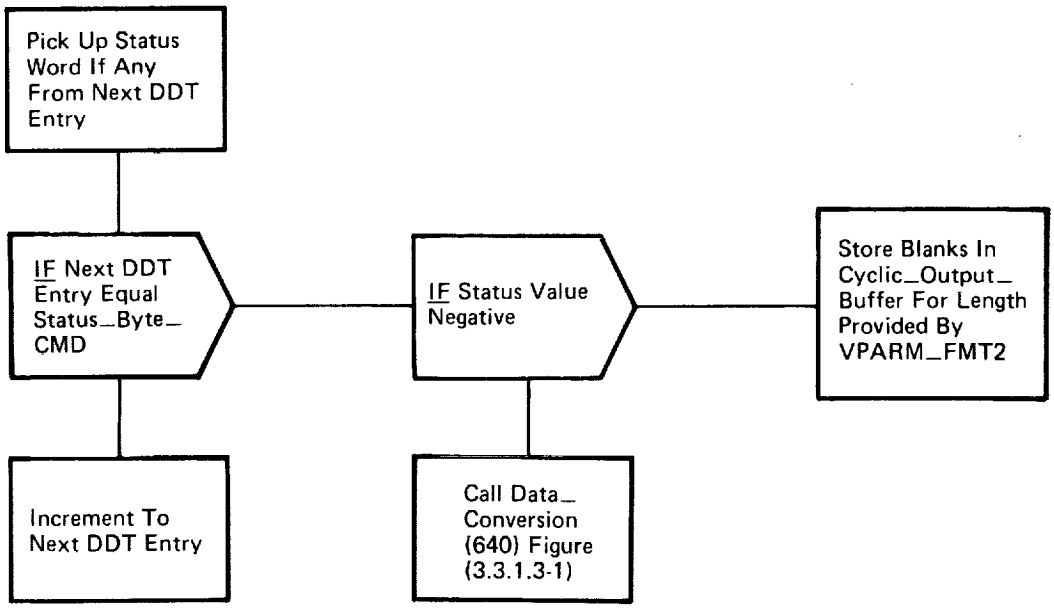


Figure 3.3.1.3-5. Data_Formatter Variable_Parameter_CMD (635.4)

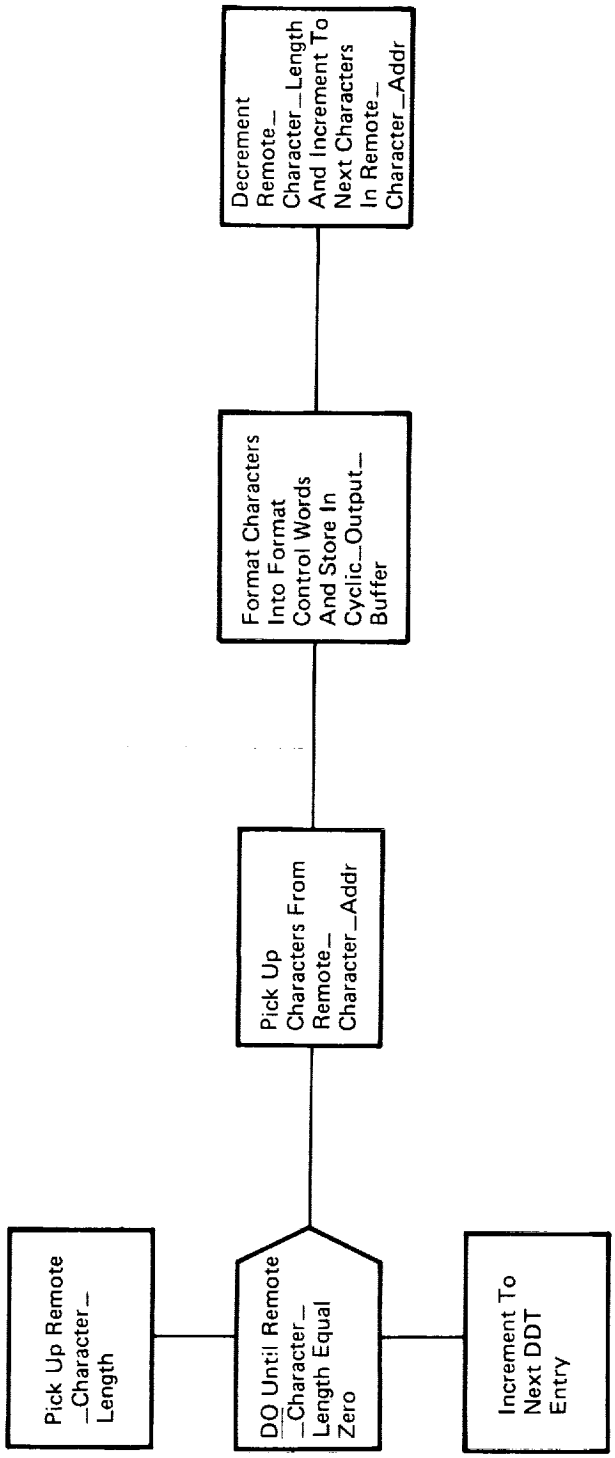


Figure 3.3.1.3-6. Data_Formatter Remote_Character (635.7)

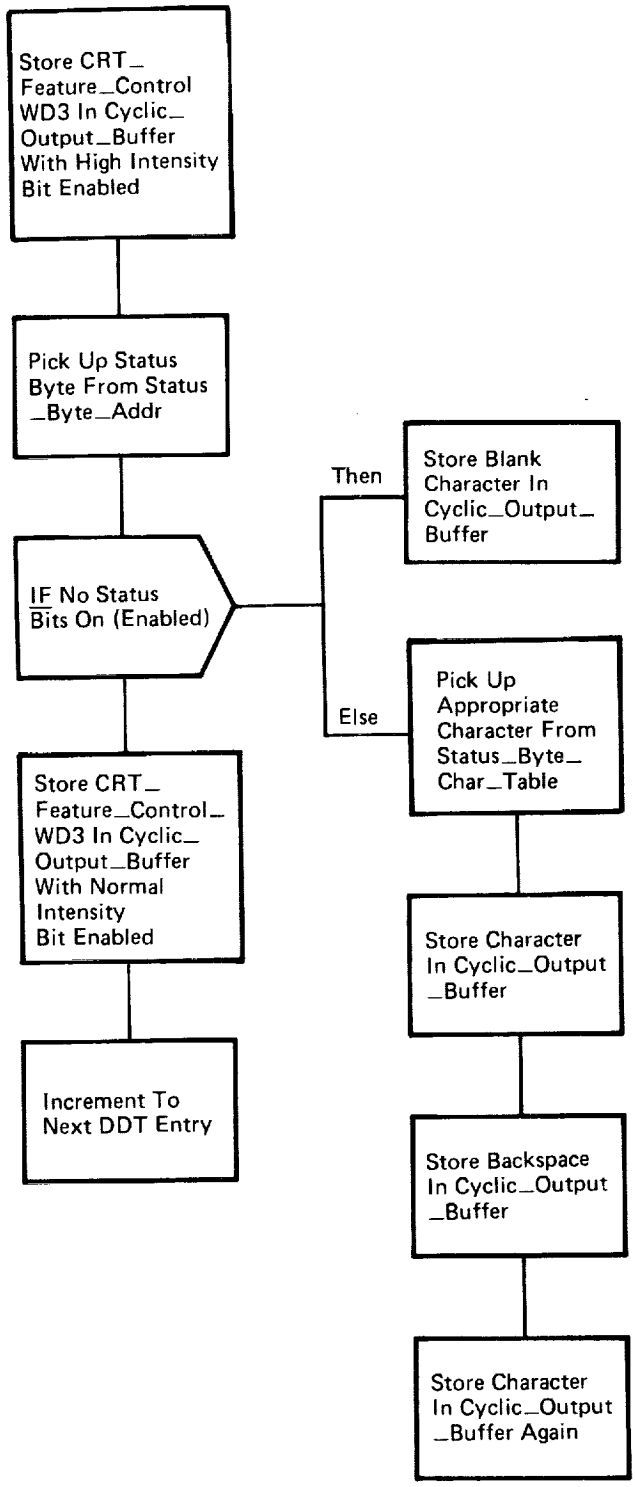


Figure 3.3.1.3-7. Data_Formatter Status_Byte_CMD (635.8)

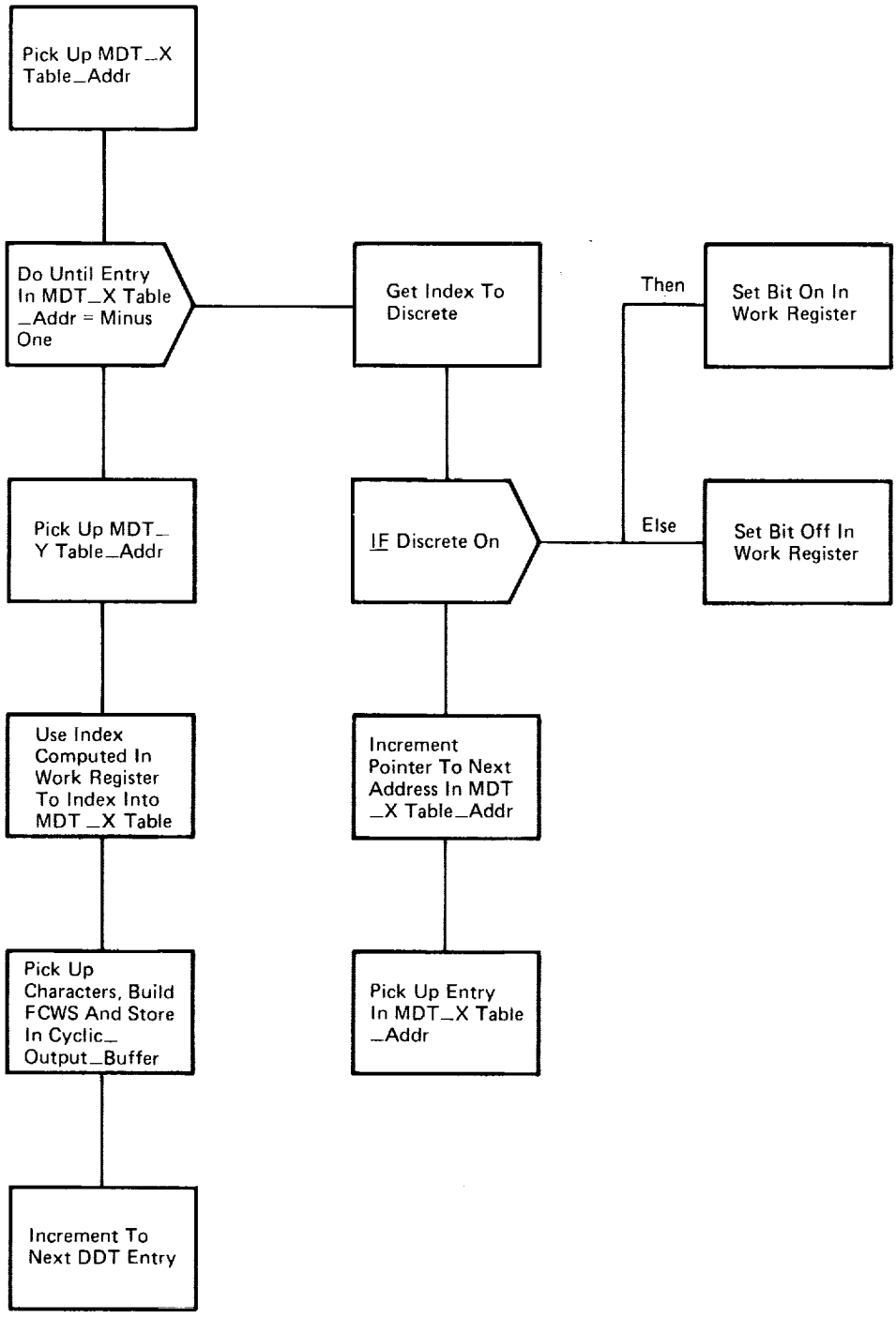


Figure 3.3.1.3-8. Data_Formatter Multiple_Discrete (635.9)

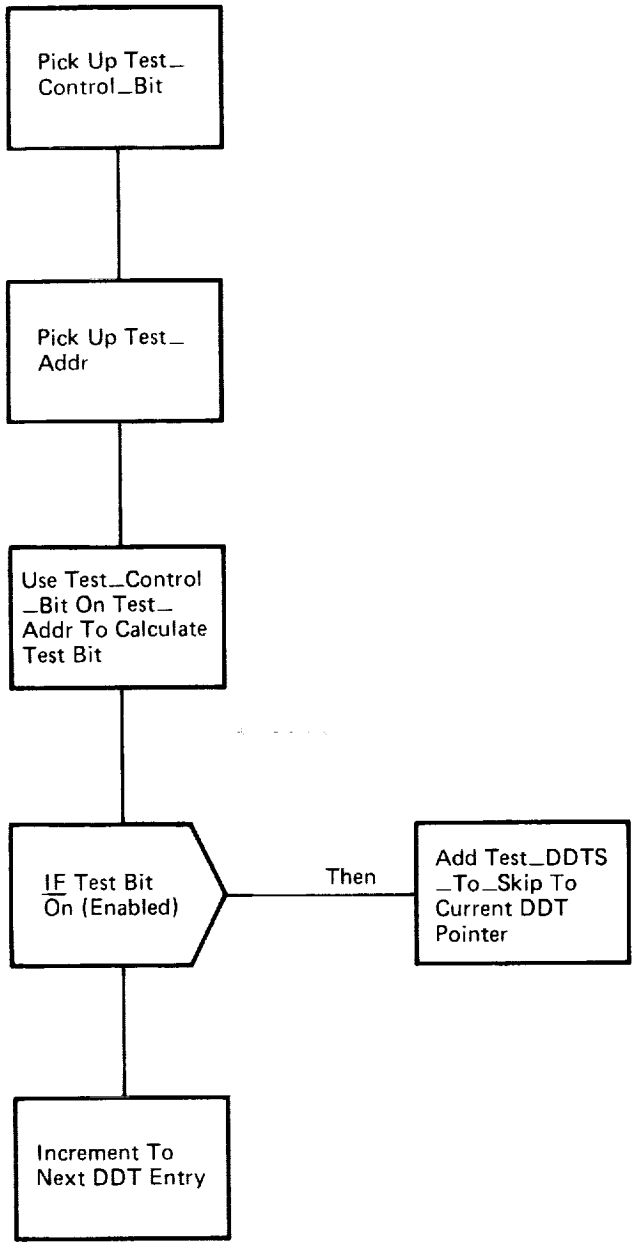


Figure 3.3.1.3-9. Data_Formatter Test_CMD (635.10)

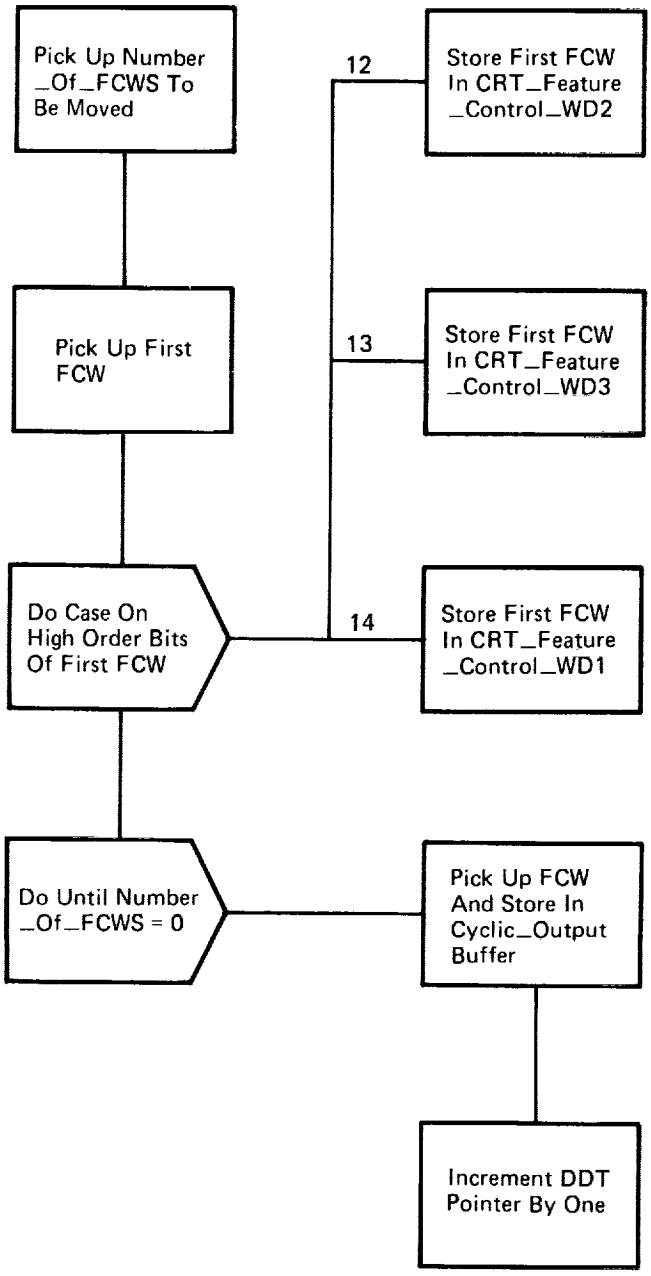


Figure 3.3.1.3-10. Data_Formatter Immediate_Data (635.11)

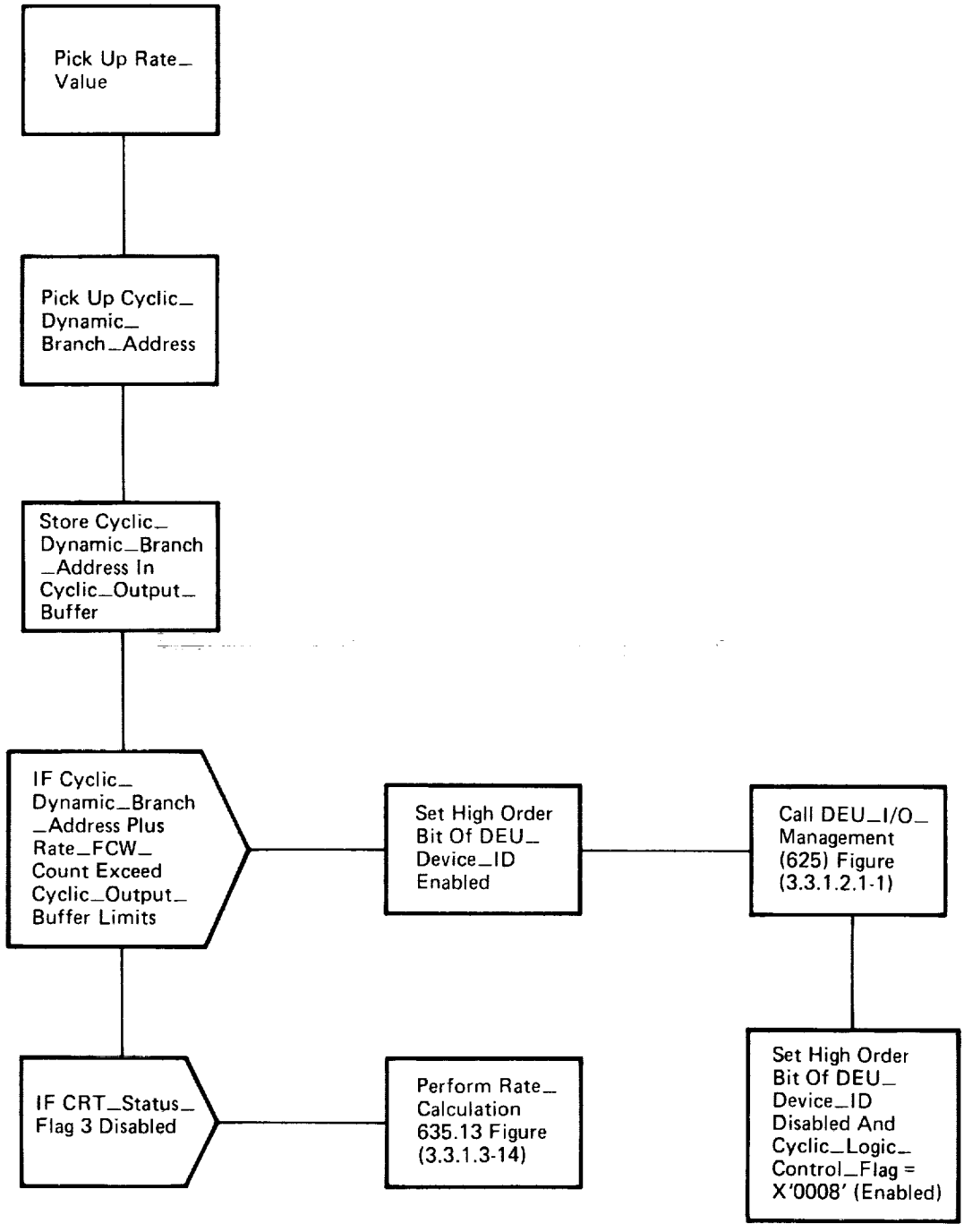


Figure 3.3.1.3-11. Data_Formatter Rate_CMD (635.12)

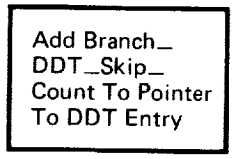


Figure 3.3.1.3-12. Data_Formatter Branch_CMD (635.11)

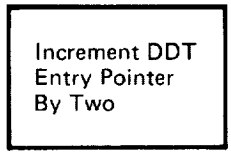


Figure 3.3.1.3-13. Data_Formatter On_Demand_CMD (635.12)

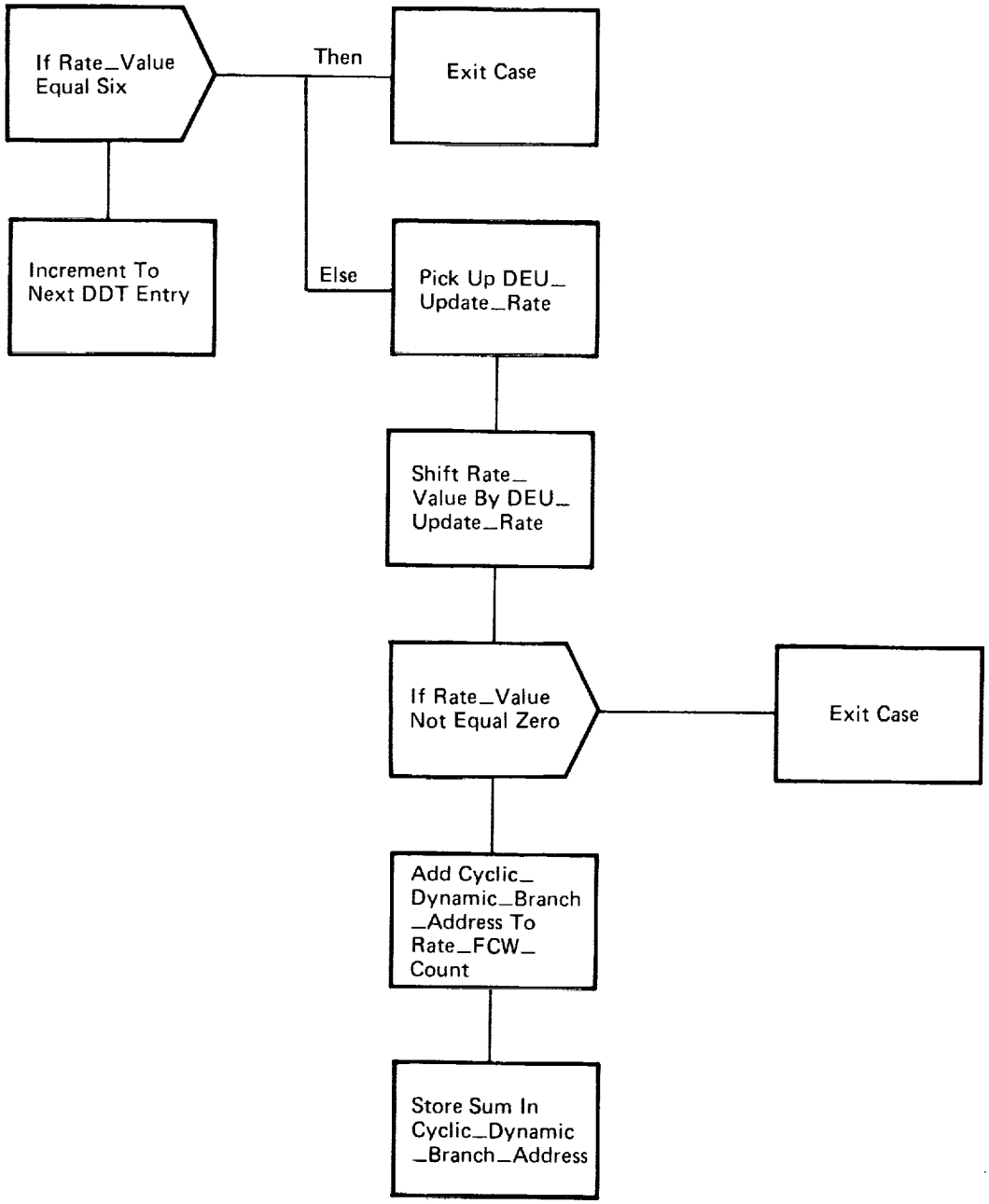


Figure 3.3.1.3-14. Data_Formatter Rate_Calculation (635.13)

3.3.1.4 Data_Conversion (DCI#CON) (640)

Provides logic to convert data from internal format to displayable ASCII according to user specified format.

a. Control Interface

1. CALL Data_Conversion
2. (a) CALLed by (620) Cyclic_Display_Processor (DCI#CYC)
   (b) CALLed by (635) Data_Formatter (DCI#FMT)
   (c) CALLed by (630) Header_Builder (DCIBHDR)

b. Input - See Table 3.3.1.4-1

c. Process Description - This module saves the current DFT_Dynamic_Data_Table entry in Temp_DDT_Address and picks up the Vparm_Output_Characteristics from the DFT_Dynamic_Data_Table entry. A DO case is then performed on the Vparm_Output_Characteristics to determine which of the following cases is to be performed;

- o TIME Conversion
- o Integer Conversion
- o Octal Conversion
- o Engineering Conversion
- o System Management Calibration

Processing for each of the cases is described below:

1. Time_Conversion - If Vparm_internal_characteristics equal X'0060' then input is fixed point else input is floating point. If floating point input then perform CVFX_Routine which is described here.

   - o CVFX_Routine - If number is negative set Cyclic_Logic_Control_Flag equal X'8000'. This setting indicates to Data_Conversion routine that the input number was negative. The routine then subtracts sixty four from the characteristic of the input number. If the remainder is positive, multiply the remainder by four and divide the input number by two raised to the power of the product. If remainder is negative complement remainder, multiply by four, subtract thirty two from product of multiply and divide input number by two raise to the power of the remainder. If input number was negative make integer portion of converted number positive. The routine then takes integer portion of number and save in Cyclic_Integer_data and proceeds to determine number of decimal places to be displayed for this DDT entry. A DO loop is established picking up sixteen fraction bits and multiplying by ten for each digit until number of decimal places equal zero. The fractional data is stored in Cyclic_Fraction_Data.

The fixed point time is saved in Total_Time and the total number of days is calculated by dividing the fixed point time by total number of seconds in a day. The routine then performs character conversion of total number of days using Character_Conversion as described below.

- Character_Conversion - Pick up number from Cyclic_Integer_Data. If Cyclic_Logic_Control_Flag equal X'8000' set Cyclic_Logic_Control_Flag equal zero and enable high order bit of Cyclic_Integer_Data. The Work_Buffer is then picked up cleared and a pointer to the last character slot is established, this is done because the conversion routine generates the characters starting with the last character. This routine then goes into a DO loop generating characters until all characters have been generated, for further information concerning this procedure see the explanation under detailed descriptions. If Cyclic_Integer_Data high order bit equal one, store minus sign in sign slot of Work_Buffer else place a plus sign in Work_Buffer. The date is then moved from Work_Buffer to Char_Buffer.

The routine then takes converted days adds a slash and store in Temp_Time_Save. The remaining number of seconds is now divided by total number of seconds in a hour to calculate number of hours and Character_Conversion as described above is performed on the number of hours. The number of hours along with semi-colons is stored in Temp_Time_Save. The number of minutes is calculated from the remaining seconds and Character_Conversion is performed, minutes and semi-colons are then stored in Temp_Time_Save. Character conversion is performed on the remaining seconds with the results stored in Temp_Time_Save. Next, the routine performs FCW_Builder on the converted characters, this process is described below.

- FCW_Builder - Pick up DDT entry and get Vparm_Sign_Status, Vparm_FMT1 and Vparm_FMT2 from DDT entry. If Cyclic_Logic_Control_Flag equal X'0020' a blank is displayed as the sign value else pick up sign from Char_Buffer and if sign equal X'2D00' (negative) then DO case Vparm_Sign_Status.

  1 - Display plus sign
  2 - Display minus sign
  3 - Display "D" as sign value
  4 - Display "U" as sign value
  5 - Display "L" as sign value
  6 - Display "R" as sign value
  7 - Display minus sign

else DO case Vparm_Sign_Status;

1 - Display minus sign
2 - Display plus sign
3 - Display "U" as sign value
4 - Display "D" as sign value
5 - Display "R" as sign value
6 - Display "L" as sign value
7 - Display blank sign

Place sign in Cyclic_Output_Buffer and continue with fraction processing. If Vparm_FMT2 not equal zero point to data in Cyclic_Fraction_Data. If number of fractional characters odd pick up character, decrement Vparm_FMT1 and Vparm_FMT2. If Vparm_FMT2 not equal zero build double character feature control word for DEU else add a decimal point to character and build feature control word for DEU. The feature control word is stored in Cyclic_Output_Buffer.

If Vparm_FMT2 not equal zero DO while Vparm_FMT2 not equal zero the following. Pick up character and decrement Vparm_FMT1 and Vparm_FMT2. If Vparm_FMT2 equal zero add decimal point, build fcw and store in Cyclic_Output_Buffer. Exit DO loop.

Pick up another character build fcw and store in Cyclic_Output_Buffer. If Vparm_FMT2 equal zero build decimal point fcw and store in Cyclic_Output_Buffer then exit DO loop else recycle through DO loop.

The routine then processes the integer portion of the data using the following procedure. Point to integer data, if Vparm_FMT1 not equal zero DO for infinity (i.e. until Vparm_FMT1 equal zero) the following. Pick up character, decrement Vparm_FMT1 and test to see if Vparm_FMT1 equal zero. If yes then build double character fcw, store fcw in Cyclic_Output_Buffer and exit DO loop. If no, pick up another character, build fcw and store in Cyclic_Output_Buffer. If Vparm_FMT1 equal zero exit DO loop else cycle back through DO loop.

2. Integer_Conversion - Get Vparm_Internal characteristics and DO case on Vparm_Internal_Characteristics performing the following processing:

(a) Vparm_SPS - Pick up input data (32 bits) and perform Set_Sign as described below;

- o Set_Sign - If input data equal zero then set Cyclic_Logic Control_Flag = X'0020'

The routine then performs CVFX_Routine, as described under Time_Conversion.

(b) Vparm_SPI - Pick up input data (16 bits) and perform Set_Sign as described in Vparm_SPS.

(c) Vparm_DPI - Pick up fullword (32 bits) of data and perform Set_Sign as described in Vparm_SPS.

(d) Vparm_DPS - Pick up two fullwords (64 bits) of data, perform Set_Sign as described in Vparm_SPS and perform CVFX_Routine as described under Time_Conversion.

Upon completion of the case processing the module performs Character_Conversion and FCW_Builder both of which are described under Time_Conversion.

3. Octal_Conversion - Get Vparm_Internal_Characteristics and DO case on Vparm_Internal_Characteristics to perform processing described below;

(a) Vparm_SPS - Pick up data (32 bits) and perform CVFX_Routine as described under Time_Conversion.

(b) Vparm_SPI - Pick up data (16 bits) and convert to full word by appending 16 high order zeroes.

(c) Vparm_DPI - Pick up a fullword of data (32 bits).

The routine then performs Set_Sign as described under Integer_Conversion and set a counter initialized to eleven. A DO loop is then entered generating octal characters until counter equals zero. The processing for generating octal characters consist of taking all 16 bits inputs and expanding to a fullword by appending 16 high order zeroes. After getting all data in 32 bit format the bits are taken three bits at a time to form characters from right to left until the full 32 bits have been exhausted. After all characters have been generated FCW_Builder is performed as described under Time_Conversion.

4. Vparm_SM_CAL - Pick up Vparm_Addr_Ptr and put in parameter list to be passed to System_Management_Calibration module for processing of a first order polynomial. The System_Management_Calibration module then passes the end result back to Data_Conversion for display. When the data is returned the CVFX_Routine, character_Conversion and FCW builder are all performed as described under Time_Conversion.

5,6,7,8,9. Engineering_Conversion - The data is picked up as single precision scalar data and a DO case on Vparm_Output_Characteristics is executed to decide which multiplication factor is to be applied to input data for conversion.

5. Vparm_Rad_To_Deg - Convert radians to degrees by multiplying radians by E'57.29577'.
6. Vparm_Kilo_To_Ft - Convert kilometers to feet by multiplying kilometers by E'3280.839'.
7. Vparm_Met_To_Ft - Convert meters to feet by multiplying meters by E'3.280839'.
8. Vparm_Met_Sec_To_Knots - Convert meters/second to knots by multiplying meters/second by E'1.9429'
9. Vparm_Ft_Nat_Miles - Convert feet to nautical miles by multiplying feet by E'.00016457883'.

The data then has Set_Sign performed on it as described under integer conversion along with CVFX_Routine, Character_Conversion and FCW_Builder as described under Time_Conversion.

d. Output - See Table 3.3.1.4-1.

e. Module References - None

f. Module Attributes - Internal procedure.

g. Template References - None

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - Detail explanations are described below:

1. Algorithm: Digits are generated one by one. Thus:
Let I = input integer. Then:

$$d_K = I_K - 10(I_K/10)$$ Integer multiply and divide.

$$I_{K+1} = (I_K - d_K)/10$$

The process terminates when $I_K = 0$. As pairs of digits are generated they are stored, right to left, in Work_Buff output area. The temporary result is then given a sign and moved to Char_Buffer.

DATA TABLE 3.3.1.4-1

NAME DATA_CONVERSTIONS (DCI#CON)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	DFT_DYNAMIC_DATA_TABLE	D150		DFG	635 640				
2	VPARM_OUTPUT_CHARACTERISTICS	D204.11		DFG	620 640				
3	VPARM_ADDR_PTR	D205.06		DFG	640				
4	VPARM_INTERNAL_CHARACTERISTICS	D204.04		DFG	640				
5	TOTAL_TIME	L255		640	640	CLOCTIM1			
6	CHAR_BUFFER	L256		640	640	CLOCBUFF			
7	CYCLIC_LOGIC_CONTROL_FLAG	L063		620	620,625 640,675	CLOCFLGS			
8	CYCLIC_FRACTION_DATA	L257		640	640	CLOCDEC			
9	CYCLIC_INTEGER_DATA	L258		640	640	CLOCARG			
10	WORK_BUFFER	L259		640	640	CLOCCHAR			
11	VPARM_SIGN_STATUS	D205.04		DFG					
12	VPARM_FMT1	D205.00		DFG	635,640				
13	VPARM_FMT2	D205.01		DFG					
14	CYCLIC_OUTPUT_BUFFER	L068		620 630	620, 625 630, 640	CLOCOUT			
15	VPARM_ZEROES_STATUS	D205.03		DFG	620 640				
16	TIME_WORK_BUFFER	L260		640	640	CLOCTIM2			

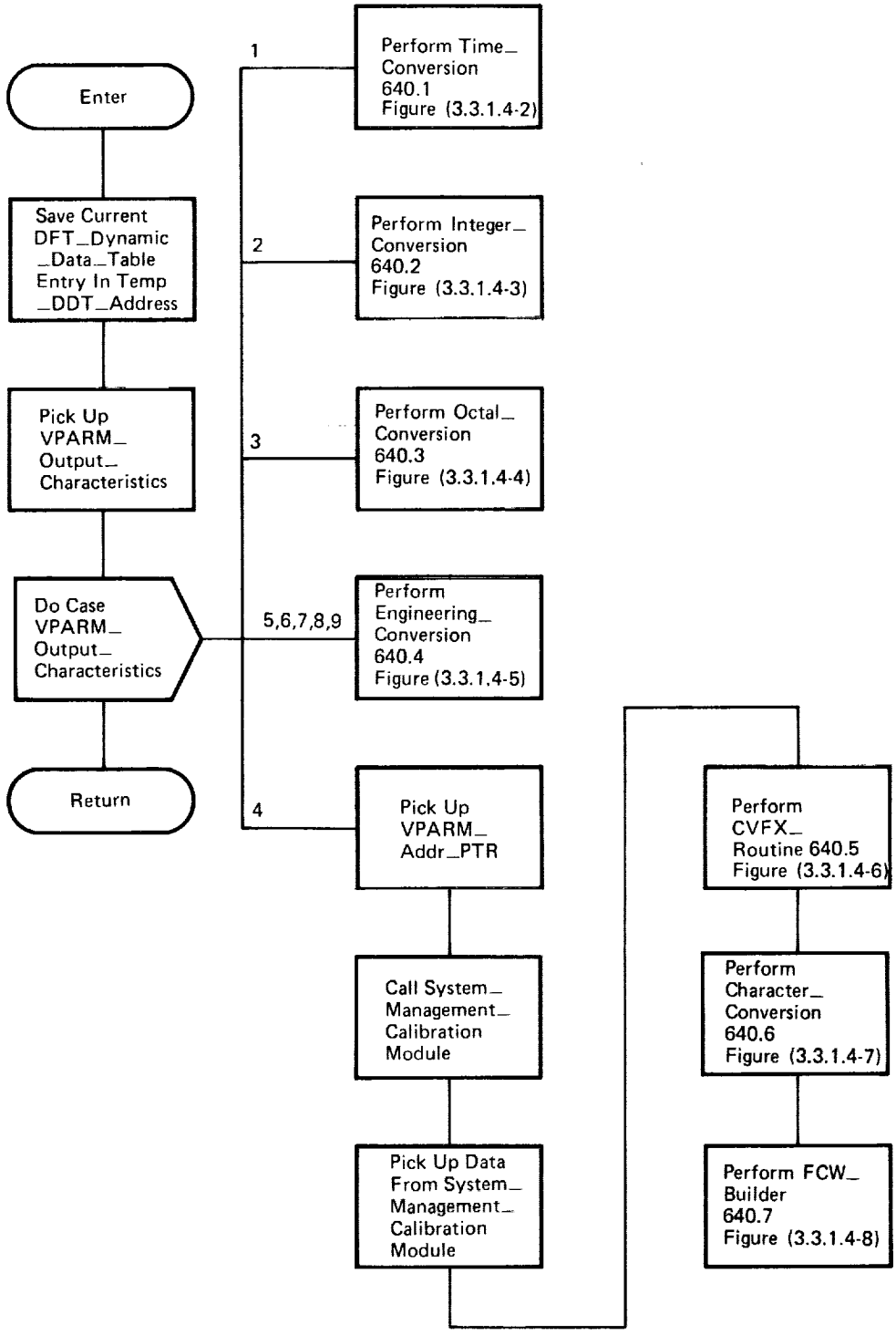


Figure 3.3.1.4-1. Data_Conversion (DCI#CON)

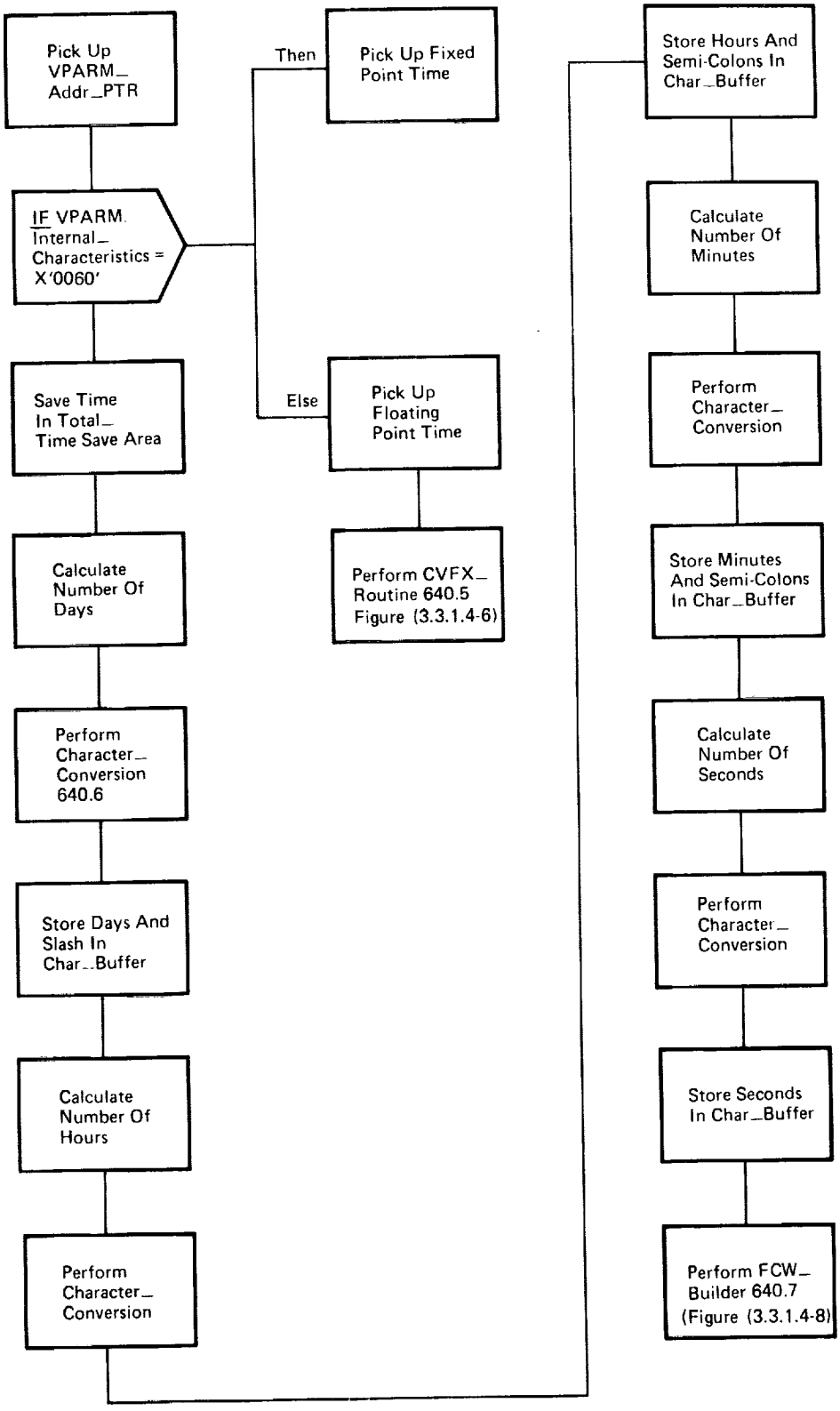


Figure 3.3.1.4-2. Data_Conversion Time_Conversion (640.1)

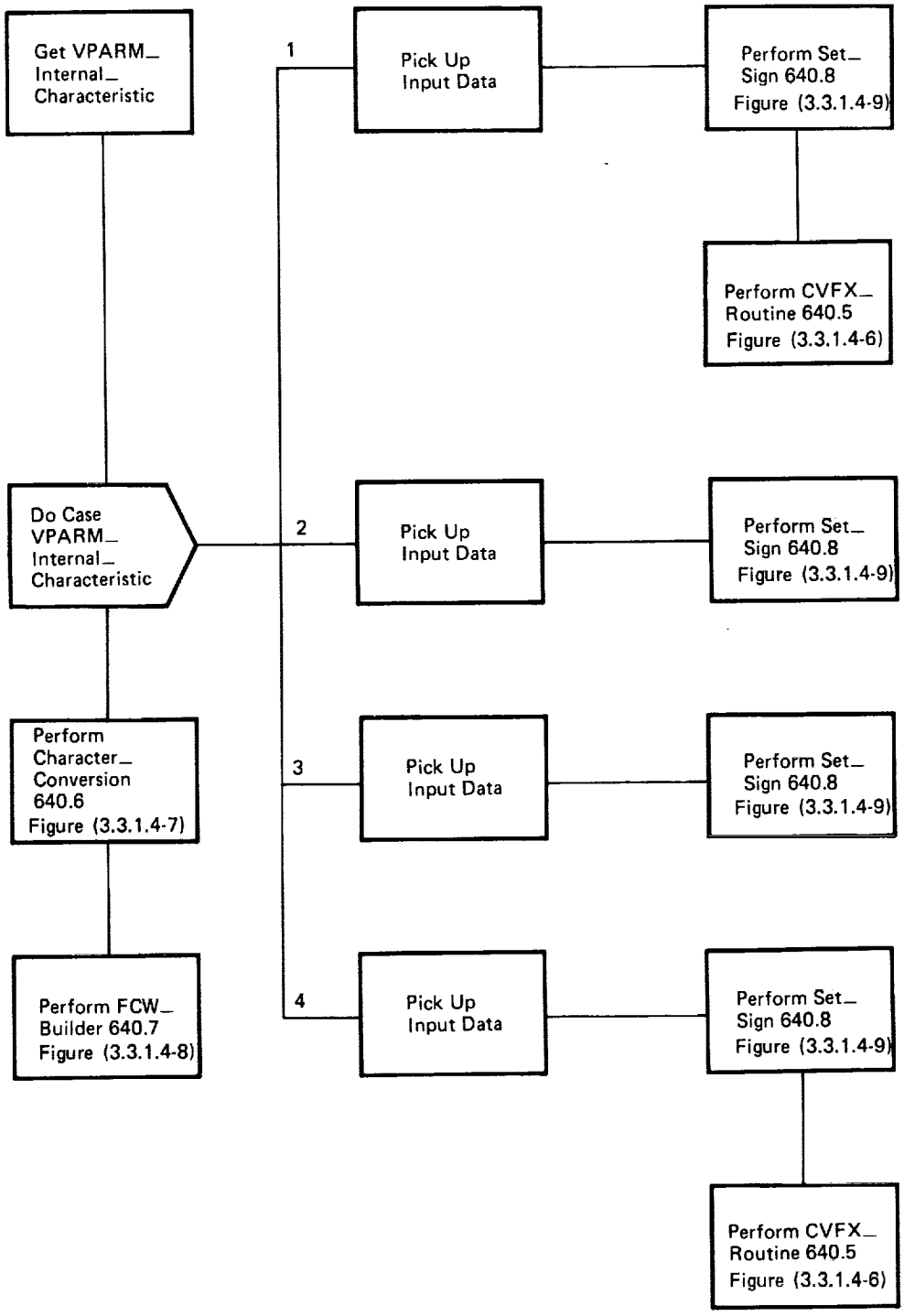


Figure 3.3.1.4-3. Data_Conversion Integer_Conversion (640.2)

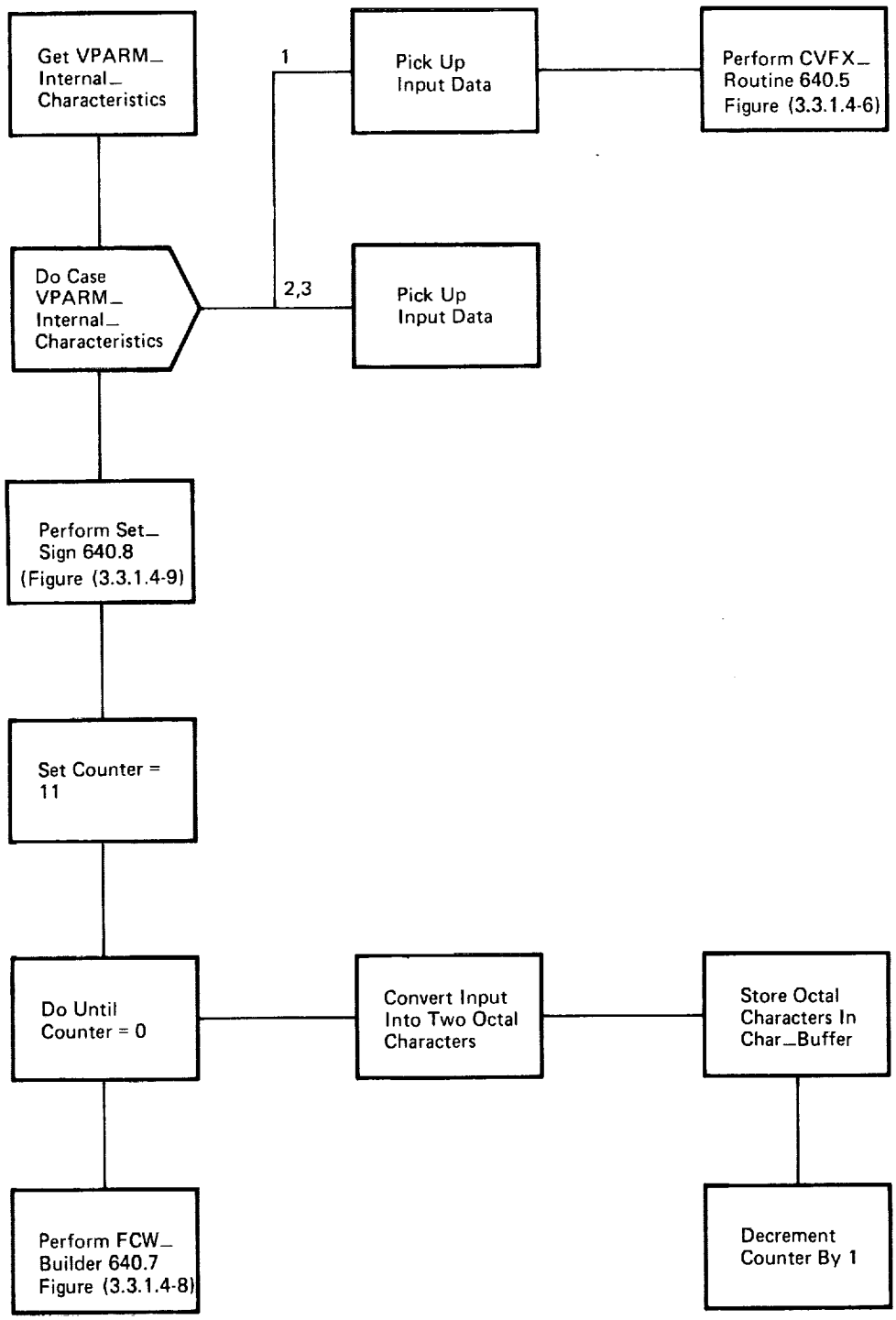


Figure 3.3.1.4-4. Data_Conversion Octal_Conversion (640.3)

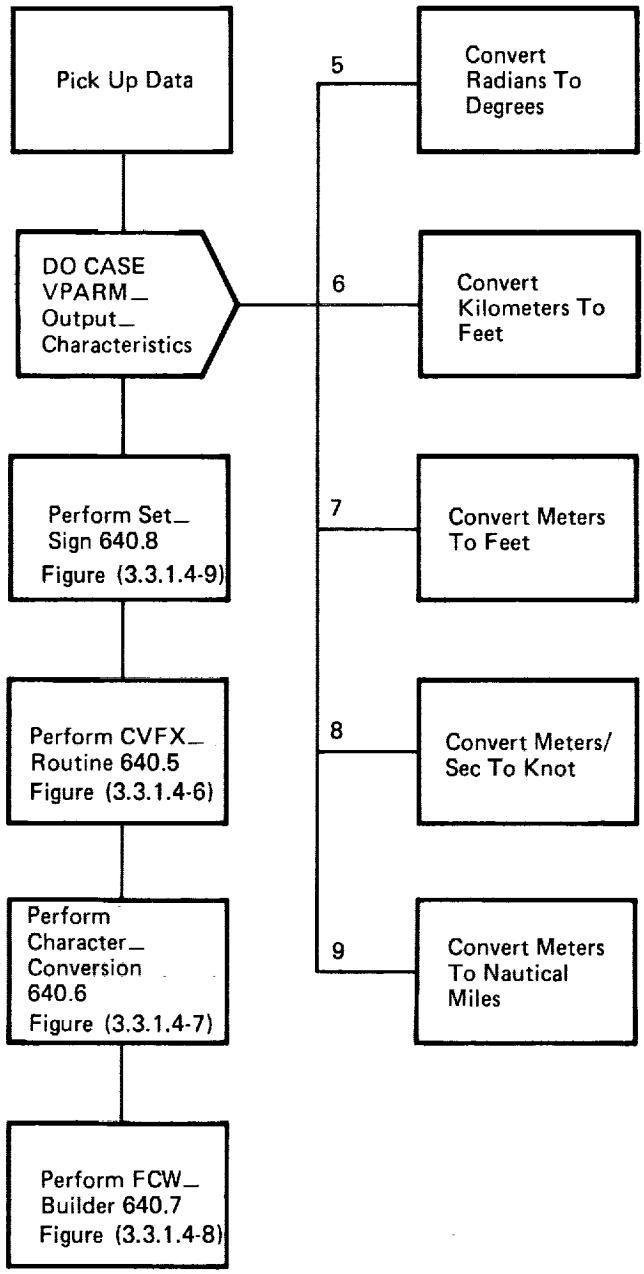


Figure 3.3.1.4-5. Data_Conversion Engineering_Conversion (640.4)

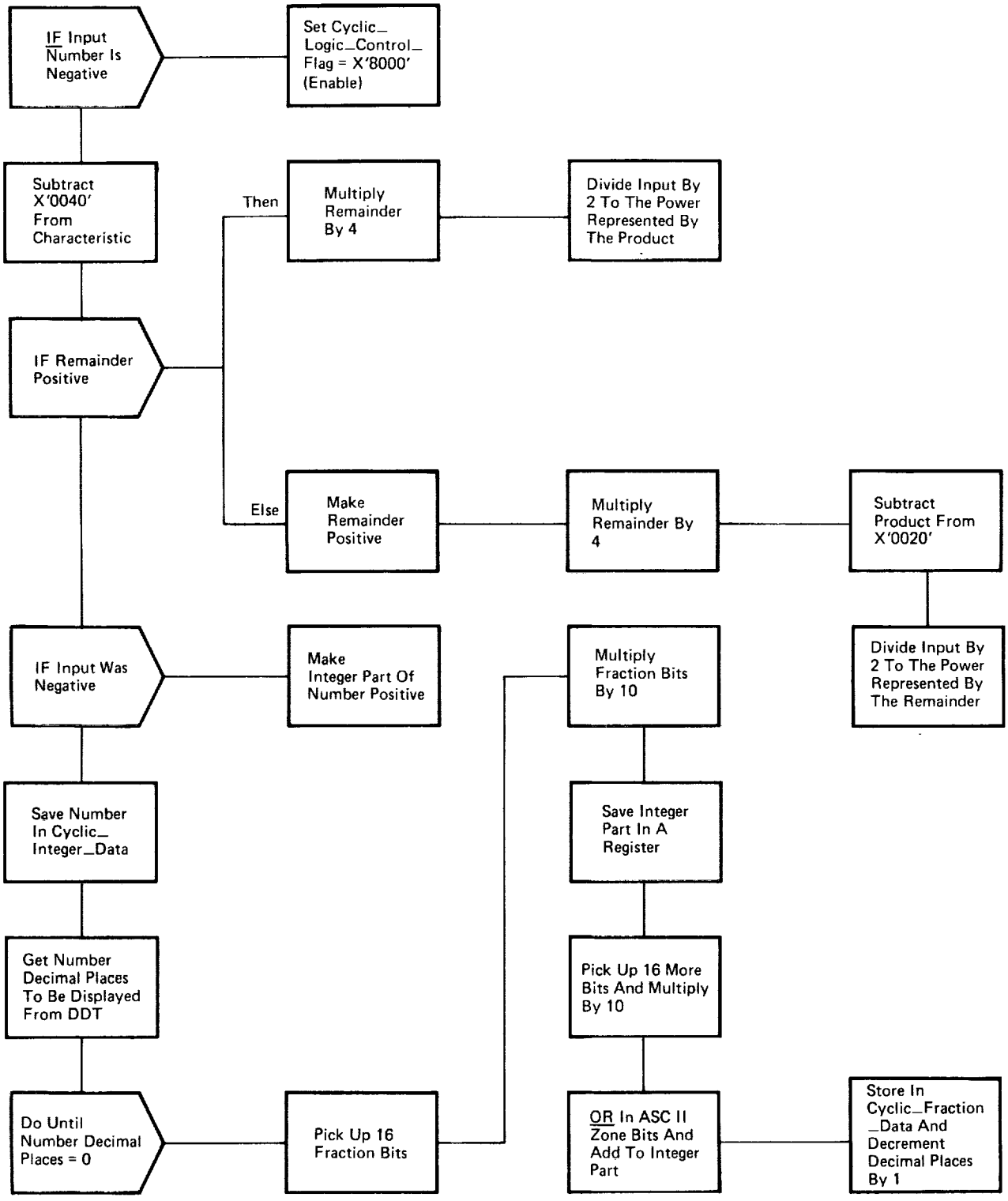


Figure 3.3.1.4-6. Data Conversion CVFX_Routine (640.5)

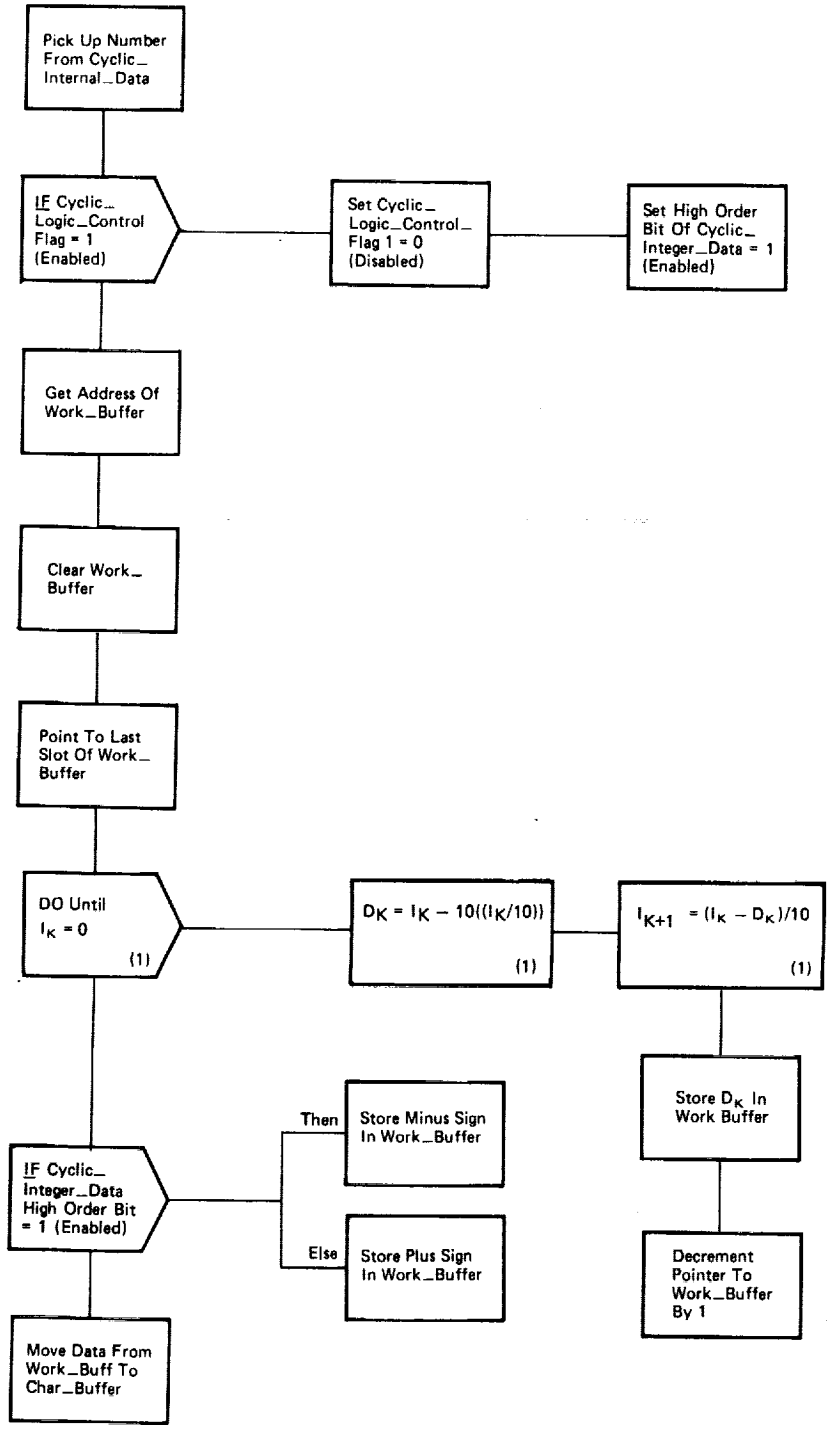


Figure 3.3.1.4-7. Data_Conversion Character_Conversion (640.6)

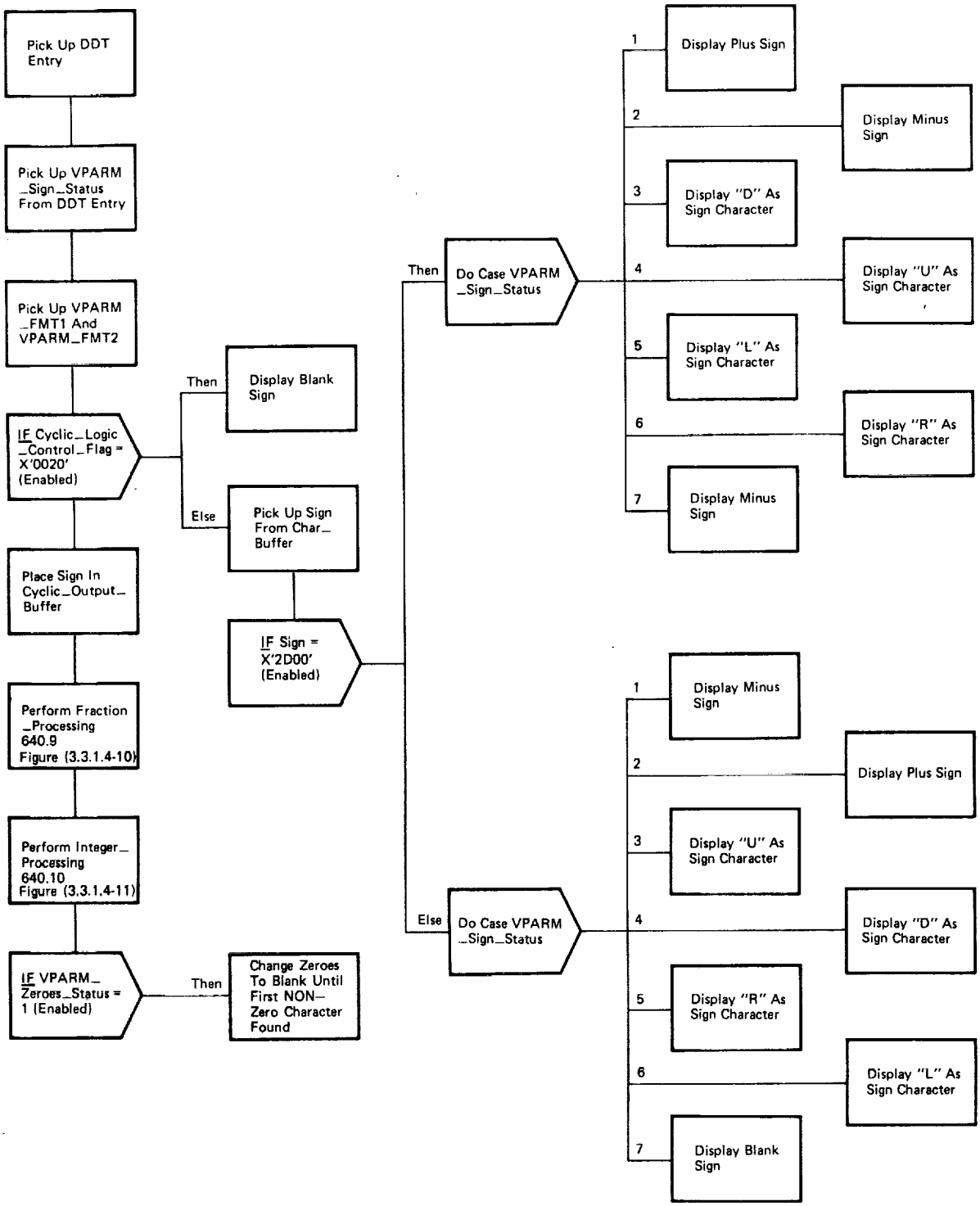


Figure 3.3.1.4-8. Data_Conversion FCW_Builder (640.7)

Figure 3.3.1.4-9. Data_Conversion Set_Sign (640.8)

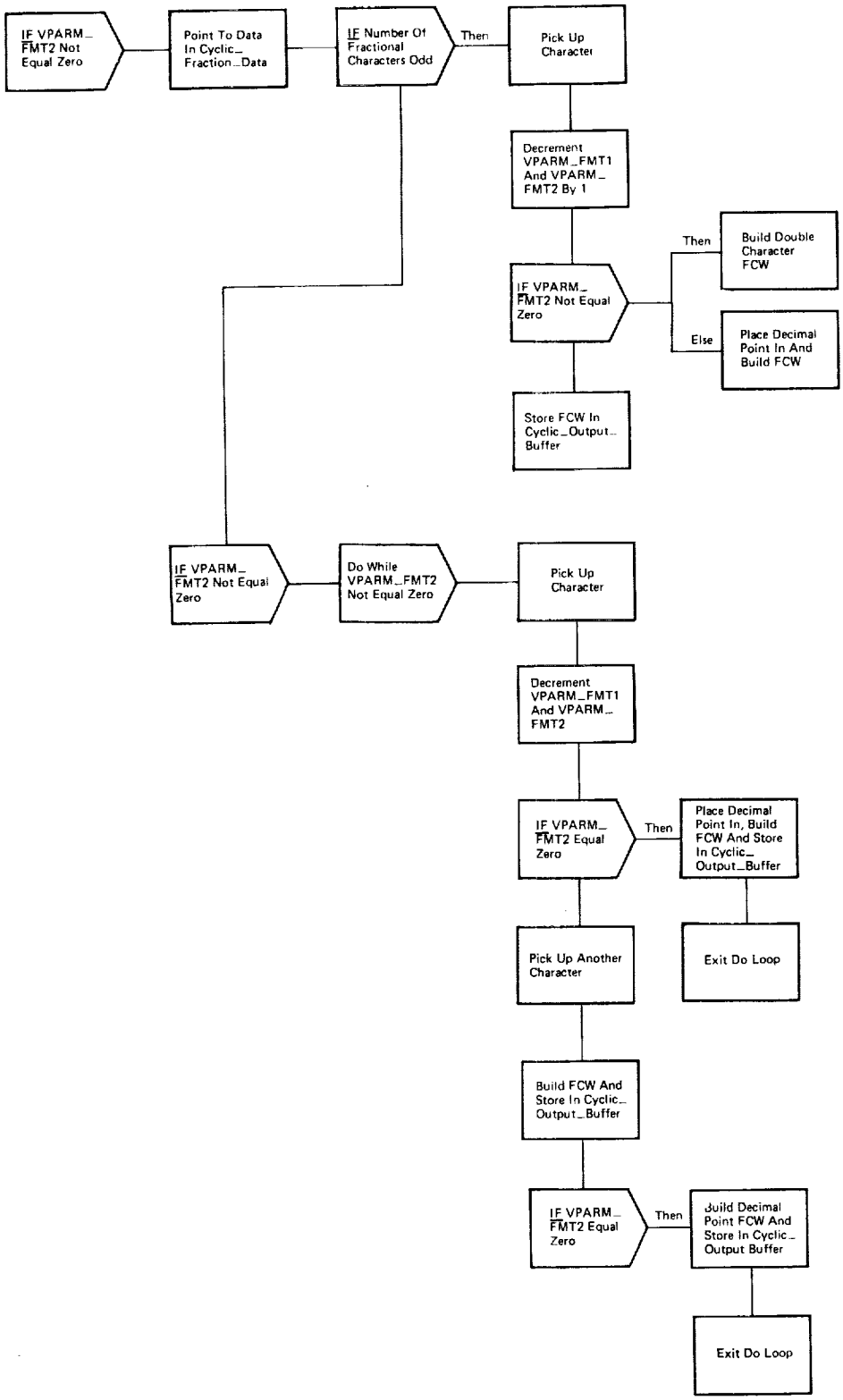


Figure 3.3.1.4-10. Data_Conversion Fraction_Processing (640.9)

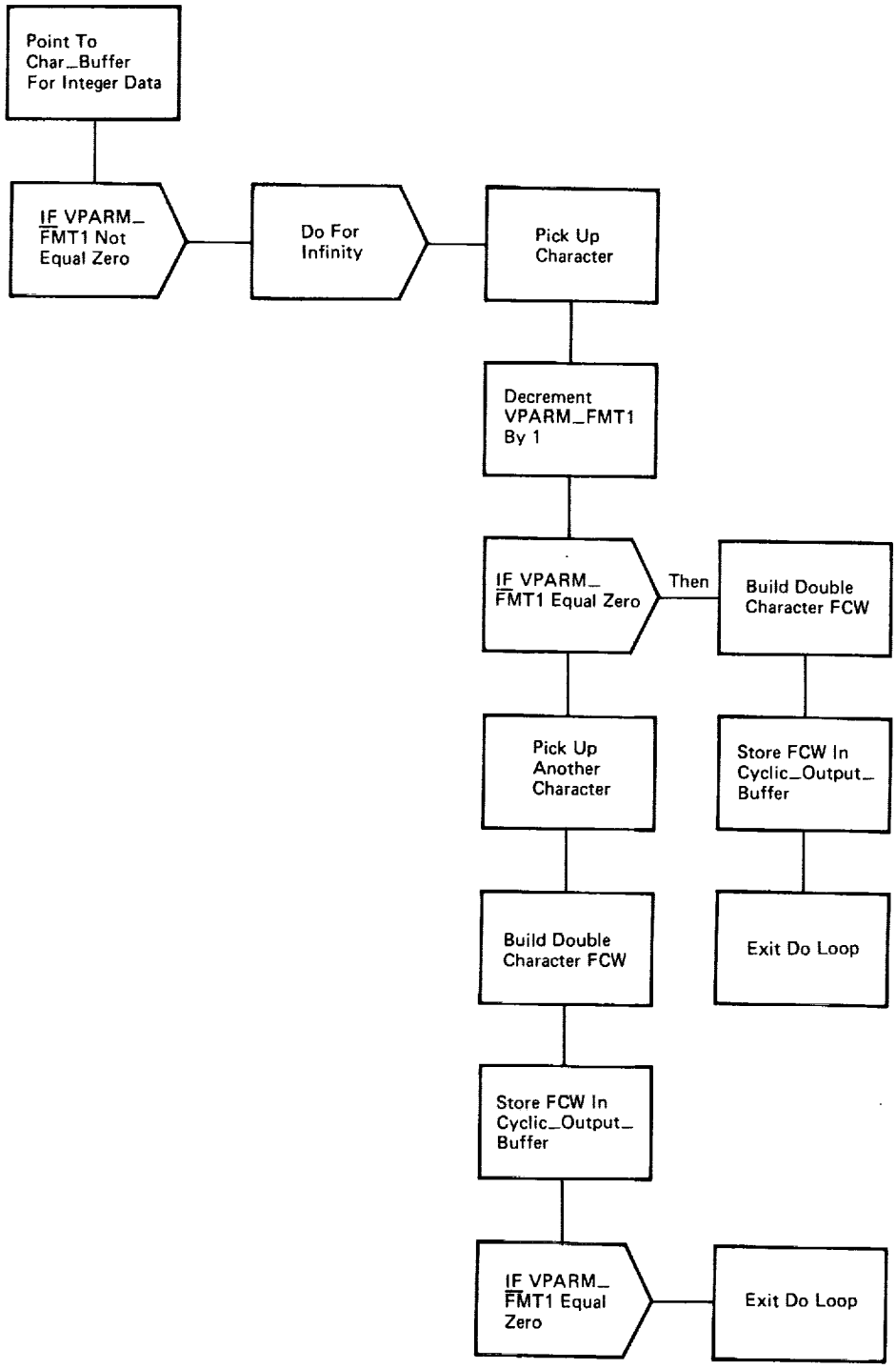


Figure 3.3.1.4-11. Data_Conversion Integer_Processing (640.10)

3.3.1.5 Message_Line_Support_Function (DMS_MSG_LSF) (650)

This module is responsible for displaying tutorial messages on the Tutorial Message Line of the CRT.

a. Control Interface -

1. Call DMS_MSG_LSF
2. Called by applications

b. Input - See Table 3.3.1.5-1.

c. Process Description - This module annunciates tutorial messages which corresponds to a specific OPS or SPEC. The application module that desires a tutorial message displayed passes a parameter list to the Message_Line_Support_Function which specifies the following:

1. Major Function ID
2. OPS or SPEC display number
3. Message_Text
4. Message_ID

The control flow for this module is presented in Figure 3.3.1.5-1.

d. Outputs - See Table 3.3.1.5-1.

e. Module References - (655) FCW_Builder (DMS_FCW_BUILDER)

f. Module Attributes - External Procedure

g. Template References - UI_General_COMPOOL (CDM_UI_COMPOOL)

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.3.1.5-1

NAME Message_Line_Support_Function (DMS_MSG_LSG)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	Deu_Destination	L095	I		620, 650	DMS_MF			
2	Display_Number	L096	I		650	DMS_DISPLAY_NUMBER			
3	Message_Text	L097	I/O		620, 650	DMS_MESSAGE_TEXT			
4	Message_ID	L098	I/O		650	DMS_ID			
5	Call_Indicator	L099	I	650	650 655	DMS_INDICATOR			
6	DEU_Counter	L100	I	650	650	DMS_COUNTER			
7	Display_Level_Page_Number	B010.35	I	500 560	650	CDMV_MAT_DISP_NUMBER			

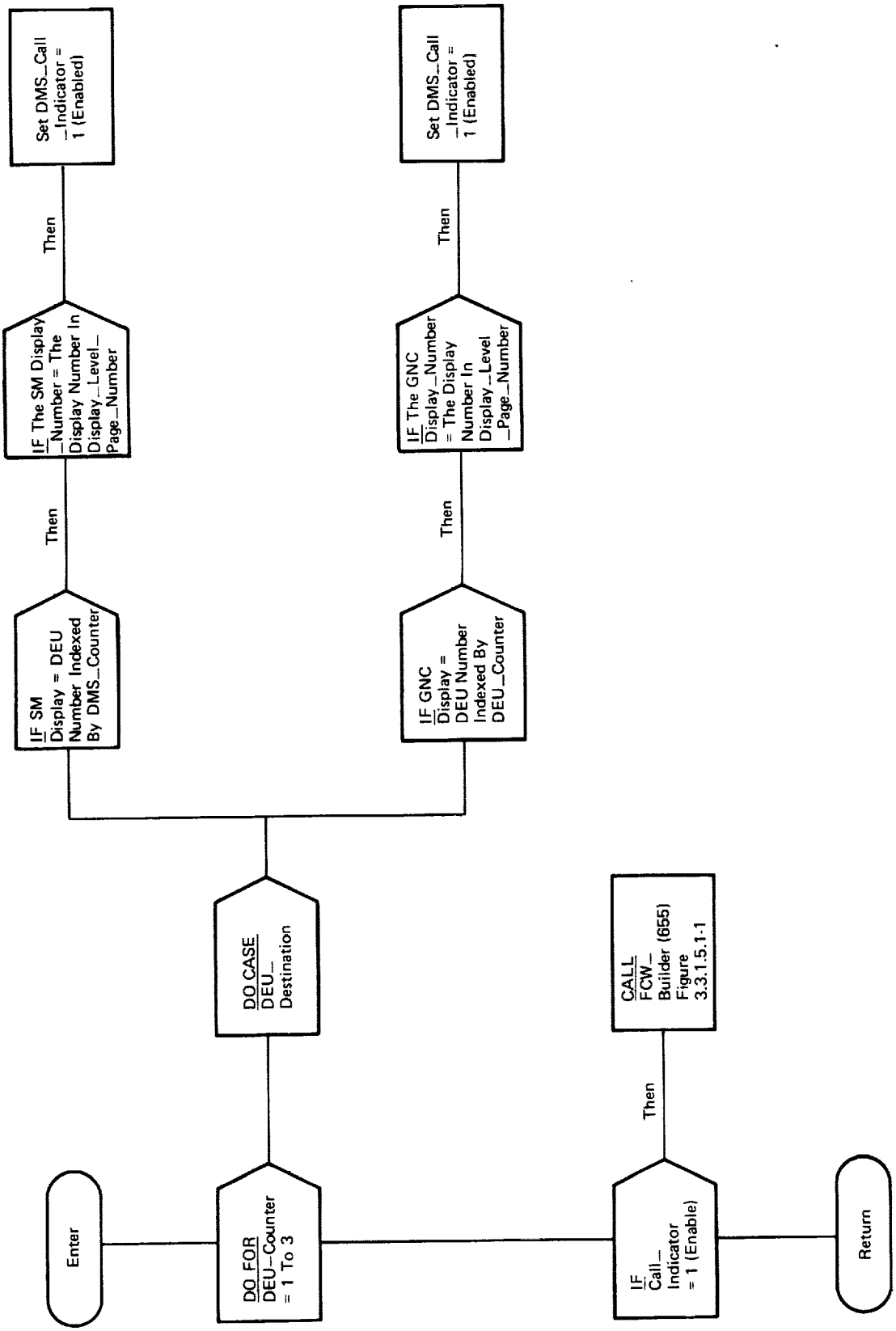


Figure 3.3.1.5-1. Message_Line_Support_Function (DMS_MSG_LSF)

3.3.1.5.1 FCW_Builder (DMS_FCW_BUILDER) (655)

This module is responsible for converting tutorial messages from characters to a displayable DEU format.

a. Control Interface -

1. Call DMS_FCW_Builder
2. Called by (650) Message_Line_Support_Function (DMS_MSG_LSF)

b. Input - See Table 3.3.1.5-1.

c. Process Description - This module builds double character FCWs and places them in the Tutorial_Message_Line_Buffer. It also sets CRT_Status_Flag7 and CRT_Status_Flag6 respectively for Cyclic_Display_Processor.

The control flow for this module is presented in Figure 3.3.1.5.1-1.

d. Outputs - See Table 3.3.1.5-1.

e. Module References - None

f. Module Attributes - Internal Procedure

g. Template References - UI_General_COMPOOL

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

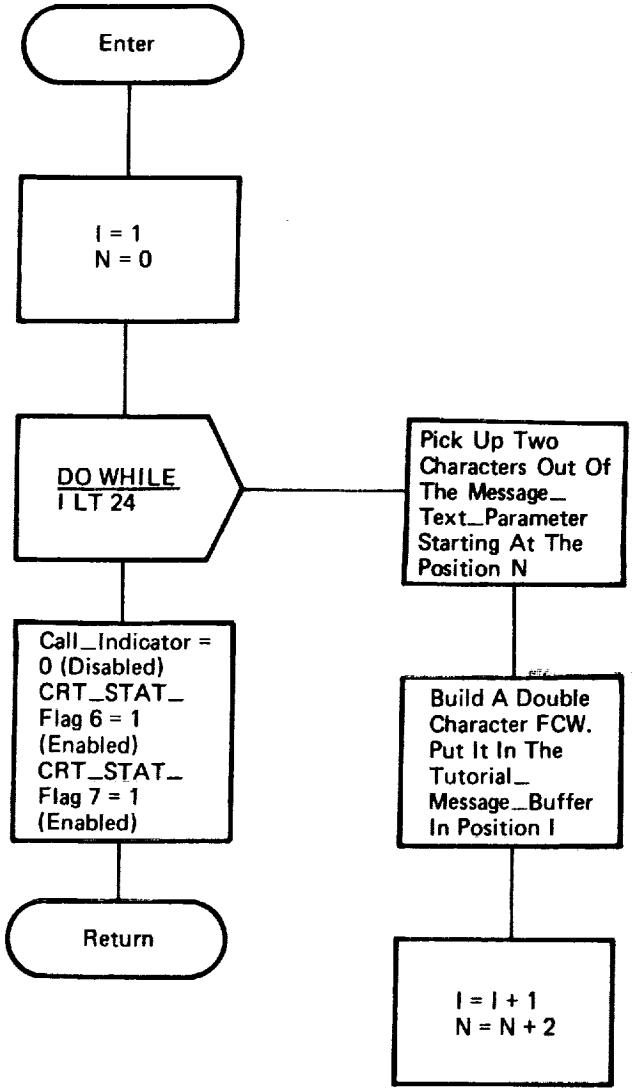
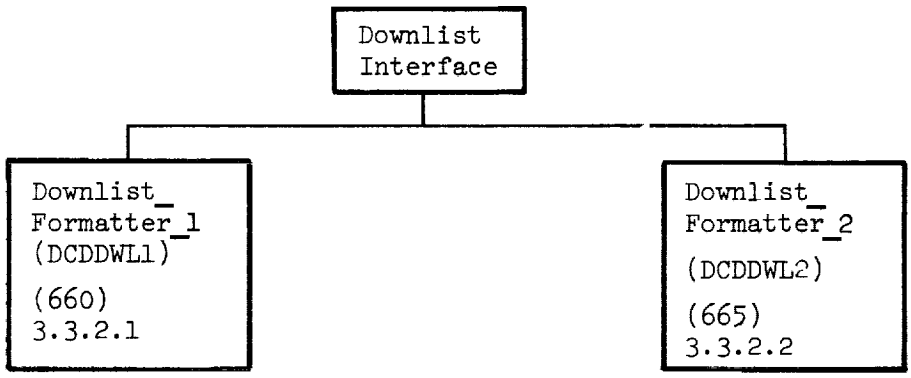


Figure 3.3.1.5.1–1. FCW_Builder (DMS_FCW_BUILDER)

3.3.2 Downlist Interface

The Downlist Interface is performed by the Downlist Formatter which controls, collects, formats, and transfers GPC downlist data (see Figure 3.3.2-1). This data may be computed quantities or raw avionics input data, made available by applications and systems software. The difference between the two Formatters is transparent to the user. The difference being the format (set of data) output.



Downlist Interface Hierarchy

Figure 3.3.2 - 1

3.3.2.1 Downlist_Formatter_1 (DCDDWL1)(660)

The Downlist Formater controls, collects, formats and transfers downlist data in OPS 0 and 1.

a. Control Interface:

1. Call DCDDWL1 (PRIO_DCD)

   a. CALLed by (750) System_Interface_Processor (AIE_SIP)
   b. CALLed by (710) GPC_Startup (AIR_GPC_STARTUP)
   c. CALLed by (820) GPC_Reconfiguration (ARC_RECONFIG)
   d. CALLed by (880) Secondary_Reconfiguration (ARH_SEC_GPC_RECONFIG)

b. Input:

The following items are data inputs to the Downlist_Formatter_1 and reside in various compools.

1. Applications and system software data to be Downlisted.
2. PCMMU I/O Table (CDWS_PMU_WRITE) - defines the GPC to PCMMU data transfer process.
3. Control indicators specifying pending changes to be effected in the downlist processes.
   - o The following indicators are made available by the Read/Write_Specialist_Function:
     (a.) Main_Memory_Dump_Req
     (b.) Main_Memory_Dump_Start_Address
     (c.) Main_Memory_Dump_Length
   - o The following indicators are made available by the UDF application programs.
     (a.) Format ID(CDWD_OP_FORMAT_ID)

See Table 3.3.2.1 - 1

c. Process Description -

On initial pass through the program the data cycle header is created and saved.

The data cycle header is mvoed into Downlist_Buffer on frames 0 and 25.

If a Main_Memory_Dump is requested it starts on frame 0. A new length is computed each frame until length is 0 or negative (in which case it is set to 0). If length is 0 and frame is not 49 then no data is moved to Downlist_Buffer. If length is 0 and frame is 49 then the dump is finished and flags are reset to resume normal Downlist on next cycle. If length is not 0 then a SVC is issued to Program_Modification to get the needed dump data and it is moved to Downlist_Bufffer.

If Main_Memory_Dump is not requested then all 25 sample per second parameters are moved to Downlist_Buffer.

A series of counters are maintained to select which set of parameters to Downlist at the various rates.

If the prime GPC has changed then the toggle buffer assigned is changed if affected.

Downlist_Buffer is written out via SVC to I/O_SVC_Service_Processor and the frame count is updated.

The control flow of this module is presented in figures 3.3.2.1-1 & 2.

d. Outputs:

1. The GPC Prime Frame Output Buffer is written to the PCMMU.
2. "End of Message" message is written to the PCMMU(Ref Table 3.3.2.1-1).

e. Module References:

(325) Program_Modification(FCMPMOD) by SVC (200) I/O_SVC_Service_Processor (FIOSVC) by SVC.

f. Module Attributes: Program

g. Template References:

1.	UI_FCOS_Shared_Compool	(CZ2-COMMON)
2.	UI_Section_Of_Common_Compool	(CZ1-COMMON)
3.	FCOS_Compool	(FCMCOM)
4.	Annunication_Compool	(CDL_ANNUN)
5.	UI-General_Compool	(CDM_UI__COMPOOL)
6.	GN & C_Compool	(CG1-GNC)
7.	CAL_Compool	(CG1-GNC)
8.	Downlist_Compool	(CDW_DOWNLIST_COMPOOL)
9.	I/O_SVC_Parameter_List	(IOMACS)
10.	I/O_SVC_Parameter_List	(PMUMACS)
11.	I/O_SVC_Parameter_List	(PMODMACS)

h. Error Handling: None

i. Constraints and Assumptions: None

j. Detailed Implementation: None

DATA TABLE 3.3.2.1-1

NAME DOWNLIST_FORMATTER_1 (DCDDWL1)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	ASSEMBLER NAME	MML	D	C
1	DOWNLIST_BUFFER	F010	Z	660,665		CDWD_DWNLST		X	
2	DOWNLIST_RATE_12_COUNTER	F030.03	Z	660,665		CDWD_CNTR_12			
3	DOWNLIST_RATE_5_COUNTER	F030.02	Z	660,665		CDWD_CNTR_5			
4	DOWNLIST_RATE_1_COUNTER	F030.01	Z	660,665		CDWD_CNTR_1			
5	DOWNLIST_FRAME_NUMBER	F100	Z	660,665		CDWD_DCDVL_FRME_CNT		X	
6	MAIN_MEMORY_DUMP_REQ	F040	Z	660,665 910		CDWD_MAIN_MEM_DMP_REQ			
7	MAIN_MEMORY_DUMP_ACTIVE	F050	Z	660,665		CDWD_MAIN_MEM_DMP_IND			
8	MAIN_MEMORY_DUMP_LENGTH	F070	Z	660,665 910	915	CDWD_MAIN_MEM_DMP_LEN			
9	MAIN_MEMORY_DUMP_START_ADDRESS	F060	Z	660,665 910	915	CDWD_MAIN_MEM_DMP_PTR			

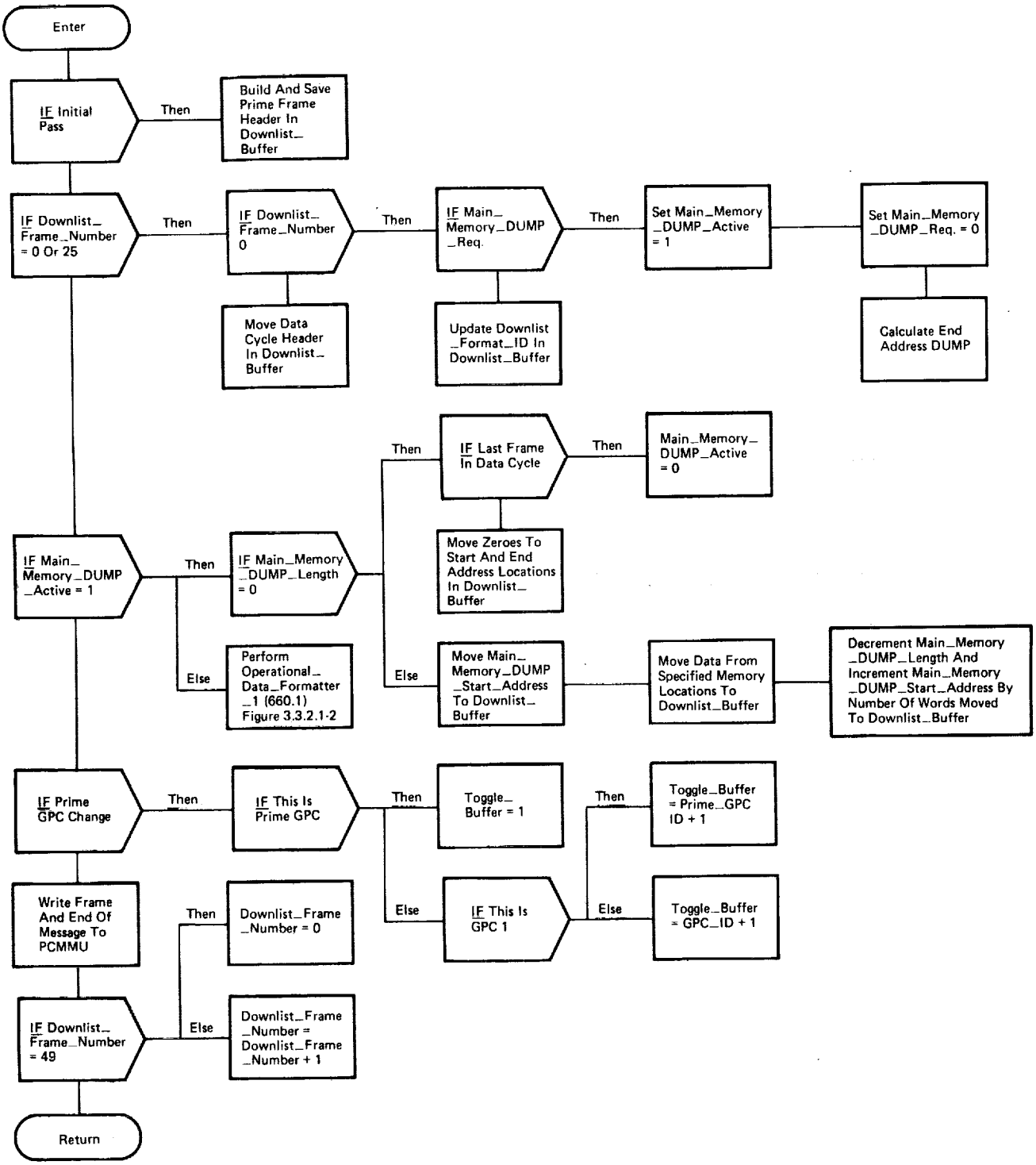


Figure 3.3.2.1-1. Downlist_Formatter_1 (DCDDWL1)

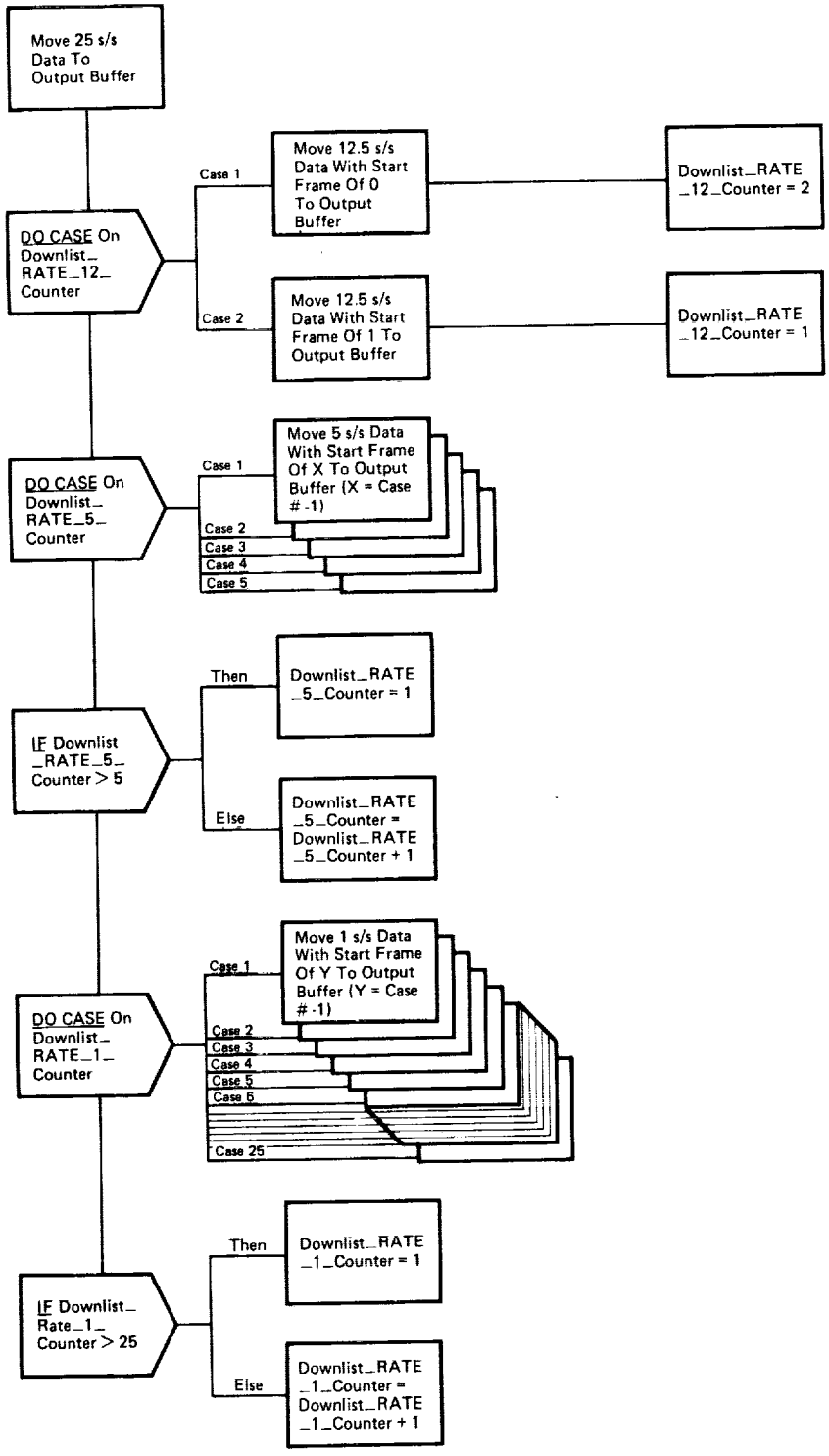


Figure 3.3.2.1-2. Downlist_Formatter_1
Operational_Data_Formatter_1 (660.1)

3.3.2.2 Downlist_Formatter_2 (DCDDWL2) (665)

The Downlist Formatter controls, collects, formats and transfers downlist data in OPS 2.

a. Control Interface:

1. CALL DCDDWL2 (PRIO_DCD)

   a. CALLed by (750) System_Interface Processor (AIE_SIP)
   b. CALLed by (710) GPC_Startup (AIR_GPC_STARTUP)
   c. CALLed by (820) GPC_Reconfiguration (ARC_RECONFIG)
   d. CALLed by (880) Secondary_GPC_Reconfiguration (ARH_SEC_GPC_RECONFIG)

b. Input: The following items are data inputs to the Downlist Formatter and reside in various compools.

   1. Applications and system software data to be Downlisted.

   2. PCMMU I/O Table (CDWS_PMU_WRITE)-defines the GPC to PCMMU data transfer process.

   3. Control indicators specifying pending changes to be effected in the downlist processes.
      - The following indicators are made available by the Read/Write_Specialist_Function:
      (a) Main_Memory_Dump_Req
      (b) Main_Memory_Dump_Start_Address
      (c) Main_Memory_Dump_Length

      - The following indicators are made available by the UDF application programs

      (a) Format ID (CDWD_OP_FORMAT_ID)

   See Table 3.3.2.2 - 1

c. Process Description:
On initial pass through the program the data cycle header is created and saved.

The data cycle header is moved into Downlist_Buffer on frames 0 and 25. If a Main_Memory_Dump is requested it starts on frame 0. A new length is computed each frame until length is 0 or negative (in which case it is set to 0). If length is 0 and frame is not 49 then no data is moved to Downlist_Buffer. If length is 0 and frame is 49 then the dump is finished flags are reset to resume normal Downlist on next cycle. If length is not 0 then a SVC is issued to Program_Modification to get the needed dump data and it is moved to Downlist_Buffer.

If Main_Memory_Dump is not requested then all 25 sample per second parameters are moved to Downlist_Buffer.

A series of counters are maintained to select which set of parameters to Downlist at the various rates.

If the prime GPC has changed then the toggle buffer assigned is changed if affected.

Downlist_Buffer is written out via SVC to I/O_SVC_Service_Processor and the frame count is updated.

The control flow of this module is presented in figures 3.3.2.2- 1 & 2

d. Outputs:
   1. The GPC Prime Frame Output Buffer is written to the PCMMU.
   2. "End of Message" message is written to the PCMMU.
      (See Table 3.3.2.2 - 1)

e. Module References:
(325)Program - Modification (FCMPMOD) by SVC (200) I/O_SVC_Service_Processor (FIOSVC)by SVC.

f. Module Attributes:
Program

g. Template References:

1.	UI_FCOS_SHARED_COMPOOL	(CZ2_COMMON)
2.	UI_SECTION_OF_COMMON_COMPOOL	(CZ1_COMMON)
3.	FCOS_COMPOOL	(FCMCOM)
4.	ANNUNCIATION_COMPOOL	(CDL_ANNUN)
5.	UI_GENERAL_COMPOOL	(CDM_UI_COMPOOL)
6.	GN & C_COMPOOL	(CG1-GNC)
7.	CAL_COMPOOL	(CGM_CAL)
8.	DOWNLIST_COMPOOL	(CDW_DOWNLIST_COMPOOL)
9.	I/O_SVC_PARAMETER_LIST	(IOMACS)
10.	I/O_SVC_PARAMETER_LIST	(PMODMACS)
11.	I/O_SVC_PARAMETER_LIST	(PMUMACS)

h. Error Handling: None

i. Constraints and Assumptions: None

j. Detailed Implementation: None

DATA TABLE 3.3.2.2-1

NAME DOWNLIST_FORMATTER_2 (DCDDWL2)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	DOWNLIST_BUFFER	F010	Z	660, 665		CDWD_DWNLST		X	
2	DOWNLIST_RATE_12_COUNTER	F030.03	Z	660, 665		CDWD_CNTR_12			
3	DOWNLIST_RATE_5_COUNTER	F030.02	Z	660, 665		CDWD_CNTR_5			
4	DOWNLIST_RATE_1_COUNTER	F030.01	Z	660, 665		CDWD_CNTR_1			
5	DOWNLIST_FRAME_NUMBER	F100	Z	660, 665		CDWD_DCDVL_FRME_CNT		X	
6	MAIN_MEMORY_DUMP_REQ	F040	Z	660,665 910		CDWD_MAIN_MEM_DMP_REQ			
7	MAIN_MEMORY_DUMP_ACTIVE	F050	Z	660, 665		CDWD_MAIN_MEM_DMP_IND			
8	MAIN_MEMORY_DUMP_LENGTH	F070	Z	660,665 910	915	CDWD_MAIN_MEM_DMP_LEN			
9	MAIN_MEMORY_DUMP_START_ADDRESS	F060	Z	660,665 910	915	CDWD_MAIN_MEM_DMP_PTR			

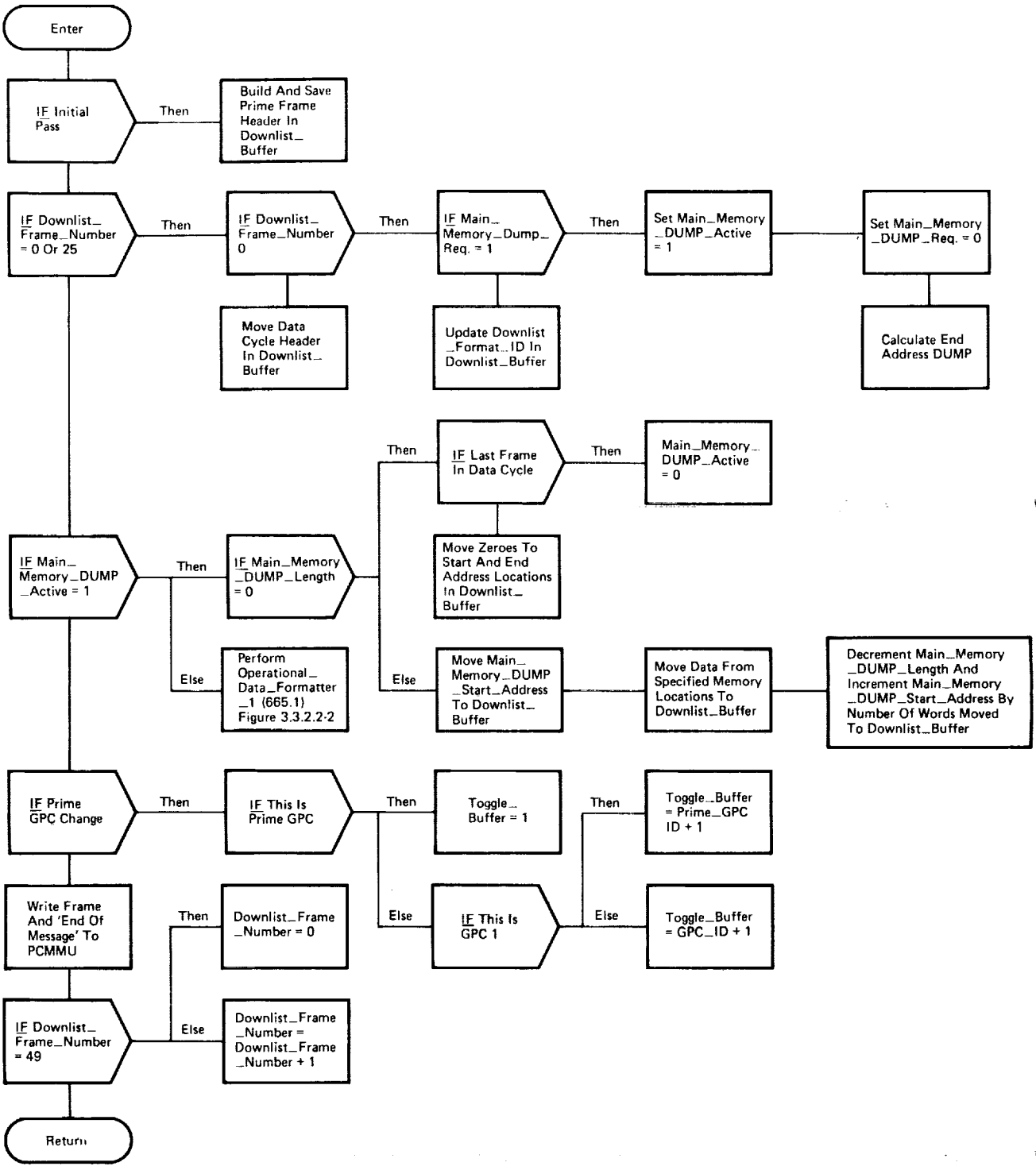


Figure 3.3.2.2-1. Downlist_Formatter_2 (DCDDWL2)

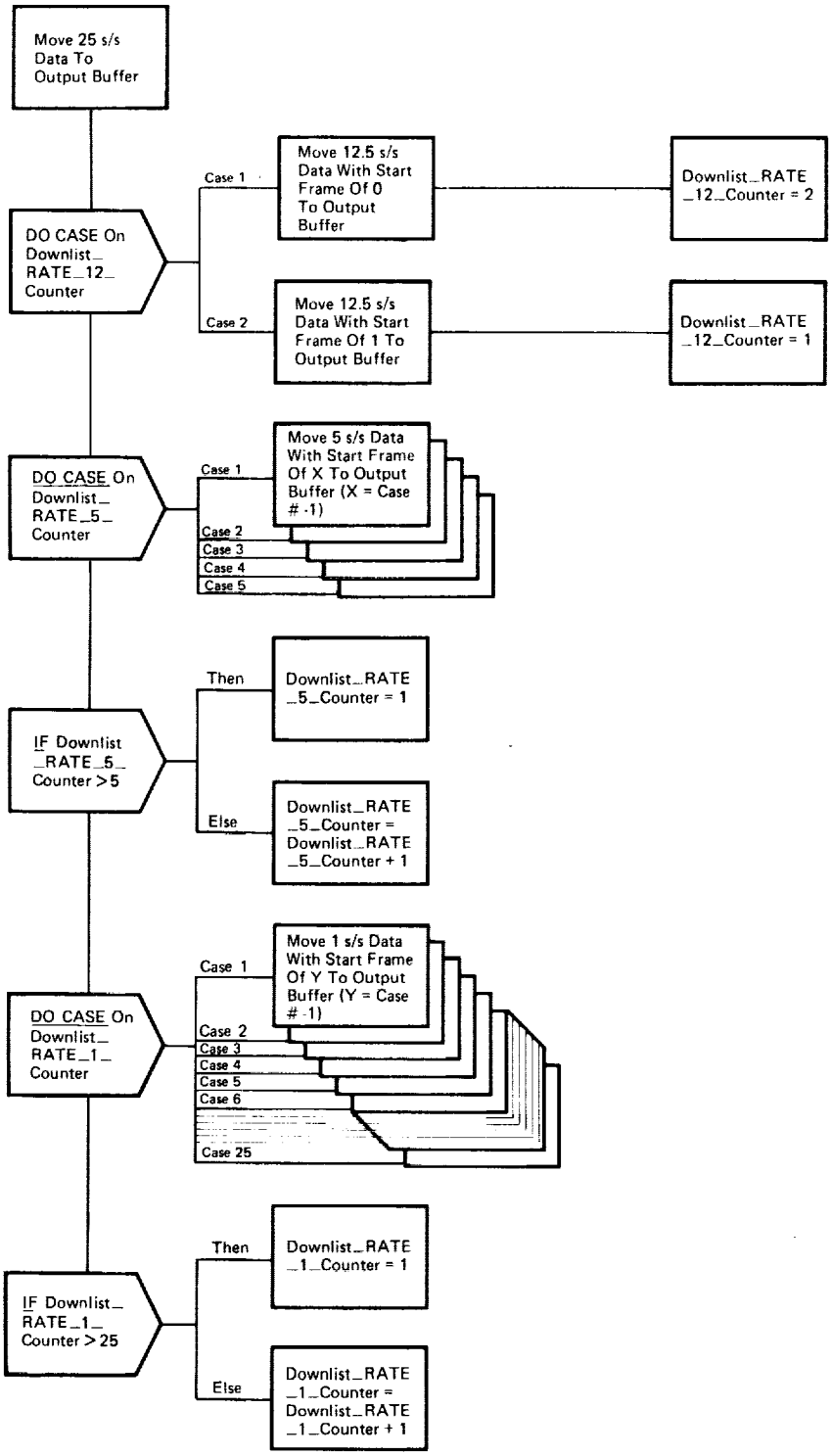


Figure 3.3.2.2-2. Downlist_Formatter_2 Operational_Data_Formatter_2 (665.1)

3.3.3 LDB Interface

For presentation continuity, the "output" portion of the LDB Interface is described in Section 3.1.2.2.

3.3.4 Annunciation Support

The Annunciation Support software provides the capability to annunciate application and system software-detected faults by controlling alarms, lights and fault summary messages. Operator error message support is also included. The Fault Summary Page display which contains a history of fault messages is maintained by this function. There are three modules included in this support function. Figure 3.3.4-1 presents a hierarchial diagram of Annunciation Support.

a. Fault_Message_Scan provides a cyclic interface to recognize enabled messages and transfer each to the ICC buffer. Annunciation Macro Interface provides support for the annunciation macros by enabling fault message flags for later processing.

b. Error_Message_Line_Support displays messages on the error message line of the CRT. This includes Caution and Warning indicator messages, alert messages, GPC-detected error messages and operator error messages. FSP_Processor controls, updates, and maintains the Fault Summary Page buffer and makes it available for display upon keyboard request.

c. Lights_and_Alarm_Processor performs the commanding of lights and alarms via the Payload MDMs in response to Caution and Warning indicator messages (Class 2) or alert messages (Class 3.)

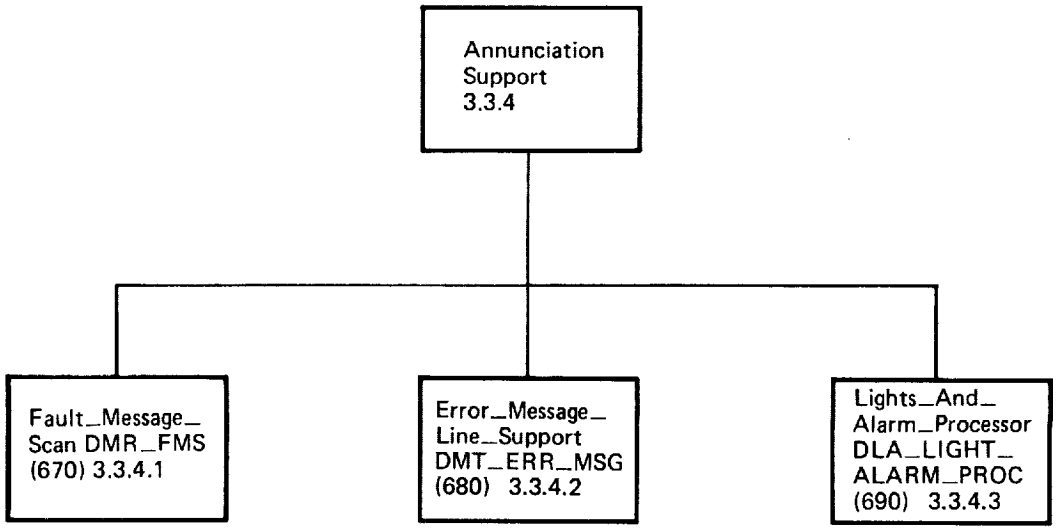


Figure 3.3.4-1. Annunciation Support Hierarchy

3.3.4.1 Fault_Message_Scan (DMR_FMS) (670)

This module is responsible for scanning FMPT_Header_Bits for bits that have been enabled, indicating there is an error message or a keyboard request to be processed. This module does time tagging and performs the interlock test on fault messages.

a. Control Interface -

1. Call DMR_FMS
2. Called by (750) System_Interface_Processor (AIE_SIP) when an annunciation request is detected by System_Interface_Processor.

b. Input - See Table 3.3.4.1-1.

c. Process Description - This module when invoked will scan the FMPT_Header_Bits for enabled bits, which corresponds to an annunciation request. The message class, downlist time, display time, and the dictionary index are put in a parameter list and a call to ICC_Collector is made.

The control flow for this module is presented in Figures 3.3.4.1-1.

d. Outputs - See Table 3.3.4.1-1 and ICC_Message_Tables.

e. Module References -

(144) TIME/DATE_Application_Requests_Processor (FPMTMHAL)
(485) ICC_Message_Collector_For_SIP (DIN_ICC_SIPCOLL)

f. Module Attributes - External Procedure

g. Template References - ANNUNCIATION_COMPOOL (CDL_ANNUN)

h. Error Handling - None

i. Constraints and Assumptions - Only two fault messages are processed per SSIP cycle.

j. Detailed Implementation - Expanded flows are used to break the module flow chart into more descriptive detail.

DATA TABLE 3.3.4.1-1

NAME Fault_Message_Scan (DMR_FMS)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	START_INDEX	L043	I	670	670	DMR_X			
2	Copy_Number_Of_Header_Words	L044	I	670	670	DMR_COPY			
3	MAX_Number_Of_MSG_Per_SSIP_Cycle	L045	I	670	670	DMR_MAX_MSG			
4	MAX_Number_Of_MSG_IN_Memory	L046	I	670	670,675 680	CDLK_HDR_BITS			
5	Least_Number_Of_MSG_IN_Memory	L047	I	670	670	DMR_I			
6	Counter_Within_DMR_X_and_DMR_Copy	L048	I	670	670	DMR_J			
7	Interlock_Value_In_SSIP_Cycles	L049	I	670	670	CDL_INTERLOCK_TIME			
8	MET_TIME	L050	I	144 670	670,680 685	DMR_CLOCKTIME			
9	TIME_DISPLAYED_ON_CRT	L051	I	670	490,620 670	DMR_DISPLAY_TIME			
10	TIME_IN_SSIP_CYCLES	L052	I	670	670	DMR_MICS512			
11	TIME_Put_IN_Downlist_Buffer	L053	I	670	670,680 685	DMR_Downlist_Buffer			
12	Call_Collector_Bit	L054	I	670	670	DMR_CALL_BIT			
13	Clocktime_Minus_Last_Mag_Time	L055	I	670	670	DMR_DELTA_TIME			
14	Last_MSG_TIME	L056	I	670	670	DMR_Last_MSG_TIME			

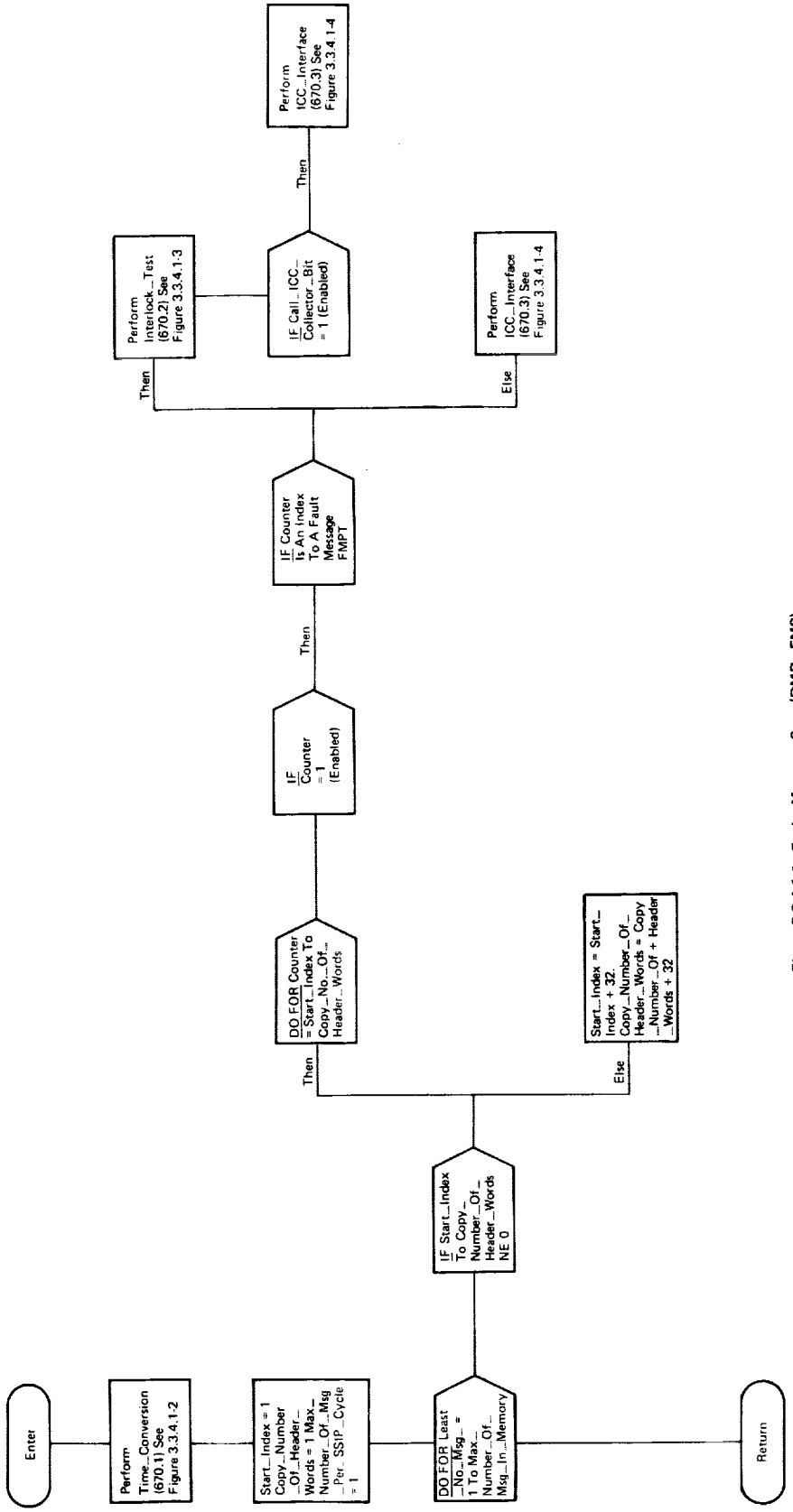


Figure 3.3.4.1-1. Fault_Message_Scan (DMR_FMS)

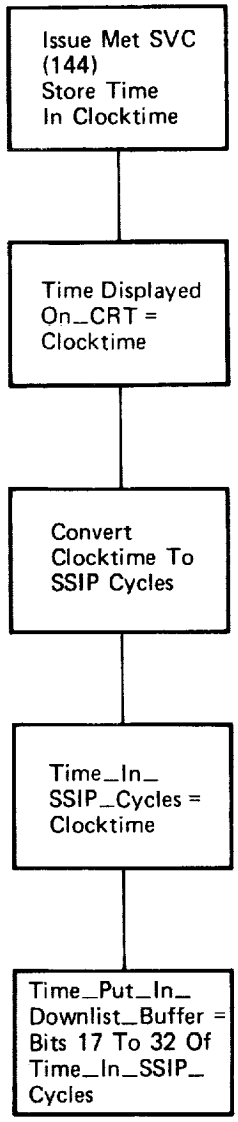


Figure 3.3.4.1–2. Fault_Message_Scan Time_Conversions (670.1)

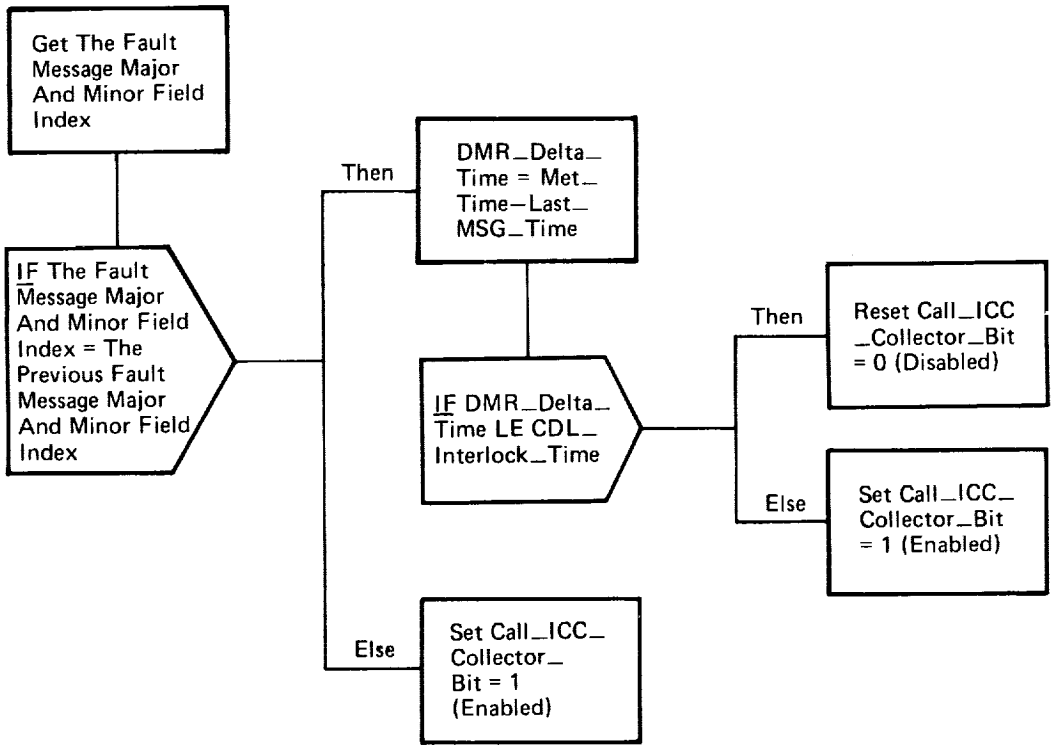


Figure 3.3.4.1–3. Fault_Message_Scan Interlock_Test (670.2)

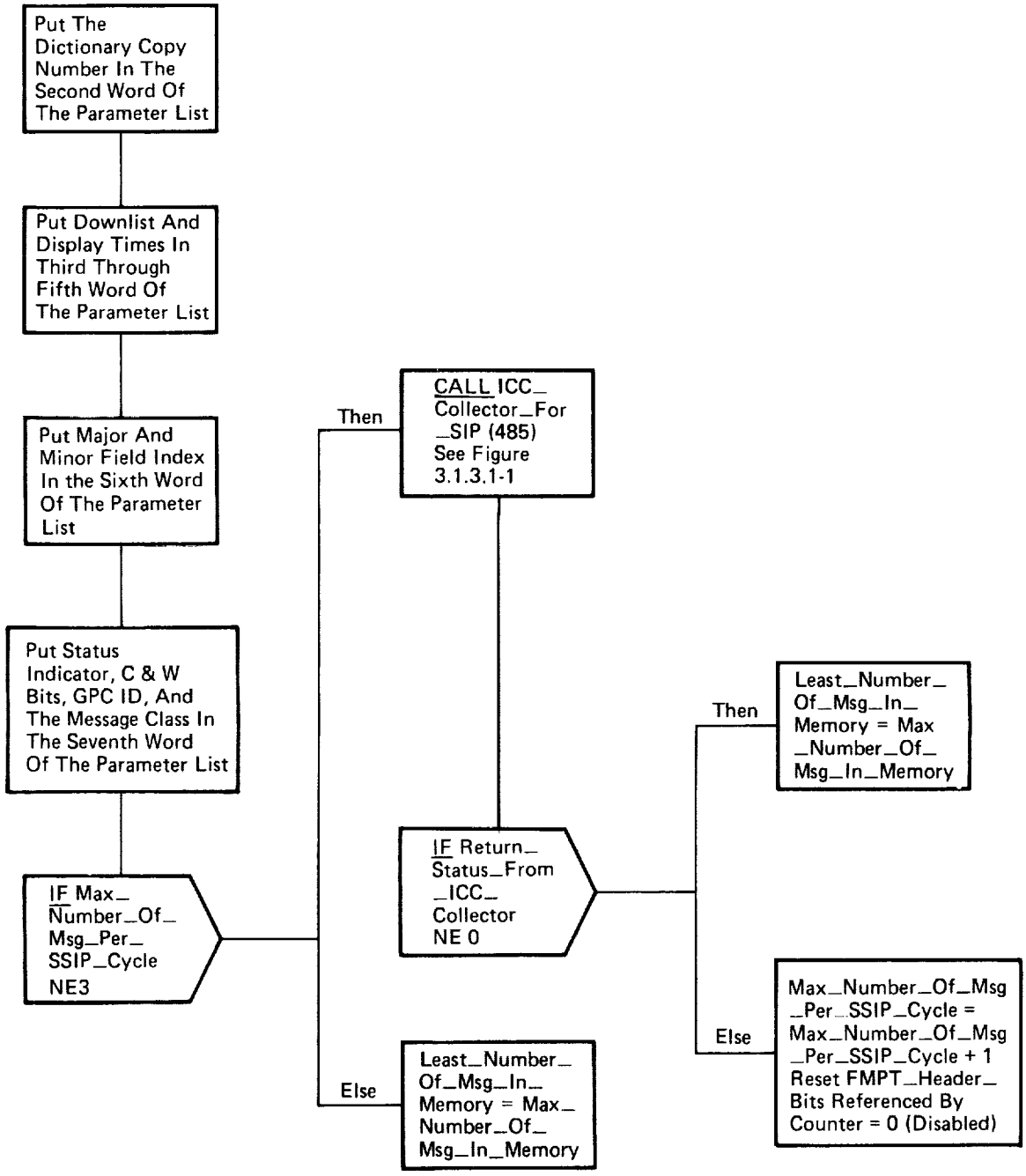


Figure 3.3.4.1–4. Fault_Message_Scan ICC_Interface (670.3)

3.3.4.1.1 Annunciation Macro Interface (DMA_MAC) (675)

This module is the interface module for applications and system services modules requesting annunciation of fault messages, operator error, or response to an annunciation related keyboard request.

a. Control Interface

1. This module is referenced as a macro.
2. It is referenced by application and system service modules.

b. Input - See Table 3.3.4.1-1.

c. Process Description - Three parameters are passed to this module. These parameters FMPT_Index,Message_Type, and Parameter_Status are used in the following manner.

The FMPT_Index is an index to the FMPT containing information that is to be annunciated. The Message_Type indicates if a message is from System Software or Application Software, and if the message is an operator error message, Message Reset, Acknowledge or DISP051 input. The Parameter_Status indicates out-of-limit status for class two fault messages.

The control flow for this module is presented in Figure 3.3.4.1-1.

d. Outputs - See Table 3.3.4.1-1.

e. Module References - None

f. Module Attributes - Program

g. Template References - Annunciation_Compool (CDL_ANNUN)

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.3.4.1.1-1

NAME Annunication_Macro_Interface (DMA_MAC)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	FMPT_Index	L026		620	620,625 640,675	DMA_I			
2	Message_Type	L025		620	620 670	DMA_D			
3	Parameter_Status	L027		675	675	DMA_S			
4	FMPT_Hdr_Bits	T070		See APP. E		CDL_FMPT_HDR			

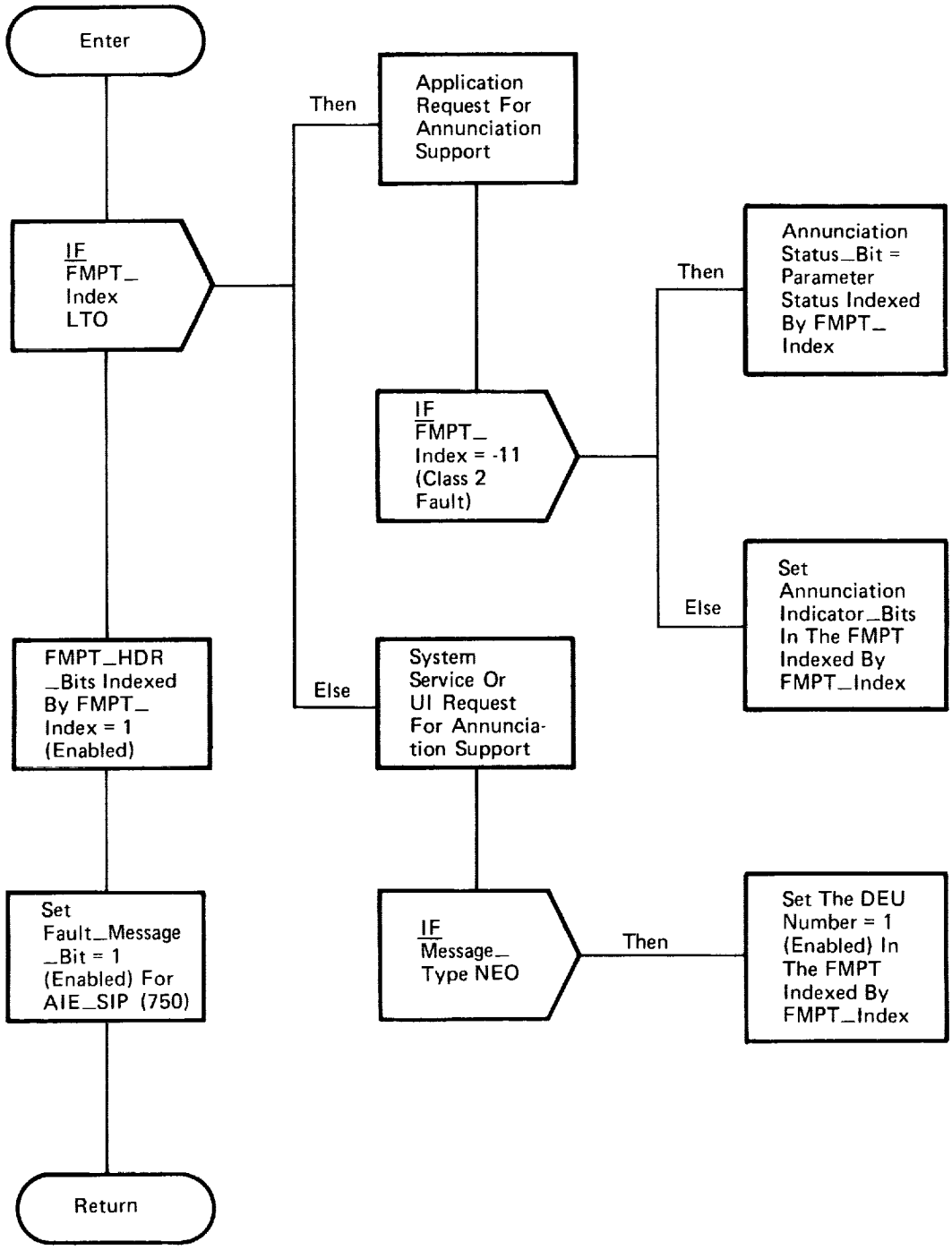


Figure 3.3.4.1.1–1. Annunciation_Macro_Interface (DMA_MAC)

3.3.4.2 Error_Message_Line_Support (DMT_ERR_MSG) (680)

This module is responsible for processing Message Reset key requests, Acknowledge Key requests, DISP051 requests, FAULT Key requests, and OPERATOR ERRORS (Class 5).

a. Control Interface -

1. Call DMT_ERR_MSG
2. Called by (490) ICC_MESSAGE_ROUTER (DME_ICC_ROUT)

b. Input - See Table 3.3.4.2-1.

c. Process Description - This module clears the fault summary page display of all fault messages and the message line of the current message when a DISP051 PRO is received via the MCDS Keyboard. When a message reset key is the input this module clears the message line of the current fault or class 5 error message, then searches the fault summary display for the oldest unannunciated class two fault message, if one exists it is displayed on the message line, otherwise it searches the FSP for the oldest unannunciated class 3 or 4 fault message and displays the fault message on the message line, it also updates the stack counter to the extreme right portion of the message line with the numerical value of unannunciated messages left in the FSP. An event is set for lights and alarms indicating the alarm associated with the previous error message should be turned off and the new light or tone invoked.

In the event of an acknowledge key depression by the user, the current message on the fault message line ceases flashing and the tones or lights associated with the fault message are commanded off by this module. Setting an event Light_and_Alarm_EVT for lights and alarm processing is the method of invocation used.

When an error condition is discovered by system or application software and the error condition warrants annunciation to the crewman, then ERROR_Message_Line_Support finds the appropriate error message in CDL_ANNUN compool, associates a time for displaying on the CRT, places the Message ID and time in a prescribed format in UI_GENERAL_COMPOOL for downlisting, fills the GPC(s) ID slot, places the message in the fault summary page display buffer, updates the fault summary page directory, increments the stack counter and sets the error message flags (DEU_UPDATE_FLAGS) for Cyclic_Display_Processor to update the message line on its next cycle. Flags are set for Cyclic_Display_Processor anytime FSP_Processor is called. ICC_Message_Router is the only caller to this module.

The control flow for this module is presented in Figure 3.3.4.2-1.

d. Outputs - See Table 3.1.1.1-1.

e. Module References -

(685) DMT_FSP_Processor (FSP_PROCESSOR) is called.
(690) CDL_LAP_EVT is set for Lights_and_Alarm_Processor

f. Module Attributes - External Procedure

g. Template References -

1. UI_General_Compool (CDM_UI_COMPOOL)
2. ANNUNCIATION_Compool (CDL_ANNUN)

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.3.4.2-1

NAME ERROR_MESSAGE_LINE_SUPPORT (DMT_ERR_MSG)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	STACK_COUNTER	L057	O	680	680 685	DMT_STACK_CT			
2	DEU_UPDATE_FLAGS	L058	O	680	620,650 680,685	CDMB_MAT_CRT_STAT			
3	NO_FLASH_FCW	L059	O	680	680 685	DMT_ NOFLASH_FCW			
4	FLASH_FCW	L060	O	680	680 685	DMT_FLASH			
5	MESSAGE_CLASSES	L061	I	670	680 685	DMT_MSG_CLASS			
6	LIGHT_ALARM_PROCESSOR_EVENT	T060	O	670,680 685	690 750	CDL_LAP_EVT			

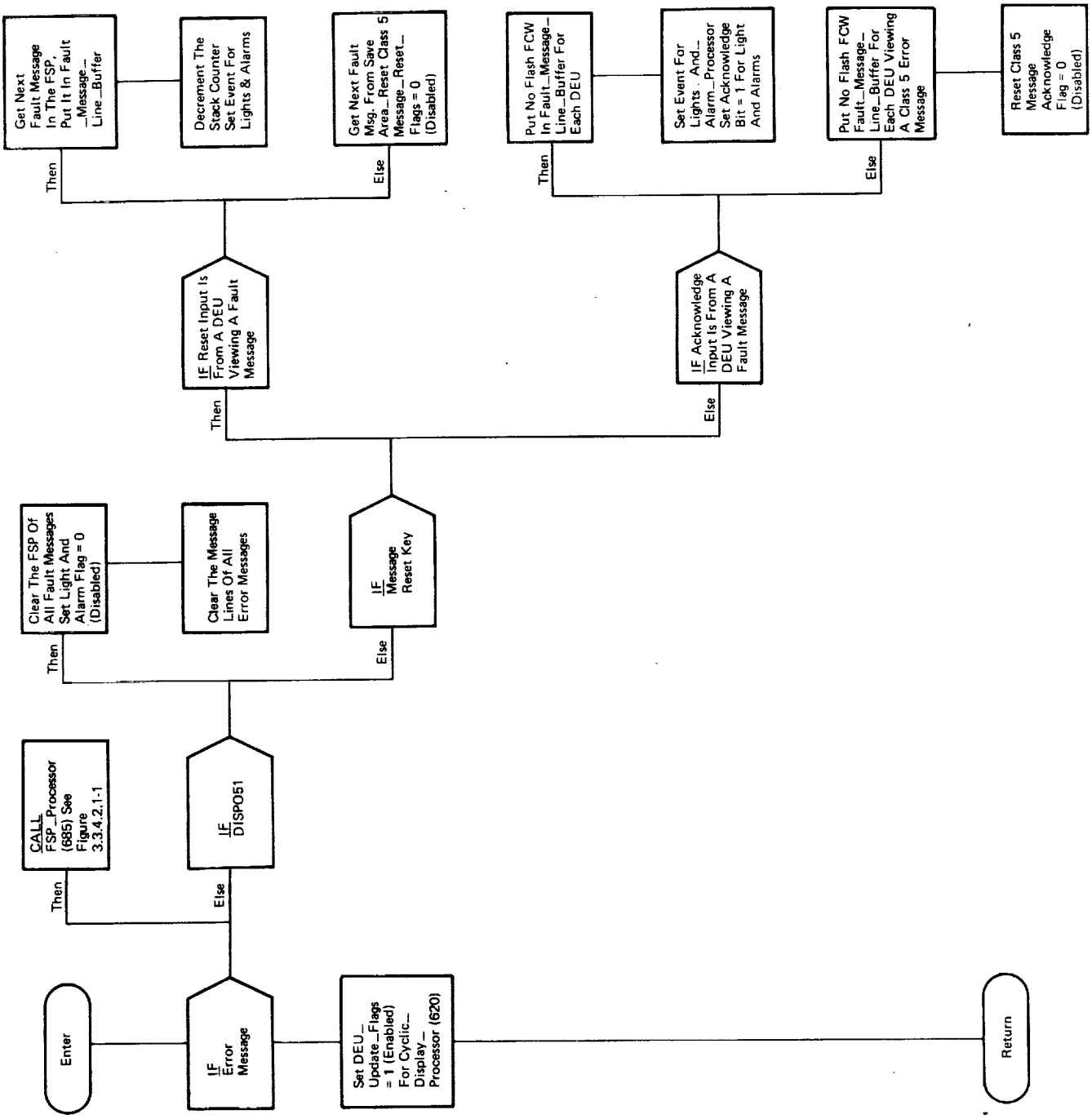


Figure 3.3.4.2-1. Error_Message_Line_Support (DMT_ERR_MSG)

3.3.4.2.1 FSP_Processor (DMT_FSP_PROCESSOR) (685)

This module is responsible for updating of the fault summary page and finding class five error messages in Annunciation_Compool (CDL_ANNUN).

a. Control Interface -

1. Call DMT_FSP_PROCESSOR
2. Called by (680) Error_Message_Line_Support (DMT_ERR_MSG)

b. Input - See Table 3.3.4.2.1-1.

c. Process Description - This module uses the major and minor field indexes to find the desired class five or fault message in the major and minor dictionaries. If the desired message is a class 5 (operator error, i.e., ILLEGAL ENTRY OPS) then the class 5 message is placed in the Fault_Message_Line Buffer and control is returned to DMT_ERR_MSG. If the desired message is a fault message then the fault message FCWS are placed in the Fault_Summary_Page and the associated time placed in the Fault_Message_Time_Buffer of ANNUNCIATION_COMPOOL. When a new fault message is added to the FSP the stack counter value is incremented and the five most current fault message IDs are placed in UI_Downlist_Buffer.

The control flow for this module is presented in Figure 3.3.4.2.1-1.

d. Outputs - See Table 3.3.4.2.1-1.

e. Module References - None

f. Module Attributes - Internal procedure

g. Template References - None

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.3.4.2.1-2

NAME FSP_Processor (DMT_FSP_PROCESSOR)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	Fault_Summary_Page_Buffer	L038	I/O	685	620 680 685	CDL_FAULT_ SUMMARY_PAGE			
2	Fault_Message_Time	L039	I/O	685	620,680 685	CDL_FAULT_ SUMMARY_TIME			
3	Message_Class	L040	I/O	670 675	485,670 680,685	CDL_MSG_CLASS			
4	Application_Interface_Table	L041	O	670,680, 685	485,670, 680,685	CDL_AIT			
5	FSP_Directory	L042	O	680 685	620,680 685	CDL_FSP_DIRECTORY			

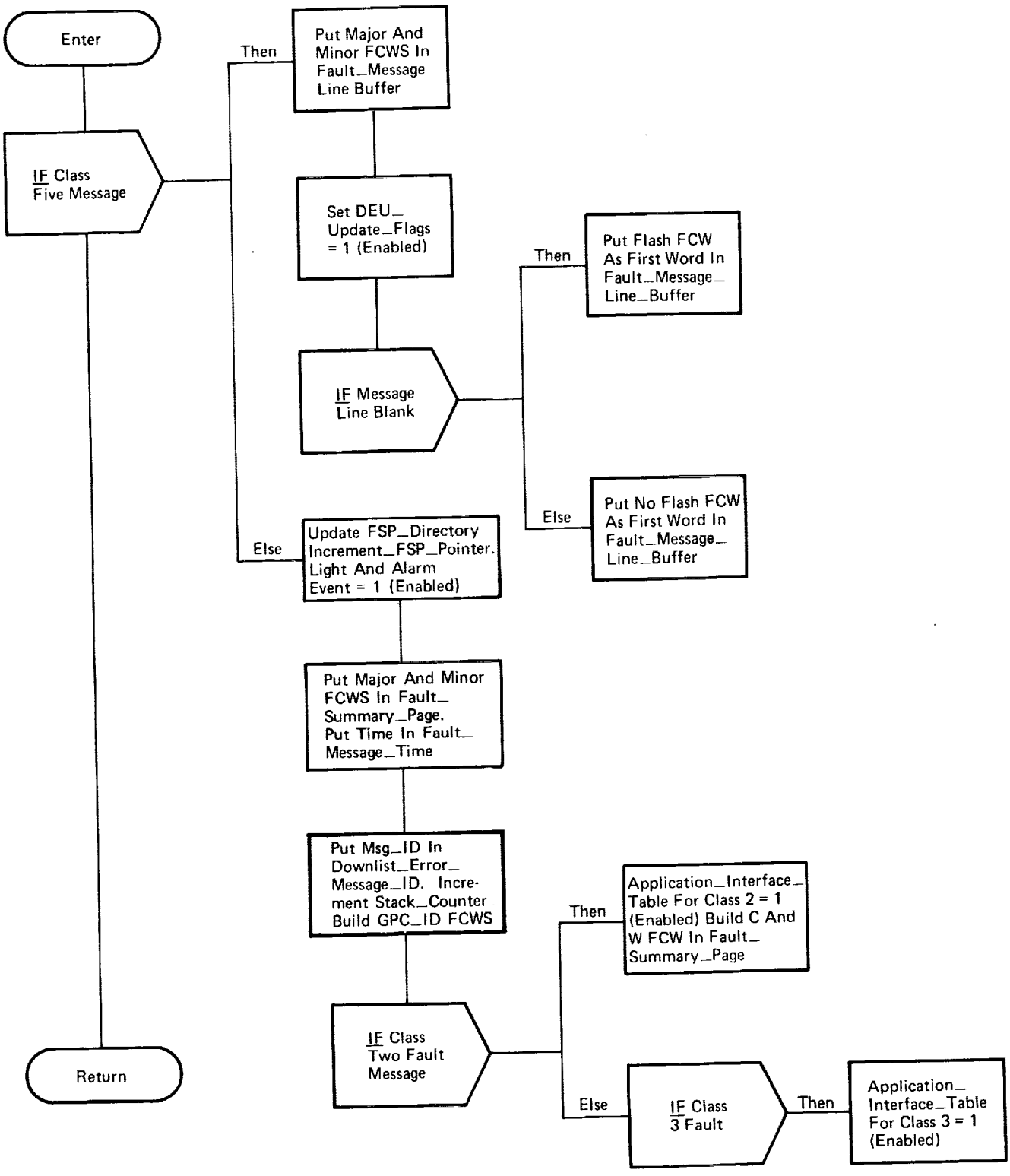


Figure 3.3.4.2.1–1. FSP_Processor (DMT_FSP_Processor)

3.3.4.3 Light_And_Alarm_Processor (DLA_LIGHT_ALARM_PROC) (690)

The Light_And_Alarm_Processor module performs the commanding of the lights and alarms to be turned off/on as a result of the entry of the Acknowledge or Message Reset key or tone processing. It also commands lights and alarms on/off upon the receipt of a Class 2 or 3 fault message.

a. Control Interface

1. CALL DLA_LIGHT_ALARM_PROC

2. (a) CALLed by (750) SYSTEM_INTERFACE_PROCESSOR (AIE_SIP)
   (b) Event LIGHT_ALARM_PROCESSOR_EVENT (CDL_LAP_EVT) set by (680) ERROR_MESSAGE_LINE_SUPPORT (DMT_ERR_MSG)

b. Input - See Table 3.3.4.3-1.

c. Process Description - Light_And_Alarm_Processor performs the logic necessary to command on a light or tone as a result of the receipt of a Class 2 or 3 fault message. It also commands the lights and tone off as a response to an 'Acknowledge' or 'Message Reset' key input or as a result of tone processing. The control flow for this module is shown in Figure 3.3.4.3

Acknowledge/Message Reset key Processing is performed as a result of user acknowledgement of an annunciated error, i.e., keyboard entry of 'Acknowledge' or 'Message Reset' key. Consequently, it is processed first in order to turn off any alarms which had been turned on by previous cycles. In order to turn off the alarms, two special FCOS command words must be formatted and then issued via an FCOS I/O macro. The format of the command words is explained in Section D.

It is also necessary to maintain a separate word for downlist (D/L) purposes to show the current status of a light or tone. Thus, if an indication is set to turn on/off a light/tone, the Lights_and_Alarms_ Processor sets indicators in the Downlist_LAM to show that a light/tone has been turned on/off.

If the 'Acknowledge' or 'Message Reset' key is entered by the crew, ERROR_MESSAGE_LINE_SUPPORT sets the REQUEST_FLAG (for the ACK key) or the MSG_RESET_FLAG (for the MR key) in the APPLICATION_INTERFACE_TABLE. This is an indication to the Lights_And_Alarm_Processor that ACK/MR_ KEY_PROCESSING must be performed. In the case of a MR key entry, the Lights_And_Alarms_Processor first sets indicators in the FCOS command words (FCOS_CWS_TO_PF1 and FCOS_CSW_to-PF1) to turn off all lights and tones. It is then necessary to set DOWNLIST_LAM to show all lights and alarms have been turned off. In the case of an ACK key entry,

indicators in the FCOS command words are set such that only the alert light and tone are turned off. These indicators are also cleared in DOWNLIST_LAM. In both cases, the FCOS command words are moved to the FCOS output buffers (FCOS_OUTPUT_BUFFER_TO_PF1 and FCOS_OUTPUT_BUFFER_TO_PF2). This is done so that FCOS can output the data to the Payload MDM's while the Lights_And_Alarm_Processor continues processing new faults. It then issues the FCOS I/O macros. Communication flags in the APPLICATION_INTERFACE_TABLE must be reset to indicate ACK/MR_KEY_PROCESSING has been accomplished.

The Lights_And_Alarms_Processor maintains timing sequences (i.e., it ensures that it is called by the System_Interface_Processor_ in subsequent SIP cycles) in the following cases:

1. A Class 2 alarm is to be enabled - A Class 2 alarm must first be turned off when the indication to enable it is first received and then 200ms later must be turned on. (This is referred to as a 'command sequence' in subsequent paragraphs.) The Class 2 timing sequences are accomplished by maintaining the COMMAND_SEQ_FLAG in the APPLICATION_INTERFACE_TABLE whenever a new Class 2 fault is to be enabled. This flag is updated and maintained by L/A. A Class 2 fault for which a new indication to annunciate is received and that is in a command sequence is <u>not</u> reannunicated.

2. Class 3 alert tone duration - The alert tone must remain on for a specified length of time. This time period is called the alert tone duration (LAP_LIMIT) and may be updated by the user. L/A ensures that it is called by the System Interface Processor while the tone is on so that it can monitor the amount of time the tone has been on and reset the tone after it has been on the specified amount of time.

If no new fault indicators have been set, the tone is off, and no class 2 alarm is in a command sequence, then there is no need to perform any other Class 2 or 3 annunciation . However, if any of these conditions is met, then the FCOS command words must be formatted and issued.

If there is any of type of annunciation to be done on the Backup C&W indicator, the FCOS command words are set to enable the alarm. Otherwise, the FCOS command words used for the alert tone and Class 2 faults are cleared for further Lights_And_Alarms_Processing. In either case, the FCOS command words used for the alert light are also cleared for further processing.

If the tone has been driven the alert tone duration time, the FCOS command words are formatted so as to turn the tone off. Tone indicators are reset in the APPLICATION_INTERFACE_TABLE and DOWNLIST_LAM to show that the tone has been turned off. It is also necessary to set the TONE_BIT to indicate that there is an alarm to be enabled through the FCOS I/O macros.

If any Class 2 alarms are in a command sequence, the FCOS command words are set to turn on these Class 2 alarms.

Before processing any new alarms, the Light_And_Alarm_Processor moves all indicators into a local hold area and zeros all the original indicators. This allows ERROR_MESSAGE_LINE_SUPPORT to set indicators for new faults while the Light_And_Alarm_Processor continues processing old faults.

To determine which Class 2 alarms (if any) are to be enabled, each CLASS 2_ALARM_IND is interrogated. If it is found to be in a command sequence, the indicator is cleared so that reannunication of this alarm does not occur. If it is a Class 2 alarm that is not in a command sequence, the appropriate indicators are set in the FCOS command words. DOWNLIST_LAM is also set to show that the alarm is enabled.

If a Class 3 fault has occurred, the alert light indicators in the FCOS command words as well as in DOWNLIST_LAM must be set. In addition, for a Class 3 fault, the alert tone indicators in the FCOS command words and DOWNLIST_LAM must be set only if the alert tone capability is not inhibited and if the tone is not presently on. The LAP_COUNT in the APPLICATION_INTERFACE_TABLE is also reinitialized for the tone.

The Lights_And_Alarms_Processor then moves the FCOS command words to the FCOS output buffers and issues the I/O macros.

If any of the new Class 2 alarms are now in a command sequence, the COMMAND_SEQ_FLAG in the APPLICATION_INTERFACE_TABLE is set to indicate to the Light_And_Alarm_Processor that there are Class 2 indicators in a command sequence. A mapping of those Class 2 alarms in a command sequence is maintained as well as a mapping of these alarms as set in the FCOS command words.

The analog LAM is processed only when System Management has indicated to the Lights_And_Alarms_Processor that the meters are to be driven. This is done by checking the METERS_IND. The meters are driven by inserting the values found in LAM_VALUE into the FCOS_ANALOG_CWS. The command is then issued via an FCOS I/O macro. NOTE: Meters Processing is N/A for ALT.

Prior to exiting, the Light_And_Alarm_Processor resets the Light_And_Alarms_Processor_Event (monitored by the System Interface_Processor to determine if Lights_And_Alarms_Processor should be called in subsequent SIP cycles) only if no additional tone or command sequence processing is necessary. In this case, it does not need to be called into execution until a new fault is to be annunicated and/or acknowledge/MR key processing is to be performed.

d. Outputs - Two different command words are formatted for the Payload MDM's (one corresponds to PF1 and the other to PF2). The Command word to PF1 contains indicators for all Class 2 and 3 alarms, i.e., the Backup C&W indicator plus the appropriate GNC or Payload indicator for Class 2 alarms, and the alert light and tone for class 3 alarms. The command word to PF2 contains indicators for the Class 2 Backup C&W indicator alone and the Class 3 alarms (alert light and tone). Each command word consists of 4 words of data as defined in the Data Descriptors Table (See FCOS_CWS_TO_PF1 and FCOS_CWS_TO_PF2).

If a light or alarm is to be turned off or on, the corresponding bit (as found in the LIGHT_ALARM_XREF_TABLE) must be set in the appropriate data word of FCOS_CWS_TO_PF1 and FCOS_CWS_TO_PF2. The following table summarizes the words used in the FCOS command words for each alarm:

DEVICE	ALARM	WORD BIT IS SET IN TO TURN ALARM OFF	WORD BIT IS SET IN TO TURN ALARM ON
PF1	Alert tone	RESET_PF1	SET_PF1
	Alert Light	ALERT_RESET_PF1	ALERT_SET_PF1
	Class 2, Indicators 1-7	RESET_PF1	SET_PF1
	Class 2, Ind 0 (SM Backup C&W Ind)	SET_PF1	RESET_PF1
PF2	Alert tone	RESET_PF2	SET_PF2
	Alert Light	ALERT_RESET_PF2	ALERT_SET_PF2
	Class 2, Indicators 1-7	—	—
	Class 2, Ind 0 (SM Backup C&W Ind)	SET_ PF2	RESET_PF2

Note: 1. The bit to be set is obtained from the LIGHT_ALARM_XREF_TABLE.

2. Class 2, Indicators 1-7 are output only via PF1.

For other Ouputs, see Table 3.3.4.3-1.

e. Module References-

1. (170) SET_EVENT_PROCESSOR (FPMSET) is called.

2. (171) RESET_EVENT_PROCESSOR (FPMRESET) is called.

3. (201) PRE_INITIALIZED_I/O_SVC_PROCESSOR (FIOSVCP) is called.

f. Module Attributes - External Procedure

g. Template References -

1. CDLANNUN (CDL_ANNUN)

h. Error Handling - None

i. Constraints and Assumptions -

1. If the bit locations to be set in the FCOS command words change, the LIGHT_ALARM_XREF_TABLE must be changed.

2. If the Payload MDM addresses change, FCOS will change, not the Light_And_Alarm_Processor.

j. Detailed Implementation - NONE

DATA TABLE 3.3.4.3-1

NAME Light_And_Alarm_Processor (DLA_LIGHT_ALARM_PROC)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	REQUEST_FLAG	T040.03	I,O	680	690	CDL_REQUEST			
2	MSG_RESET_FLAG	T040.04	I,O	680	690	CDL_MSG			
3	FCOS_CWS_TO_PF1	L268	L	690		DLA_PF1			
4	FCOS_CWS_TO_PF2	L269	L	690		DLA_PF2			
5	DOWNLIST_LAM	T050.09	O	690		CDL_DL_LAM	V92M8975P	X	
6	LIGHT_ALARM_XREF_TABLE	L270	L		690	DLA_LXR			
7	ALERT_LITE_DL_IND	T050.11	O	690		CDL_DL_LAM	V92X2400X	X	
8	ALERT_TONE_DL_IND	T050.10	O	690		CDL_DL_LAM	See Appendix E	X	
9	FCOS_OUTPUT_BUFFER_TO_PF1	L271	L	690		PF1_OUT			
10	FCOS_OUTPUT_BUFFER_TO_PF2	L272	L	690		PF2_OUT			
11	APPLICATION_INTERFACE_TABLE	T040	I,O	680, 685,690	690	CDL_AIT			
12	TONE_IND	T040.01	I,O	690	690	CDL_TONE_IND			
13	RECEIVE_FLAG	T040.02	O	690		CDL_RECEIVE			
14	BACKUP_C&W_ALARM_IND	T050.06	I,O	685	690	CDL_LAM_ALARMS	V72X7143Y V72X7144Y		
15	LAP_COUNT	T040.07	I,O		690	CDL_LAP_COUNT			
16	LAP_DECREMENT_VALUE	T040.08	I		690,750	CDL_LAP_DEC			
17	TONE_BIT	T050.03	I,O	685	690	CDL_LAM_ALARMS	V72X7141X V72X7142X		
18	COMMAND_SEQ_FLAG	T040.05	I,O	690	690	CDL_CMD_SEQ			
19	LAM_ALARMS	T050.02	I,O	685	690	CDL_LAM_ALARMS			
20	CLASS2_MAP_TO_PF1	L266	L	690	690	DLA_RPF1			

DATA TABLE 3.3.4.3-1 (cont'd)

NAME Light_And_Alarm_Processor (DLA_LIGHT_ALARM_PROC)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
21	CLASS2_MAP_TO_PF2	L267	L	690	690	DLA_RPF2			
22	LAM_ALARMS2	L265	L	690	690	DLA_LAM_ALARMS2			
23	CLASS2_ALARM_IND	T050.05	I,O	685	690	CDL_LAM_ALARMS	See Appendix E		
24	CLASS2_DL_IND	T050.12	O			CDL_DL_LAM	See Appendix E		
25	BACKUP_C&W_DL_IND	T050.13	O	690		CDL_DL_LAM	V92X2405X	X	
26	LAP_LIMIT	T040.06	I	ILD	690	CDL_LAP_LIMIT	V92W2005C		
27	LAM_ALARMS3	T050.08	I,O	690	690	CDL_LAM_ALARMS3			
28	METERS_IND	T055	I,O		690	CDL_METERS_IND			
29	LAM_VALUE N/A for ALT	T050.07	I		690	CDL_LAM_VALUE	V7296001V V7296002V V7296003V		
30	FCOS_ANALOG_CWS	L273	L		690	DLA_APU_WRITE			
31	LIGHT_ALARM_PROCESSOR_EVENT	T060	O	670,680 685	690 750	CDL_LAP_EVT			

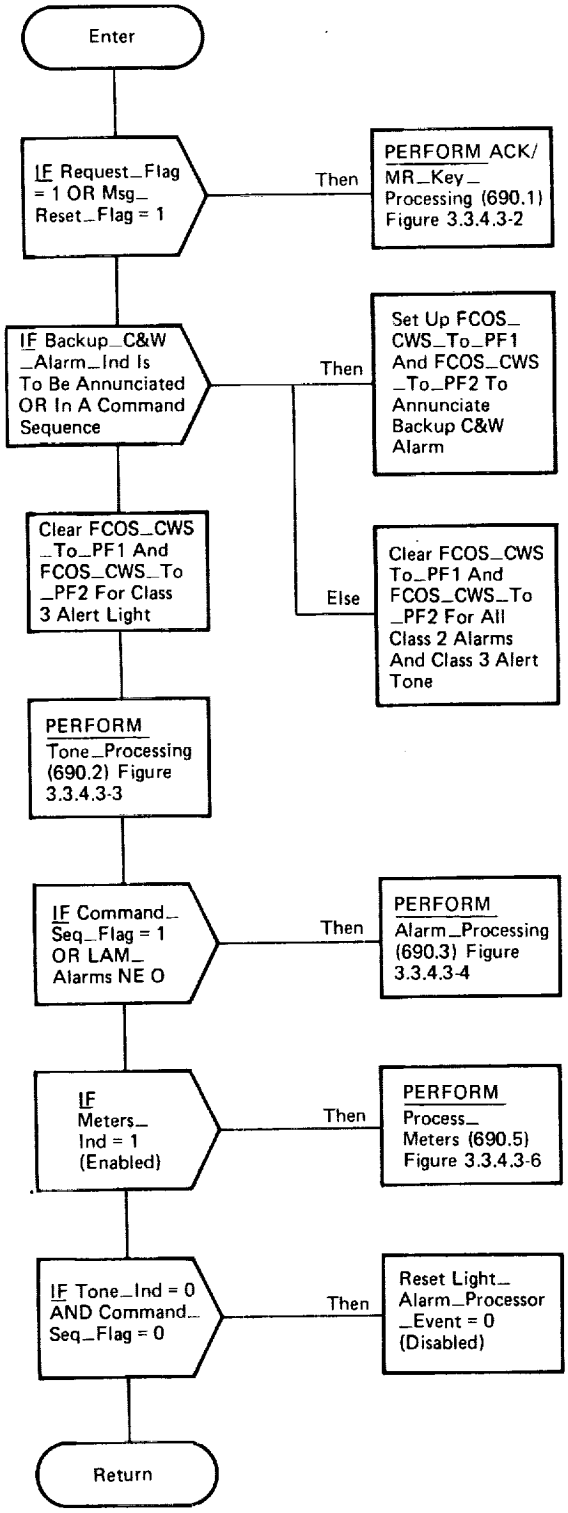


Figure 3.3.4.3-1. Lights_And_Alarm_Processor (DLA_Light_Alarm_PROC)

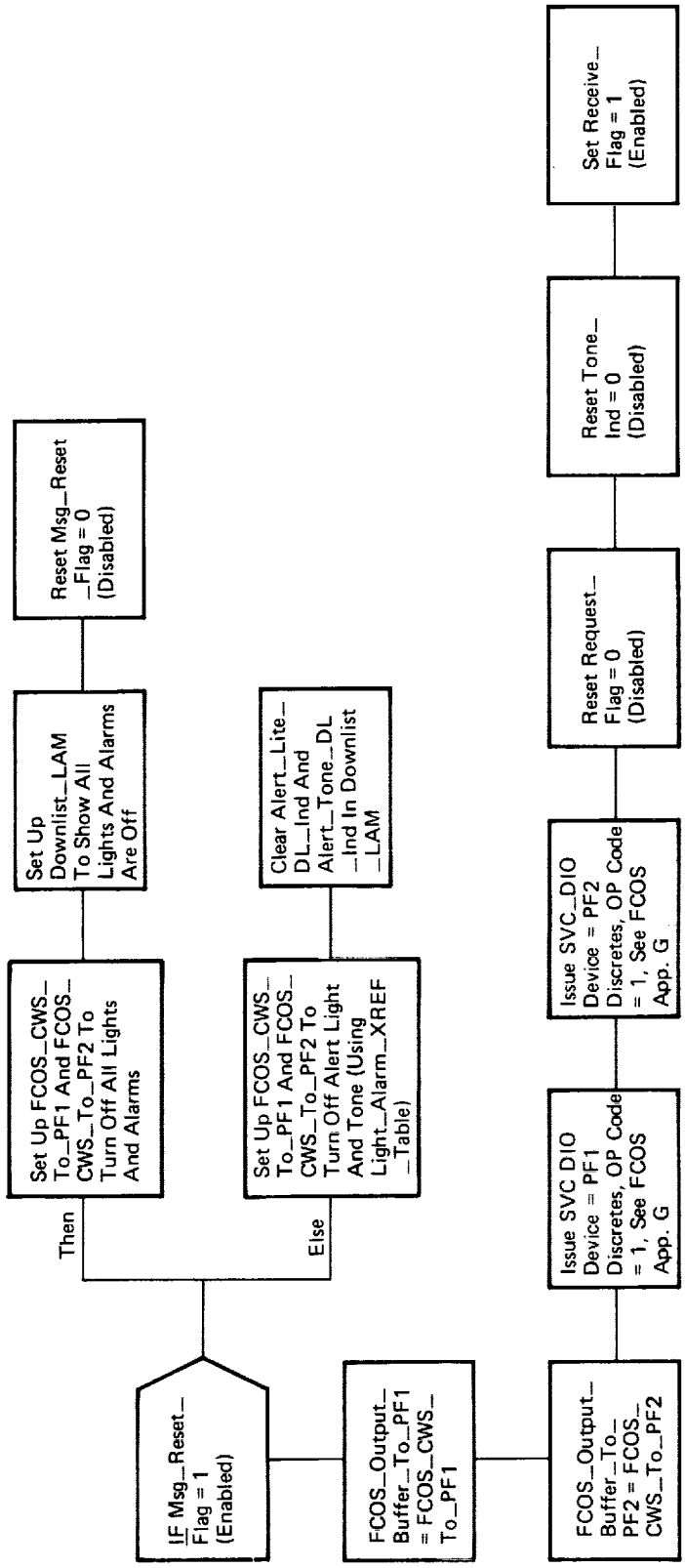


Figure 3.3.4.3-2. Lights_And_Alarm_Processor ACK/MR_Key_Processing(690.1)

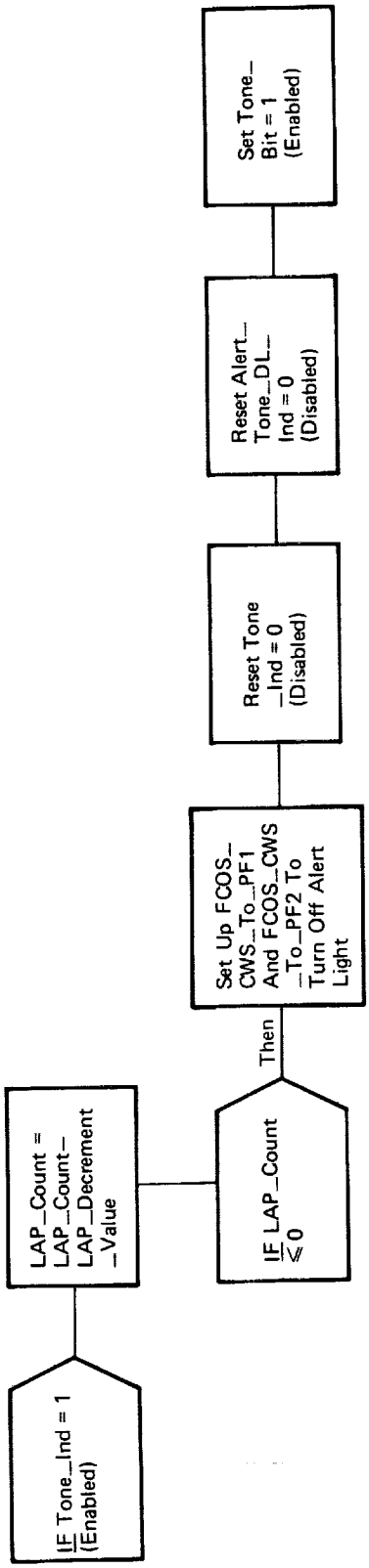


Figure 3.3.4.3-3. Lights_And_Alarm_Processor Tone_Processing (690.2)

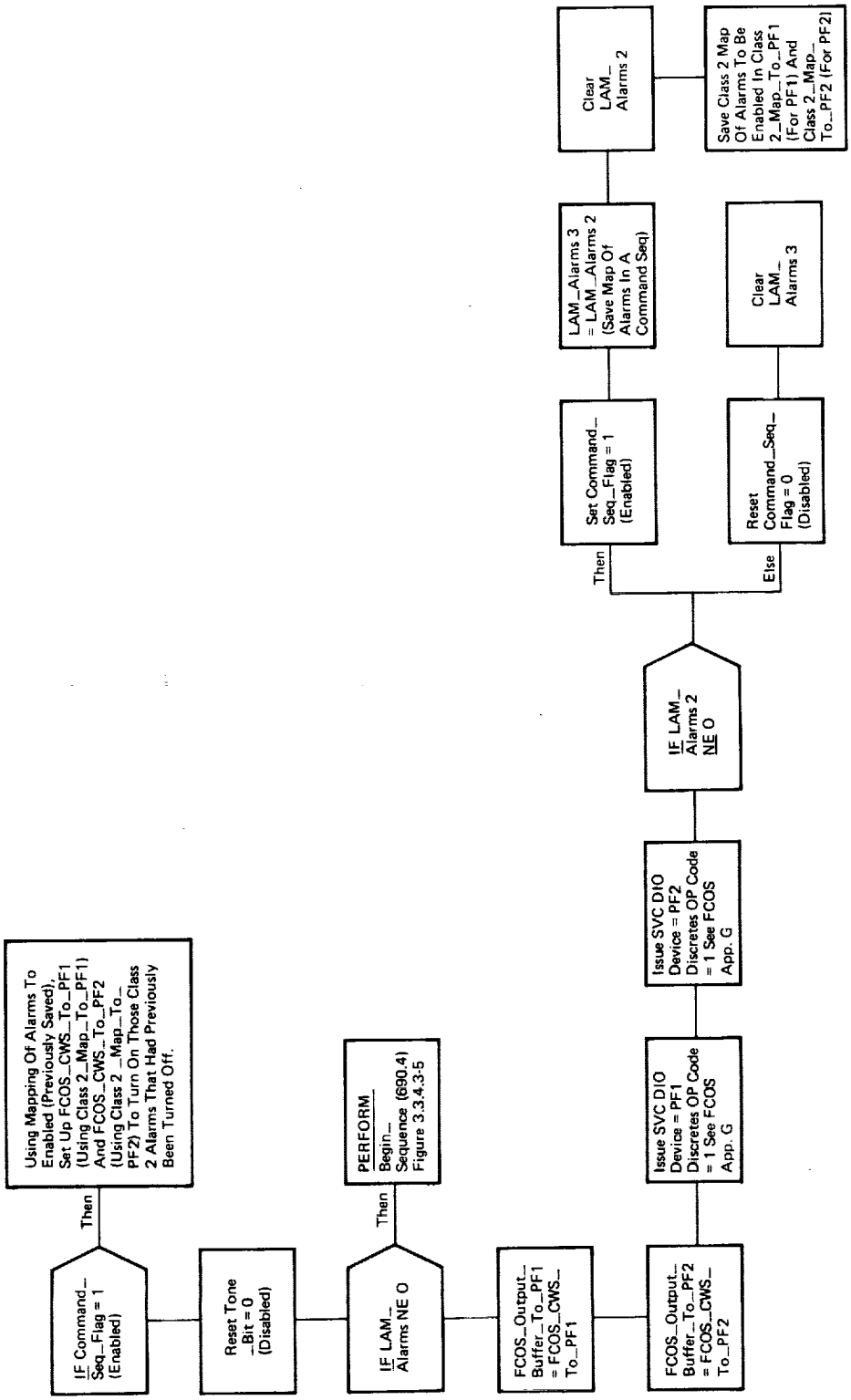


Figure 3.3.4.3-4. Lights_And_Alarm_Processor Alarm_Processing (690.3)

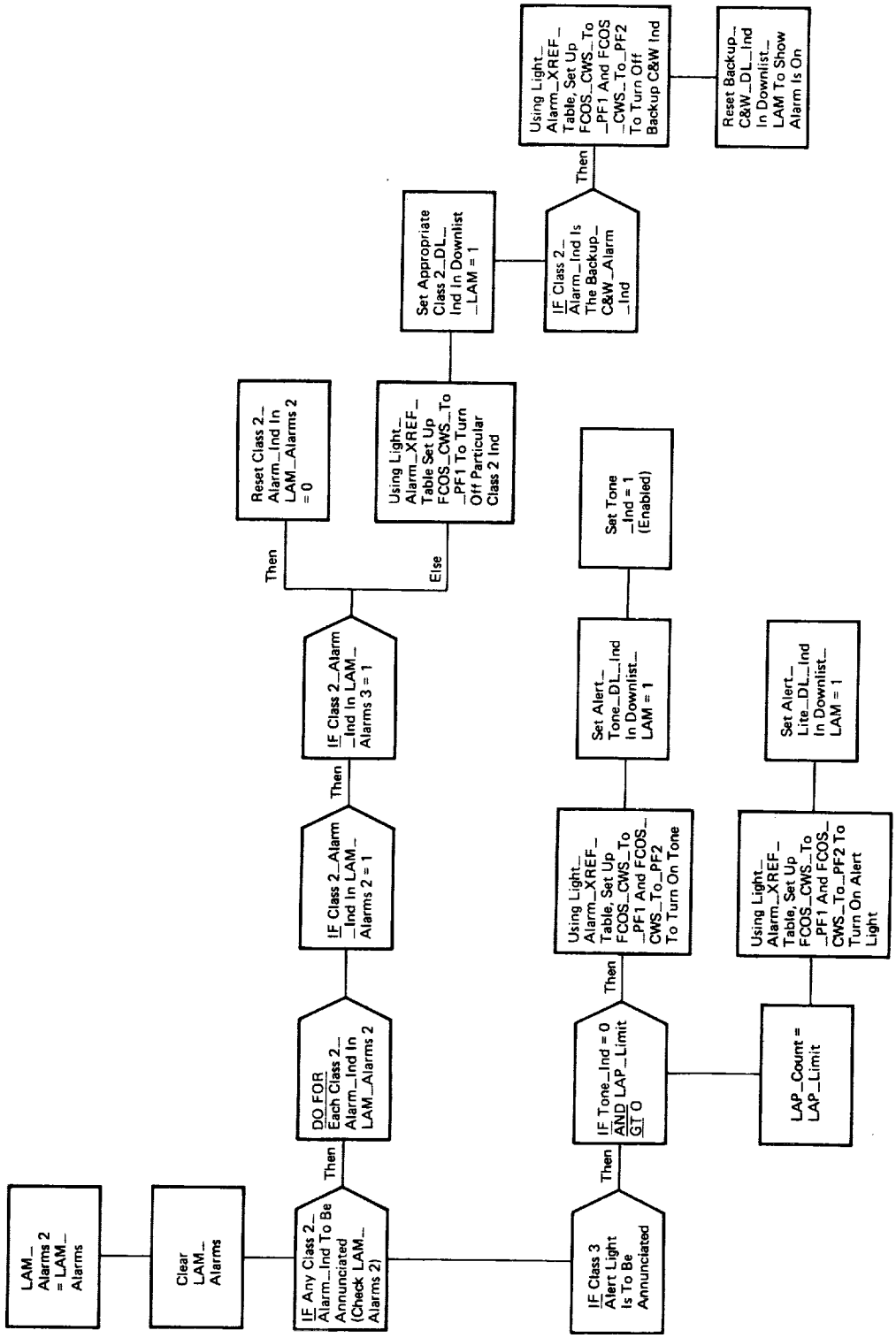


Figure 3.3.4.3-5. Lights_And_Alarm_Processor Begin_Sequence (690.4)

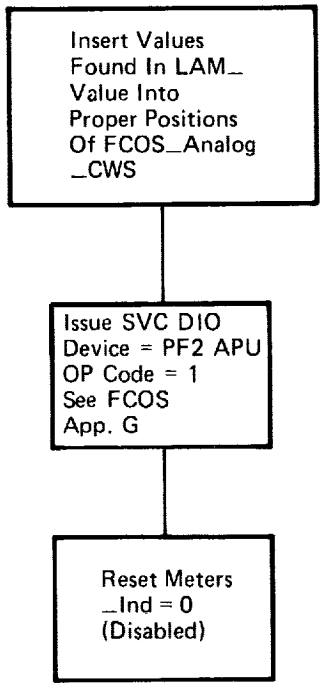


Figure 3.3.4.3-6. Lights_And_Alarm_Processor Process_Meters (690.5) (N/A For Alt.)

3.4 CRT DISPLAYS

3.4.1 Fault Summary

This display provides a historical summation of the most current fault messages. The display functions as a push-down list. The most recent fault message appears on the top; the next most recent, one row below, etc. The display is limited, at a maximum, to the last 20 messages. After 20 fault messages have been generated, the oldest (bottom) message on the display will disappear when the next fault occurs. The display is independent of major-function selection and can be called by the FAULT key at anytime. The display can also be called by DISPLAY page number; however, this reinitializes the display by clearing all messages.

a. Inputs - Inputs are specified in TAble 3.4.1-1.

b. Process Description

1. Each line on the fault summary is composed of five fields: DIS, FAULT, C&W, GPC, and TIME. The DIS and FAULT fields comprise a total of 20 characters, which are referred to as the major field of a fault message. The right-most four characters of the fault field are consequently referred to as the minor field of the message. The DIS part of the major field when used is intended to aid the crewman in determining the display on which the fault is displayed and the major function in which the display resides. Major functions will be defined as (G) GNC, (P) payload, (S) system management and will be identified by the first character of the major field when used. A (C) will denote system level or common fault. The major field identifies the problem (caution and warning). The minor field identifies the particular subsystem or string where the problem occured. The following examples illustrates the DIS and FAULT fields:

EXAMPLE 1.

```
DIS  FAULT

S701 MN BUS      A
Major Field      Minor Field
```

EXAMPLE 2.

```
DIS  FAULT
I/O ERROR        DEU1
Major Field      Minor Field
```

The GPC ID field is used to identify the source GPC or GPC's that generated the fault message as an aid in trouble shooting internal GPC or I/O type errors. The C&W field is a single character status indicator used with caution and warning (class 2) fault messages only. This status indicator will indicate to the crew that the fault is C&W-related and out of limits high (↑) or low (↓). The TIME field specifies the time the fault message was detected by annucation support and like the DIS, FAULT and GPC field is included as part of every fault message. The following example illustrates a fault message:

DIS	FAULT	C&W	GPC	TIME
		↓		
I/O ERROR	FF1		12	14/17:42:22
Major Field	Minor Field			

All messages on this display have been or will be enabled for output to the CRT fault message line (line 25) of all CRT's independently of major function. The messages on the fault message lines are identical to the corresponding message on the Fault Summary Display except for the truncation of days in the TIME field on the CRT fault message line.

2. The capability exists to prevents repetition of identical fault messages. When a fault message is generated, its major and minor fields are compared to the same field of the last fault message detected by this GPC. IF the fields agree and the new message has occurred within X.XX minutes of the top displayed message, the output of the new message is inhibited. The X.XX quanitity is called the FSP interlock time and may be changed as a constant by the table maintenance display or the Read/write spec.

3. ALL fault messages can be removed from the display. This capability is initiated by calling the fault summary page by entering DISP 051 PRO. This display is shown in figure 3.4.1-1.

DATA TABLE 3.4.1-1

NAME Fault Summary Display

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	Fault_Error_Message_Maj	L033	I	685	670,680 685	CDLB_FMPT_MAJOR		X	X
2	Fault_Error_Message_Minor	L034	I	685	670,680 685	CDLB_FMPT_MINOR		X	X
3	Caution_And_Warning_Parameter_Status	L035	I	685	670,680 685	CDLB_FMPT_IND			
4	GPC_Source	L036	I	685	485,680 685	DMS_GPC_CODE		X	
5	Time_Of_Message_Detection	L037	I	144 670	620,680 670,685	CDL_FAULT_MSG_TIME		X	

```
XXXX/XXXX/0051               FAULT SUMMARY              X  DDD/HH:MM:SS
                                                           DDD/HH:MM:SS

DIS        FAULT                C&W        GPC        TIME

XXXX   XXXXXXXXXXXXXXXXXX                XXXXX    XXX/XX:XX:XX
XXXX   XXXXXXXXXXXXXXXXXX                XXXXX    XXX/XX:XX:XX
XXXX   XXXXXXXXXXXXXXXXXX                XXXXX    XXX/XX:XX:XX
XXXX   XXXXXXXXXXXXXXXXXX                XXXXX    XXX/XX:XX:XX
XXXX   XXXXXXXXXXXXXXXXXX                XXXXX    XXX/XX:XX:XX
XXXX   XXXXXXXXXXXXXXXXXX                XXXXX    XXX/XX:XX:XX
XXXX   XXXXXXXXXXXXXXXXXX                XXXXX    XXX/XX:XX:XX
XXXX   XXXXXXXXXXXXXXXXXX                XXXXX    XXX/XX:XX:XX
XXXX   XXXXXXXXXXXXXXXXXX                XXXXX    XXX/XX:XX:XX
XXXX   XXXXXXXXXXXXXXXXXX                XXXXX    XXX/XX:XX:XX
XXXX   XXXXXXXXXXXXXXXXXX                XXXXX    XXX/XX:XX:XX
XXXX   XXXXXXXXXXXXXXXXXX                XXXXX    XXX/XX:XX:XX
XXXX   XXXXXXXXXXXXXXXXXX                XXXXX    XXX/XX:XX:XX
XXXX   XXXXXXXXXXXXXXXXXX                XXXXX    XXX/XX:XX:XX
XXXX   XXXXXXXXXXXXXXXXXX                XXXXX    XXX/XX:XX:XX
XXXX   XXXXXXXXXXXXXXXXXX                XXXXX    XXX/XX:XX:XX
XXXX   XXXXXXXXXXXXXXXXXX                XXXXX    XXX/XX:XX:XX
XXXX   XXXXXXXXXXXXXXXXXX                XXXXX    XXX/XX:XX:XX
XXXX   XXXXXXXXXXXXXXXXXX                XXXXX    XXX/XX:XX:XX
XXXX   XXXXXXXXXXXXXXXXXX                XXXXX    XXX/XX:XX:XX

                                                        (XX)
```

Figure 3.4.1-1

CPDS - Detailed Design Cross Reference

APPENDIX A

SSDS	CPDS VOL 1 BOOK 2
3.1.1	5.3.3.1
3.1.1.1	4.6.4.3.1 , 5.3.5.2, 5.3.5.2.1, 5.3.5.2.4
3.1.1.2	4.6.2.7, 4.6.3.1.10, 4.6.4.3.1, 5.2, 5.3.3.1.1, 5.3.5.1.1, 5.3.5.1.2, 5.3.5.1.3, 5.3.5.2, 5.3.5.2.2, 5.3.5.2.4, 5.3.5.4, 5.3.5.8, 5.7.3.7, 5.8.1.3.3
3.1.2	N/A
3.1.2.1	4.6.4.3.5 , 5.3.5.4
3.1.2.2	4.6.4.3.5 , 5.3.5.4
3.1.2.3	4.6.4.3.5, 5.3.5.4
3.1.3	4.6.3.3.5
3.1.3.1	4.6.3.3.5
3.1.3.2	4.6.3.3.5
3.1.3.3	4.6.3.3.5
3.1.3.3.3	4.6.3.3.5
3.2	5.2
3.2.1	5.3.1.1, 5.3.1.2, 5.3.1.3, 5.3.1.4, 5.3.3.1
3.2.1.1	5.2, 5.3.2.3.1, 5.3.3.1.1, 5.3.4.1, 5.3.5.5, 5.3.5.9, 5.8.1.1
3.2.1.1.1	5.3.2.3.1, 5.3.3.1.1 , 5.3.3.1.2 , 5.3.3.2.1, 5.3.3.2.7, 5.3.3.2.8, 5.3.3.2.9, 5.3.3.3 , 5.3.3.4.1 , 5.3.3.4.2, 5.3.3.4.3, 5.3.3.5, 5.3.3.7, 5.3.3.7.1, 5.3.3.8, 5.3.3.9, 5.3.3.9.4, 5.3.4.1, 5.3.5.3, 5.3.5.6, 5.3.5.6.4, 5.3.5.7.4, 5.9.1
3.2.1.1.2	5.3.3.1.1, 5.3.3.1.2, 5.3.3.2.1, 5.3.3.2.8, 5.3.3.2.9, 5.3.3.5, 5.3.3.5.1 , 5.3.3.5.2, 5.3.5.6.4

SSDS	CPDS VOL. 1 BOOK 2
3.2.1.1.3	5.3.2.3.1, 5.3.2.3.2, 5.3.2.3.3 , 5.3.3.1.2 , 5.3.5.3, 5.3.5.6, 5.3.5.6.1
3.2.2	5.3.1.1 , 5.3.1.2, 5.3.1.3, 5.3.1.4
3.2.2.1	5.3.2.3.1, 5.3.4.1, 5.3.4.2, 5.3.4.2.3, 5.3.5.6
3.2.2.2	5.3.4.1, 5.3.4.2, 5.3.4.2.1 , 5.3.4.2.2 , 5.3.5.5 , 5.3.5.9, 5.3.5.9.4
3.2.3	4.6.2.2.2, 4.6.2.2.3, 5.3.1.4
3.2.3.1	4.6.2.2.5, 5.3.1.1, 5.3.1.2, 5.3.1.4, 5.3.3.4.1, 5.3.5.3 , 5.3.5.9.1 , 5.3.5.9.2, 5.3.5.9.4
3.3	5.2
3.3.1	5.3.2.4
3.3.1.1	5.3.2.1, 5.3.2.3.2, 5.3.2.3.3, 5.3.3.2.3, 5.3.5.6, 5.3.5.6.1, 5.3.5.8
3.3.1.2	4.6.2.4.5, 4.6.2.4.6, 4.6.2.4.8, 5.3.2.1, 5.3.2.2.3, 5.3.2.6.2, 5.3.3.1.1, 5.3.5.7.4
3.3.1.2.1	5.3.2.1, 5.3.2.2.1, 5.3.2.2.4
3.3.1.2.2	5.3.2.2.1.
3.3.1.3	5.3.2.1, 5.3.2.2.1, 5.3.2.2.3, 5.3.2.2.4, 5.3.2.4.1, 5.3.2.4.2, 5.3.2.4.3, 5.3.2.4.4
3.3.1.4	5.3.5.6.2
3.3.1.5	5.3.2.2.1, 5.3.2.2.3, 5.3.5.7.3, 5.3.5.8
3.3.1.5.1	5.3.2.2.3, 5.3.5.7.3
3.3.2	5.7.2
3.3.2.1	5.7.2
3.3.2.2	5.7.2

SSDS	CPDS VOL 1 BOOK 2
3.3.3	N/A
3.3.4	5.3.5.7
3.3.4.1	5.3.5.7.1, 5.3.5.7.2, 5.3.5.7.4, 5.9.1
3.3.4.1.1	5.3.5.7.1, 5.3.5.7.2, 5.3.5.7.4, 5.9.1
3.3.4.2	5.3.2.2.1, 5.3.2.2.3, 5.3.3.1.2, 5.3.5.7, 5.3.5.7.1, 5.3.5.7.4, 5.7.3.7, 5.9.1
3.3.4.3	5.3.5.7, 5.3.5.7.1, 5.3.5.7.5

Resource Allocation Summary

Appendix B

VI Resource Allocation Summary

Item	Main Memory Words	
Command Input Processing		2059
Keyboard Interface	769	
MCDS Input Processor (DMI_MCDS_IN)	271	
MCDS Message Processor (DMM_MCDS_PROCESSOR)	498	
LDB Interface	674	
LDB I/O Processor (DGI_LDB_IO)	569	
LDB Message Router (DGI_LDB_ROUT)	(***)	
Mass Memory Message Processor (DMP_MM_MSG_PROC)	(***)	
LDB Output Message Coordinator (DGO_LDB_COORD)	105	
ICC Interface	616	
ICC Collector for SIP (DIN_ICC_COLLECTOR)	141	
ICC Message Collector (DIM_ICC_COLLECTOR)	72	
ICC Message Router (DME_ICC_ROUT)	336	
ICC/Idle OPS Processor (DID_IDLE_OPS_ICC)	67	
Operations Control		3531
User Interface Control	2307	
User Interface Control Supervisor (DMC_SUPER)	850	
MCDS Functions Processing (DMC_FUNCTIONS) *	334	
MCDS ITEM Processor (DMC_ITEM_PROC) *	667	
MCDS Display Coordination (DMC_DISPLAY) *	456	
Application Control	207	
Display Presentation and Control (DIS_PLAY)	93	
Application Moding and Sequencing (DNX_BMS)	114	
Memory Overlay Coordination	1017	
Sequence Request Processing (DMC_SEQ_REQ_PROC) *	1017	
Output Message Processing and Coordination		5463
CRT Interface	1862	
New Display Processing (DMC_NEW_DISPLAY) *	115	
Cyclic Display Processing (DMI#CYC)	1604	
DEU I/O Management (DCIBIO) **		
Header Build (DCIBHDR) **		
Data Formatting (DCI#FMT) **		
Data Conversion (DCI#CON) **		
Message Line Support Function (DMS_MSG_LSF)	85	
FCW Builder (DMS_FCW_BUILDER)	58	
Downlist Interface	1534	
Downlist Formatter (DCD_DOWNLIST)	1534	
LDB Interface	2067	
Annunciation Support		
Fault Message Scan (DMR_FMS)	228	
Annunciation Macro Interface (DMA_MAC)	69	
Error Message Line Support (DMT_ERR_MSG)	538	

* Data included with DMC_SUPER
** included with DCI#CYC
*** Size not included because item is not permanently resident.

VI Resource Allocation Summary

Item	Main Memory Words
FSP Processor (DMT_FSP_PROCESSOR)	426
Lights and Alarm Processing (DLA_LIGHT_ALARM_PROC)	806

Compool Data	Main-Memory Words	
TOTAL	1775	1775
Downlist (CDW_DOWNLIST)	76	
Annunciation (CDL_ANN)	1101	
User-Interface (CDM_UI_COMPOOL)	166	
Common (CZ1_COMMON)	219	
UI_SUPERVISOR_TABLE (SUPTAB)	66	
LDB (CDVCOM)	147	

APPENDIX C
USER INTERFACE MODULE LIST

Module ID	Module Name	HAL/s Name	Section
400	MCDS_Input_Processor	DMI_MCDS_IN	3.1.1.1
405	MCDS_Message_Processor	DMM_MCDS_Process	3.1.1.2
440	LDB_I/O_Processor	DGI_LDB_IO	3.1.2.1
460	LDB_Output_Message_Coordinator	DGO_LDB_COORD	3.1.2.2
480	ICC_Message_Collector	DIM_ICC_Collector	3.1.3.2
485	ICC_Message_Collector_For_SIP	DIN_ICC_SIPCOLL	3.1.3.1
490	ICC_Message_Router	DME_ICC_ROUT	3.1.3.3
495	ICC_Idle_OPS_Processor	DID_IDLE_OPS_ICC	3.1.3.3.1
500	User_Interface_Control_Supervisor	DMC_SUPER	3.2.1.1
505	MCDS_Functions_Processor	DMC_FUNCTIONS	3.2.1.1.1
510	MCDS_ITEM_Processor	DMC_ITEM_PROC	3.2.1.1.2
520	MCDS_Display_Coordinator	DMC_DISPLAY	3.2.1.1.3
540	Display_Presentation_and_Control	DIS_PLAY	3.2.2.1
550	Application_Moding_and_Sequencor	DNX_BMS	3.2.2.2
560	Sequence_Request_Processor	DMC_SEQ_REQ_PROC	3.2.3.1
600	New_Display_Processor	DMC_NEW_DISPLAY	3.3.1.1
620	Cyclic_Display_Processor	DCI#CYC	3.3.1.2
625	DEU_I/O_Management	DCIBIO	3.3.1.2.1
630	Header_Builder	DCIBHDR	3.3.1.2.2
635	Data_Formatter	DCI#FMT	3.3.1.3
640	Data_Conversion	DCI#CON	3.3.1.4
650	Message_Line_Support_Function	DMS_MSG_LSF	3.3.1.5
655	FCW_Builder	DMS_FCW_BUILDER	3.3.1.5.1
660	Downlist_Formatter_1	DCDDWL1	3.3.2.1
665	Downlist_Formatter_2	DCDDWL2	3.3.2.2
670	Fault_Message_Scan	DMR_FMS	3.3.4.1
675	Annunciation_Macro_Interface	DMA_MAC	3.3.4.1.1
680	Error_Message_Line_Support	DMT_ERR_MSG	3.3.4.2
685	FSP_Processor	DMT_FSP_Processor	3.3.4.2.1
690	Light_and_Alarm_Processor	DLA_LIGHT_ALARM_PROC	3.3.4.3

APPENDIX D
ERROR CONDITIONS

ALT SYSTEM SOFTWARE ERRORS

NO. BY TYPE	ERROR CONDITION	SOFTWARE RESPONSE	FMPT ID	TEXT		CLASS
				MAJOR	MIN	
OT001	OPS UNDEFINED OR OPS NOT RESIDENT AND NOT OVERLAY INITIATOR	REQ REJ	K1	ILLEGAL ENTRY	MC	5
OT002	OPS NOT IN ACTIVE GPC AND MMU OFF/BUSY	REQ REJ	M1	MMU OFF/BUSY		3
OT003	NO TARGET GPC IN CONFIGURATION TABLE WHEN OPS SELECTED	REQ REJ	K1	ILLEGAL ENTRY	MC	5
OT010	OPS REQUEST CONTAINING INVALID MODE NUMBER	REQ REJ	K2	ILLEGAL ENTRY	OPS	5
OS018	ATTEMPT TO ASSIGN LDB TO NON-GPCp IN RS	REQ REJ	K5	ILLEGAL ENTRY	CONF	5
DU001	DEU 1 BITE INDICATION	NONE	B1	BITE	DEU1	3
DU002	DEU 2 BITE INDICATION	NONE	B2	BITE	DEU2	3
DU003	DEU 3 BITE INDICATION	NONE	B3	BITE	DEU3	3
DU004	ERROR IN DEU 1 MSG OR BITE REQ	REQ REJ	R1	I/O TRANSIENT	DEU1	3
DU005	ERROR IN DEU 2 MSG OR BITE REQ	REQ REJ	R2	I/O TRANSIENT	DEU2	3
DU006	ERROR IN DEU 3 MSG OR BITE REQ	REQ REJ	R3	I/O TRANSIENT	DUE3	3

ALT SYSTEM SOFTWARE ERRORS (CONT'D)

NO. BY TYPE	ERROR CONDITION	SOFTWARE RESPONSE	FMPT	ID	TEXT		CLASS
					MAJOR	MIN	
DU006	ERROR IN DEU 3 MSG OR BITE REQ	REQ REJ	R3		I/O TRANSIENT	DEU3	3
DU007	SELECTION OF SPEC ALREADY ACTIVE	FIRST REQ REJ	K7		ILLEGAL ENTRY	ACT	5
DU008	EXTERNAL DATA OUT LIMITS (PER DFG)	FRIST INP REJ	K3		ILLEGAL ENTRY	LIM	5
DU009	GPC DETECTED INPUT MSG SYNTAX ERROR	INP REJ	K4		ILLEGAL ENTRY	SYN	5
DU010	SELECTION OF DISPLAY NOT ASSOCIATED WITH CURRENT ØPS	REQ REJ	K2		ILLEGAL ENTRY	OPS	5
DU011	RESUME WHEN SPEC OR DISPLAY NOT ACTIVE	REQ REJ	K7		ILLEGAL ENTRY	ACT	5
DU012	SPEC NOT ALLOWED IN CURRENT OPS	REQ REJ	K2		ILLEGAL ENTRY	OPS	5
DU013	INVALID ITEM NO. FOR DATA LOAD	REQ REJ	K8		ILLEGAL ENTRY	ITEM	5
DU014	EXEC NOT COMPATIBLE WITH CURRENT FORMAT OR STEP IN CONTROL SEGMENT	REQ REJ	K4		ILLEGAL ENTRY	SYN	5
DU015	PRO NOT COMPATIBLE WITH CURRENT FORMAT OR STEP IN CONTROL SEGMENT	REQ REJ	K4		ILLEGAL ENTRY	SYN	5
DU016	TWO ACTIVE NONRESIDENT FORMATS WHEN ANOTHER REQUESTED	REQ REJ	K7		ILLEGAL ENTRY	ACT	5

ALT SYSTEM SOFTWARE ERRORS (CONT'D)

NO. BY TYPE	ERROR CONDITION	SOFTWARE RESPONSE	FMPT ID	TEXT		CLASS
				MAJOR	MIN	
DU017	ATTEMPT TO LOAD DATA VIA DISPLAY	REQ REJ	K4	ILLEGAL ENTRY	SYN	5
DU026	REQUEST SPEC WHEN TWO ARE ACTIVE (PER MF)	REQ REJ	K7	ILLEGAL ENTRY	ACT	5
DU027	PRO, EXEC OR ITEM WHEN NEW FORMAT IS BEING RETRIEVED	REQ REJ	K4	ILLEGAL ENTRY	SYN	5
DU028	INPUTS OTHER THAN MSG RESET, CLEAR, OR ACK IN MIDST OF OPS TRANSITION	REQ REJ	K4	ILLEGAL ENTRY	SYN	5
DU029	FORMAT NOT FOUND IN BUFFER FOLLOWING RETRIEVAL FROM MMU	REQ REJ	D1	DISPLAY SYSTEM	ERR	4
SC008	REQUEST FOR DEU INIT WHEN GPC NOT IN OPS 00 OR SM 8	REQ REJ	K5	ILLEGAL ENTRY	CONF	5

APPENDIX E

DATA DESCRIPTION TABLE

The Data Descriptors Appendix provides detailed information about each parameter. Parameters in the Data Descriptors can be cross referenced to parameters in the modular Data Tables via a parameter identification tag (ID). The definition of each field in the table follows.

1. ID - A parameter ID has been assigned to each data item in the format

   ZXXX [. XX]

   where:

   Z - Single character denoting the function of the parameter, chosen from one of the following:

   - A - Moding of Control Segments (See Vol II, P2)
   - D - Display Formats (See Vol II, P3)
   - B - Mass Memory (See Vol II, P2)
   - T - Annunciation (See Vol II, P2)
   - V - LDB (See Vol II, P2)
   - F - Downlist (See Vol II, P2)
   - W - Read/Write of Main Memory (See Vol II, P3)
   - H - Time Management (See Vol II, P3)
   - I - FCOS/CI Common Compool (See Vol II, P3)
   - J - Common (Miscellaneous) (See Vol II, P2)
   - L - User Interface Local Variable (See Vol II, P2)
   - X - System Control Local Variable (See Vol II, P3)
   - Y - FCOS I/O Management (See Vol II, P1)
   - @ - FCOS Process Management (See Vol II, P1)
   - O - FCOS Configuration Management (See Vol II, P1)
   - # - FCOS COMPOOL (See Vol II, P1)
   - Q - FCOS Control Blocks (See Vol II, P1)
   - S - FCOS SVC Parameter List (See Vol II, P1)
   - & - Hardware Parameters (See Vol II, P1)

   XXX - A unique three digit number assigned to each major division. (A major division is a structure, table, array, or a single parameter not contained in one of the previously mentioned items.

   [.XX]- Second level qualifier for elements of arrays, parts of structures or tables or the bits in a flag or discrete word.

2. ITEM - A unique meaningful English_Equivalent_Name for the data item.

3. HAL/Assembler Name - The HAL/S or Assembler Name used in coding. The column is blank if no name is assigned as is the case with bits in a flag word. (User Interface and System Control column name is "HAL NAME" FCOS column is "ASSEMBLER NAME").

4. Description - A concise statement of the nature of the parameter and its purpose.

5. Source - A listing of the three digit ID's of all modules updating the parameter and/or one of the following special codes:

   ILD - The parameter is of the Initial Load (I-LOAD) type (see Level A CPDS, Table 6-4).

   MRW - The parameter is available to memory read write (see Level A CPDS, Table 6-4).

6. Destination - A list of the three digit ID's of all modules referencing the parameter and/or one of the following special codes:

   DL - The parameter is available for downlink.
   CRT - The parameter is available for display.

7. Attributes - The attributes of the parameter are as follows:

   a. Data Type - The HAL/S declaration type or assembler language type

ST(n)	-	Structure with n copies
A(m,n),k	-	Array of m,n dimensions of HAL parameter type k
BS(n)	-	Bit string of length n
BT	-	Bit
C(n)	-	Character string of length n
SC	-	Scalar/Assembler Language Floating Point
I	-	Integer/Assembler Language Fixed Point Halfword
BO	-	Boolean
M(n,m)	-	Matrix of n,m dimensions
V(n)	-	Vector of n elements
E	-	Event
Y	-	Halfword Address Constant
AC	-	Fullword Address Constant
Z	-	Fullword Indirect Address Constant
X	-	Hexadecimal
F(n)	-	Fullword Fixed Point n Copies

NOTE: Double length of a parameter type is denoted by prefixing a 'D' to the above characters.

b. Initial Value - The initialization value if applicable;

INIT(value)

c. Units of Measure - The units of measure of the parameter if applicable:

UM(unit)

d. Other - Any of the attributes such as REPLACE, LOCU(h), etc.

8. MML - The MML number associated with the parameter. The column is blank if none is defined.

In part 1 (FCOS) bits in a word will be numbered starting with 0. In parts 2 and 3 (UI and SC) bits in a word will be numbered starting with 1.

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
A010	AMT_BUFFER_NAME_TABLE	CDAV_BNAME	STRUCTURE FOR DFB ADDRESSES ARE MM DIRECTING ADDRESS	DFG	560	ST(1)	
A010.10	DFB1_ADDRESS	CDAV_NBUF1	CONTAINS THE ADDRESS OF DISPLAY FORMAT BUFFER 1	DFG	560	Y	
A010.20	DFB2_ADDRESS	CDAV_NBUF2	CONTAINS THE ADDRESS OF DISPLAY FORMAT BUFFER 2	DFG	560	Y	
A010.30	DFB3_ADDRESS	CDAV_NBUF3	CONTAINS THE ADDRESS OF DISPLAY FORMAT BUFFER 3	DFG	560	Y	
A010.40	DFB4_ADDRESS	CDAV_NBUF4	CONTAINS THE ADDRESS OF DISPLAY FORMAT BUFFER 4 (DISPLAY MASS MEMORY BUFFER)	DFG	560	Y	
A010.50	DFB5_ADDRESS	CDAV_NBUF5	CONTAINS THE ADDRESS OF DISPLAY FORMAT BUFFER5 (DISPLAY MASS MEMORY BUFFER)	DFG	560	Y	
A010.60	DISPLAY_MASS_MEMORY_DIRECTORY_ADDRESS	CDAV_NMMDX	CONTAINS THE ADDRESS OF THE DISPLAY MASS MEMORY DIRECTORY	DFG	560	Y	
A070	MAJOR_FUNCTION_LOCATING_INFORMATION	CDAV_MF	CONTAINS OPS INFORMATION PER MAJOR FUNCTION	DFG	560	ST(3)	
A070.10	NUMBER_LEGAL_OPS	CDAV_MF_NOPS	THE NUMBER OF LEGAL OPS PER MAJOR FUNCTION	DFG	560	I	
A070.20	FIRST_OPS_COPY_NUMBER	CDAV_MF_FOPS	THE FIRST CDAV_OPS STRUCTURE COPY CONTAINING OPS DATA FOR THIS MAJOR FUNCTION	DFG	560	I	
A080	OPS_DATA	CDAV_OPS	CONTAINS OPS RELATED INFORMATION	DFG	550 560	ST(6)	
A080.10	OPS_NUMBER	CDAV_OP_OPSID	OPS ID/NUMBER	DFG	560	I	
A080.20	NUMBER_MODES	CDAV_OP_NMODES	NUMBER OF MODES IN THIS OPS	DFG	550 560	I	
A080.30	MODE_DATA_ARRAY_POINTER	CDAV_OP_MODXXFR_ST	THE CDAV_MOD_VAL_ARR ARRAY NUMBER POINTING TO MODE ONE (1) OF THIS OPS	DFG	550	I	
A080.40	NUMBER_SPECS	CDAV_OP_NSPEC	THE NUMBER OF SPEC LEGAL FOR THIS OPS	DFG	560	I	
A080.50	FIRST_SPEC_COPY	CDAV_OP_SPEC_ST	THE CDAV_SPEC STRUCTURE COPY FOR THE FIRST SPEC LEGAL FOR THIS OPS	DFG	560	I	

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
A080.60	OPS_TO_OPS_TRANSFER_DATA	CDAV_OP_XFR	THE LEGAL OPS TO OPS TRANSFER DATA	DEG		BS(16)	
A080.70	OPS_PRIORITY	CDAV_OP_PRIOR	THE PRIORITY OF THIS OPS	DEG	560	I	
A090	SPEC_DATA	CDAV_SPEC	CONTAINS SPEC RELATED DATA	DEG	560	ST(12)	
A090.10	SPEC_NUMBER	CDAV_SP_ID	SPEC ID/NUMBER	DEG	560	I	
A090.20	SPEC_BLOCKS	CDAV_SP_BLOCKS	NUMBER OF BLOCKS ASSOCIATED WITH THIS SPEC	DEG	560	I	
A090.30	SPEC_PRIORITY	CDAV_SP_PRIOR	PRIORITY OF THIS SPEC	DEG	560	I	
A100	OPS_NAMES_TABLE	CDAV_OPNAME	CONTAINS NAMES OF ALL OPS'S	DEG	560	ST(6)	
A100.10	OPS_PROGRAM_NAME	CDAV_OP_PROGNM	NAMES OF ALL OPS PROGRAMS USED TO SCHEDULE AN OPS	DEG	560	Y	
A110	SPEC_NAME_TABLE	CDAV_SPNAME	CONTAINS NAMES OF ALL SPECS	DEG	560	ST(12)	
A110.10	SPEC_PROGRAM_NAME	CDAV_SP_SPECNM	NAMES OF ALL SPEC PROGRAMS USED TO SCHEDULE A SPEC	DEG	560	Y	
A120	OPS_MODING_DATA	CDAV_MOD_VAL_ARR	CONTAINS OPS MODING INFORMATION *THE SIZE OF THIS ARRAY IS A FUNCTION OF THE TOTAL NUMBER OF MODES FOR ALL OPS DEFINED IN THIS COMPOOL BITS 1-28 ARE FOR MODE TO MODE VALIDITY CHECKING (NOT USED) AND BITS 29-32 HAS THE INTEGER VALUE OF THE NUMBER OF BLOCKS FOR THIS MODE	DEG	550	A(*),BS(32)	

MODING OF CONTROL SEGMENTS

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
B010	MCDS_ALLOCATION_TABLE	CDMV_MAT_TABLE	TABLE USED TO CONTROL DISPLAY PRESENTATION TO THE CRT			ST(3)	
B010.05	MAT_FULLWORD_ALIGNMENT	CDMV_MAT_FILLER	ALIGNS MAT TABLE ON FULLWORD BOUNDARY	620	620	I	
B010.10	MAT_MAJOR_FUNCTION_SETTING	CDMV_MAT_MF	MAJOR FUNCTION SETTING USED FOR DISPLAY	505 520	505 560 600 620	I	
B010.15	DEU_DISPLAY_SUPPORT_LEVEL	CDMV_MAT_LEVEL	LEVEL OF THE DEU DISPLAY SUPPORT WHERE 0-BETWEEN LEVELS 1-OPS 2-SPEC 3-DISP	505 520 620	505 560 600	I	
B010.20	DEU_UPDATE_RATE	CDMV_MAT_DEU_RATE	DEU UPDATE RATE	500 620 820 880	620 625	BS(16) INIT 000F 000F 0000	
B010.25	PROCESSING_DISPLAY_DFB_INDEX	CDMV_MAT_DFB_INDEX	INDEX TO THE DISPLAY FORMOT BUFFER FOR THE SPECIFIED DISPLAY LEVEL	520 540 560	540 600 620	A(3)	
B010.30	DFB NUMBER FOR PROCESSING DISPLAY	CDMV_MAT_DFB_NUMBER	DFB NUMBER FOR DISPLAY LEVEL	520 540 560	540 600 620	A(3)	
B010.35	DISPLAY_LEVEL_PAGE_NUMBER	CDMV_MAT_DISP_NUMBER	PAGE NUMBER FOR EACH DISPLAY LEVEL	505 520 560	505 620	A(3)	
B010.40	CRT_STATUS_FLAGS	CDMB_MAT_CRT_STAT	FLAGS FOR CRT,STATUS WHERE:	400 405 505 560 600 620 650 790 820	400 505 560 600 620 625 680 790 880	BS(16)	
B010.41	CRT_STATUS_FLAG1		1-DISPLAY FROZEN	400 600 505	505 620 625		
B010.42	CRT_STATUS_FLAG2		2-I/O SUPPRESSED	400 505	505 620		
B010.43	CRT_STATUS_FLAG3		3-UPDATE ALL PARAMETERS ONE TIME	400	625		
B010.44	CRT_STATUS_FLAG4		4-FAST PAPMS PARMS PRESENT IN	400	600		

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
			CURRENT DISPLAY	600 650	620 680		
B010.45	CRT_STATUS_FLAG6		6-MESSAGE LINE READY FOR OUTPUT TO THE DEU	620	620		
B010.46	CRT_STATUS_FLAG7		7-TUTORIAL MESSAGE LINE UPDATE IN PROGRESS 8-SPARE	650	620		
B010.47	CRT_STATUS_FLAG9		DISPLAY PROCESSED	620	620		
B010.48	CRT_STATUS_FLAG10		10-FAULK MESSAGE LINE UPDATE IN PROGRESS	650	620		
B010.49	CRT_STATUS_FLAG11		11-PERFORM TIME CONVERSION FOR THE CORRESPONDING FAULT MESSAGE 12-16 NOT USED	650	620		
B010.50	I/O_ERROR_RETURN_BUFFER	COMV_MAT_ERROR_RETURN	AREA USED FOR I/O ERROR RETURN UPON SVC ISSUED IN CYCLIC DISPLAY PROCESSOR	244	620	A(2)	
B010.52	DEU_MESSAGE_HEADER_WORD1	COMV_MAT_HEADER1	DEU MESSAGE HEADER WORD FOR MESSAGE BUFFER	620	620	I	
B010.55	DEU_MESSAGE_HEADER_WORD1	COMV_MAT_HEADER2	DEU MESSAGE HEADER WORD FOR MESSAGE BUFFER	620	620	I	
B010.60	TUTORIAL_MESSAGE_LINE BUFFER	COMV_MAT_TUTORIAL_BUF	BUFFER FOR TUTORIAL MESSAGE LINE CONTENTS	650	620	A(29)	
B010.65	FAULT_MESSAGE_LINE_BUFFER	COMV_MAT_MSG_LINE_BUF	BUFFER FOR FAULT MESSAGE LINE CONTENTS	620 680	620	A(35)	
B010.70	MESSAGE_BUFFER_TRAILER_RECORD	COMV_MAT_TRAILER	TRAILER RECORD FOR MESSAGE	620	620	I,INIT X'FFFF0'	
D010.75	FAULT_MESSAGE_LINE_TIME	COMV_MAT_TIME	TIME FIELD FOR FAULT MESSAGE LINE TIME PARAMETER	680	620	DI	
B010.80		COMV_MAT_SPARE		620	620	DI	
B020	UI_DOWNLIST_PARAMETER BUFFER	COMV_UI_DOWNLIST	STORAGE AREA FOR UI'S DOWNLIST PARAMETERS	400 600 680	660	S(3)	
B020.05	TUTORIAL_MESSAGE_ID_CODE	COMV_UI_DOWNLIST_TUTORIAL	TUTORIAL MESSAGE ID CODE	650	660	DI	V92M4027P V92M4044P V92M4061P
B020.10	STACK_COUNTER_VALUE	DOWNLIST_STACK_CT	STACK COUNTER VALUE CURRENTLY BEING DISPLAYED ON THE FAULT MESSAGE LINE	680	660	DI	V92Q4008C

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
B020.15	DOWNLIST_ERROR_MESSAGE	CDMV_UI_DOWNLIST_ERROR_MSG	THE ID CODE ASSOCIATED WITH THE ERROR MESSAGE TO BE DOWNLISTED	680	660	DI	
B020.20	DISPLAY_NUMBER_WITH_MAJOR_FUNCTION_SETTING	CDMV_UI_DOWNLIST_FORMAT_ID	DISPLAY NUMBER, AND, MAJOR FUNCTION SETTING ASSOCIATED WITH THAT DISPLAY	520	660	BS(116)	V92X5065X V92X5066X V92X5067X
B030	CONTROL_SEGMENT_INDEX_TO_MACT	CDMV_MACT_INDEX	MACT INDEX FOR CONTROL SEGMENT	560	550 560	I	
B040	MACT_INDEX_EVENT	CDME_MACT_INDEX	MACT INDEX EVENT FOR CONTROL SEGMENT INPUT	550 560	560	E	
B050	APPLICATION_SERVICE_REQUEST_EVENT	CDME_APP_SERVICE	APPLICATION SERVICE REQUEST EVENT FOR DMC SUPER	540 820 870 880	660 810 820	E	
B060	MM_READ_COMPLETE_EVENT	CDME_ID_MM_EVENT	EVENT FOR MM READ COMPLETE FOR THE DFT	500	505	E	
B070	DOWNLIST_ANNUNCIATION_MESSAGE_STRUCTURE	CDMV_ANNUN_DOWNLIST	STRUCTURE FOR DOWNLISTING UP TO THE FIVE MOST CURRENT ANNUNCIATION MESSAGES	685	660	ST(5)	
B070.05	MAJOR_MINOR_FAULT_MESSAGE_ID	CDMV_WORD1	MINOR AND MAJOR FAULT MESSAGE ID	685	660	DI	V91M4077P V91M4083P V91M4092P V91M4098P V91M4104P
B070.10	MESSAGE_DOWNLIST_TIME	CDMV_TIME	DOWNLIST TIME ASSOCIATED WITH EACH MAJOR AND MINOR ID	685	660	DI	V91W4078C V91W4084C V91W4093C V91W4099C V91W4105C
B080	CYCLIC_PROCESS_STATUS	CDMV_POSTEVENT	A FLAG TO INDICATE THE CYCLIC DISPLAY PROCESSOR STATUS, WHERE 0=CYCLIC IDLE 1=CYCLIC RUNNING 2=CYCLIC FINISHED PROCESSING	620	500	I	
B090	CYCLIC_IN_PROCESS_EVENT	CDME_CYCLIC_EVENT	EVENT SET BY CYCLIC DISPLAY PROCESSOR TO INDICATE THE SUBROUTINE IS RUNNING	620	500 505 520 560	E	
B100	DFB_NAME_STRUCTURE	CDMV_B_NAMES	STRUCTURE CONTAINING NAMES OF THE FIVE DFB'S AND THE MASS MEMORY DIRECTORY	560	560 620	ST(6)	
B100.5	DFB_NAMES_WITH_MM	CDMV_NBUFS	NAMES OF THE FIVE DFBS AND MASS	560	560	Y	

ID	ITEM HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
DIRECTORY		MEMORY DIRECTORY		620		

UI GENERAL COMPOOL

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
F010	DOWNLIST_BUFFER	COMD_DWNLST	DOWNLIST BUFFER AREA	660 665	DL	A(128),I,INIT (-5232,1,440),1 22 4(-21946))	
F020	DOWNLIST_SPARE	COMD_NME_SPARE	SPARE (NOT USED)			I	
F030	DOWNLIST_VARIABLE_DUMP	COMD_VAR_DMP	DOWNLIST VARIABLE DUMP DATA AND ADDRESS AREA	660 665		ST(1)	
F030.01	DOWNLIST_RATE_1_COUNTER	COMD_CNTR_1	DOWNLIST COUNTER TO CONTROL RATE 1 PARAMETERS	660 665		I,INIT(1)	
F030.02	DOWNLIST_RATE_5_COUNTER	COMD_CNTR_5	DOWNLIST COUNTER TO CONTROL RATE 5 PARAMETERS	660 665		I,INIT(1)	
F030.03	DOWNLIST_RATE_12_COUNTER	COMD_CNTR_12	DOWNLIST COUNTER TO CONTROL RATE 12.5 PARAMETERS	660 665		I,INIT(1)	
F030.04	VARIABLE_DUMP_NAME1	COMD_PARM_DUMP1	DOWNLIST VARIABLE DUMP ADDRESS SLOT		DL	I,INIT,NAME(COM D_DWNLST$128)	V92U5500C
F030.05	VARIABLE_DUMP_NAME2	COMD_PARM_DUMP2	DOWNLIST VARIABLE DUMP ADDRESS SLOT		DL	I,INIT,NAME(COM D_DWNLST$128)	V92U5502
F030.06	VARIABLE_DUMP_NAME3	COMD_PARM_DUMP3	DOWNLIST VARIABLE DUMP ADDRESS SLOT		DL	I,INIT,NAME(COM D_DWNLST$128)	V92U5504
F030.07	VARIABLE_DUMP_NAME4	COMD_PARM_DUMP4	DOWNLIST VARIABLE DUMP ADDRESS SLOT		DL	I,INIT,NAME(COM D_DWNLST$128)	V92U5506
F030.08	VARIABLE_DUMP_NAME5	COMD_PARM_DUMP5	DOWNLIST VARIABLE DUMP ADDRESS SLOT		DL	I,INIT,NAME(COM D_DWNLST$128)	V92U5508
F030.09	VARIABLE_DUMP_NAME6	COMD_PARM_DUMP6	DOWNLIST VARIABLE DUMP ADDRESS SLOT		DL	I,INIT,NAME(COM D_DWNLST$128)	V92U5510
F030.10	VARIABLE_DUMP_NAME7	COMD_PARM_DUMP7	DOWNLIST VARIABLE DUMP ADDRESS SLOT		DL	I,INIT,NAME(COM D_DWNLST$128)	V92U5512
F030.11	VARIABLE_DUMP_NAME8	COMD_PARM_DUMP8	DOWNLIST VARIABLE DUMP ADDRESS SLOT		DL	I,INIT,NAME(COM D_DWNLST$128)	V92U5514
F030.12	VARIABLE_DUMP_NAME9	COMD_PARM_DUMP9	DOWNLIST VARIABLE DUMP ADDRESS SLOT		DL	I,INIT,NAME(COM D_DWNLST$128)	V92U5516
F030.13	VARIABLE_DUMP_NAME10	COMD_PARM_DUMP10	DOWNLIST VARIABLE DUMP ADDRESS SLOT		DL	I,INIT,NAME(COM D_DWNLST$128)	V92U5518
F040	MAIN_MEMORY_DUMP_REQ	COMD_MAIN_MEM_DMP_REQ	MAIN MEMORY DUMP REQUESTED FLAG	660 665		BT,INIT(OFF)	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
F050	MAIN_MEMORY_DUMP_ACTIVE	CDWD_MAIN_MEM_DMP_IND	MAIN MEMORY DUMP ACTIVE FLAG	910 660 665		BT,INIT(OFF)	
F060	MAIN_MEMORY_DUMP_START_ADDRESS	CDWD_MAIN_MEM_DMP_PTR	CURRENT START ADDRESS FOR NEXT FRAME OF MAIN MEMORY DUMP	660 665 910	915	ID,INIT(0)	
F070	MAIN_MEMORY_DUMP_LENGTH	CDWD_MAIN_MEM_DMP_LEN	CURRENT LENGTH LEFT TO DUMP FOR MAIN MEMORY DUMP	660 665 910	915	ID,INIT(0)	
F080	DOWNLIST_FORMAT_ID	CDWD_DP_FORMAT_ID	DOWNLIST ACTIVE FORMAT NUMBER	660 665	DL	I,INIT(4)	
F090	DOWNLIST_MAIN_MEMORY_DUMP_ID	CDWD_DMP_FMT_ID	MAIN MEMORY DUMP FORMAT NUMBER		660 665 DL	I,INIT(90)	
F100	DOWNLIST_FRAME_NUMBER	CDWD_DCOVL_FRME_CNT	CURRENT DOWNLIST FRAME COUNTER	660 665	DL	I,INIT(0)	
F110	DOWNLIST_SPARE2	CDWD_REQUEST_PROCESSED	NOT USED			I,EVENT	

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
J002	NUMBER_OF_MAJOR_FUNCTIONS	CZ1K_D_NUMBER_MAJOR_FUNCTIONS	THE MAXIMUM NUMBER OF MAJOR FUNCTIONS THAT MAY BE SUPPORTED BY A GPC			REPLACE(2)	
J004	NUMBER_OF_DEUS	CZ1K_D_NUMBER_DEUS	THE MAXIMUM NUMBER OF DEUS THAT MAY BE SUPPORTED BY A GPC		400 405 500 510 520 700 800 815	REPLACE(3)	
J006	MACT_COPIES	CZ1K_D_MACT_COPIES	THE NUMBER OF MACT COPIES REQUIRED TO SUPPORT THE MAJOR FUNCTIONS-OPS AND SPEC CONTROL SEGMENTS		505 560	REPLACE(7)	
J010	PHASE_ELAPSED_TIMES	CZ1V_D_PHASE_ELAPSED_TIME	ME PHASE ELAPSED TIMES			A(3),DSC	
J010.01	PLM_PHASE_ELAPSED_TIME		PLM ELAPSED TIME			DSC	
J010.02	GN&C_PHASE_ELAPSED_TIME		GNC PHASE ELAPSED TIME			DSC	
J010.03	SM_PHASE_ELAPSED_TIME		SM PHASE ELAPSED TIME			DSC	
J020	KEYBOARD_MESSAGE_READY_EVENT	CZ1E_D_MCDS_EVENT	REQUEST KEYBOARD MESSAGE PROCESSING	405 710 800 830	500 815	E	
J030	NULL EVENT	CZ1E_D_NULL_EVT	EVENT USED IN WAIT ALGORITHMS WHEN NO EVENT ADVANCEMENT IS DESIRED		810 910 950	E	
J040	MISCELLANEOUS_APPLICATIONS_CONTROL_TABLE	CZ1V_D_MACT	MISCELLANEOUS APPLICATIONS CONTROL TABLE USED TO PASS CONTROL AND DATA BETWEEN UI AND APPLICATIONS CONTROL SEGMENTS	505 510 540 550 560	505 510 540 550 560	ST(7)	
J040.01	MACT_SERVICE_REQUEST_FLAGS	CZ1B_D_FLAGS	CONTROL SEGMENT SERVICE REQUEST FLAGS.(BITS 1,3,4,5,7,10,14,15,16 ARE NOT USED)	540 550 820 870 880	505 540 810 820	BS(16)	
J040.03	NEW_DISPLAY_		BIT 2=1(ENABLED)-IS A NEW DISPLAY	505	505	BT	

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
	REQUEST_FLAG		REQUEST	540			
J040.07	ITEM_PROCESSING_ COMPLETE_FLAG		BIT 6=1(ENABLED)-INDICATES ITEM PROCESSING COMPLETION BY THE CONTROL SEGMENT	505 540	505	BT	
J040.09	CONTROL_SEGMENT_ TERMINATION_FLAG		BIT 8=1(ENABLED)=IS A CONTROL SEGMENT TERMINATION REQUEST	505 550	505	BT	
J040.10	CONTROL_SEGMENT_ FIRST_DISPLAY_ ENTRY_FLAG		BIT 9=0(DISABLED)-IS A FIRST DISPLAY ENTRY REQUEST	540	540	BT	
J040.12	INIT_BLOCK_FLAG		BIT 11=1(ENABLED)-IS AN INIT BLOCK FLAG TO ALLOW MODE REINITIALIZATION UPON MODE REENTY	540 550	540	BT	
J040.13	GPC RECONFIGURATION_ COMPLETION		BIT 12=1(ENABLED)=GPC RECONFIGURATION PROCESSING IN THE RS GPC(S) HAS COMPLETED	505 820 870 880	505 560	BT	
J040.14	GPC_SECONDARY_ RECONFIGURATION_ REQUEST		BIT 13=1(ENABLED)_SECONDARY GPC RECONFIGURATION REQUEST	505 820 870	505	BT	
J040.18	MACT_APPLICATION_ CONTROL_FLAGS	CZ1B_D_FLAG2	APPLICATION CONTROL FLAGS 4-16 NOT USED	505 550 560	550	BS(16)	
J040.19	CONTROL_SEGMENT_ BLOCK_TERMINATION_ REQUEST		BIT 1=1(ENABLED)-BLOCK TERMINATION REQUEST	505 550 560	550		
J040.20	CONTROL_SEGMENT_ MODE_TERMINATION_ REQUEST		BIT 2=1(ENABLED)-MODE TERMINATION REQUEST	505 550 560	550		
J040.21	CONTROL_SEGMENT_ TERMINATION_ REQUEST		BIT 3=1(ENABLED)-SEGMENT TERMINATION REQUEST				
J040.35	MAJOR_FUNCTION_ID	CZ1V_D_MF_ID	MF ID OF CONTROL SEGMENT (CS) ASSIGNED TO THIS MACT COPY	560	505 560	I	
J040.36	CURRENT_DISPLAY_ NUMBER	CZ1V_D_DISP	CURRENT DISPLAY TO BE UPDATED FOR THIS CS	505 540 550 560	505 560	I	
J040.37	CURRENT_OPS_NUMBER	CZ1V_D_OPS	CURRENT OPS NUMBER FOR THIS MF (SEE GRAMMAR MAPCD D_OPS_NUMBER IN	550 560	440 550	I	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
			APPENDIX H)				
J040.38	CURRENT_MODE_NUMBER	CZ1V_D_MOD	CURRENT MODE FOR THIS OPS.(SEE GRAMMAR MACRO D_MODE_NUMBER IN APPENDIX H)	540 550	550 810 910 950	I	
J040.39	CURRENT_BLOCK_NUMBER	CZ1V_D_BLK	CURRENT BLOCK NUMBER OF THE CONTROL SEGMENT.(SEE GRAMMAR MACRO D_BLOCK_NUMBER IN APPENDIX H)	550	810 910 950	I	
J040.40	NEW_DISPLAY_NUMBER	CZ1V_D_NEW_DISP	REQUEST (NEW) DISPLAY NUMBER FOR THIS CONTROL SEGMENT	505 540	505	I	
J040.41	NEW_OPS_NUMBER	CZ1V_D_NEW_OPS	REQUESTED OPS NUMBER WHEN THE CURRENT OPS IS BEING TERMINATED. (SEE GRAMMAR MACRO D_NEW_OPS_NUMBER IN APPENDIX H)	550 560	550	I	
J040.42	NEW_MODE_NUMBER	CZ1V_D_NEW_MOD	REQUESTED MODE NUMBER UPON A MODE CHANGE.(SEE GRAMMAR MACRO D_NEW_MODE_NUMBER IN APPENDIX H)	540 550 560	550		
J040.43	NEW_BLOCK_NUMBER	CZ1V_D_NEW_BLK	REQUESTED BLOCK NUMBER UPON A BLOCK CHANGE	540 550 560	550	I	
J040.44	DFB_NDX	CZ1V_D_DFB_NDX	DFB INDEX FOR CURRENT DISPLAY (NOT USED)			I	
J040.45	NAME_OF_KEYBOARD_MESSAGE_EVENT	CZ1E_D_KYBD_MSG_EVT	NAME OF EVENT ARRAY MEMBER.USED TO SIGNAL A KEYBOARD MESSAGE IMPACT	505 510 540 550 560 810	540	Y	
J040.46	OPS_ADVANCE_EVENT	CZ1E_D_OPS_ADV_EVT	NAME OF EVENT TO INDICATE OPS ADVANCEMENT	810 900 910 950	540	Y	
J040.47	MODE_ADVANCE_EVENT	CZ1E_D_MOD_ADV_EVT	NAME OF EVENT TO INDICATE MODE ADVANCEMENT	810	540	Y	
J040.48	BLOCK_ADVANCE_EVENT	CZ1E_D_BLK_ADV_EVT	NAME OF EVENT TO INDICATE BLOCK ADVANCEMENT	810 900 910 950	540	Y	
J040.49	KEYBOARD_MESSAGE_ID	CZ1V_D_KEY_ID	ID IDENTIFYING THE KEYBOARD MESSAGE ENTERED.(SEE GRAMMAR MACRO KEY IN	505 510 540	540 810 900	I	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
			APPENDIX H)		910		
J040.50	ITEM_NUMBER	CZ1V_D_ITEM_NO	ITEM NUMBER FOR ITEM INPUTS.(SEE GRAMMAR MACRO ITEM_NO IN APPENDIX H)	510	860 910 950	I	
J040.51	KEYBOARD_INPUT_DEU_NUMBER	CZ1V_D_DEU_NUMBER	THE DEU AT WHICH THE LAST KEYBOARD INPUT WAS MADE.(SEE GRAMMAR MACRO D_DEU_NUMBER IN APPENDIX H)	505 510 560	505 810 900 910 960	I	
J040.52	SCALAR_ITEM_INPUT	CZ1V_D_ITEMS	SCALAR ITEM DATA INPUT(SEE GRAMMAR MACRO ITEMS IN APPENDIX H)	510	950	S	
J040.53	INTEGER_ITEM_INPUT	CZ1V_D_ITEM_I	INTEGER ITEM DATA INPUT (SEE GRAMMAR MACRO ITEM_I IN APPENDIX H)	510	860 910 950	I	
J040.54	BINARY_ITEM_INPUT	CZ1V_D_ITEM_O_B_1	BINARY ITEM DATA INPUT(SEE GRAMMAR MACRO ITEM_O IN APPENDIX H)	510		BS(32)	
J040.55	SECONDARY_BINARY_ITEM_INPUT	CZ1V_D_ITEM_O_B_2	SECOND BINARY ITEM DATA INPUT (NOT USED)			BS(32)	
J040.56	TERMINATION_SEQUENCE_INDEX	CZ1V_D_SEQUENCE_INFO	NUMBER OF INDEX TO INDICATE THE NUMBER OF BLOCKS IN A MODE OR SPEC OR THE NUMBER OF MODES IN AN OPS	560	540 550 560		
J050	DEU_MESSAGE_READY_FLAGS	CZ1B_D_DIT_MSG_READY	DEU INPUT TABLE MESSAGE READY FLAGS	405 505 815 830	500 505	BS(16)	
J050.01	DEU_MESSAGE_READY_FLAG		BIT 1=1(ENABLED)-DEU1 MESSAGE READY IN DIT				
J050.02	DEU2_MESSAGE_READY_FLAG		BIT 2=1(ENABLED)-DEU2 MESSAGE READY IN DIT				
J050.03	DEU3_MESSAGE_READY_FLAG		BIT 3=1(ENABLED)-DEU3 MESSAGE READY IN DIT BIT 4-16 -NOT USED				
J060	DEU_INPUT_TABLE	CZ1V_D_DIT	DEU MESSAGE INPUT TABLE.(DIT)	400 405 670 790 815 830	500 505 510 560 660 665	ST(3)	
J060.01	DIT_STATUS	CZ1V_D_DIT_STAT	DEU INPUT STATUS	400	400	A(4),BS(16)	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES
				440	660	
					665	
J060.02	DEU_MESSAGE_HEADER	CZ1B_D_BIT_MSG_HEADR(FIOD8F1P, FIOD8F2P, FIOD8F3P)	DEU MESSAGE HEADER FOR THE GIVEN DEU	400	505	BS(16)
				406	510	
				670	563	
				790	660	
				815	665	
				830		
J060.03	DEU_MESSAGE_TYPE		BITS 1-4 WHERE, 0=NOT USED 1=KEYBOARD MESSAGE 2=BITE STATUS 3=MODE STATUS REPLY 4=DEU MEMORY DUMP 5-15=NOT USED	560		
J060.08	DEU_ID		BITS 6-8 ARE THE DEU ID OF THIS DEU INPUT			
J060.11	DEU_MF_SWITCH_SETTING		BITS 9-10 ARE THE MF SWITCH SETTING FOR THIS DEU WHERE 0=PLM 1=GNC 2=SM 3=INVALID SETTING		500	
J060.13	DEU_FREEZE_FLAG		BIT 11=1(ENABLED) INDICATES FREEZE DISPLAY,0(DISABLED) INDICATES UNFREEZE (UPDATE) DISPLAY			
J060.15	DEU_KEYBOARD_PENDING_FLAG		BIT 13=1(ENABLED) INDICATES A KEYBOARD MESSAGE IS PENDING IN THIS DEU			
J060.16	DEU_STAND_ALONE_SELF_TEST		BIT 14=1(ENABLED) INDICATES A STAND ALONE SELF TEST IS IN PROGRESS IN THE DEU			
J060.17	DEU_BITS_STATUS_AVAILABLE		BIT 15=1(ENABLED) INDICATES BITE STATUS IS AVAILABLE IN THE DEU			

COMMON COMPOOL(CZ1)

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	NNL
J060.18	DEU_INITIALIZATION		BIT 16=1(ENABLE) INDICATES INITIALIZATION IS REQUIRED BY THIS DEU				
J060.19	DEU_NUMBER_OF_KEYSTROKES	CZIV_D_NUMOFKEYS	THE NUMBER OF KEYSTROKE/CODES CONTAINED IN THIS KEYBOARD MESSAGE	400 405 670 740 815 830	500 505 515 660 665	1	V93M22102P V93M22202P
J060.20	KEYBOARD_MESSAGE	CZIV_D_DIT_KYBD_MSG	KEYSTROKE CODES MAKING UP THIS KEYBOARD MESSAGE. THE FOLLOWING IS A TABLE OF VALID CODES. # DEU CODE — FUNCTION 0 1A7 — ZERO 1 21F — ONE 2 2FE — TWO 3 45E — THREE 4 55E — FOUR 5 56E — FIVE 6 5EE — SIX 7 76C — SEVEN 8 8F8 — EIGHT 9 9F9 — NINE 10 C09 — MINUS(-) 11 C68 — PLUS(+) 12 D05 — PERIOD(.) 13 D87 — FAULT 14 DDC — (NOT USED) 15 E45 — (NOT USED) 16 E83 — ACK 17 E8A — OPS 18 EC7 — SPEC 19 CF7 — DISPLAY 20 D06 — ITEM 21 A06 — (NOT USED) 22 A03 — MAS RESET 23 DDA — (CLEAR DEU ONLY) 24 E3 — (NOT USED) 25 FA — CMPTR/CRT 26 F5 — (NOT USED) 27 D9 — (NOT USED) 28 E7 — ENTER 29 EC — CMPTR/BUS 30 E6 — EXEC 31 — PRO	A(40),1	500 505 510 560 665		V93M20023P V93M20024P V93M20025P V93M20026P V93M20027P V93M20028P V93M20029P V93M20030P V93M20031P V93M20032P V93M20033P V93M20034P V93M20035P V93M20036P V93M20037P V93M20038P V93M20039P V93M20040P V93M20041P V93M20042P V93M21101P V93M21102P

COMMON COMPOOL (CZ11)

ID	ITEM/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
						V93M21103P
						V93M21104P
						V93M21105P
						V93M21106P
						V93M21107P
						V93M21108P
						V93M21109P
						V93M21110P
						V93M21111P
						V93M21112P
						V93M21113P
						V93M21114P
						V93M21115P
						V93M21116P
						V93M21117P
						V93M21118P
						V93M21119P
						V93M21120P
						V93M21121P
						V93M21122P
						V93M21123P
						V93M21124P
						V93M21125P
						V93M21126P
						V93M21127P
						V93M21128P
						V93M21129P
						V93M21130P
						V93M21131P
						V93M21132P
						V93M21133P
						V93M21134P
						V93M21135P
						V93M21136P
						V93M21137P
						V93M21138P
						V93M21139P
						V93M21140P
						V93M21141P
						V93M21142P
						V93M22201P
						V93M22202P
						V93M22203P
						V93M22204P
						V93M22205P
						V93M22206P
						V93M22207P
						V93M22208P
						V93M22209P
						V93M22210P
						V93M22211P
						V93M22212P
						V93M22213P

COMMON COMPOOL(CZ1)

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
J062	DEU_POLL_FLAGS	CZIB_D_DEU_POLL_FLAG	FLAGS TO INDICATE POLLING OF THE DEU'S	405 400 700 790 880	820 400 915 405 560 790	BS(16)	V93M22140P
J062.01	DEU_POLL_FLAG		BIT 1=1(ENABLED)-POLL DEU1	815		BT	V93M22150P
J062.02	DEU2_POLL_FLAG		BIT2=1(ENABLED)=POLL DEU2			BT	V93M22160P
J062.03	DEU3_POLL_FLAG		BIT3=1(ENABLED)-POLL DEU3			BT	V93M22170P
J064	KEYBOARD_MESSAGE_READY_EVENT_ARRAY	CZIF_D_KYBD_MSG_ARRAY	MACT APPLICATION KEYBOARD MESSAGE READY EVENT ARRAY	550	550 560	E(7)	V93M22180P
J065	APPLICATION_MACT_INDEX	CZIV_D_MACT_INDEX	THE MACT INDEX PASSED TO A CONTROL SEGMENT FOR THE EXCLUSIVE USE OF A GIVEN CONTROL SEGMENT	563	810 900 910 950	I	V93M22190P
J068	DEU_LOAD_TABLE	CZIB_D_DLT	THE DEU LOAD TABLE WITH A BIT SET (=1)CORRESPONDING TO THE DEU TO	790 405	790	BS(16)	V93M22200P
							V93M22210P
							V93M22220P
							V93M22230P
							V93M22240P
							V93M22250P
							V93M22260P
							V93M22270P
							V93M22280P
							V93M22290P
							V93M22300P
							V93M22310P
							V93M22320P
							V93M22330P
							V93M22340P
							V93M22350P
							V93M22360P
							V93M22370P
							V93M22380P
							V93M22390P
							V93M22400P
							V93M22410P
							V93M22420P

COMMON COMPOOL(CZ1)

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	OST	ATTRIBUTES	MML
			BE LOADED				
J068.01	DEU1_LOAD_FLAG		BIT 1=1(ENABLED) TO LOAD DEU1			BT	
J068.02	DEU2_LOAD_FLAG		BIT 2=1(ENABLED) TO LOAD DEU2			BT	
J068.03	DEU3_LOAD_FLAG		BIT 3=1(ENABLED) TO LOAD DEU3				
J070	DEU_BITE_STATUS	CZ1B_D_BITE	DEU BITE STATUS TABLE	405 790	405 790	ST(3)	
J070.01	DEU_BITE_TRANSMISSION_STATUS	CZ1V_D_XMIS_STAT	DEU BITE TRANSMISSION STATUS	400 405 790	400 405 790	A(4),I	
J070.05	DEU_BITE_MESSAGE_HEADER	CZ1B_D_DEU_MSG_HDR/FIODBF1B, FIODBF2B, FIODBF3B	DEU BITE MESSAGE HEADER	261 405 790	405 790	BS(16)	
J070.21	DEU_BITE_STATUS_MESSAGE	CZ1V_D_BITE_STAT	DEU BITE STATUS MESSAGE	405 790	405 790	A(4),BS(15)	
J070.22	DEU_HARDWARE_REGISTER_ONE		DEU HARDWARE REGISTER ONE			BS(16)	
J070.23	DEU_IPL_PERFORMED		BIT 1=IPL HAS BEEN PERFORMED			BT	
J070.24	DEU_IPL_ERROR		BIT 2=IPL ERROR			BT	
J070.25	DEU_IPL_CIRCUIT_ERROR		BIT 3=PERIODIC CHECK OF IPL CICUIT ERROR			BT	
J070.70	DEU_SOFTWARE_REGISTER		DEU SOFTWARE REGISTER			BS(16)	
J070.72	DEU_INITIALIZATION PERFORMED		BIT 2= INITIALIZATION PERFORMED			BT	
J070.81	DEU_CHECK_SUM_ERROR		BIT 11= CHECKSUM ERROR			BT	
J080	TIME_OF_SIP	CZ1V_A_TSIP	TIME AT WHICH AIE_SIP IS SCHEDULED	700	495 700 710 830	DS	
J090	BFCS_ENGAGE/FIOBFCSE	CZ1V_A_BFCS_ENGAGE	BFCS ENGAGE			I	
J100	IN_EVENT_TIME_STATUS	CZ1V_A_IN_ET_STAT	TIME MANAGEMENT DISPLAY EVENT TIME STATUS	800		I	
J110	SUM_WORD	CZ1B_G_SUM_WD	GPC REDUNCY MANAGEMENT SUM WORD	GPP	750	DI	

ID	ITEM	HAL/NAME	DESCRIPTION (SEE GNRC FCS COMMANDS PROCESSOR)	SPC # 750	DST #	ATTRIBUTES	MNL
J120	VCD_OPS_FLAG	CZ1B_V_VALID_OPS	FLAG SET WHEN VEHICLE CHECKOUT OPS ARE ACTIVE		440 460	R0	
J130		CZ58_V_HOA				B0	
J140		CZ1B_R_O_RDYFLG				B0	
J150	FC_BUSES_COMFAULT_WORD	CZEB_COMM_FAULT (FIDBCFSI)	I/O COMFAULT WORD FOR THE FLIGHT CRITICAL (FC) BUSES	205		BS(32)	
J160	MACT SERVICE REQUEST READY EVENTS	CZ1E_D_MACT_EVTS	MACT SERVICE REQUEST READY EVENTS USED TO SYNC APPLICATION REQUESTS	505 540 550 820 870	505 810	A(71,F	

COMMON COMPOOL (CZ1)

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
L002	MESSAGE_HEADER_MASK	DMMV_MSG_HDR	AN OVERLAY FOR BITS 9-24 OF CZ18_0_BIT_MSG_HEADER,USED TO REFERENCE ONLY THE 16 DATA BITS OF THE DEU TO GPC DATA WORD	405	405	BS(16)	
L003	KEYSTROKE_STORAGE	DMMVL_KEY	USED TO STORE THE 8-BIT KEYSTROKE CODE	405	405	BS(8)	
L004	INTERNAL_KEYSTROKE CODE	DMMKL_NEW_KEY	THE DECIMAL REPRESENTATION OF THE KEYSTROKE	405	405	A(32)BIT8 INIT (DEC'31,22,16,28,0, 1,2,3,4,5,6,7,8 ,9,11,10,12,20, 30,18, 27,7,19,13,25,29,15,21,23, 14,24,26)	
L005	HEX_KEYSTROKE_CODE	DMMKL_KEY_5BIT	8 BIT CODE ARRAY OF ALL LEGAL KEYSTROKES COMPARE AGAINST THE KEYBOARD MESSAGE FOR CONVERSION TO A DECIMAL NO.	405	405	A32 BIT8 INIT HEX EC,86,83, D9,A7,1F,2F,3F, 4F,5E,6E,7C, 8F,9F,CB,CD,D5, CE,EC,BC,E5, BA,C7,B9,E3,E9, B5,D6,D3,DC, DA,EA	
L006	HEADER_MESSAGE_TYPE		BITS 1-4=MESSAGE TYPE 0=NOT USED 1=KEYBOARD MESSAGE RESPONSE 2=BITE STATUS RESPONSE 3=MODE STATUS RESPONSE 4=MEMORY FILL BIT 5 - NOT USED				
L007	HEADER_DEU_NO		BIT6-8 DEU NUMBER 5=DEU1 6=DEU2 7=DEU3				
L008	HEADER_MF_SETTING		BIT 9-10-MAJOR FUNCTION SWITCH SETTING 0=PAYLOAD MAINTENCE 1=GN&C 2=SYSTEMS MAINTAINENCE 3=INVALID SETTING BIT 11 NOT USED				
L009	HEADER_FREEZE		BIT 12-FREEZE BIT INDICATOR 0=UNFREEZE 1=FREEZE				
L010	HEADER_KEYBOARD		BIT13_KEYBOARD MESSAGE PENDING 0=NO MESSAGE IN MODE STATUS RESPONSE 1=MESSAGE PENDING				

USER INTERFACE LOCAL DATA

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES
L011	HEADER_SAST		BIT14-STAND-ALONE-SELF TEST (SAST) 0-NO SAST 1-SAST ACTIVE FOR THIS DEU			
L012	HEADER_DEU_BITE		BIT15-INDICATES WHETHER THERE IS A BITE ERROR IN ONE OF THE DEU'S REGISTERS 0=NO BITE DETECTED 1=BITE AVAILABLE FOR DOWNLISTING			
L013	HEADER INITIALIZATION		BIT 16-INDICATES WHETHER DEU IS REQUESTING INITILIZATION 0-NO 1-INITIALIZATION REQUESTED			
L014	SAME_DEU_BITE	DMMB_SAME BITE	FLAG SET WHEN A BITE ERROR IS DETECTED, USED FOR ANNUNCIATION SUPPRESSION ON SUCCESSIVE BITE ERRORS FROM THE SAME DEU	405	405	BS(16) INIT (HEX'0000')
L015	INITIALIZATION_ ERROR_FLAG	DMMV_INIT_ERR	FLAG USED FOR ANNUNCIATION SUPPRESSION ON SUCCESSIVE INITIALIZATION ERRORS	405	405	I
L016	MF_TIMING_NEEDED	DMMB_MF_TIMING	A FLAG WHEN ON INDICATES MAJOR FUNCTION SWITCH LOGIC IS REQUIRED	405	405	BS(16) INIT (HEX'0000')
L017	MF_3CYCLE_TIMER	DMMV_MF_TIMR	A COUNTER FROM 1TO3 INDICATING THE 600 MILLISEC CYCLE FOR MF SWITCH SETTING	405	405	A
L018	BYPASS_LOGIC	DMMV_BYPASS		405	405	I
L019	CURRENT_MF_SETTING	DMMV_MF_NEW	STORAGE FOR THE CURRENT SETTING OF THE MAJOR FUNCTION SWITCH FOR THE DEU BEING PROCESSED	405	405	A
L020	PREVIOUS_FREEZE_ BIT_SETTING	DMMV_OLD_FRZ	PREVIOUS FREEZE BIT	405	405	BS(16) INIT(HEX'0000')
L021	CURRENT_FREEZE BIT_ SETTING	DMMV_NEW_FRZ	CURRENT FREEZE BIT SETTING FROM THE DEU POLL RESPONSE	405	405	BS(16) INIT HEX '0000'
L022	MF_CHANGE_FOR_ICC	DMMB_MF_ICC	USED AS PARAMETER PASSED TO ICC_ MESSAGE_COLLECTOR TO INDICATE CHANGE IN MAJOR FUNCTION	405	405	A(2)BS(16)INIT(HEX'1402 0101')
L024	ICC_ACCEPTANCE_ STATUS	DMMB_ICC_ST	FLAG SET BY ICC_MESSAGE_COLLECTOR INDICATES WHEN MAJOR FUNCTION CHANGED IS ACCEPTED BY COLLECTOR	405	405	BS(16) INIT (HEX'FF00')
L025	MODULE_IDENT	DMA_D	USED A PARAMETER PASSED BY MODULES REQUESTING ANNUNCIATION SUPPORT TO DISTINGUISH APPLICATIONS FROM SYSTEM SERVICE AND USER INTERFACE	675	675	I

USER INTERFACE LOCAL DATA

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
L026	ANNUNCIATION_MSG_NUMBER	DMA_I	A VALUE PASSED BY ANNUNCIATION USERS INDICATES ERROR MESSAGE THEY WISH TO BE DISPLAYED ON THE FSP	675	675	I	
L027	CAUTION_WARNING_PARM	DMA_S	BIT PASSED TO ANNUNCIATION SUPPORT INDICATING THE CAUTION AND WARNING PARAMETER I.E. UP ARROW ↑ OR DOWN ARROW ↓ TO DISPLAY WITH THE REQUESTED CLASS TWO FAULT MESSAGE	675	675	BT	
L028	ID_CODE	DMS_ID_CODE	NUMBER ASSOCTD WITH EACH TUTORIAL MESSAGE NEEDED FOR DOWN LIST	650	650	I	
L029	MSG_PRIORITY	DMS_PRIORITY	NOT USED	650	650	I	
L030	DISPLAY_NUMBER	DMS_DISPLAY_NUMBER	NUMBER ASSOCIATED WITH THE OPS OR SPEC DISPLAY WITH TUTORIAL MESSAGE IS TO BE PLACED OR MAJOR FUNCTION I.D.	650	650	I	
L031	DEU_DESTINATION	DMS_DEU_DESTINATION	MAJOR FUNCTION WITH WHICH THE DISPLAY IS ASSOCIATED I.E. 1=SM 2=GNC	650	650	I	
L032	MESSAGE_TEXT	DMS_MESSAGE_TEXT	CHARACTER STRING OF 48 CHARACTERS REPRESENTING THE TUTORIAL MESSAGE THE APPLICATION MODULE DESIRES DISPLAYED ON THE CRT		650 655	C(48)	
L033	FAULT_ERROR_MESSAGE_MAJ	CDLB_FMPT_MAJOR	THE MAJOR FIELD OF THE CURRENT MESSAGE IN THE FSP	685	670 680 685	BS(10)	
L034	FAULT_ERROR_MESSAGE_MINOR	CDLB_FMPT_MINOR	THE MINOR FIELD OF THE CURRENT FAULT MESSAGE IN THE FSP	685	670 680 685	BS(7)	
L035	CAUTION_AND_WARNING_PARAMETER_STATUS	CDLB_FMPT_IND	OUT OF LIMIT INDICATOR USED FOR CLASS TWO FAULT MESSAGES	685	670 680 685	BS(1)	
L036	GPC_SOURCE	DMS_GPC_CODE(CDLB_GPC_ID)	THE FIELD INDICATING THE GPC'S IN WHICH THE FAULT MESSAGE WAS DETECTED	685	485 680 685	BS(5)	
L037	TIME_OF_MESSAGE_DETECTION	CDL_FAULT_MSG_TIME	THE PART OF FAULT MESSAGE FIELD DISPLAYING THE TIME THE FAULT WAS DETECTED BY ANNUNCIATION	144 670	620 670 680 685	I	
L038	FAULT_SUMMARY_PAGE_BUFFER	CDL_FAULT_SUMMARY_PAGE	THE PART OF ANNUNCIATION_COMPOOL THAT CONTAINS ALL THE DATA	685	620 680	ST(20)	

USER INTERFACE LOCAL DATA

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
			NECESSARY TO DISPLAY THE FAULT_ SUMMARY_PAGE ON THE CRT		685		
L039	FAULT_MESSAGE_TIME	CDL_FAULT_SUMMARY_ TIME	THE PART OF ANNUNCIATION_COMPOOL THAT CONTAINS THE MISSION ELAPSED TIME FOR EACH FAULT MESSAGE IN THE FAULT_SUMMARY_PAGE_BUFFER	685	620 680 685	I(20)	
L040	MESSAGE_CLASS	CDL_MSG_CLASS(DMS_ MSG_CLASS,CDL8_ EMPT_CLASS)	THIS PARAMETER IS CALLED BY EACH MODULE THAT REFERENCES IT BY A DIFFERENT NAME, BUT ITS VALUE IS THE SAME. IT GIVES THE CLASS OF CURRENT FAULT MESSAGE BEING PROCESSED IN FSP DISPLAY. IT IS REFERENCED TO DETERMINE THE ORDER OF ANNUNCIATION ON THE CRT AND THE APPROPRIATE LIGHTS AND ALARMS TO ENABLE ON THE PAYLOAD MDMS	670 675	485 670 680 685	BS(4)	
L041	APPLICATION_ INTERFACE_TABLE	CDL_AIT	PROVIDES COMMUNICATION BETWEEN L/A PROCESSING CYCLES AND BETWEEN L/A PROC. AND UI/APPLICATION MODULES	670 680 685	485 670 680 685	BS(16)	
L042	FSP_DIRECTORY	CDL_FSP_DIRECTORY	PROVIDES INFORMATION AS TO EACH FAULT MESSAGE IN THE FSP	680 685	620 680 685	ST(20)	
L042.02	ANNUNCIATED_OR_NOT	CDL_ANNUNCIATED_OR_ NOT	IF ENABLED THIS BIT MEANS THAT THE CORRESPONDING FAULT MESSAGE HAS ALREADY BEEN SEEN AND ACKNOWLEDGED BY THE CREW	685	680 685	BT	
L042.04	MESSAGE_CLASS	CDL_MSG_CLASS	GIVES THE FAULT CLASS I.E. 2,3, OR 4 OF THE CORRESPONDING MESSAGE IN THE FSP	685	680 685	BS(4)	
L042.06	DISPLAY_POSITION	CDL_FSP_DISPLAY _ POSITION	GIVES THE POSITION ON THE FSP OF WHICH THE CORRESPONDING FAULT MESSAGE IS TO BE DISPLAYED	685	620 680 685	BS(5)	
L042.08	CONVERSION_BIT	CDL_CONVERSION_ BIT	THIS BIT WHEN ENABLED IS AN INDICATION FOR CYCLIC_DISPLAY_ PROCESSOR THAT THE TIME FIELD ASSOCIATED WITH THIS FAULT MESSAGE HAS TO BE CHANGED INTO FCW FORMAT BEFORE IT CAN BE DISPLAYED ON THE CRT VIA THE FAULT KEY	685	620 680 685	BT	
L043	START_INDEX	DMR_X	USED AS THE START INDEX INTO CDLK_ HDR_BITS FOR COMPARING THE FOLLOWING 32 BITS TO ZERO	670	670	I	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
L044	COPY_NUMBER_OF_HEADER_WORDS	DMR_COPY	USED AS END INDEX INTO CDLK_HDR BITS FOR COMPARING THE PRECEEDING 32 BITS TO ZERO	670	670	I	
L045	MAX_NUMBER_OF_MSG_PER_SSIP_CYCLE	DMR_MAX_MSG	THIS VALUE SET TO TWO TO PREVENT ANNUNCIATION FROM PROCESSING MORE THAN TWO MESSAGES PER SSIP/CYCLE. THIS HELPS PREVENT CYCLE OVERRUNS DUE TO AN ABUNDANCE OF FAULT MESSAGES THAT REQUIRE PROCESSING	670	670		
L046	MAX_NUMBER_OF_MSG_IN_MEMORY	CDLK_HDR_BITS	A VALUE REPRESENTING THE MAXIMUM NUMBER OF FAULT MESSAGES IN THIS MEMORY CONFIGURATION	670	670 675 680	I	
L047	LEAST_NUMBER_OF_MSG_IN_MEMORY	DMR_I	THE MINIMUM NUMBER OF ERROR MESSAGES IN ANY MEMORY CONFIGURATION	670	670	I	
L048	COUNTER_WITHIN_DMR_X_AND DMR_COPY	DMR_J	WHEN THE VALUE OF CDLK_HDR_BITS BETWEEN DMR_X AND DMR_COPY IS NOT EQUAL TO ZERO, THEN DMR_J IS USED AS THE POINTER TO ENABLED BITS BETWEEN DMR_X AND DMR_COPY	670	670	I	
L049	INTERLOCK_VALUE_IN_SSIP_CYCLES	COL_INERLOCK_TIME	FSP INTERLOCK TIME IN SSIP CYCLES	670	670	I	
L050	MET_TIME	DMR_CLOCKTIME	MISSION ELAPSED TIME USED FOR TIME TAGGING OF FAULT MESSAGES AND CALCULATION OF THE INTERLOCK TEST	144 670	670 680 685	DI	
L051	TIME_DISPLAYED_ON_CRT	DMR_DISPLAY_TIME	MET IN A FORMAT THAT CAN BE CONVERTED INTO A CRT DISPLAYABLE FORMAT BY CYCLIC DISPLAY PROCESSOR	670	400 620 670	I	
L052	TIME_IN_SSIP_CYCLES	DMR_MICS512	MET REPRESENTED IN SSIP CYCLES	670	670	DI	
L053	TIME_PUT_IN_DOWNLIST_BUFFER	DMR_DOWNLIST_TIME	LEAST SIGNIFICANT 16 BITS OF DMR_MICS512 L0052 USED FOR DOWNLISTING	670	670 680 685	I	
L054	CALL_COLLECTOR_BIT	DMR_CALL_BIT	WHEN THIS BIT IS ENABLED BY ONE OF THE SEGMENTS OF DMR_FMS IT IS AN INDICATION TO THE ICC_INTERFACE SEGMENT OF DMR_FMS THAT THE PARAMETER LIST FOR PROCESSING OF THE MESSAGE IS TO BE FILLED WITH DATA AND A CALL TO THE ICC_ COLLECTOR IS TO BE MADE FOR A DESCRIPTION OF THE DATA IN THIS PARAMETER LIST SEE ICC DATA TABLES	670	670	BT	

USER INTERFACE LOCAL DATA

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
L055	CLOCKTIME_MINUS_ LAST_MSG_TIME	DMR_DELTA_TIME	THE DIFFERENCE IN TIME BETWEEN TWO CONSECUTIVE CYCLES OF PROCESSING BY DMR_FMS(FAULT_MESSAGE_SCAN)	670	670	I	
L056	LAST_MSG_TIME	DMR_LAST_MSG_TIME	TIME OF WHICH THE LAST FAULT MESSAGE WAS PROCESSED BY DMR_FMS	670	670	I	
L057	STACK_COUNTER	DMT_STACK_CT	INTEGER REPRESENTING THE NUMBER OF FAULT MESSAGES IN FSP THAT HAVE NOT BEEN DISPLAYED ON THE FAULT MESSAGE LINE. THE STACK COUNTER IS ALSO DISPLAYED ON THE CRT	680	680 685	I	
L058	DEU_UPDATE_FLAGS	COMB_MAT_CRT_STAT	ENABLING OF THESE BITS INDICATE TO CYCLIC DISPLAY PROCESSOR THAT AN UPDATE TO THE MESSAGE LINES IS REQUIRED. SEE UI_GENERAL_COMPOOL B010.40 FOR A DETAILED DESCRIPTION	680	620 650 680 685	BS(16)	
L059	NO_FLASH_FCW	DMT_NOFLASH_FCW	A DEU CONTROL WORD WHICH IS A COMMAND TO NOT FLASH A SET OF CHARACTERS ON THE CRT	680	680 685	BS(16)	
L060	FLASH_FCW	DMT_FLASH	A DEU CONTROL WORD WHICH IS A COMMAND TO FLASH A SET OF CHARACTERS ON THE CRT	680	680 685	BS(16)	
L061	MESSAGE_CLASSES	DMT_MSG_CLASS	A VALUE PASSED IN THE PARAMETER LIST TO DMT_ERR FROM DMR_FMS VIA ICC_ROUT. VALID MESSAGE CLASSES ARE 2_CLASS TWO FAULTS MESSAGES 3_CLASS THREE FAULTS MESSAGES 4_CLASS FOUR FAULTS MESSAGES 5_OPERATOR ERROR MESSAGES 6_MESSAGE RESET KEY MESSAGES 7_ACKNOWLEDGE KEY MESSAGES 8_DISP_051_PRO MESSAGES	670	680 685	BS(4)	
L062	CRT_INIT_DYNAMIC_ FILE	DCIBLOCH	THIS IS THE ADDRESS OF BEGINNING OF THE DYNAMIC DATA ON THE DEU.	620	620 675	I, INIT(041C)	
L063	CYCLIC_LOGIC_ CONTROL_FLAG	CLOCFLGS	FLAGS FOR DYNAMIC DATA PROCESSING	620	620 625 640 675	BS(16) INIT(0)	
L063.10			1-INPUT NUMBER IS NEGATIVE				
L063.20			2-BUILD NORMAL INTENSITY FEATURE CONTROL WORD				

USER INTERFACE LOCAL DATA

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
L063.30			3-NOT USED				
L063.40			4-NOT USED				
L063.50			5-NOT USED				
L063.60			6-ANGLE BEING PROCESSED				
L063.70			7-NOT USED				
L063.80			8-NOT USED				
L063.90			9-BUILD END OF REFRESH FEATURE CONTROL WORD				
L063.92			10-NOT USED				
L063.94			11-INDICATE BLANK SIGN TO BE DISPLAYED				
L063.96			12-16 -NOT USED				
L065	CRT_FEATURE_CONTROL_WD1	CLOCFS01	DEU FEATURE CONTROL WORD ONE. USED BY CRT SOFTWARE FOR INTERNAL CONTROL	620	620	BS(16) INIT(3800)	
L066	CRT_FEATURE_CONTROL_WD2	CLOCFS02	DEU FEATURE CONTROL WORD TWO.USED BY CRT SOFTWARE FOR INTERNAL CONTROL	620	620	BS(16) INIT(3006)	
L067	CRT_FEATURE_CONTROL_WD3	CLOCFS03	DEU FEATURE CONTROL WORD THREE. USED BY CRT SOFTWARE FOR INTERNAL CONTROL	620	620	BS(16) INIT(3400)	
L068	CYCLIC_OUTPUT_BUFFER	CLOCOUT	BUFFER USED TO FORMAT FEATURE CONTROL WORDS FOR OUTPUT TO THE DEU	620 630	620 625 630 640	512I	
L069	CYCLIC_NUMBER_DEUS	DCIS#DEU	NUMBER OF DEUS SUPPORTED	620	620	REPLACE (3)	
L070	CRT_NEXT_DYNAMIC_FILL	CLOCNXTA	ADDRESS OF NEXT DYNAMIC I/O TO DEU	620	620 625	Y	
L071	CYCLIC_TIME_CONVERSION_DOT	CLOCDDTC	DUMMY DYNAMIC DATA TABLE ENTRY FOR TIME CONVERSION	620	620	3I	
L072	MET_BUFFER	CLOCTMES	SAVE AREA FOR FORMATTED MET	620 630	620 630	4I	
L073	CYCLIC_DFT_SAVE_AREA	CLOCDFT	TEMPORARY SAVE AREA FOR ADDRESS OF DFT BEING PROCESSED	620	620	1	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
L074	CYCLIC_DEU_SAVE_AREA	CLOCDEU	TEMPORARY SAVE AREA FOR NUMBER OF DEU BEING PROCESSED	620	620	I	
L075	CYCLIC_DYNAMIC_BRANCH_ADDRESS	CLOCBRAN	ADDRESS DEU SOFTWARE IS TO BRANCH TO WHEN PROCESSING DYNAMIC DATA	620	620 625	X, INIT (14434)	
L076	REQUIRED_DISPLAY_NUMBER	DIS_DISPLAY_NUM	DISPLAY NUMBER PROVIDED BY THE CONTROL SEGMENT REQUESTING DISPLAY PROCESSING	810 910 915 950	540	I	
L077	MACT_INDEX	DIS_MACT_INDEX	INDEX TO MACT COPY ASSIGNED TO CALLING CONTROL SEGMENT	810 910 915 950	540		
L078	APPLICATION_SERVICE_EVENT_FLAG	DIS_FLAG_APP_SERVICE_EVT	FLAG TO SIGNAL ONE OR MORE REQUESTS FOR APPLICATION SERVICES	540	540		
L079	CS_PROCESS_LEVEL	DNX_LEVEL	PROCESS LEVEL OF THE INVOKING CONTROL SEGMENT 1-OPS/SPEC 2-MODE 3-BLOCK	810 910 950	550		
L080	CS_MACT_INDEX	DNX_MACT_INDEX	INDEX TO THE MACT COPY ASSIGNED TO CALLING CONTROL SEGMENT	550 810 910 950	550 800 810 910 950		
L081	BLOCK_INDEX	DNX_MVA_I	INDEX USED TO DETERMINE THE NUMBER OF BLOCKS IN GIVEN MODE OF AN OPS CONTROL SEGMENTS	550	550		
L082	OPS_PAGE_X_COORDINATE	DCIBXC1	THE X COORDINATE OF POSITION ON THE CRT TO DISPLAY THE FIRST CHARACTER OF THE OPS NUMBER	630	630		
L083	OPS_PAGE_Y_COORDINATE	DCIBYC1	THE Y COORDINATE ON THE CRT TO DISPLAY OPS NUMBER IN HEADER	630	630		
L084	OPS_PAGE_NUMBER	CMATPAGE	THE OPS NUMBER STORED IN THE MAT	630	630		
L085	SPEC_PAGE_NUMBER	CMATPAGE+1	THE SPEC NUMBER, IF ANY, IN THE MAT	630	630		
L086	DISPLAY_PAGE_NUMBER	CMATPAGE+2	THE DISPLAY NUMBER IF ANY STORED IN THE MAT	630	630		
L087	PSEUDO_DOT_ENTRY	CLOCDDTP	AREA TO ASSEMBLE DOT ENTRIES TOGETHER FOR OUTPUT	630	630		
L088	RETURN_STATUS_FLAG	DMC_STATUS	A RETURN CODE FROM AN I/O ATTEMPT THAT IS PASSED TO MCOS_DISPLAY	520	520 600	I	

USER INTERFACE LOCAL DATA

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	HAL
			COORDINATEOR				
L089	MC_ZERO	IPL_MC	CONTAINS THE CONSTANT '0' FOR MC=0	405 500	405 600		
L090	MC_THREE	SM_&_MC	CONTAINS THE CONSTANT '3' FOR MC=3	405	405		
L091	DMI_LOCAL_DEU_NUMBER	DMMVL_DEUNO	CONTAINS THE DEU BEING PROCESSED BY A POLL FROM MCOS_INPUT_ PROCESSOR	400 405	405		
L092	TEMPORARY_I_LOOP	DMMVL_I	USED AS A POINTER TO KEYSTROKE BEING PROCESSED BY 405	405	405		
L093	TEMPORARY_J_LOOP	DMMVL_J	A COUNTER FROM 1 TO 32 TO SCAN ALL POSSIBLE VALID KEYSTROKE ENTRIES	405	405		
L094	DEV_DEVICE_NO	DMMV_BT_STAT. DDEV	CONTAINS THE DEVICE ID FOR CALLING THE FCOS SVC DIO	405	405		
L095	DEU_DESTINATION	DMS_MF	A PERAMETER INDICATING THAT THE TUTORIAL MESSAGE IS TO BE DISPLAYED ONLY ON CRTS SUPPORTING SPECIFIC MAJOR FUNCTION A DEU_ DESTINATION VALUE OF ONE=&SM,A TWO =GNC		620 650	I	
L096	DISPLAY_NUMBER	DMS_DISPLAY_NUMBER	PARAMETER INDICATING OPSOLSPEC NUMBER WITH WHICH TO DISPLAY THE TUTORIAL MESSAGE		650	I	
L097	MESSAGE_TEXT	DMS_MESSAGE_TEXT	CHARACTER STRING OF UP TO 46 CHARACTERS. THESE CHARACTERS ARE CONVERTED INTO DEU DISPLAYABLE FORMAT AND PLACED ON CRT AS A TUTORIAL MESSAGE		620 650	C(46)	
L098	MESSAGE_ID	DMS_ID	A NUMBER CORRESPONDING TO THE TUTORIAL MESSAGE BEING DISPLAYED. THIS PARAMETER IS USED FOR DCMNLISTING		650	I	
L099	CALL_INDICATOR	DMS_INDICATOR	A BIT ENABLED BY MESSAGE_LINE_ SUPPORT_FUNCTION AND LATER TESTED IF REQUEST FOR TUTORIAL MESSAGE IS VALID FCM BUILDER IS INVOKED VIA CALL BY MESSAGE_LINE_ SUPPORT_FUNCTION AS RESULT OF CALL_INDICATOR BEING ENABLED	650	650 655	BT	
L100	DEU_COUNTER	DMS_COUNTER	A INTERNAL PARAMETER USED TO LOOP FROM CNF TO THE MAXIMUM NUMBER OF	650	650	I	

USER INTERFACE LOCAL DATA

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	OST	ATTRIBUTES	MML
			DEUS WHICH IS THREE				
L101	NOOP_FCW	DCIBNOOP	A HALFWORD NOOP TO ACT AS BLANK FILLER FOR CERTAIN DISPLAY POSITIONS	630	630		
L102.00	NAME_TABLE	DIN_ICC_NAMES	THIS STRUCTURE IS USED TO DEVELOP ADDRESSES OF DATA ITEMS WITH DIFFERENT ATTRIBUTES TO BE INITIALIZED.THIS STRUCTURE CONTENT IS INITIALIZED IN DIN_ICC_SIPCOLL& MOVED TO CZ2V_LNAME TO ALLOW INDEXING BY STRUCTURE COPY.THE VALVES WITHIN PARENTHESIS ARE THE INITIALIZED ADDRESSES	485		ST(1)	
L102.01	NAME_01	DIN_XN01	NAME OF SCALAR DOUBLE(CZ2V_TZERO)			Y	
L102.02	NAME_02	DIN_XN02	NAME OF INTEGER (CZ2V_CAM-CNTRS &(1,1))			Y	
L102.03	NAME_03	DIN_XN03	NAME OF SCALAR(CZ2V_DUTY_CYCLE &(1))			Y	
L102.04	NAME_04	DIN_XN04	NAME OF BIT (16) (CZ2V_DP_DISP &(1,1))			Y	
L102.05	NAME_05	DIN_XN05	NAME OF SCALAR(CZ2V_DUTY_CYCLE &(1))			Y	
L102.06	NAME_06	DIN_XN06	NAME OF INTEGER (CZ2V_CURRENT_OPS&(1))			Y	
L102.07	NAME_07	DIN_XN07	NAME OF BIT(16)(CZ2B_ACT_XMITR &(1))			Y	
L102.08	NAME_08	DIN_XN08	NAME OF BIT(16)(CZ2B_CS&(1))			Y	
L102.09	NAME_09	DIN_XN09	NAME OF BIT(16)(CZ2B_STATUS&(1))			Y	
L102.10	NAME_10	DIN_XN10	NAME OF CHARACTER(2)(CZ2V_MF_OPS&(1))			Y	
L102.11	NAME_11	DIN_XN11	NAME OF BIT(16)(CZ2B_OPS_STATUS)			Y	
L102.12	NAME_12	DIN_XN12	NAME OF BIT(32)(CZ2B_STRNG_DEF$(1))			Y	
L102.13	NAME_13	DIN_XN13	NAME OF BIT(16)(CZ2B_BTU_BITE_FC&(1))			Y	
L102.14	NAME_14	DIN_XN14	NAME OF BIT(16)(CZ2B_NOM_STRNG&(1))			Y	

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
L102.15	NAME_15	DIN_XN15	NAME OF BIT(16)(CZ2B_FAIL_SYNC_SET))			Y	
L102.16	NAME_16	DIN_XN16	NAME OF BIT(16)(CZ2B_ICC_FRC_0&(1))			Y	
L102.17	NAME_17	DIN_XN17	NAME OF INTEGER(CZ2V_CNTRS$(1,1))			Y	
L102.18	NAME_18	DIN_XN18	NAME OF INTEGER(CZ2V_CAM_CNTRS$(2,1))			Y	
L102.19	NAME_19	DIN_XN19	NAME OF INTEGER(CZ2V_CAM_CNTRS$(3,1))			Y	
L102.20	NAME_20	DIN_XN20	NAME OF INTEGER(CZ2V_CAM_CNTRS$(4,1))			Y	
L102.21	NAME_21	DIN_XN21	NAME OF BIT(16)(CZ2B_IO_ERR_GPC$(1,1))			Y	
L102.23	NAME_23	DIN_XN23	NAME OF BIT(16)(CZ2B_IO_ERR_GPC$(2,1))			Y	
L102.24	NAME_24	DIN_XN24	NAME OF BIT(16)(CZ2B_IO_ERR_GPC$(3,1))			Y	
L102.25	NAME_25	DIN_XN25	NAME OF BIT (16)(CZ2B_IO_ERR_GPC$(4,1))			Y	
L102.26	NAME_26	DIN_XN26	NAME OF BIT(16)(CZ2B_IO_ERR_GPC$(5,1))			Y	
L102.27	NAME_27	DIN_XN27	NAME OF BIT(16)(CZ2B_ERR_GPC$(1,1))			Y	
L102.28	NAME_28	DIN_XN28	NAME OF BIT(16)(CZ2B_ERR_GPC$(2,1))			Y	
L102.29	NAME_29	DIN_XN29	NAME OF BIT(16)(CZ2B_ERR_GPC$(3,1))			Y	
L102.30	NAME_30	DIN_XN30	NAME OF BIT(16)(CZ2B_ERR_GPC$(4,1))			Y	
L102.31	NAME_31	DIN_XN31	NAME OF BIT(16)(CZ2B_ERR_GPC$(5,1))			Y	
L102.32	NAME_32	DIN_XN32	NAME OF INTEGER(CZ2V_LDB_GPV_REQ)			Y	
L102.33	NAME_33	DIN_XN33	NAME OF INTEGER(CZ2V_LDB_GPC_REQ)			Y	
L102.34	NAME_34	DIN_XN34	NAME OF INTEGER(CZ2V_MC_REQ)			Y	

USER INTERFACE LOCAL DATA

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
L102.35	NAME_35	DIN_XN35	NAME OF INTEGER(CZ2V_MC_FRZ)			Y	
L102.36	NAME_36	DIN_XN36	NAME OF INTEGER(CZ2V_DEU_MF$(1)			Y	
L102.37	NAME_37	DIN_XN37	NAME OF INTEGER(CZ2V_DEU_MF$(2)			Y	
L102.38	NAME_38	DIN_XN38	NAME OF INTEGER(CZ2V_DEU_MP$(3)			Y	
L102.39	NAME_39	DIN_XN39	NAME OF BIT(16)(CDMV_UI_DOWNLIST_FORMAT_IO$(1)			Y	
L102.40	NAME_40	DIN_XN40	NAME OF BIT(16)(CDMV_UI_DOWNLIST_FORMAT_IO$(2)			Y	
L102.41	NAME_41	DIN_XN41	NAME OF BIT(16)(CDMV_UI_DOWNLIST_FORMAT_IO$(3)			Y	
L102.42	NAME_42	DIN_XN42	NAME OF INTEGER DOUBLE(CDMV_UI_DOWNLIST_TUTORIAL$(1)			Y	
L102.43	NAME_43	DIN_XN43	NAME OF INTEGER DOUBLE(CDMV_UI_DOWNLIST_TUTORIAL$(2)			Y	
L102.44	NAME_44	DIN_XN44	NAME OF INTEGER DOUBLE(CDMV_UI_DOWNLIST_TUTORIAL$(3)			Y	
L102.45	NAME_45	DIN_XN45	NAME OF SCOLAR(CZ2V_DUTY_CYCLE$(2)			Y	
L102.46	NAME_46	DIN_XN46	NAME OF BIT(16)(CZ2B_DP_DISP$(2,1))			Y	
L102.47	NAME_47	DIN_XN47	NAME OF SCOLAR(CZ2V_DUTY_CYCLE$(2))			Y	
L102.48	NAME_48	DIN_XN48	NAME OF INTEGER(CZ2V_CURRENT_OPS$(2,1))			Y	
L102.49	NAME_49	DIN_XN49	NAME OF BIT(16)(CZ2B_ACT_XM1TR$(2))			Y	
L102.50	NAME_50	DIN_XN50	NAME OF BIT(16)(CZ2B_CS$(2))			Y	
L102.51	NAME_51	DIN_XN51	NAME OF BIT(16)(CZ2B_STATUS$(2))			Y	
L102.52	NAME_52	DIN_XN52	NAME OF CHARACTER(2)(CZ2V_MF_DPS$(2))			Y	
L102.53	NAME_53	DIN_XN53	NAME OF SCALOR(CZ2V_DUTY_CYCLE$(3))			Y	
L102.54	NAME_54	DIN_XN54	NAME OF BIT(16)(CZ2B_DP_DISP$(3,1))			Y	

USER INTERFACE LOCAL DATA

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
L102.55	NAME_55	DIN_XN55	NAME OF SCALOR(CZ2V_DUTY_CYCLE $(3))			Y	
L102.56	NAME_56	DIN_XN56	NAME OF INTEGER(CZ2V_CURRENT_OPS $(3,1))			Y	
L102.57	NAME_57	DIN_XN57	NAME OF BIT(16)(CZ2B_ACT_XMITR $(3))			Y	
L102.58	NAME_58	DIN_XN58	NAME OF BIT(16)(CZ2B_CS$(3))			Y	
L102.59	NAME_59	DIN_XN59	NAME OF BIT(16)(CZ2B_STATUS$(3))			Y	
L102.60	NAME_60	DIN_XN60	NAME OF CHARACTER(2)(CZ2V_MF_DPS $(3))			Y	
L102.61	NAME_61	DIN_XN61	NAME OF SCALAR(CZ2V_DUTY_CYCLE $(4)			Y	
L102.62	NAME_62	DIN_XN62	NAME OF BIT(16)(CZ2B_DP_DISP $(4,1))			Y	
L102.63	NAME_63	DIN_XN63	NAME OF SCALAR(CZ2V_DUTY_CYCLE $(4))			Y	
L102.64	NAME_64	DIN_XN64	NAME OF INTEGER(CZ2V_CURRENT_OPS $(4,17)			Y	
L102.65	NAME_65	DIN_XN65	NAME OF BIT(16)(CZ2B_ACT_XMITR $(4))			Y	
L102.66	NAME_66	DIN_XN66	NAME OF BIT(16)(CZ2B_CS$(4))			Y	
L102.67	NAME_67	DIN_XN_67	NAME OF BIT(16)(CZ2B_STATUS$(4))			Y	
L102.68	NAME_68	DIN_XN_68	NAME OF CHARACTER(2)(CZ2V_MF_DPS $(4))			Y	
L102.69	NAME_69	DIN_XN_69	NAME OF BIT(16)(NULL)			Y	
L102.70	NAME_70	DIN_XN_70	NAME OF BIT(16)(NULL)				
L103	ASSIGN VALUE	DIN_ASSIGN	CALLER ASSIGN VALUE.. DEFINED IN (DIN_ICC_SIPCOLL) BUT EXISTS IN CALLER PROGRAM	485 670 750	485 670 750	BS(16)	
L104	ASSIGN_CALLER		BITS 1-8 OF CALLER ASSIGN VALUE SPECIFIES USER..HEX'00'=AIESIP	670 750	485 670 750	BS(8)	
L105	ASSIGN_REQUEST_ STATUS		BITS 9-16 OF CALLER ASSIGN VALUE SPECIFIES STATUS OF REQUEST	485	485 670	BS(8)	

USER INTERFACE LOCAL DATA

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
			HEX'00'=MESSAGE PROCESSED (GOOD STATUS) HEX'01'=MESSAGE NOT PROCESSED (BUFFER FROZEN & CALLER NOT AIESIP HEX'02'=MESSAGE NOT PROCESSED (MESSAGE NOT FIT IN BUFFER) HEX'03'=MESSAGE NOT PROCESSED (BAD PARAMETER PASSED)		750		
L106	PASSED_ARRAY	DIN_ARRAY	CALLER PASSED DATA ARRAY...DEFINE IN(DIN_ICC_SIPCOLL) BUT EXIST IN CALLER PROGRAM	670 750	485 670 750	A(*),BIT(16)	
L107	INITIAL_VALUE	DIN_INITIAL	INITALIZATION OF DATA NAMES VALUE 0=INITIALIZATION REQUIRED 1=INITALIZATION NOT REQUIRED	485	485	I INIT(0)	
L108	POINTER	DIN_BUF_POINTER	VALUE POINTING TO NEXT LOCATION IN ICC BUFFER WHERE DATA IS PLACED	485	485	I	
L109	MESSAGE_HEADER	DINB_BITVAL	THE PASSED ARRAY MESSAGE HEADER WHICH IS WORD 1 OF PASSED_ARRAY	485	485	BS(16)	
L110	MESSAGE_SIZE	DIN_MSG_SIZE	NUMBER OF WORDS CONTAINED IN PASSED MESSAGE....INTEGER EQUIV OF BITS 1 TO 8 OF MESSAGE HEADER	485	485	I	
L111	MESSAGE_INDEX	DIN_MSG_INDEX	INDEX TO PROPER COPY OF ICC_MSG_TABLE(CZ2V_ICC_MSG_TAB) INTEGER EQUIV OF BITS 10 TO 16 OF DIN_MSG_HEADER	485	485	I	
L112	GPC_ICC_BUFFER_NAME	DIN_NICCBUF	NAME OF PROPER ICC MESSAGE BUFFER STRUCTURE COPY FOR THIS GPC	485	485	Y	
L113	ROUTING_CODE	DIN_RCODE	INTEGER EQUIV OF BITS 1 TO 4 OF ICC_MSG_TABLE(CZ2V_ICC_MSC_TAB) SPECIFIES HOW MSG DATA IS TO BE COLLECTED	485	485	I	
L114	ADDRESS_INDEX	DIN_ADRINX	INTEGER EQUIV OF BITS 1 TO 16 OF PASSED_ARRAY(2) WHICH IS INDEX TO PROPER ADR OF DATA TO BE MOVED FOR THIS MSG	485	485	I	
L115	ADDRESS_NAME	DIN_Q	ADR VALUE EXTRACTED FROM ICC_ADR_NAME TO MOVE DATA FOR INDIRECT DATA TYPE MESSAGE	485	485	Y	
L116	I	I	TEMPORARY VALUE USED IN DO LOOPS	485	485	I	
L117	GPC_ID_X COORDINATE	DCI8XC3	X COORDINATE FOR GPC_ID FOR DISPLAY IN HEADER	630	630		

USER INTERFACE LOCAL DATA

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
L118	CONTROL_SEGMENT_ REQUEST_BLOCK	DMCV_REQBLK	STRUCTURE USED TO DESCRIBE ALL ACTIVITY CONCERNING A MAJOR FUNCT COPY 1 REFERS TO MF=PL COPY 2 REFERS TO MF=GN&C COPY 3 REFERS TO MF=SM	500 560	500 560	ST(3)	
L118.10	WAIT_COUNT	DMCV_RQWTC	NUMBER OF CONTROL SEGMENTS ACTIVE	560	560	I	
L118.20	PROCESS_TYPE	DMCV_RQPROC_TY	TYPE ACTIVITY ASSOC WITH MAJ FUNCT 1=OPS TRANSACTION 2=MODE TO MODE TRANSACTION 3=SCHEDULE	560	500 505 510 520 560	I	
L118.30	ACTIVE_OPS_AMT_ INDEX	DMCV_RQAMT_IN	APPLICATION MODING TABLE(AMT) INDEX POINTING TO DATA FOR OPS	560	560	I	
L118.40	REQUESTED_MACT_OPS_ INDEX	DMCV_XXMACT_I	MISC APPLICATIONS MODING TABLE (MACT) INDEX TO BE USED FOR REQUESTED OPS	560	560	I	
L118.50	REQUESTED_OPS_ NUMBER	DMCV_RQOSNBR	OPS NUMBER REQUESTED FOR MAJOR FUNCTION	560	560	I	
L118.60	REQUESTED_MODE_ NUMBER	DMCV_ RQMODE	MODE NUMBER REQUESTED FOR MAJOR FUNCTION	560	560	I	
L118.70	ACTIVE_CONTROL_ SEGMENT_MACT_INDEX	DMCV_RQMACT_IN	MACT COPIES CONTROLLING CONTROL SEGMENTS FOR MAJOR FUNCTION	560	505 560	A(4)I	
L119	ACTION_TYPE	DMC_TYPE	PARAMETER SPECIFYING TYPE OF PROCESSING SEQUENCE_REQUEST_ PROCESSOR IS TO PERFORM (LOADED BY CALLER)	505	505 560	I	
L120	RECONFIGURATION_IN_ PROGRESS	DMC_REC_IN_ PROGRESS	VALUE SPECIFYING CONDITION OF RECONFIGURATION ACCOMPLISHED 0=NO 1,2,3=YES FOR MF=PL,GN&C,SM -100 FOR SECONDARY GPC IN REDUNDANT SET FORMATION	560	500 510 520 560	I INIT(0)	
L121	SEQUENCE_TEST1	DMC_TVAL1	VALUE USED FOR VARIOUS EXCLUSIVE PROCESSES (STATUS OF RECONFIG, TYPE OF KEYBOARDED REQUESTS)	560	560	I	
L122	SEQUENCE_MAJOR_	DMC_MFI	SEQUENCE_REQUEST_PROCESSOR VALUE	560	560	I	

USER INTERFACE LOCAL DATA

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
	FUNCTION		OF MAJOR FUNCTION				
L123	SEQUENCE_PROCESS_TYPE	DMC_PROCTYPE	SPECIFIC STATUS OF CURRENT EXECUTION OF SEQUENCE_REQUEST PROCESSOR 0=ILLEGAL REQUEST 1=OPS TRANSITION 2=MODE TO MODE TRANSITION 3=SCHEDULE 6=CANCEL OLD SPEC&START NEW SPEC 7=RECONFIGURATION REQUIRED	560	560	I	
L124	TERMINATOR_INDEX	DMC_TERMI	DETERMINES OPS/SPEC TERMINATION 1=OPS TERMINATED 2=SPEC TERMINATED ON DEU#1 3=SPEC TERMINATED ON DEU#2 4=SPEC TERMINATED ON DEC#3	560	560	I	
L125	TEST_INDEX	DMC_TINDX	VALUE USED FOR VARIOUS TEST CONDITIONS & LOOP CONTROLS	560	560	I	
L126	MCDS_MACT_INDEX	DMC_MACT_INDEX		505 510 520 560	505 510 520 560	I	
L127	DEU_NUMBER	DMC_DIT_INDEX		500 505 510 520 560	500 505 510 520 560	I	
L128	DEU_REQUEST_MACT_INDEX	DMCV_DEU_MACT	SEVEN VALUES CORRESPING TO THE 7 COPIES OF THE MACT SPECIFYING: WHEN=0 THAT AN OPS (LEVEL 1) DISPLAY IS BEING USED BY THE MACT COPY OR NOT USED AND WHEN NE 0 SPECIFIES A SPEC (LEVEL 2) DISP IS BEING USED BY THE MACT COPY 1=DEU #1 2=DEU #2 3=DEU #3	560	500 505 560	A(7)I,INIT(7#0)	

USER INTERFACE LOCAL DATA

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
L129	SPEC_ZERO_MACT	DMC_SZFMACT	SINCE 00 IS THE SAME CONTROL SEGMENT AS OPS 0 00 IT IS ALWAYS CONTROLLED BY MACT COPY #1... HOWEVER A "FALSE MACT COPY" IS GIVEN TO IT TO REMOVE A MACT FROM THE TABLE OF AVAILABLE MACT COPIES THAT CAN BE USED	560	560	I	
L130	SEQUENCE_AMT_INDEX	DMC_AMT_I		560	560	I	
L131	FILTER_MASK	DME_FILX	A 16 BIT MASK TO INDICATE WHICH IMT INDEXES HAVE BEEN PROCESSED ONCE ALREADY. ALL OTHER MESSAGES WITH THAT INDEX ARE IGNORED	490	490	BS(16), INIT=0380	
L132	FAULT_FILTER_VALUE	DME_NFFF1	A COUNT OF THE NUMBER OF MESSAGES WITH AN IMT INDEX OF ONE AND UNIQUE COMBINATIONS OF FMPT AND MAJOR_MINOR FIELDS. USED AS AN INDEX TO POINT TO ENTRIES IN ARRAYS FOR DME_FBN,DME_FIN AND DMEB_GPC	490	490	I	
L133	NON_FAULT_FILTER_VALUE	DME_NFFF2	A COUNT OF THE NUMBER OF MESSAGES WITH AN IMT INDEX OF TWO HAVING UNIQUE MESSAGE CLASSES. USED AS AN INDEX IN MESSAGE_CLASS ARRAY FOR MATCH COMPARISONS	490	490	I	
L134	NUMBER_OF_UNIQUES	DME_NBR_UNIQ	A COUNT OF THE NUMBER OF UNIQUE DATA WORDS FOR EACH IMT INDEX OF 20,21,22 OR 23	490	490	I	
L135	BUFFER_NUMBER	DME_I	CONTAINS VALUE OF ICC BUFFER BEING PROCESSED FROM 1 TO 4	490	490	I	
L136	ICC_MESSAGE_POINTER	DME_A	CONTAINS RELATIVE DISPLACEMENT IN WORDS OF MESSAGE CURRENTLY BEING PROCESSED	490	490	I	
L137	BUFFER_END	DME_B	SET TO ONE WHEN END OF BUFFER IS ENCOUNTERED OR THE NUMBER OF WORDS EXCEEDS 121	490	490	I	
L138	COMMON_SET_MEMBER	DME_C	POINTER WITHIN COMMON_SET_MASK	490	490	I	
L139	ICC_MESSAGE_HEADER	DME_MSGHEAD	FIRST WORD OF EACH ICC MESSAGE CONTAINS ICC MESSAGE TABLE INDEX (IMT) AND LENGTH OF ICC MESSAGE BITS 1-8 IMT INDEX BITS 10-16 LENGTH	490	490	BS(16)	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
L140	IMT_INDEX	DME_IMTI	CONTAINS BITS OF THE ICC_MESSAGE_HEADER AND IS THE IMT_INDEX	490	490	I	
L141	MESSAGE_SIZE	DME_MSGS	THE SIZE OF THE ICC MESSAGE CONTAINED IN BITS 10-16 OF ICC_MESSAGE_HEADER	490	490	I	
L142	ICC_MESSAGE_TABLE_VALUE	DME_MSGTABV	THE CONTENTS STORED IN AN ICC_MESSAGE_TABLE ENTRY	490	490	BS(16)	
L143	CONTENTION_FLAG	DME_MSGTABV	BIT 16 OF ICC_MESSAGE_TABLE VALUE	490	490		
L144	FILTERED_MESSAGE_MATCH	DME_MATCH	A FLAG WHEN SET TO ONE MEANS THAT CONDITIONS ARE SUCH THAT THE MESSAGE BEING PROCESSED IS IGNORED	490	490	I	
L145	CURRENT_FMPT_COPY_NUMBER	DME_CURFMPT	VALUE OF THE FMPT COPY NUMBER FOR COMPARING TWO MESSAGES FOR FILTERING	490	490	BS(16)	
L146	SAVED_BUFFER_NUMBER	DME_BF	THE ICC BUFFER NUMBER OF THE MESSAGE WITH IMT INDEX=1 SAVED FOR COMPARISON	490	490	I	
L147	SAVED_INDEX_NUMBER	DME_IN	THE POINTER TO SAVED MESSAGE	490	490	I	
L148	CURRENT_MAJOR_MINOR_INDEX	DME_CURMJMN	THE CURRENT MAJOR-MINOR INDEX FOR THE ICC MESSAGE BEING PROCESSED	490	490	BS(16)	
L149	COMBINED_GPC_MASK	DME8_GP	LOCATION WHERE THE GPC'S ARE OK'ED TOGETHER TO DETERMINE FINAL ROUTING OF THE ICC MESSAGE	490	490	BS(16)ARRAY 21	
L150	TEMPORARY_BUFFER_NUMBER	DME_FBN	THE BUFFER NUMBER OF THE MESSAGE WHICH HAS A UNIQUE COMBINATION OF FMPT AND MAJOR-MINOR FIELDS IS SAVED HERE	490	490	BS(16)ARRAY 21	
L151	TEMPORARY_INDEX	DME_FIN	THE INDEX INTO THE ICC BUFFER TO THE MESSAGE SAVED FOR ITS UNIQUE COMBINATION OF FMPT AND MAJOR-MINOR FIELDS	490	490	BS(16)ARRAY 21	
L152	CURRENT_MESSAGE_CLASS_INDICATOR	DME_CURINDMC	THE CURRENT MESSAGE WITH IMT INDEX =2 HAS ITS MESSAGE CLASS SAVED HERE FOR COMPARISON	490	490	BS(16)	
L153	SAVED_MESSAGE_CLASS_INDICATOR	DME_INDMC	THE MESSAGE CLASS(SAVED FOR COMPARISON) OF THE SAVED MESSAGE IS STORED HERE IN ARRAY FORMAT	490	490	BS(16)ARRAY 10	
L154	STRUCTURE_COPY_NUMBER	DME_YY1	THE MESSAGES WITH IMT INDEX BETWEEN 19 AND 23 HAVE THE INDEXES	490	490	I	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
			REDUCED TO A NUMBER 1 TO 4 FOR INDEXING AN ARRAY				
L155	SAVE_NUMBER_OF_UNIQUES	DME_YY2	THE NUMBER OF UNIQUE ONE WORD MESSAGES FOR EACH MESSAGE TYPE WITH IMT INDEX 20,21,22 OR 23 IS STORED HERE	490	490	I	
L156	ONE_WORD_MESSAGE_CONTENTS	DME_MSG_VAL	THE CONTENTS OF THE MESSAGE SAVED WHOSE IMT INDEX IS 20,21,22 OR 23 TO COMPARE WITH THE CURRENT MESSAGE FOR A MATCH OF THE SAME WORD IF THE CURRENT MESSAGE HAS THE APPROPRIATE IMT INDEX	490	490	BS(16)ARRAY	
L157	PROCESSING_VALUE	DME_PROCV	VALUES MEAN 1 CALL GPC_RECONFIGURATION_MESSAGE_HANDLER 2 CALL ICC_IDLE_OPS_PROCESSOR 3 CALL BUS_CONFIGURATION CHANGE	490	490	I	
L158	ROUTING_CODE	DME_RCODE	THE ROUTING CODE VALUES TO PROCESS ICC MESSAGES WHERE: 0-2 NOT USED 3=CALL A PROCEDURE AND POINT TO DATA 4-13-NOT USED 14 CALL A PROCEDURE AND POINT TO DATA VIA INDEX 15 - MOVE DATA ONLY	490	490	I	
L159	GET DATA_VIA_INDEX	DME_J	POINTER TO THE DATA TO BE MOVED FOR RCODE=15	490	490	I	
L160	DATA_MOVE_INDEX	DME_LOCI	INDEX TO THE DATA TO MOVE FOR A ROUTING_CODE=15	490	490	I	
L161	DATA_MOVE_LENGTH	DME_T	THE NUMBER OF DATA WORDS TO MOVE FOR RCODE=15	490	490	I	
L162	RECEIVING_ADDRESS	DME_Q	THE ADDRESS WHERE TO MOVE THE DATA FOR ROUTING_CODE=15	490	490	I	
L163	MOVE_LOOP	DME_X	THE COUNT=TO THE NUMBER OF WORDS TO MOVE FOR THE DO LOOP	490	490	I	
L164	PROCESSING_GPC	DME_GPCI	CONTAINS THE NUMBER OF THE GPC BEING PROCESSED	490	490		
L165	INITIALIZATION_REQUEST_CONTROL_BLOCK	DMCV_IRFQBLK	A STRUCTURE USED TO INITIALIZE ALL COPIES OF THE CONTROL_SEGMENT REQUEST_BLOCK TO THEIR REQUIRED INITIAL VALUES		500 560	ST(1)	

```
*****************************************************************************************************************
*  ID   *          ITEM          *          HAL/NAME          *          DESCRIPTION          * SPC * DST *  ATTRIBUTES  *  MML  *
*****************************************************************************************************************
*       *                        *                            * IT IS DEFINED AS BEING A STRUCTURE *     *     *              *       *
*       *                        *                            * OF TYPE CONTROL_SEGMENT_REQUEST    *     *     *              *       *
*       *                        *                            * BLOCK(LOO13) ;;*THEREFORE ALL HAL  *     *     *              *       *
*       *                        *                            * NAMES WITHIN THE STRUCTURE ARE THE *     *     *              *       *
*       *                        *                            * SAME AS THOSE OF CONTROL_SEGMENT_  *     *     *              *       *
*       *                        *                            * REQUEST_BLOCK.DMCV_IPEG BLK.DMCV_RQMPCT_*     *     *              *       *
*       *                        *                            * EXAMPLE-DMCV_IPEG BLK.DMCV_RQMPCT_ *     *     *              *       *
*       *                        *                            * IN                                 *     *     *              *       *
*       *                        *                            * THE STRUCTURE IS INITIALIZATED TO  *     *     *              *       *
*       *                        *                            * SPECIFY OPS_00 IN SCHEDULE         *     *     *              *       *
*       *                        *                            *                                    *     *     *              *       *
* L165.10 * INITIALIZATION_      * (DMCV_RQWTC)               *                                    *     * I INIT(1)          *       *
*       * WAIT_COUNT             *                            *                                    *     *     *              *       *
* L165.20 * INITIALIZATION_      * (DMCV_RQPROC_TY)           *                                    *     * I INIT(3)          *       *
*       * PROCESS_TYPE           *                            *                                    *     *     *              *       *
* L165.30 * INITIALIZATION_      * (DMCV_RQAMT_IN)            *                                    *     * I INIT(0)          *       *
*       * ACTIVE_OPS_AMT_        *                            *                                    *     *     *              *       *
*       * INDEX                  *                            *                                    *     *     *              *       *
* L165.40 * INITIALIZATION_      * (DMCV_XXMACT_I)            *                                    *     * I INIT(1)          *       *
*       * REQUESTED_MACT_OPS_    *                            *                                    *     *     *              *       *
*       * INDEX                  *                            *                                    *     *     *              *       *
* L165.50 * INITIALIZATION_OPS_  * (DMCV_RQOSNBR)             *                                    *     * I INIT (0)         *       *
*       * NUMBER                 *                            *                                    *     *     *              *       *
* L165.60 * INITIALIZATION_      * (DMCV_RQMODE)              *                                    *     * I INIT(1)          *       *
*       * MODE_NUMBER            *                            *                                    *     *     *              *       *
* L165.70 * INITIALIZATION_      * (DMCV_RQMACT_IN)           *                                    *     * A(4)!INIT(1,0,0    *       *
*       * MACT_INDEX_BLOCK       *                            *                                    *     *     *,0)           *       *
*       *                        *                            *                                    *     *     *              *       *
* L166  * NAME_REQUEST_BLOCK     * DMC_NRQBLK                 * NAME VARIABLE FOR THE PROPER COPY  * 560 * 560 * V            *       *
*       *                        *                            * OF THE STRUCTURE CONTROL_SEGMENT_  *     *     *              *       *
*       *                        *                            * REQUEST_BLOCK.IT IS USED TO REDUCE *     *     *              *       *
*       *                        *                            * CODE REQUIRED TO ADDRESS DATA      *     *     *              *       *
*       *                        *                            * WITHIN THAT STRUCTURE              *     *     *              *       *
*       *                        *                            *                                    *     *     *              *       *
* L167  * SEQUENCE_MACT_         * DMC_MACT_I                 * CALCULATED MACT COPY NUMBER USED   * 560 * 560 * I            *       *
*       * INDEX                  *                            * WITHIN SEQUENCE_REQUEST PROCESSOR  *     *     *              *       *
*       *                        *                            * TO INDICATE THE MACT COPY FOR      *     *     *              *       *
*       *                        *                            * PROCESSING AN OPS OR SPEC CONTROL  *     *     *              *       *
*       *                        *                            * SEGMENT                            *     *     *              *       *
*       *                        *                            *                                    *     *     *              *       *
* L168  * SAVE_SPEC_NUMBERS      * DMC_SPIDS                  * STRUCTURE BY MAJOR FUNCTION (COPY= * 560 * 560 * ST(3) INIT     *       *
*       *                        *                            * PLI;COPY2=GNCC,COPY3=SM) USED TO   *     *     * 9#(500)      *       *
*       *                        *                            * SAVE THOSE SPECIFIC NUMBERS OF SPECS*     *     *              *       *
*       *                        *                            * THAT ARE REQUESTED WHEN ANOTHER    *     *     *              *       *
*       *                        *                            * SPEC IS ACTIVE ON THAT OEU AND     *     *     *              *       *
*****************************************************************************************************************
```

U/I LOCAL DATA

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
			MLST BE CANCELLED PRIOR TO SCHEDULING THE NEW REQUESTED SPEC IT IS INITIALIZED TO REFLECT NO WAITING SPECS (IE 500)				
L168.10	NEXT_SPEC_NUMBER	DMC_SPECID	THE NEXT SPEC ID TO BE SCHEDULED AFTER CANCELLATION OF ACTIVE SPEC ON DEU DEU 1'S DATA IN ARRAY (1) DEU 2'S DATA IN ARRAY (2) DEU 3'S DATA IN ARRAY (3)	560	560	A(3)I	
L169	NEXT_SPEC_MACT_AMT_INDEXES	DMC_SPEC_T	STRUCTURE BY MAJOR FUNCTION (COPY1=PL,COPY 2=GNEC AND COPY3=SM) OF AMT INDEXES AND MACT COPY NUMBER THAT WILL BE USED TO SCHEDULE A SPEC AFTER CANCELLATION OF ACTIVE SPEC ON A DEU			ST(3) INIT (1#0)	
L169.10	NEXT_SPEC_MACT	DMC_MINXTS	THE MACT COPY THAT WILL CONTROL THE WAITING SPEC WHEN IT IS SCHEDULED	560	560	A(3)I	
L169.20	NEXT_SPEC_AMT	DMC_AINXTS	THE AMT INDEX TO USE TO GET ALL SPEC SCHEDULING DATA FOR THE NEW SPEC TO BE SCHEDULED WHEN LESS THAN 100 IT IS INDEX FOR SYSTEM SPECS (SEE L170 & L171) WHEN GREATER THAN 100 MORE THAN INDEX INTO OPS SPECS IN AMT (SEE A090 & A110)	560	560	A(3)I	
L170	SYSTEM_SPEC_VERIFICATION_DATA	DMCV_SYS_SPEC	STRUCTURE USED TO PROVIDE DATA NECESSARY TO VERIFY AND IDENTIFY A REQUESTED SPEC AS BEING A SYSTEM SPEC (IE A SPEC THAT IS LEGAL FOR OPS) IN ANY MAJOR FUNCTION			ST(2) INIT(2,1,PRIO_ASC), 4,1,PRIO_ASB	
L170.10	SYSTEM_SPEC_ID	DMCV_SYS_ID	THE ID (SPEC NUMBER) ASSOCIATED WITH THE SPEC DATA	560	560	I	
L170.20	SYSTEM_SPEC_BLOCK	DMCV_SYS_BLOCKS	THE NUMBER OF BLOCKS CONTAINED IN THIS SYSTEM SPEC		560	I	
L170.30	SYSTEM_SPEC_PRIORITY	DMCV_SYS_PRIOR	THE PRIORITY ASSOCIATED WITH THIS SYSTEM SPEC		560	I	
L171	SYSTEM_SPEC_NAMES_TABLE	DMCVL_SYS_SPEC NAME	STRUCTURE CONTAINING THE NAMES OF ALL SYSTEM SPECS		560	ST(2)INIT(NAME(ASC (TIME_MGMT),NAME(ASB_RD_WPT))	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	CST	ATTRIBUTES	MML
L171.10	SYSTEM_SPEC_ PROGRAM_NAME	DMCV_SYS_SPNAME	THE NAME VARIABLES FOR THE NAMES OF ALL SYSTEM SPECS		560	Y	
L172	NAME_MACT_COPY	DMC_NMACT	THE NAME OF THE PROPER MACT COPY TO ACCESS WHEN STOWING OR RETREIVING DATA FROM THE MACT.IT IS USED TO REDUCE CODE WHEN REFERENCING DATA IN THE MACT	560	560	Y	
L173	SEQUENCE_PROGRAM_ NAME	DMC_PROG_NAME	A PROGRAM NAME VARIABLE USED TO SCHEDULE AN OPS OR SPEC CONTROL SEGMENT	560	560	Y	
L174	SEQUENCE_PROGRAM_ PRIORITY	DMC_PROG_PRIOR	A VALUE CONTAINING THE PRIORITY ASSOCIATED WITH THE SCHEDULING OF AN OPS OR SPEC CONTROL SEGMENT	560	560	I	
L175	SEQUENCE_REQUEST_ OPS/SPEC	DMC_REQOSNBR	VALVE IN SEQUENCE_REQUEST PROCESSOR WHICH CONTAINS THE REQUESTED OPS NUMBER OR THE REQUESTED SPEC NUMBER	560	560	I	
L176	OPS_ZERO_PAGE	DMC_OZPAGE_VAL	VALVE USED TO INDICATE THE STATUS OF THE OPS0_00 PAGE DISPLAY 1=PAGE IS VALID 0=PAGE IS INVALID...(GN&C OPS2 OVERLAY OR OVERLAY ERROR)	560	560	I INIT(1)	
L177	AMT_TABLE_VALID	DMC_AMTVALID	VALVE USED TO REPRESENT THE STATUS OF THE APPLICATIONS MODING TABLE (AMT). 1=AMT IS VALID 0=AMT IS INVALID (OVERLAY ERROR	560	520 560	I INIT(1)	
L178	DISPLAY_BUFFER_ NAME_TABLE	DMC_BUFNAMES	STRUCTURE CONTAINING NAMES OF DISPLAY FORMAT BUFFERS(DFB)1,2,3,4 ,5 AND THE MASS MEMORY DIRECTORY	560	560	ST(6)	
L178.10	DISPLAY_BUFFER_ NAME	DMC_BUF	NAME OF DFB'S 1-5 ARE MASS MEMORY DIRECTORY	560	510 520 560 600	Y	
L179	SEQUENCE_REQUEST_ MODE	DMC_REQMODE	VALUE FOR SEQUENCE_REQUEST PROCESSOR WHICH CONTAIN THE REQUESTED MODE ASSOCIATED WITH AN OPS TRANSITION REQUEST OR MODE TO MODE REQUEST AND =-1 FOR A SPEC REQUEST	560	560	I	
L180	GPC_OPS_ZERO_ KEYSTROKE	DMC_GPCZ	THE NUMBER OF KEYSTROKES ASSOCIATED WITH AN OPS KEYBOARD	560	560	I	

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
			REQUEST (NOMINALLY IT IS 5 KEYSTROKES,HOWEVER WHEN IT IS 6 KEYSTROKES AND THE REQUEST IS FOR OPS000PRO IT INDICATES ALL MAJOR FUNCTION ARE REQUESTED TO GO TO OPS000)(IE.GPC OPSO-00)				
L181	SEQUENCE_VALUE	DMC_SEQVAL	VALUE CONTAINING SEQUENCE TERMINATING DATA TO BE INPUT INTO THE MACT CONTROLLING THE CONTROL SEGMENT SCHEDULED 0=VALUE FOR OPSO-00 $ SPEC00 WHEN AN OPS IS SCHEDULED THE AMT INDEX FOR THE OPS DATA IS PLACED IN THIS VALUE SO THAT THE NUMBER OF BLOCKS PER MODE CAN BE LOCATED. WHEN A SPEC IS SCHEDULED THE VALUE IS THE NUMBER OF BLOCKS ASSOCIATED WITH THIS SPEC	560	560	I	
L182	SEQUENCE_WAIT_ COUNT	DMC_WTCT	VALUE CONTAINING THE NUMBER OF ACTIVE CONTROL SEGMENTS	560	560	I	
L183	SEQUENCE_CURRENT_ OPS	DMC_CUR_OP	VALUE CONTAINING THE CURRENT OPS NUMBER FOR THE REQUESTED MAJOR FUNCTION	560	560	I	
L184	OLD_OPS	DMC_SAVE	VALUES BY MAJOR FUNCTION OF OPS BEING CANCELLED	560	560	A(3)I,INIT(0,0,0)	
L185	GRT_INDEX	DMC_GRTIX	VALUE CONTAINING THE INDEX TO THE GRT WHEN AN OPS IS REQUESTED THAT WILL ALLOW OVERLAY 0=OVERLAY NOT ALLOWED	560	560	I	
L186	SEQUENCE_RUN_BITS	DMCB_RUNBITS	VALUE USED TO DETERMINE THE RUN STATUS OF THE GPC'S THAT ARE TO BE PART OF A REDUNDANT BIT 1 = GPC1 BIT 2 = GPC2 BIT 3 = GPC3 BIT 4 = GPC4 BIT 5 = GPC5 BIT 6-16 = 0 IF 0 NOT IN RUN	560	560	BS(16)	
L188	SEQUENCE_SECONDARY_ ICC_MESSAGE	DMC_REC_IMSG	THE ICC MESSAGE SENT TO INDICATE THAT A REQUEST FOR A SECONDARY GPC THAT WILL BE PART OF A REDUNDANT SET HAS COMPLETED ALL PROCESSING REQUIRED BY SEQUENCE REQUEST_ PROCESSOR WORD(1) HEX "1303"_ SPECIFIES IMT INDEX OF 19 (HEX 13)	560	485 560	A(3),BS(16) INIT (HEX "1303",HEX "0000","0000" HEX "0000")	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
L189	SEQUENCE_ASSIGN_VALUE	DMC_ASSIGN	AND 3 WORDS OF DATA. HEX "00" WORD(2) WILL CONTAIN THE VALUE OF RECONFIGURATION STATUS (1540) AND WORD 3 WILL CONTAIN THE VALUE GPC_ID (L190) THE ASSIGN VALUE REQUIRED FOR A CALL TO THE ICC_MESSAGE_COLLECTS (480) BITS 1-8 CONTAIN CALLER TYPE 'FF' CALLER IS SEQUENCE_RFQUEST_ PROCESSOR BITS 9-16 REQUEST STATUS	485 560	485 560	BS(16) INIT('HEX'FF00')	
L189.10	SEQUENCE_ASSIGN_STATUS		BITS 9 TO 16 OF SEQUENCE_ASSIGN_VALUE '00'=REQUEST ACCEPTED AND PROCESSED NE 0 =REQUEST NOT PROCESSED	485	560	BS(8)	
L190	ID_GPC	DMC_GPCID	VALUE REPRESENTING THE GPC NUMBER OF THE COMPUTER EXECUTING THIS LOAD; IT IS LOADED BY MCDS FUNCTIONS PROCESSOR (505) TO REDUCE CODE REQUIRED TO ACCESS THIS VALUE	505	505 520 560	I	
L191	FOUND_VALUE	DMC_FOUND	VALUE USED TO DETERMINE MODE VALIDITY OF AN OPS REQUEST 0=REQUESTED MODE VALID NE 0 REQUESTED MODE NOT VALID	560	560	I	
L192	MCDS_MAJOR_FUNCTION	DMC_MF	THE MAJOR FUNCTION VALUE USED BY ALL PROCEDURES WITHIN USER INTERFACE_CONTROL_SUPERVISOR (500)	500 505 560	500 505 510 520 560 600	I	
L193	RECONFIGURATION_IN_INITIALIZATION	DMC_REC_IP2	VALUE SPECIFYING RECONFIGURATION BEING STARTED 0=NOT STARTED NE 0 RECONFIGURATION BEGUN	500 505 560	505 560	I	
L194	DISPLAY_REQUEST_LEVEL	DMCVL_D_LVLN	LEVEL NUMBER PASSED BY ALL CALLERS OF MCDS DISPLAY COORDINATION (520) SPECIFYING LEVEL OF DISPLAY REQUESTED 1=OPS1 LEVEL DISPLAY REQUESTED 2=SPEC LEVEL DISPLAY REQUESTED 3=DISP LEVEL DISPLAY REQUESTED	505 510 520 560	505 510 520 560	I	
L195	DISPLAY_REQUEST	DMCVL_D_DISP	THE DISPLAY NUMBER REQUESTED BY	505	505	I	

U/I LOCAL DATA

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
	NUMBER		ALL CALLERS OF MCOS_DISPLAY_ COORDINATION (520)	510 520 560	510 520 560		
L196	INITIALIZATION_ FLAG	DMC_INITIAL	VALUE SPECIFYING INITIAL CONDITIONS 0=INITIAL CONDITIONS NE 0 IS NON INITIAL CONDITION	500 560	500 560	!	
L197	ASSIGN_MACT_COPY	DMCV_MACT_MF	THE MACT COPIES TO BE ASSIGNED TO CPS REQUESTS BY MAJOR FUNCTION (1)=PL=0=NOT SUPPORTED (2)=GNC=2=SUPPORTED IN COPY 2 (3)=SM=5=SUPPORTED IN COPY 5	560	560	A(3)! INIT(0,2,5)	
L198	SEQUENCE_UPDATE_ INDEX	DMC_UNDX	VALUE USED TO CALCULATE OPS MODE NUMBER OR SPEC NUMBER FROM A KEYBOARD REQUEST	560	560	!	
L199	SEQUENCE_TESTER	DMC_TVAL	VALUE USED FOR VARIOUS LOOP CONTROLS AND INTERMEDIATE CALCULATIONS	560	560	!	
L200	SEQUENCE_TEST2	DMC_TVAL2	END LOOP VALUE FOR SEARCHING SPEC'S ASSOCIATED WITH A GIVEN OPS	560	560	!	
L201	SEQUENCE_TEST3	DMC_TVAL3	VALUE USED FOR VARIOUS LOOP CONTROLS AND CALCULATIONS	560	560	!	
L202	SEQUENCE_TEST4	DMC_TVAL4	VALUE USED FOR CALCULATIONS AND LOOP CONTROLS PASSED TO PROCEDURES INTERNAL TO SEQUENCE_REQUEST_ PROCESSOR (560)	560	560	!	
L203	I	I	TEMPORARY LOOP CONTROL VARIABLE	560	560	!	
L204	J	I	TEMPORARY LOOP CONTROL VARIABLE	560	560	!	
L217	ITEM_REQUEST_BLOCK	DMCVL_ITEM_RQBLK	STRUCTURE DESCRIBING ALL ITEM ACTIVITY ON A DEU	510	510	ST(3)	
L217.10	SAVED_ITEM_NUMBER	DMCVL_RITEMNO	ITEM NUMBER OF ITEM CURRENTLY BEING PROCESSED			!	
L217.20	SAVED_ITEM_POINTER	DMCVL_RDPTRV	LOCATION OF THE START OF ITEM DATA WITHIN THE KEYBOARD_MESSAGE (SEE J060.04)				
L217.30	LIMIT_PENDING	DMCVL_RTYPX					
L217.40	SAVED_ITEM_TOTAL_ KEYSTROKES	DMCVL_RNINK	THE NUMBER OF KEYSTROKES ASSOCIATED WITH THE ITEM REQUEST			!	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
L218	REQUEST_STATUS	DMCVL_KEYID	VALUE USED TO DESCRIBE THE STATUS OF ITEM REQUESTS 20=ITEM WITH DATA...LEGAL STATUS 33=ITEM_EXECUTE...LEGAL STATUS 100=ITEM_COMPLETE...LEGAL STATUS 101=LIMIT ERROR...ERROR CONDITION 102=FORMAT ERROR...ERROR CONDITION 103=KEYBOARD ERROR...ERROR CONDITION	510	510	I	
L219	MCDS_FUNCTION	DMC_CASE	VALUE USED TO SPECIFY THE TYPE OF PROCESSING REQUIRED BY MCDS FUNCTION PROCESSOR WHEN CALLING AN INTERNAL PROCEDURE	500 505	500 505 510		
L220	ITEM_VERIFICATION_BUFFER_DATA	DMCV_BUF_DATA	STRUCTURE OF ALL DATA BY DEU USED TO VERIFY ITEM INPUTS	520	510 520	ST(3)	
L220.10	ITEM_START_OFFSET	DMCV_ITEM_START	THE INDEX TO ITEM 1	520	510		
L220.20	SCALAR_LIMITS_OFFSET	DMCV_SL_OST	INDEX TO THE START OF ALL SCALAR LIMITS THIS DISPLAY	520	510		
L220.30	GENERAL_LIMITS_OFFSET	DMCV_GL_OST	INDEX TO THE START OF ALL GENERAL LIMITS THIS DISPLAY	520	510		
L220.40	KEYBOARD_VERIFICATION_TABLE_DATA	DMCB_KVT_DATA	TOTAL ITEMS FOLLOWED FOR THIS DISPLAY + PRO, EXEC ALLOWED FLAGS	520	505 510 520	BS(16)	
L220.40	KEYBOARD_VERIFICATION_TOTAL_ITEMS		TOTAL ITEMS FOR THIS DISPLAY		510	BS(8)	
L220.42	PRO_ALLOWED_FLAG		PRO ALLOWED ON THIS DISPLAY IF =1		505	BS(1)	
L220.43	EXEC_ALLOWED_FLAG		EXECUTE ALLOWED ON THIS DISPLAY IF =1		505	BS(1)	
L221	ITEM_FORMAT_WORD1	DMCBL_FORMW1	1ST WORD OF ITEM FORMAT VERIFICATION DESCRIBING OPTION ALLOWED, ITEM TYPE, UPDATING REQUIRED AND LIMITS PRESENT EXTRACTED FROM THE DISPLAY_FORMAT_TABLE (DFT)			BS(16)	
L221.10	FORMAT_EXECUTE_ALLOWED		ITEM WITH EXECUTE OPTION ALLOWED BIT 2 OF ITEM_FORMAT_WORD1 0=ITEM_EXECUTE OPTION NOT ALLOWED 1=ITEM_EXECUTE OPTION ALLOWED			BS(1)	
L221.20	FORMAT_DATA_TYPE		ITEM WITH DATA ALLOWED AND TYPE OF			BS(3)	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
			DATA BITS 3 TO 5 OF ITEM_FORMAT_WORD1 000=ITEM WITH DATA OPTION NOT ALLOWED 001=SCALAR DATA TYPE ALLOWED 010=INTEGER DATA TYPE ALLOWED 100=OCTAL DATA TYPE ALLOWED 101=OCTAL/SCALAR DATA TYPE ALLOWED				
L221.30	FORMAT_LIMITS		LIMIT ON ITEM AND ITS RELATIVE LOCATION BITS 6 TO 12 OF ITEM_FORMAT_WORD1 0=NO LIMITS FOR THIS ITEM NE 0=LIMITS ARE PRESENT FOR THIS ITEM AND THIS NON ZERO VALUE SPECIFIES THEIR RELATIVE OFFSET FROM SCALAR LIMIT OFFSET (SEE) IN DISPLAY_FORMAT_TABLE OR GENERAL_LIMIT_OFFSET IN DISPLAY_FORMAT_TABLE DEPENDING ON FORMAT_DATA_TYPE VALUE			BS(7)	
L221.40	FORMAT_UPDATE_DISPLAY_EXEC		ONE TIME UPDATE DISPLAY FOR ITEM EXECUTE BIT 13 OF ITEM_FORMAT_WORD1 0=NO			BS(1)	
L221.50	FORMAT_UPDATE_DISPLAY_DATA		ONE TIME UPDATE DISPLAY FOR ITEM DATA BIT 14 OF ITEM_FORMAT_WORD1 0=NO			BS(1)	
L222	ITEM_FORMAT_WORD2	DMCVL_FORMW2	CONTAINS THE NUMBER OF INPUT CHARACTERS WHEN THE FORMAT_DATA_TYPE (SEE) =2 (INTEGER) OR 4 (OCTAL) WHEN THE FORMAT_DATA_TYPE=1 (SCALAR) OR 5 (OCTAL/SCALAR) CONTAINS 100 X THE NUMBER OF CHARACTERS ALLOWED BEFORE THE DECIMAL POINT + THE NUMBER OF CHARACTERS ALLOWED AFTER THE DECIMAL POINT	510	510	I	
L223	NEXT_INPUT	DMCV_NXT_INP	AN ARRAY BY DEU WHICH IS USED FOR CORRESPONDENCE BETWEEN MCDS_FUNCTIONS_PROCESSOR AND ANY_USERS OF DEU INPUT. FUNCTIONS UPDATE THIS VALUE AFTER EACH KEYBOARD INPUT...MCDS_ITEM_PROCESSOR SET IT TO 1 ON A LIMIT ERROR AND EXPECTS ITS VALUE TO BE 2 TO PERFORM A	505 510	505 510	A(3),I	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
			LEGAL ITEM LIMIT OVERRIDE				
L224	ITEM_PROCESSOR_ ITEM_NUMBER	DMCVL_ITEMNO	ITEM NUMBER ASSOCIATED WITH AN ITEM REQUEST	510 690	510	I	
L225	ITEM_PROCESSOR_ TOTAL_KEYSTROKES	DMCVL_MAXKY	TOTAL KEYSTROKES ASSOCIATED WITH AN ITEM REQUEST	510 690	510	I	
L226	LAST_KEYSTROKE_ VALUE	DMCVL_L_KEY	THE LAST KEYSTROKE INPUT WITH AN ITEM REQUEST EITHER ITEM (20) OR EXECUTE (30)	510	510 690	I	
L227	TEST_VALUE	DMCVL_TV	VALUE USED TO SELECT INDIVIDUAL KEYSTROKE INPUTS FROM AN ITEM REQUEST KEYBOARD MESSAGE	510	510	I	
L228	ITEM_PROCESSOR_ POINTER	DMCVL_POINTI	VALUE USED ON AN INDEX TO CURRENT KEYSTROKE TO BE PROCESSED FOR A SPECIFIC ITEMS REQUEST	510	510	I	
L229	INPUT_TEST_FLAG	DMCVL_I	VALUE SPECIFYING THE STATUS OF AN ITEM REQUEST WHEN THE LAST KEYSTROKE OF THE REQUEST HAS BEEN PROCESSED	510	510	I	
L230	LAST_KEYBOARD_ MESSAGE_VALUE	DMC_DIT_LAST_KYBD_ MSG	STRUCTURE USED TO STORE ANY KEYBOARD MESSAGE REQUIRING MATCHING	510	510	ST(3)	
L230.10	OLD_KEYBOARD	DMCV_DIT_KYBD_MSG	THE AREA USED TO SAVE AN EXACT COPY OF ALL KEYSTROKES OF A KEYBOARD MESSAGE THAT MUST BE SAVED IN ORDER TO MATCH AGAINST A NEW KEYBOARD MESSAGE TO DETERMINE IF ITS AN EXACT COPY	510	510	A(40)	
L231	ITEM_BUFFER_NUMBER	DMCVL_BUFFX	VALUE USED AS AN INDEX TO SPECIFY WHICH BUFFER ALL ITEM VERIFICATION DATA WILL BE USED TO PROCESS AN ITEM REQUEST (1-5)	510	510	I	
L232	ITEM_BUFFER_OFFSET	DMCVL_OFFS	VALUE USED AS AN INDEX TO SPECIFY THE OFFSET WITHIN A SPECIFIC BUFFER THAT WILL BE USED TO PROCESS ALL ITEM VERIFICATION DATA	510	510	I	
L233	UPDATE_DISPLAY_ VALUE	DMCVL_IPROC	VALUE USED TO SPECIFY THE STATUS OF PERFORMING A "ONE-TIME" UPDATE OF THE DISPLAY BASED ON THE ITEM REQUIRING THIS VIA THE DISPLAY_ FORMAT TABLE (DFT) ITEM	510	510	I	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
			VERIFICATION				
L234	ITEM_DATA_TYPE	DMCVL_ITEM_TYPE	VALUE TAKEN FROM AN INDIVIDUAL ITEMS VERIFICATION ENTRY IN THE OFT TO SPECIFY AN ITEMS DATA TYPE 0=ITEM DATA NOT ALLOWED 1=SCALAR DATA THIS ITEM 2=INTEGER DATA THIS ITEM 4=OCTAL DATA THIS ITEM 5=OCTAL/SCALAR DATA THIS ITEM	510	510	I	
L235	ITEM_LIMITS	DMCVL_LIMITS	VALUE USED TO REFLECT THE STATUS OF LIMITS PRESENT FOR ITEM BEING PROCESSED 0=LIMITS NOT PRESENT NE 0=LIMITS PRESENT AND THIS NON ZERO VALUE REPRESENTS THEIR RELATIVE POSITION WITH THE LIMITS TABLE OF THE DISPLAY_FORMAT_TABLE (DFT) SCALAR OR GENERAL (I.E., A VALUE OF 2 FOR AN ITEM OF SCALAR SPECIFIES IT IS THE 2ND ENTRY OF SCALAR LIMITS)	510	510	I	
L236	GENERAL_LIMIT_VALUES	DMC_GENLIM	ARRAY CONTAINING LOWER & UPPER INTEGER LIMIT VALUES	510	510	A(2),DI	
L236.10	LOWER_INTEGER_LIMITS		LOWER LIMIT VALUE ASSOCIATED WITH AN ITEM SPECIFYING LIMITS	510	510	DI	
L236.20	UPPER_INTEGER_LIMIT		UPPER LIMIT VALUE ASSOCIATED WITH AN ITEM SPECIFYING LIMITS	510	510	DI	
L237	SCALAR_LIMIT_VALUES	DMCV_SCALIM	ARRAY CONTAINING LOWER & UPPER SCALAR LIMITS	510	510	A(2),DI	
L237.10	LOWER_SCALAR_LIMIT		LOWER LIMIT VALUE ASSOCIATED WITH AN ITEM DEFINED AS SCALAR & CONTAINING LIMITS	510	510	DI	
L237.20	UPPER_SCALAR_LIMIT		UPPER LIMIT VALUE ASSOCIATED WITH AN ITEM DEFINED AS SCALAR & CONTAINING LIMITS	510	510	DI	
L238	INPUT_CHARACTER_ALLOWED	DMCVL_FNIC	VALUE CONTAINING THE NUMBER OF A CHARACTER OF INPUT ALLOWED FOR AN ITEM OF TYPE INTEGER OR OCTAL OR THE NUMBER OF OCTAL CHARACTER ALLOWED WHEN THE ITEM IS OCTAL/SCALAR	510	510	I	
L239	INPUT_CHARACTERS_	DMCVL_NIC	VALUE CONTAINING THE NUMBER OF	510	510	I	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
	FOUND		INPUT CHARACTERS EXTRACTED FROM THE ITEM REQUEST.THIS VALUE IS UPDATED AS EACH KEYSTROKE FOR AN ITEM IS PROCESSED...FOR ITEM OF INTEGER,OCTALS OR OCTAL/SCALAR ONLY				
L240	CHARACTER_BEFORE_DECIMAL_ALLOWED	DMCVL_FNBDP	VALUE CONTAINING THE MAXIMUM NUMBER OF INPUT CHARACTERS ALLOWED IN A SCALOR ITEM BEFORE THE DECIMAL POINT IS FOUND	510	510	I	
L241	CHARACTER_AFTER_DECIMAL_ALLOWED	DMCVL_FNADP	VALUE CONTAINING THE MAXIMUM NUMBER OF INPUT CHARACTER ALLOWED IN A SCALAR ITEM AFTER THE DECIMAL POINT IS FOUND	510	510	I	
L242	DECIMAL_POINT_FOUND	DMCVL_DPF	VALUE SPECIFYING THE STATUS OF FINDING THE DECIMAL POINT FOR SCALAR ITEMS(0=NOT FOUND , NON-0 =FOUND DECIMAL POINT	510	510	I	
L243	CHARACTERS_FOUND_BEFORE_DECIMAL	DMCVL_NCBDP	VALUE CONTAINING THE TOTAL NUMBER OF INPUT CHARACTER READ FROM THE ITEM REQUEST KEYBOARD MESSAGE BEFORE FINDING A DECIMAL POINT	510	510	I	
L244	CHARACTERS_FOUND_AFTER_DECIMAL	DMCVL_NCADP	VALUE CONTAINING THE TOTAL NUMBER OF INPUT CHARACTERS READ FROM THE ITEM REQUEST KEYBOARD MESSAGE AFTER FINDING A DECIMAL POINT	510	510	I	
L245	OCTAL/SCALAR_OCTAL_ERROR	DMCVL_OSI	STATUS OF THE OCTAL PART OF AN OCTAL/SCALAR ITEM:0 STATUS IS OK, NON-0 STATUS IS BAD TOO MANY CHARACTERS OR NON OCTAL VALUE-SCALAR PROCESSING CONTINUES	510	510	I	
L246	OCTAL/SCALAR_SCALE_ERROR	DMCVL_OSIX	STATUS OF SCALE POINT OF OCTAL/SCALAR ITEM =0 STATUS OK NE 0-STATUS BAD-TOO MANY CHAR. BEFORE DECIMAL OR AFTER DECIMAL	510	510	I	
L247	DECIMAL_VALUE	DMCVL_S1	VALUE CONTAINING THE SCALAR EQUIVALENT OF ALL ITEM DATA KEYSTROKES PROCESSED BEFORE A DECIMAL POINT WAS FOUND AND WILL CONTAIN THE SCALAR EQUIVALENT OF ALL ITEM DATA KEYSTROKES AFTER PROCESSING ENTIRE ITEM DATA INPUT	510	510	DSC	

U/I LOCAL DATA

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
L248	AFTER_DECIMAL_VALUE	DMCVL_S2	VALUE CONTAINING THE SCALAR EQUIVALENT OF ALL ITEM DATA KEYSTROKE PROCESSED AFTER A DECIMAL POINT WAS FOUND	510	510	DSC	
L249	OCTAL_DATA_WORD	DMCVL_QWORD	VALUE CONTAINING THE PROCESSED OCTAL KEYSTROKES RIGHT JUSTIFIED (LEADING ZERO OCTAL CHARACTER)	510	510	BS(32)	
L250	INTEGER_INPUT_VALUE	DMCVL_D	VALUE CONTAINING THE INTEGER EQUIVALENT OF ALL ITEM KEYSTROKES PROCESSED FOR AN ITEM OF INTEGER TYPE	510	510	DI	
L251	FIRST_OCTAL_CHARACTER	DMCVL_FOICH	VALUE CONTAINING THE DECIMAL EQUIVALENT OF THE FIRST OCTAL CHARACTER PROCESSED(USED TO TEST AGAINST AN 11 CHARACTER OCTAL VALUE SO THAT THE FIRST CHARACTER MUST NOT EXCEED THE VALUE OF 3 SINCE IT WOULD NOT FIT INTO A 32 BIT OCTAL FIELD)	510	510	I	
L252	DELIMITOR_VALUE	DMCVL_DELIM	VALUE USED TO DETERMINE WHEN A DELIMITOR HAS BEEN FOUND IN AN ITEM KEYBOARD REQUEST(+,-,OR AT LEAST KEYSTROKE)OR SET TO(=1 UPON A FORMAT VIOLATION)	510	510	I	
L253	TEST_INDEX	DMCVL_TX	VALUE USED FOR VARIOUS LOOP CONTROLS	510	510	I	
L254	ICC/IDLE_OPS_PROCESSOR_LOCAL_DATA						
L254.1	INTERNAL_ICC_DATA_WORD	DID_ICC_DATA_WORD	STORAGE FOR ICC MESSAGE	495	495	I	
L254.2	INTERNAL_ICC_BUFFER_NUMBER	DID_ICC_BUFNO	ICC BUFFER NUMBER	495	495	I	
L254.3	INTERNAL_ICC_INDEX_1	DID_ICC_INDEX	MESSAGE WORD INDEX		495	I	
L254.4	INTERNAL_ICC_INDEX_2	DID_ICC_INDEX_2	MESSAGE WORD INDEX TWO	495	495	I	
L255	TOTAL_TIME	CLOCTIM1	AREA USED TO SAVE TIME VALUE IN WHEN DURING A TIME CONVERSION IN DATA_CONVERSION	640	640	2I	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
L256	CHAR_BUFFER	CLOCBUFF	AREA DATA IS PLACED IN FOR FCW_ BUILDER ROUTINE'S INPUT	640	640	7I	
L257	CYCLIC_FRACTION_ DATA	CLOCDEC	SAVE AREA FOR CONVERTED FRACTIONAL DATA	640	640	8I	
L258	CYCLIC_INTEGER_ DATA	CLOCARG	TEMP SAVE AREA FOR FIXED POINT INTEGER DATA	640	640	2I	
L259	WORK_BUFFER	CLOCCHAR	WORK AREA FOR CONVERTING INPUT TO ASCII REPRESENTATION	640	640	8I	
L260	TIME_WORK_BUFFER	CLOCTIM2	WORK BUFFER USED BY TIME_CONVERSION	640	640	2I	
L261	WORD_COUNT_ TRANSFERRED	CLOCCNT	NUMBER OF WORDS TO BE TRANSFERRED TO THE CRT VIA THE SVC DIO (24)	625	625	I	
L262	DEU_FILL_ADDRESS		ADDRESS OF AREA ON CRT DATA IS TO BE PLACED VIA THE SVC DIO (24)	625	625	I	
L263	DATA_ADDRESS	CLOCADDR	ADDRESS OF DATA TO BE OUTPUT TO CRT VIA THE SVC DIO (24)	625	625	I	
L264	DEU_DEVICE_ID	CLOCDEU	DEVICE ID FOR CRT THAT DATA IS TO BE SENT TO VIA THE SVC DIO (24)	625	625	I	
L265	LAM_ALARMS2	DLA_LAM_ALARMS2	LOCAL DATA AREA USED TO STORE THE MAPPING OF THE LIGHTS & TONES TO BE TURNED OFF/ON. LAM_ALARMS2 IS MAPPED LIKE LAM_ALARMS (I.E, THERE IS A BIT BY BIT CORRESPONDENCE BETWEEN LAM_ALARMS & LAM_ALARMS2)	690	690	BS(16)	
L266	CLASS2_MAP_TO_PF1	DLA_RPF1	LOCAL DATA AREA USED TO STORE THE MAPPING OF THE CLASS 2 ALARMS AS IT RESIDES IN THE FCOS COMMAND WORD TO PF1. THIS MAPPING IS USED IN THE NEXT L/A CYCLE TO TURN ON THOSE CLASS 2 ALARMS THAT HAD PREVIOUSLY BEEN TURNED OFF	690	690	BS(16)	
L267	CLASS2_MAP_TO_PF2	DLA_RPF2	LOCAL DATA AREA USED TO STORE THE MAPPING OF THE CLASS 2 ALARMS AS IT RESIDES IN THE FCOS COMMAND WORD TO PF2. THIS MAPPING IS USED IN THE NEXT L/A CYCLE TO TURN ON THOSE CLASS 2 ALARMS THAT HAD PREVIOUSLY BEEN TURNED OFF	690	690	BS(16)	
L268	FCOS_CWS_TO_PF1	DLA_PF1	COMMAND WORDS FORMATTED TO DRIVE BITS & ALARMS ON/OFF VIA PF1	690		ST(1)	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
L268.10	SET_PF1	DLA_FCOS_SPF1	FIRST SIGNIFICANT WORD IN THE FCOS CW TO PF1. BITS ARE SET IN THIS WORD TO TURN ON THE CLASS3 ALERT TONE AND ALL CLASS2 BITES WITH THE EXCEPTION OF THE BACKUP C&W LITE. A BIT IS SET IN THIS WORD TO TURN OFF THE B/U C&W LITE	690		BS(16)	
L268.20	RESET_PF1	DLA_FCOS_RPF1	SECOND SIGNIFICANT WORD IN THE FCOS CW TO PF1. BITS ARE SET IN THIS WORD TO TURN OFF THE CLASS 3 ALERT TONE AND ALL CLASS 2 LITES WITH THE EXCEPTION OF THE B/U C&W LITE. A BIT IS SET IN THIS WORD TO TURN ON THE B/U C&W LITE	690		BS(16)	
L268.30	ALERT_SET_PF1	DLA_ALERT_SPF1	THIRD SIGNIFICANT WORD IN THE FCOS CW TO PF1. BITS ARE SET IN THIS WORD TO TURN ON THE CLASS3 ALERT LITE	690		BS(16)	
L268.40	ALERT_RESET_PF1	DLA_ALERT_RPF1	FOURTH SIGNIFICANT WORD IN THE FCOS CW TO PF1. BITS ARE SET IN THE WORD TO TURN OFF THE CLASS 3 ALERT LITE	690		BS(16)	
L269	FCOS_CWS_TO_PF2	DLA_PF2	COMMAND WORDS FORMATTED TO DRIVE LITES & ALARMS ON/OFF VIA PF2	690		ST(1)	
L269.10	SET_PF2	DLA_FCOS_SPF2	FIRST SIGNIFICANT WORD IN THE FCOS CW TO PF2. BITS ARE SET IN THIS WORD TO TURN ON THE CLASS 3 ALERT TONE. A BIT IS SET IN THIS WORD TO TURN OFF THE B/U C&W BITE	690		BS(16)	
L269.20	RESET_PF2	DLA_FCOS_RPF2	SECOND SIGNIFICANT WORD IN THE FCOS CW TO PF2. BITS ARE SET IN THIS WORD TO TURN OFF THE CLASS 3 ALERT TONE. A BIT IS SET IN THIS WORD TO TURN ON THE B/U C&W LITE	690		BS(16)	
L269.30	ALERT_SET_PF2	DLA_ALERT_SPF2	THIRD SIGNIFICANT WORD IN THE FCOS CW TO PF2. BITS ARE SET IN THIS WORD TO TURN ON THE CLASS3 ALERT LITE	690		BS(16)	
L269.40	ALERT_RESET_PF2	DLA_ALERT_RPF2	FOURTH SIGNIFICANT WORD IN THE FCOS CW TO PF2. BITS ARE SET IN THIS WORD TO TURN OFF THE CLASS 3 ALERT LITE	690		BS(16)	
L270	LIGHT_ALARM_XREF_TABLE	DLA_LXR	TABLE INDICATING THE BIT TO BE SET IN THE FCOS COMMAND WORDS FOR EACH		690	ST(16)	

```
*****************************************************************************************************************
*  ID   *        ITEM        *                                                 * SRC * DST *   ATTRIBUTES   *  MML  *
*       *      HAL/NAME      *                    DESCRIPTION                  *     *     *                *       *
*****************************************************************************************************************
*       *                    * LITE AND ALARM. THE TABLE CONTAINS             *     *     *                *       *
*       *                    * AN ENTRY FOR EACH ALARM AND IS                 *     *     *                *       *
*       *                    * MAPPED AS FOLLOWS:                             *     *     *                *       *
*       *                    *   ENTRY  1:CLASS3 ALERT TONE BIT               *     *     *                *       *
*       *                    *          LOCATION                              *     *     *                *       *
*       *                    *   ENTRY  2:CLASS3 ALERT LITE BIT               *     *     *                *       *
*       *                    *          LOCATION                              *     *     *                *       *
*       *                    *   ENTRY  3:CLASS2 INDICATOR 1 BIT              *     *     *                *       *
*       *                    *          LOCATION                              *     *     *                *       *
*       *                    *   ENTRY  4:CLASS2 INDICATOR 2 BIT              *     *     *                *       *
*       *                    *          LOCATION                              *     *     *                *       *
*       *                    *   ENTRY  5:CLASS2 INDICATOR 3 BIT              *     *     *                *       *
*       *                    *          LOCATION                              *     *     *                *       *
*       *                    *   ENTRY  6:CLASS2 INDICATOR 4 BIT              *     *     *                *       *
*       *                    *          LOCATION                              *     *     *                *       *
*       *                    *   ENTRY  7:CLASS2 INDICATOR 5 BIT              *     *     *                *       *
*       *                    *          LOCATION                              *     *     *                *       *
*       *                    *   ENTRY  8:CLASS2 INDICATOR 6 BIT              *     *     *                *       *
*       *                    *          LOCATION                              *     *     *                *       *
*       *                    *   ENTRY  9:CLASS2 INDICATOR 7 BIT              *     *     *                *       *
*       *                    *          LOCATION                              *     *     *                *       *
*       *                    *   ENTRY 10:CLASS2 INDICATOR 0 BIT              *     *     *                *       *
*       *                    *          LOCATION                              *     *     *                *       *
*       *                    *   ENTRY 11-16:SPARE                            *     *     *                *       *
*       *                    * A TABLE EXISTS FOR EACH PAYLOAD                *     *     *                *       *
*       *                    * MDM                                            *     *     *                *       *
*       *                    *                                                *     *     *                *       *
*       *                    * BUFFER USED TO OUTPUT DATA TO PF1              *     *     *                *       *
*       *                    * VIA FCOS I/O MACRO. DATA IS MOVED              *     *     *                *       *
*       *                    * FROM THE FORMATTED COMMAND WORD                *     *     *                *       *
*       *                    * (FCOS_CW_TO_PF1) PRIOR TO ISSUING              *     *     *                *       *
*       *                    * THE FCOS I/O MACRO. THIS IS DONE               *     *     *                *       *
*       *                    * SO THAT LIGHT AND ALARM MAY                    *     *     *                *       *
*       *                    * CONTINUE TO PROCESS NEW FAULTS                 *     *     *                *       *
*       *                    * WHILE FCOS OUTPUTS THE DATA FROM               *     *     *                *       *
*       *                    * THE OLD FAULTS AND/OR LIGHT AND                *     *     *                *       *
*       *                    * ALARM PROCESSING. THE BUFFER IS                *     *     *                *       *
*       *                    * MAPPED LIKE THE FCOS_CWS_TO_PF1                *     *     *                *       *
*       *                    * SUCH THAT FOR EACH ENTRY IN FCOS               *     *     *                *       *
*       *                    * CWS_TOPF1, THERE IS A CORRESPONDING            *     *     *                *       *
*       *                    * ENTRY IN FCOS_OUTPUT_BUFFER_TO_PF1             *     *     *                *       *
*       *                    *                                                *     *     *                *       *
*       *                    * BUFFER USED TO OUTPUT DATA TO PF2              *     *     *                *       *
*       *                    * VIA FCOS I/O MACRO. DATA IS MOVED              *     *     *                *       *
*       *                    * FROM THE FORMATTED COMMAND WORD                *     *     *                *       *
*       *                    * FCOS_SWS_TO_PF2 PRIOR TO ISSUING               *     *     *                *       *
*       *                    * THE FCOS I/O MACRO. THIS IS DONE SO            *     *     *                *       *
*       *                    * THAT L/A MAY CONTINUE TO PROCESS               *     *     *                *       *
*       *                    * NEW FAULTS WHILE FCOS OUTPUTS THE              *     *     *                *       *
*       *                    * DATA FROM THE OLD FAULTS AND/OR                *     *     *                *       *
*       *                    * LIGHT AND ALARM PROCESSING. THE                *     *     *                *       *
*       *                    * BUFFER IS MAPPED LIKE THE FCOS_CWS             *     *     *                *       *
*       *                    * TO_PF2 SUCH THAT FOR EACH ENTRY                *     *     *                *       *
*****************************************************************************************************************
* L271  * FCOS_OUTPUT_BUFFER_ * PF1_OUT                                       * 690 * ST(1)                      *
*       * TO_PF1              *                                                *     *                           *
*****************************************************************************************************************
* L272  * FCOS_OUTPUT_BUFFER_ * PF2_OUT                                       * 690 * ST(1)                      *
*       * TO_PF2              *                                                *     *                           *
*****************************************************************************************************************
```

U/I LOCAL DATA

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
			IN FCOS_CWS TO PF2,THERE IS A CORRESPONDING ENTRY IN FCOS_OUTPUT _BUFFER_TO_PF2				
L273	FCOS_ANALOG_CWS	OLA_APU_WRITE	BUFFER USED TO OUTPUT VALUES TO THE APU METERS.N/A FOR ALT	SM	690	ST(1)	
L274	STATUS_BYTE_CHAR_ TABLE	CLOCSB	TABLE CONTAINING LEGAL STATUS INDICATORS	635	635	SI INIT: "MM" "HE" "LE" " "	
L280	INITIALIZATION_ REQUEST_CONTROL_ BLOCK	DMCV_IREQBLK	INITIALIZATION DATA FOR CONTROL SEGMENT-REQUEST-CONTROL BLOCK (SEE L118)	500	560	ST,INIT(1,3,0,1 ,0,1,1,0,0,0)	
L281	MM/DFT_IO_ COMPLETION	DMC_IO	INTEGER VALUE TO INDICATE COMPLETION OF MASS MEMORY READ OF DISPLAY FORMAT TABLE. (DFT)	500 505 520	505 520	I	
L282	BLANK_SCRATCH_PAD_ LINE_FLAG	DMC_CLEAR_SPL	FLAG (=0,DISABLED) TO CLEAR SCRATCH PAD LINE UPON OUTPUT OF A NEW DISPLAY	500 505	600		
L283	DEU_MESSAGE_LENGTH_ TABLE	DMC_DML	TABLE OF INDICATERS GIVING A) LENGTH OF GIVEN KEYBOARD MESSAGE FOR VALID INPUTS. B)-1 IF INVALID INPUT C) 0 IF VARIABLE LENGTH MESSAGE	500	500	A(20),I INIT(2,1,-1,-1, 1,0,5,0,0, -1,1,-1,-1,4, -1,1,-1,4,1,1	
L284	FUNCTIONS_CASE_ OFFSET	DMC_CASE_OFFSET	DIFFERENCE BETWEEN KEYBOARD FUNCTION INPUT AND CASE ID IN DO-CASE STATEMENT	505	505	9	
L285	MF_CHANGE_FUNCTION_ NUMBER	DMCKL_MF_CHANGE	FUNCTION ID EQUAL TO CASE NUMBER PLUS OFFSET	505	505	12	
L286	ANNUNCIATION_ERROR_ ID	DMC_EMPT_ERR	ID OF ERROR TO BE PASSED TO ANNUNCIATION_MACRO_INTERFACE TO FLAG AN ERROR VIA ERROR MESSAGE LINE OF THE DEU	505	505	I	
L287	DEU_NO	DMC_DEU_NUMBER	THE DEU NUMBER SAVED FROM DMC_DIT_ INDEX	505	505	I	
L288	ICC_STATUS_WORD	DMC_ICC_STATUS	ICC STATUS PASS BACK AFTER CALL TO ICC_MESSAGE_COLLECTOR (FIGURE 3.1.3.2-1)	505 480	505 480	B(16)	
L289	LOOP_COUNTER	DMC_I	COUNTER FOR LOCAL DO LOOP.	505	505		

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
L290	ICC_MESSAGE_FOR_CMPTR/CRT_KEY	DMC_CMPTR_CRT_MSG	MESSAGE BUILT FROM CMPTR/CRT KEY DESTINED FOR BUS_CONFIGURATION_CHANGE	505	480 830	BS(2)	
L291	ICC_MESSAGE_FOR COMPTR/BUS KEY	DMC_CMPTR_BUS_MSG	MESSAGE BUILT FROM CMPTR/BUS KEY DESTINED FOR BUS_CONFIGURATION_CHANGE	505	480 830	BS(2)	
L292	ITEM_PRO_EXEC_ALLOW	DMCVC_ITEM PRO_EXEC_ALLOW	FLAG TO PROHIBIT OR ALLOW ITEM/PRO /EXEC KEY ENTRIES	505	505		
L293	EMPT_ERROR_ID	DMC_EMPTE	DISPLAY REQUEST ERROR CODE, TO BE PASSED TO ANNUNCIATOR_MACRO_INTERFACE (675)	520	675		
L294	DISPLAY_REQUEST_STATUS	DMCVL_D_STATUS	ERROR INDICATOR FOR DISPLAY-REQUEST STATUS	520	520		
L295	I_INDEX	DMC_I	INTERNALLY USED LOOP INDEX	520	520		
L296	DFB_POINTER	DMC_BUFP	POINTER TO DFB IN DFB SEARCH FOR A DFT	520	520		
L297	DFB_STORE_INDEX	DMC_B_STORI	INDEX TO BUFFER FOR MM READ FOR DFT	520	520	I INIT(1)	
L298	DFB_READ_STATUS	DFB_STAT_I	INDICATOR OF THE STATUS OF BUFFER (DFB40P 5) FOR A MM READ.	520	520	I	
L299	J_INDEX	DMC_J	INTERNALLY USED LOOP INDEX	520	520	I	
L300	SAVE_MACT_INDEX	DMC_SAVMACT	MCDS_MACT INDEX ASSOCIATED WITH EACH OF THE TWO DFB'S INVOLVED IN A MM READ UPON DISPLAY IO REQUEST	520	520	A(2),I	
L301	DFB_BUFFER_NUMBER	DMC_BUFF_I	DFB INDEX (1 TO 5) FOR A GIVEN DFT SEARCH	520	520	I	
L302	IO_DFT/MM_READ_DATA	DMC_READ_BUF	DATA SAVED FOR A DISPLAY REQUEST WITH DFT ON MM.	520	520	ST(2)	
L302.10	IO_DISPLAY_BUFFER_COUNT	DMC_RBUF_COUNT	IO DFT USE COUNT-USED BY DEUS.	520	520	I	
L302.20	IO_DISPLAY_LEVEL	DMC_RBUF_LVLN	SUPPORT LEVEL IN MAT OF DISPLAY READ FROM THE MM.	520	520	I	
L303.30	IO_DISPLAY_DEU	DMC_RBUF_DEU	DEU NUMBER OF A DFT READ FROM THE MM	520	520	I	
L302.40	IO_DISPLAY_MF	DMC_RBUF_MF	MF OF A DFT READ FROM THE MM	520	520	I	

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
L303	KVT_DATA_SAVE	DMC3_SAV_KVT	AREA USED TO STORE KEYBOARD VERIFICATION TABLE DATA WHILE A MM IO IS IN PROCESS	520	520	A(3),BS(16)	
L304	DISPLAY_FOUND_ID	DMC_FOUND	INDICATOR AND INDEX TO FLAG THE STATUS OF A DFB SEARCH FOR A GIVEN DFT	520	520	I	
L305	DFB_NUMBER	DMCVC_D_BUFFNO	NUMBER OF A DFB IN WHICH A DFT IS FOUND	520	520	I	
L306	DFB_INDEX	DMC_BUFFI	INDEX TO A GIVEN DFB	520	520	I	
L307	F_INDEX	DMC_F	INTERNALLY USED LOOP INDEX	520	520	I	
L308	DFB_OFFSET_INDEX	DMCVL_D_OFFSET	INDEX WITHIN A DFB TO A GIVEN DFT	520	520	I	
L309	NUMBER_OF_DISPLAYS	DMC_NBRDISP	NUMBER OF DISPLAYS(DFT) IN THE DMMD TO BE CHECKED	520	520	I	
L310	DFT_DMF	DMC_DMF	MF ID FOR A DFT WITHIN A DFB OR IN THE DMMD	520	520	I	
L311	BLOCK_STRING	DMCB_BLK_STRING	BINARY STRING USED TO TRANSFER THE DFB MM ADDRESS FROM THE DMMD TO IO PARAMETER LIST FOR MM READ	520	520	BS(16)	
L312	VALID_DEU	DMC_VDEU	THE NUMBER OF A DEU AVAILABLE TO OUTPUT A DFT SEARCH OR A NEW DISPLAY EPROP MESSAGE	520	520	I	
L313	NUMBER_OF_DFTS	DMC_NDFT	NUMBER OF DFTS FOUND IN THE SEARCH OF A DFB	520	520	I	
L314	DFB_MAX_SIZE	DMC_DFBSZE	THE MAXIMUM SIZE ALLOWED FOR THE INDEX INTO A DFB FOR EACH DFB	520	520	A(5),I,INIT=(3500,21000, 7000,2049, 2048)	
L315	DFT_LENGTH	DMC_INCR	THE LENGTH OF A DFT (TAKEN FROM THE SECOND 16 BIT WORD OF A DFT)	520	520	I	
L316	BIT_VALUE	DMC_TVALB	INTERMEDIATE BIT AREA USED FOR BIT MANIPULATIONS	520	520	BS(16)	
L317	DFT_DISPLAY	DMC_DDISP	THE NUMBER OF A DISPLAY FOUND IN A DFB TO BE CHECKED	520	520	I	
L319	K_INDEX	DMC_K	INTERNALLY USED LOOP INDEX	520	520	I	
L320	START_DEU	DMC_SDEU	THE STARTING DEU OF A DISPLAY KEY DEU SEARCH	520	520	I	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
L321	END_DEU	DMC_EDEU	THE ENDING DEU OF A DISPLAY BY DEU SEARCH	520	520	I	
L322	READ_PENDING_ INDICATOR	DMC_ RPEND	INDICATOR USED TO FLAG A PENDING MM READ WHEN A SECOND REQUEST IS MADE	520	520	I	
L323	DFB_4_READ_TABLE	DMC_MMD_PLIS4	MM/DFT READ I/O TABLE (SEE TABLE 3.2.1.1.3-1)	520	520	ST	
L323.10	DFB_4_READ_MM_ ADDRESS	DMC_MMRD PLIS4.DBLOCK	ADDRESS OF THE DFT ON THE MM	520	520	I	
L324	DFB_5_READ_TABLE	DMC_MMRD_PLIS5	MM/DFT READ I/O TABLE (SEE TABLE 3.2.1.1.3-1)	520	520	I	
L324.10	DFB_5_READ_MM_ ADDRESS	DMC_MMRD_ PLIS5.DBLOCK	ADDRESS OF DFT ON THE MM	520	520	I	
L340	DMI_LOCAL_DEU_ NUMBER	DM1VL_DEU	STORES THE DEU NUMBER CURRENTLY BEING PROCESSED BY MCDS INPUT PROCESSOR AND MCDS MESSAGE PROCESSOR	400	400 405	I	
L341	POLLING_ERROR_FLAG	DM1V_POLL_ERR	FLAG WITH TWO SETTINGS WHERE: 0=NO POLLING ERROR HAS OCCURRED 1=POLLING ERROR OCCURRED ON POLL (FLAG USED TO UNSUPPRESS I/O WHEN POLLING AGAIN SUCCESSFUL FOLLOWING A FAILURE	400	400	BS(16) INIT('0000'X)	

U/I LOCAL DATA

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
T010	FAULT_SUMMARY_PAGE	CDL_FAULT_SUMMARY_PAGE	FAULT SUMMARY PAGE BUFFER AREA	680 685 880	620 680 685 CPT	ST(20),A(20),BS(16)	
T020	FAULT_MESSAGE_TIME	CDL_FAULT_MESSAGE_TIME	MISSION ELAPSED TIME ASSOCIATED WITH EACH FAULT MESSAGE BEING DISPLAYED ON THE FAULT SUMMARY PAGE DISPLAY	680 685	620 680 685 CPT	ST(20),F(20)	
T030	FSP_DIRECTORY	CDL_FSP_DIRECTORY	INFORMATION REGARDING THE STATUS OF EACH FAULT MESSAGE IN THE FSP I.E.,CLASS,POSITION IN THE FSP AND WHETHER OR NOT THE MESSAGE HAS BEEN ANNUNCIATED ON THE MESSAGE LINE	680 685	620 680 685	ST(20)	
T030.02	TIME_CONVERSION_BIT	CDL_CONVERSION_BIT	INFORMATION FOR CYCLIC DISPLAY PROCESSOR INDICATING WHETHER OR NOT A TIME CONVERSION ON A PARTICULAR MESSAGE IN THE FSP IS NEEDED	685	635 680 685	BIT (1)	
T040	APPLICATION_INTERFACE_TABLE	CDL_AIT	PROVIDES COMMUNICATION BETWEEN L/A PROCESSING CYCLES & BETWEEN L/A PROC AND UI/APPLICATIONS MODULES	680 685 690		ST(1)	
T040.01	TONE_IND	CDL_TONE_IND	INDICATION THAT TONE IS CURRENTLY BEING DRIVEN	690	690	BT	
T040.02	RECEIVE_FLAG	CDL_RECEIVE	INDICATION THAT AN ERROR HAS BEEN ANNUNCIATED_MONITORED BY THE SM APPLICATION_BEFORE REANNUNCIATING ANY I/O FAULTS	690		BT	
T040.03	REQUEST_FLAG	CDL_REQUEST	INDICATION OF KEYBOARD ENTRY OF ACKNOWLEDGE KEY	680	690	BT	
T040.04	MSG_RESET_FLAG	CDL_MSG	INDICATION OF KEYBOARD ENTRY OF MESSAGE RESET KEY	680	690	BT	
T040.05	COMMAND_SEQ_FLAG	CDL_CMD_SEQ	INDICATION THAT CLASS 2 COMMAND SEQUENCE IS ACTIVE	690	690	BT	
T040.06	LAP_LIMIT	CDL_LAP_LIMIT	ALERT TONE DURATION (NUMBER OF SSIP CYCLES TONE SHOULD BE DRIVEN)	ILO	690	I	V92W2005C
T040.07	LAP_COUNT	CDL_LAP_COUNT	CYCLE COUNTER FOR TONE		690	I	
T040.08	LAP_DECREMENT_VALUE	CDL_LAP_DEC	DECREMENT VALUE FOR TONE CYCLE COUNTER		690 750	I	

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
T040.09	SCAN_COUNT	CDL_SCAN_COUNT	CYCLE COUNTER FOR SCAN ROUTINE	750	750	I	
T040.10	SCAN_DECREMENT	CDL_SCAN_DEC	SCAN ROUTINE DECREMENT VALUE		750	I	
T040.11	FSP_INTERLOCK_TIME	CDL_INTERLOCK	FSP INTERLOCK TIME IN SSIP CYCLES	ILD	670	I	V92W2000C
T050	LIGHT_AND_ALARM_MATRIX	CDL_LAM	LIGHTS & ALARMS INDICATORS USED FOR DETERMINING WHICH ALARM IS TO BE ENABLED WITH THE CORRESPONDING FAULT MESSAGE	685 690	690 CPT	ST(1)	
T050.01	NUMBER_OF_ALARMS	CDL_LAM_NUM_ALARMS_PFI	NUMBER OF ALARMS BEING COMMANDED VIA PFI			I	
T050.02	LAM_ALARMS	CDL_LAM_ALARMS	INDICATION OF LIGHTS AND TONES TO BE TURNED OFF/ON	685	690	BS(16)	
T050.03	TONE_BIT		BIT 1 OF LAM_ALARMS. INDICATES THE ALERT TONE IS TO BE PROCESSED	685	690		V72X7141X V72X7142X
T050.04	ALERT_LITE_BIT		BIT 2 OF LAM_ALARMS. INDICATES THAT CLASS 3 ALARMS ARE TO BE ENABLED				V72X7147X V72X7148X
T050.05	CLASS2_ALARM_IND		BITS 3-9 OF LAM_ALARMS. THESE ARE THE CLASS 2 ALARMS (EXCEPT THE BACKUP C&W) THAT ARE TO BE DRIVEN THE BITS ARE MAPPED AS FOLLOWS: BIT 3-CLASS 2,IND 1 (FLT CONTR SYS SATURATION BIT 4-CLASS 2,IND 2 (IMU RM FAILURE) BIT 5-CLASS 2,IND 3 (GYRO/ACC FAILURE) BIT 6-CLASS 2,IND 4(L RHC FAILURE) BIT 7-CLASS 2,IND 5(R RHC FAILURE) BIT 8-CLASS 2,IND 6(FLT CONTR CHANNEL FAILURE) BIT 9-CLASS 2,IND 7(NAV SENSOR FAILURE	685	690		V72X4555Y V72X4560Y V72X4590Y V72X4575Y V72X4570Y V72X4550Y V72X4580Y
T050.06	BACKUP_C&W_ALARM_IND		BIT 10 OF LAM_ALARMS. INDICATES THAT THE SM BACKUP C&W INDICATOR IS TO BE ENABLED	685	690		V72X7143Y V72X7144Y
T050.07	LAM_VALUE	CDL_LAM_VALUE	PCM VALUES USED TO DRIVE THE APU METERS (NOT USED IN ALT)	SM	690	A(3),I	V72Q6001V V72Q6002V V72Q6003V
T050.08	LAM_ALARMS3	CDL_LAM_ALARMS3	MAPPING OF CLASS 2 ALARMS TO BE TURNED ON IN THE NEXT LIGHT AND ALARM CYCLE. LAM_ALARMS3 IS MAPPED LIKE LAM_ALARMS(I.E.,THERE IS A	690	690	BS(16)	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
			BIT BY BIT CORRESPONDENCE BETWEEN LAM_ALARMS & LAM_ALARMS3)				
T050.09	DOWNLIST_LAM	CDL_DL_LAM	INDICATION OF THE CURRENT STATUS OF THE LIGHTS AND TONE (FOR DOWNLIST PURPOSES)	690	DL	BS(16)	V92M8975P
T050.10	ALERT_TONE_DL_IND		BIT 1 OF DOWNLIST_LAM.INDICATES CURRENT STATUS OF ALERT TONE 1=ON 0=OFF	690	DL		
T050.11	ALERT_LITE_DL_IND		BIT 2 OF DOWNLIST_LAM.INDICATES CURRENT STATUS OF ALERT LIGHT 1=ON 0=OFF	690	DL		V92X2400X
T050.12	CLASS2_DL_IND		CLASS 2 INDICATORS IN DOWNLIST_LAM COMPRISE BITS 3-THROUGH 9.0=OFF 1=ON.THE BITS ARE MAPPED AS FOLLOWS:				V92X2409X V92X2411X V92X2419X V92X2415X V92X2413X V92X2407X V92X2417X
			BIT 3-CLASS 2,IND 1 (FLT CONTR SYS SATURATION) BIT 4-CLASS 2,IND 2 (IMU RM FAILURE) BIT 5-CLASS 2,IND 3,(GYRO/ACC FAILURE BIT 6-CLASS 2,IND 4 (L RHC FAILURE BIT 7-CLASS 2,IND 5 (R RHC FAILURE BIT 8-CLASS 2,IND 6 (FLT CONTR CHANNEL FAILURE) BIT 9-CLASS 2,IND 7 (NAV SENSOR FAILURE)				
T050.13	BACKUP_C&W_DL_IND		BIT 10 OF DOWNLIST_LAM.INDICATES CURRENT STATUS OF BACKUP C&W LIGHT 1=OFF 0=ON	690	DL		V92X2405X
T055	METERS_IND	CDL_METERS_IND	INDICATION TO DRIVE THE APU FUEL GAUGE METERS.(NOT USED IN ALT)	SM	690	E(INIT=FALSE)	
T060	LIGHT_ALARM_ PROCESSOR_EVENT	CDL_LAP_EVT	EVENT USED TO INVOKE THE LIGHT & ALARM PROCESSOR	670 680 685	690 750	E,INIT(FALSE)	
T070	FMPT_HDR_BITS	CDL_FMPT_ HDR?CDLBERRM	BITS REPRESENTING AN INDEX INTO THE FMPTS CORRESPONDING TO A UNIQUE FAULT MESSAGE	142 180 181 142 330 375 670	182 240 242 244 375 670 675	ST(5),BS(16),IN IT(X'0000')	

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
				675	750		
T070.02	MAX_HDR_WORDS	CDLK_HDR_BITS	REPRESENTS MAXIMUM NUMBER OF FAULT MESSAGES.USED AS THE UPPER LIMIT FOR DETERMINING THE MAXIMUM NUMBER OF FMPT_HDR_BITS TO SCAN		670	I	
T080	FAULT_MESSAGE_PROCESSING_TABLES	CDL_FMPT	CONTAINS THE FOLLOWING FAULT MESSAGE INFORMATION	670 675 MRW	670 CRT	ST(210),BS(16)	
T080.02	RESERVE_BIT	CDLB_FMPT_DIC	1-NOT USED	670 675 MRW	670 CRT	B(1)	
T080.04	ANNUNCIATION_GPC_CODE	CDLB_FMPT_GPC	2.USED FOR GPC IDENTIFICATION			BS(5)	
T080.06	ANNUNCIATION_MAJOR_FIELD	CDLB_FMPTMAJOR	3.USED AS THE MAJOR FIELD INDEX INTO THE MAJOR DICTIONARY	670 675 MRW	670 CRT	BS(10)	
T080.08	ANNUNCIATION_MINOR_FIELD	CDLB_FMPT_MINOR	4.USED AS THE MINOR FIELD INDEX INTO THE MINOR DICTIONARY	670 675 MRW	670 CRT	BS(7)	
T080.10	ANNUNCIATION_STATUS_BIT	CDLB_FMPT_PSI	5.USED WITH CLASS TWO MESSAGES INDICATING A PARAMETER IS OUT OF LIMIT HIGH OR LOW	670 675 MRW	670 CRT	B(1)	
T080.12	ANNUNCIATION_INDICATOR_BITS	CDLB_FMPT_IND	6.USED TO DETERMINE WHICH CAUTION AND WARNING INDICATOR ARE TO BE DRIVEN IN ASSOCIATION WITH THE CORRESPONDING FAULT MESSAGE AND FOR KEYBOARD MESSAGES INDICATES THE DEU NUMBER OF THE SOURCE DEU	670 675 MRW	670 CRT	BS(4)	
T080.14	ANNUNCIATION_MESSAGE_CLASS	CDLB_FMPT_CLASS	7.THE FAULT MESSAGE CLASS CONSISTING OF CLASS TWO, CLASS THREE,OR CLASS FOUR FAULT MESSAGES	670 675 MPW	670 CRT	BS(4)	
T090	MAJOR_DICTIONARY	CDLB_A_MAJOR	CONTAINS THE MAJOR FIELDS OF ALL FAULT MESSAGES IN A DEU DISPLAYABLE FORMAT	670 675	685 DL	ST(89),A(20),BS(16)	
T090.5	MAJ_FIELD_ID	CDL_MAJ_ID	USED FOR DOWNLISTING OF THE FIVE MOST CURRENT MAJOR FIELD ON THE FSP	670 675 MRW	685 DL	BS(16)	
T100	MINOR_DICTIONARY	CDLBA_MINOR	CONTAINS THE MINOR FIELD OF ALL	670	685	ST(109),BS	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
			FAULT MESSAGES IN A DEU DISPLAYABLE FORMAT	MPM MPM	DL		
T100.5	MIN_FIELD_ID	COLB_MIN_ID	USED FOR DOWNLISTING OF THE FIVE MOST CURRENT MINOR FIELDS ON THE FSP	670 675 MPM	685	BS(9)	
T110	FAULT_MESSAGE_BIT	COL_MSG_BIT	INDICATOR USED TO INVOKE FAULT MESSAGE SCAN FOR PROCESSING FAULT MESSAGES	675 750	670 750	BO	
T120	ANNUNCIATION_UP_ARROW	COC_UP_ARROW	DEU DISPLAYABLE FCW REPRESENTING AN UP ARROW ( ) USED WITH CLASS 2 FAULT MESSAGES	680	685	BS(16)	DPU1103
T130	ANNUNCIATION_DOWN_ARROW	COC_DOWN_ARROW	DEU DISPLAYABLE FCW REPRESENTING A DOWN ARROW ( ) USED WITH CLASS 2 FAULT MESSAGES	680	685	BS(16)	DPU1104

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
V010	LDB_TRANSACTION_STATUS_WORD	CDVV_LDB_FSW_ITSW	STATUS OF LDB INPUT TRANSACTIONS	440	440	DI	
V020	INPUT_RESIDUAL_WORD_COUNT	CDVV_LDB_FSW_IRES	RESIDUAL WORD COUNT ON LDB INPUT TRANSACTION ERRORS			I	
V030	INPUT_ERROR_LOG_POINTER	CDVV_LDB_FSW_IERR	POINTER TO I/O ERROR LOG ON LDB INPUT TRANSACTION ERRORS			I	
V040	LDB_INPUT_BUFFER	CDVV_LDB_FSW_IBUF	BUFFER FOR LDB INPUT TRANSACTIONS PROCESSED IN NON-VEHICLE CHECKOUT OPS	440	440	ST(1)	
V040.05	INPUT_DESTINATION	CDVV_DEST	IDENTIFIES APPLICATION TO WHICH INPUT MESSAGE IS TO BE ROUTED	440	440	BS(5)	
V040.10	INPUT_TRANSACTION_ID	CDVV_TRANSACT_ID	TRANSACTION ID OF INPUT MESSAGE	440	440	BS(11)	
V040.15	MM_OPERATION_CODE	CDVV_MM_OPCOD	IDENTIFIES MM OPERATION TO BE PREFORMED BY MM UTILITY APPLICATION	440	440	BS(4)	
V040.20	SPARE_1	CDVV_SPARE1	SPARE			BS(1)	
V040.25	LDB_DEU_NUMBER	CDVV_LDB_DEU	IDENTIFIES DEU TO BE ADDRESSED BY EQUIVALENT DEU MESSAGE	440	440	BS(3)	
V040.30	SPARE_2	CDVV_SPARE2	SPARE			BS(2)	
V040.35	MM_BLOCK_NUMBER	CDVV_LDB_MM_BLOCK	IDENTIFIES BLOCK ON MM UTILITY WRITE REQUEST	440	440	BS(5)	
V040.40	SPARE_3	CDVV_SPARE3	SPARE			BS(1)	
V040.45	CONTROL_HEADED_WORD_2	CDVV_CHW2	APPLICATION DEPENDENT CONTROL INFORMATION	440	440	I	
V040.50	INPUT_DATA_WORDS	CDVV_LDB_IN_DATA	DATA WORDS IN LDB INPUT MESSAGE	440	440	A(131),I	
V050	OUTPUT_BUFFER_ALIGN	CDVV_BUFR_ALIGN	ALIGNS OUTPUT BUFFER ON DOUBLE WORD BOUNDARY			DI	
V060	OUTPUT_RESIDUAL_WORD_COUNT	CDVV_LDB_FSW_ORES	RESIDUAL WORD COUNT ON LDB OUTPUT MESSAGES			I	
V070	OUTPUT_ERROR_LOG_POINTER	CDVV_LDB_FSW_OERR	POINTER TO ERROR LOG ENTRY ON LDB OUTPUT ERRORS			I	
V080	LDB_OUTPUT_BUFFER	CDVV_LDB_OUT_DATA	BUFFER FOR LDB OUTPUT MESSAGES	460	440	A(137),I	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
			PROCESSED IN NON VEHICLE CHECKOUT OPS		460		
V090	LDB_ERROR_COUNT	CDVV_ERR_COUNT	COUNT OF CONSECUTIVE ERRORS ON LDB TRANSACTIONS	440	440	I,INIT(0)	
V100	OUTPUT_MESSAGE_LENGTH	CDVV_OUT_MSG_LENG	NUMBER OF WORDS IN MESSAGE TO BE TRANSMITTED TO THE GROUND	440 460	440 460	I,INIT(0)	
V110	INPUT_MESSAGE_LENGTH	CDVV_IN_WDCOUNT	NUMBER OF WORDS IN MESSAGE RECEIVED FROM THE GROUND.	440	440	I,INIT(0)	
V120	INPUT_MESSAGE_STATUS_WORD	CDVV_LDB_STAT	TRANSMITTED TO THE GROUND TO INDICATE STATUS OF GROUND TO GPC MESSAGE	440	440	BS(16)	
V130	OLD_TRANSACTION_ID	CDVV_OLD_TRANSACT	TRANSACTION ID OF LAST GROUND TO GPC MESSAGE PROCESSED	440	440	I,INIT(-1)	
V140	TCSS_TRANSACTION_ID	CDVV_TCSS_TRANSACT	TRANSACTION ID OF LAST TCS SEQUENCE MESSAGE PROCESSED	440	440	I,INIT(-1)	
V150	INTERROGATE_RESPONSE_STATUS	CDVV_LDB_IRW_TSW	STATUS OF LDB INTERROGATE TRANSACTION	440	440	DI	
V160	INTERROGATE_RESPONSE_RESIDUAL_WORD_COUNT	CDVV_LDB_IRW_IRES	RESIDUAL WORD COUNT ON LDB INTERROGATE TRANSACTION ERRORS			I	
V170	INTERROGATE_ERROR_POINTER	CDVV_LDB_IRW_IERR	POINTER TO ERROR LOG ENTRY ON INTERROGATE RESPONSE TRANSACTION ERRORS	440	440	I	
V180	INTERROGATE_RESPONSE_WORD	CDVV_LDB_IRW(TFCMCIII TFCMLSRI)	GROUND RESPONSE TO LDB INTERROGATE COMMAND. ALSO USED FOR GROUND RESPONSE TO STATUS REQUEST COMMAND	263 440	440	ST(1)	
V180.05	INTERROGATE_RESPONSE_CODE	CDVV_OPCODE	GROUND RESPONSE CODE INDICATING ACTION TO BE TAKEN BY GPC 0011-NO NEED FOR BUS 1011-SEND GPC DATA 0110-GROUND DATA TO BE READ	440	440	BS(4)	
V180.10	GROUND_MESSAGE_WORD_COUNT	CDVV_IN_MSG_LENG	NUMBER OF WORDS IN GROUND TO GPC MESSAGE TO BE READ, RECEIVED IN CONJUNCTION WITH CDVV_OPCODE=0110 NON-ZERO BINARY CODE	440	440	BS(12)	
V190	OUTPUT_PENDING_FLAG	CDVB_XMIT_FLAG	INDICATES THAT A GPC TO GROUND MESSAGE IS READY FOR TRANSMISSION	440 460	440 460	BO,INIT(FALSE)	
V200	OUTPUT_CONTENTION_	CDVE_LDB_ACTV	CONTROLS USAGE OF LDB OUTPUT	440	440	F	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MNL
	EVENT		BUFFER	460	460		
V210	INPUT_BUFFER_ ADDRESS	DGI_IBUF_ADDR	ADDRESS OF LDB INPUT BUFFER TO BE USED IN CURRENT MEMORY CONFIGURATION	440	440	Y	
V220	OUTPUT_BUFFER_ ADDRESS	DGI_OBUF_ADDR	ADDRESS OF LDB OUTPUT BUFFER TO BE USED IN CURRENT MEMORY CONFIGURATION	440	440	Y	
V230	VCO_INPUT_BUFFER	CVSV_LDB_IBUF	LDB INPUT BUFFER USED IN VEHICLE CHECKOUT OPS. CONTAINED IN VCO SHARED COMPOOL (CVS_SHARED_ COMPOOL)	440	440	ST(1)	
V240	VCO_OUTPUT_BUFFER	CVSV_LDB_OBUF	LDB OUTPUT BUFFER USED IN VEHICLE CHECKOUT OPS'S. CONTAINED IN VCO SHARED COMPOOL (CVS_SHARED_ COMPOOL)	460	440 460	ST(1)	
V250	IO_STATUS_POINTER	DGI_IOERR_CODE	POINTER TO STATUS INFORMATION ON LDB INPUT TRANSACTIONS	440	440	Y	
V260	VCO_TRANSACTION_ STATUS_WORD	CVSV_LDB_TSW	TRANSACTION STATUS WORD FOR LDB INPUT MESSAGES PROCESSED IN VEHICLE CHECKOUT OPS'S	440	440	DI	
V270	ERROR_DATA_ STRUCTURE	DGI_ERR_STRUC	USED TO ADDRESS CONTENTS OF ERROR LOG ENTRY ON ERROR ENCOUNTERED ON LDB STATUS REQUEST COMMANDS	440	440	ST(1)	
V270.05	ERROR_DATA_POINTER	DGI_ERP_PTR	USED TO ADDRESS CONTENTS OF ERROR LOG ENTRY ON ERRORS ENCOUNTERED ON LDB STATUS REQUEST COMMANDS	440	440	Y	
V280	CHECKSUM_SAVE_AREA	DGI_TEMP_SUMCHECK	CHECKSUM WORD RECEIVED WITH INPUT MESSAGE IS SAVED HERE	440	440	I	
V290	END_MESSAGE_ POINTER	DGI_ENDDATA	ADDRESS OF LAST WORD IN INPUT MESSAGE	440	440	Y	
V300	INPUT_CHECKSUM_ COUNT	DGI_SUMCHECK_COUNT	NUMBER OF WORDS TO BE INCLUDED IN CHECKSUM OF INPUT MESSAGE	440	440	I	
V310	CURRENT_ TRANSACTION_ID	DGI_CURR_TRANSACT	TRANSACTION ID OF INPUT MESSAGE BEING PROCESSED	440	440	I	
V320	CURRENT_MESSAGE_ DESTINATION	DGI_FUNC_DEST	FUNCTIONAL DESTINATION OF MESSAGE BEING PROCESSED	440	440	I	
V330	CURRENT_DEU_NUMBER	DGI_DEU_ID	NUMBER OF DEU ADDRESSED IN CURRENT EQUIVALENT DEU MESSAGE	440	440	I	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
V340	MASKED_STATUS_WORD	DGI_ERR_TEST	RESULT OF TESTING INPUT MESSAGE STATUS WORD FOR I/O ERROR	440	440	BS(16)	
V350	ERROR_MASK	DGI_ERR_MASK	MASK USED IN TESTING INPUT MESSAGE WORD FOR I/O ERRORS		440	BS(16) CONSTANT(HEX'0820')	
V360	TCS_BUSY_FLAG	CVSA_BUSY	INDICATES AVAILABILITY OF TCS SEQUENCE ACQUISITION PROCESSOR TO ACCEPT A MESSAGE	440	440	B	
V370	TCS_BUFFER_READY	CVAA_READY	INDICATES AVAILABILITY OF A TCS SEQUENCE BUFFER		440	BO	
V380	TCS_BLOCK_FLAG	CVSA_SAP_INPUT	INDICATES WHETHER THE LDBI/O PROCESSOR IS PASSING A TCS SEQUENCE INITIAL BLOCK OR CONTINUATION BLOCK	440	440	I	
V390	SACS_INPUT_BUFFER	CVAV_LDB_DATA	MESSAGES TO BE PROCESSED BY SACS APPLICATION ARE STORED HERE	440		A(135)I	
V400	TCS_INPUT_BUFFER	CVSA_BUFFER	BUFFER FOR MESSAGES TO BE PROCESSED BY THE TCS SINGLE COMMAND APPLICATION AND GPC TO GROUND RESPONSES	440	440 460	A(6,100)I	
V410	MM_HEADER	CDHV_REQUEST	THREE WORD HEADER CONTAINING CONTROL INFORMATION FOR MM READ AND WRITE REQUESTS	440	440	ST(1)	
V410.05	MM_ROUTER_HEADER_WORD	CDHV_RHW	ROUTER HEADER WORD ON MM UTILITY MESSAGES	440	440	BS(16)	
V410.10	MM_INPUT_BLOCK_ID	CDHB_NDX	BLOCK NUMBER USED IN MM REQUESTS INVOLVING MULTIPLE BLOCKS	440	440 460	BS(5)	
V410.15	MM_BLOCK_COUNT	CDHB_N_BLK	NUMBER OF BLOCKS IN MM UTILITY REQUEST	440	440	BS(5)	
V410.20	MM_REQUEST	CDHB_OPCD	CODE IDENTIFYING MM UTILITY OPERATION TO BE PERFORMED	440	440 460	BS(4)	
V420	INITIAL_MM_ROUTER_HEADER_WORD	CDHB_RHW_ST	SAVE AREA FOR ROUTER HEADER WORD ON INITIAL BLOCK OF A MULTI-BLOCK MM WRITE REQUEST	440	440	BS(16)	
V430	MM_PATCH_HEADER	CDHV_PATCH_REQUEST	SEVEN WORD HEADER CONTAINING CONTROL INFORMATION FOR MM PATCH REQUEST MESSAGES	440 460	440 460	ST(1)	
V440	MM_RESPONSE	CDHV_PATCH_RESPONSE	RESPONSE BUFFER FOR GPC TO GROUND LDB MESSAGES FROM MM UTILITY		460	A(7)I	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
			APPLICATION				
V450	MM_READ_WRITE_BUFFER	CDHV_BLOCKS	INPUT BUFFER FOR MM UTILITY READ AND WRITE REQUESTS	440 460	440 460	A(32,512)I	
V460	MM_PATCH_BUFFER	CDIV_RW_BUFR	INPUT BUFFER FOR MM PATCH REQUESTS	440	440	A(32,512)I	
V470	MM_PATCH_WORD_COUNT	CDHV_LDB_COUNT	COUNT OF PATCH DATA WORDS	440	440	I	
V480	MM_DUMP_ID	CDHV_DUMP_BLOCK_ID	BLOCK ID ON MM UTILITY DUMP OF MULTIPLE BLOCK LOAD BLOCKS		460	I	
V490	MM_OUTPUT_BLOCK_INDEX	DGO_MM_BLOCK	BLOCK INDEX INTO MM READ/WRITE BUFFER DUMP TO GROUND	460	460	I	
V500	MM_INPUT_BLOCK_INDEX	DLM_MM_BLOCKID	BLOCK INDEX INTO MM READ/WRITE BUFFER ON GROUND TO GPC TRANSMISSION	440	440	I	
V510	TCS_SEQUENCE_BUFFER	CVSA_DATA	INPUT BUFFER FOR TCS SEQUENCE MESSAGES	440	440 460	A(512)I	
V520	OUTPUT_BUFFER_ID	DGO_BUFFER_ID	BUFFER ID PASSED TO LDB OUTPUT MESSAGE COORDINATOR BY APPLICATION PROCESSORS		460	I	
V530	OUTPUT_WORD_COUNT	DGO_WDCOUNT	WORD COUNT PASSED TO LDB OUTPUT MESSAGE COORDINATOR BY APPLICATIONS PROCESSORS, INDICATES NUMBER OF WORDS TO BE TRANSMITTED TO GROUND		460	I	
V540	SACS_RESPONSE	CVAV_RESPONSE_DATA	SACS GPC TO GROUND RESPONSE BUFFER		460	A(7)I	
V550	WAVEFORM_GENERATOR_ERROR_RESPONSE	CVAV_V_ERR_RESP	WAVEFORM GENERATOR GPC TO GROUND ERROR RESPONSE BUFFER		460	A(7)I	
V560	RAMP_ERROR_RESPONSE	CVRV_ERROR_RESPONSE_DATA	RAMP GENERATOR GPC TO GROUND ERROR RESPONSE BUFFER		460	A(7)I	
V570	OUTPUT_CHECKSUM_COUNT	DGO_SUMCHECK_COUNT	NUMBER OF WORDS TO BE INCLUDED IN CHECKSUM OF OUTPUT MESSAGE	460	460	I	
V580	INTERROGATE_WORD_COUNT	DGI_IWD_PARMS.DDCNT	I/O PARAMETER LIST LOCATION OF THE COUNT OF WORDS AVAILABLE IN THE GPC FOR TRANSMISSION TO THE GROUND	440	440	I	
V590	STATUS_MASK	DGI_STAT_PARMS.DDMASK	I/O PARAMETER LIST LOCATION OF THE STATUS WORD SENT TO THE GROUND ON STATUS COMMANDS	440	440	I	
V600	VCO_SEND_WORD_	DGI_VCO_TREN_	I/O PARAMETER LIST LOCATION OF	440	440	I	

ITEM ID	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
	COUNT	PARMS.DWDCT				
V610	VCO_SEND_NZB_COUNT	DGI_VCO_TREN_ PARMS.DDCNT	I/O PARAMETER LIST LOCATION OF WORD COUNT IN NON-ZERO BINARY FORMAT ON TRANSMISSION ENABLE COMMAND	440	440	I
V620	LDB_SEND_WORD_ COUNT	DGI_TREN_ PARMS.DWDCT	I/O PARAMETER LIST LOCATION OF WORD COUNT ON TRANSMISSION ENABLE COMMAND	440	440	I
V630	LDB_SEND_NZB_COUNT	DGI_TREN_ PARMS.DDCNT	I/O PARAMETER LIST LOCATION OF WORD COUNT IN NZB FORMAT ON TRANSMISSION ENABLE COMMAND	440	440	I
V640	VCO_RECEIVE_WORD_ COUNT	DGI_VCO_GO_ PARMS.DWDCT	I/O PARAMETER LIST LOCATION OF THE WORD COUNT ASSOCIATED WITH LDB 'GO AHEAD' COMMANDS USING VCO INPUT BUFFER	440	440	I
V650	LDB_RECEIVE_WORD_ COUNT	DGI_GO_PARMS.DWDCT	I/O PARAMETER LIST LOCATION OF THE WORD COUNT ASSOCIATED WITH LDB 'GO AHEAD' COMMANDS USING LDB INPUT BUFFER	440	440	I
V660	LDB_COMMAND_BUS	DGI_CMD_MODE.VAR_ DATA1	SVC PARAMETER LIST LOCATION OF NUMBER OF THE BUS TO BE PLACED IN COMMAND MODE ON BUS CONFIGURATION REQUEST	440	440	I
V670	LDB_LISTEN_BUS	DGI_LSTN_MODE.VAR_ DATA1	SVC PARAMETER LIST LOCATION OF NUMBER OF THE BUS TO BE PLACED IN LISTEN MODE ON BUS CONFIGURATION REQUESTS	440	440	I
V680	LDB_PRIME_ID	DGI_LSTN_MODE.VAR_ DATA2	SVC PARAMETER LIST LOCATION OF PRIME GPC ID ON LISTEN MODE BUS CONFIGURATION REQUESTS	440	440	I

ICC Messages

Appendix F

(To Be Provided)

APPENDIX G

Time Lines

KEYBOARD MESSAGE TIME LINE

The attached time lines represent the sequence of processing for two MCDS keyboard entries from their introduction to the GPC until a new display has been transferred to the MCDS. The flight software has been loaded, initialized and is executing an Operations Sequence (OPS) within a Major Function.

The charts outline the sequence of modules that are executed to process two keyboard messages. The two messages selected for illustration are an OPS message requesting an Operational Sequence in a new program overlay and a PRO message requesting a change of display within the current OPS or SPEC. The following is the environment in which this processing takes place:

- The MCDS_Input_Processor is cyclically polling the MCDS and, at the beginning of the time line, a request to transmit a keyboard message has been issued to the MCDS.

- Display_Presentation_and_Control has been called by the OPS Control Segment when a display was requested and it is now in the WAIT state until an entry is received, and event occurs, or a time elapses.

- User_Interface_Control_Supervisor which was scheduled at system initialization is now in the WAIT state until an MCDS input is received, or an application control segment requests a service, or a load from Mass Memory is completed.

- Cyclic_Display_Processor has been scheduled at system initialization and is (as the name implies) cyclically generating Format Control Word Buffers for the MCDS.

"PROGRAM OVERLAY (OPS)"

Time Line

#	Software Processor	Sub-system	Action
1.	DEU software	DEU	assembles keyboard message and transmits it when commanded by GPC
2.	I/O_Completion_Processor	FCOS	locates and sets associated event and calls . . .
3.	Process_Dispatcher	FCOS	passes control to . . .
4.	MCDS_Input_Processor	UI	receives keyboard message, sets routing flag, calls . . .
5.	MCDS_Message_Processor	UI	adds data to DIT table, sets event (SVC) for . . .
6.	User_Interface_Control_ Supervisor	UI	recognizes an OPS request and calls . . .
7.	Sequence_Request_ Processor	UI	verifies overlay request and sets event (SVC) for ... (and WAITS)
8.	Display_Presentation_ and_Control	UI	determines termination requested, returns control to . . .
9.	Block cleanup segment of Application Control Segment	APP	executes application block cleanup code, calls . . .
10.	Application_Moding_ and_Sequencor	UI	determines mode termination requested, returns control to . . .
11.	Mode cleanup segment of Application Control Segment	APP	executes application mode cleanup code, calls . . .
12.	Application_Moding_and_ Sequencor	UI	determines OPS termination requested, returns control to . . .
13.	OPS cleanup of Control Segment	APP	executes application OPS cleanup code, calls . . .

"PROGRAM OVERLAY (OPS)" (Cont'd)

#	Software Processor	Sub-system	Action
14.	Application_Moding_and_ Sequencor	UI	termination processing completed, sets event (SVC) for . . .
15.	User_Interface_Control_ Supervisor	UI	determines OPS processing completed and calls . . .
16.	Sequence_Request_ Processor	UI	verifies overlay request, checks if OPS is resident and, if not resident, schedules . . .
17.	GPC Reconfiguration	SC	checks if required GPCs are available and WAITS after issuing an SVC for . . .
18.	GPC_Reconfiguration	SC	reconfigures busses, if required, creates a pseudo-keyboard message and sets an event for . . . (and CLOSEs)
19.	Program Overlay	FCOS	reads overlay from Mass Memory and returns to . . .
20.	User_Interface_Control_ Supervisor	UI	recognizes an OPS request and calls . . .
21.	Sequence_Request_ Processor	UI	schedules requested OPS Control Segment (SVC) and WAITS for Applications Input Communications event.
22.	OPS Control Segment	APP	begins execution, calls . . .
23.	Application_Moding_and_ Sequencor	UI	sets appropriate indicators in MACT table, sets event (SVC) for Sequence_Request_Processor, returns control to the OPS Control Segment (Step #26).
24.	Sequence_Request_Processor	UI	returns control to . . .
25.	User_Interface_Control_ Supervisor	UI	returnx to WAIT state.

"PROGRAM OVERLAY (OPS)" (Cont'd)

#	Software Processor	Sub-system	Action
26.	OPS Control Segment	APP	executes initialization code, executes DISPLAY statement which calls . . . (with a display request)
27.	Display_Presentation_ and_Control	UI	recognizes new display request, sets flag in MACT table for processing by Application Control Interface, sets event for . . . (and WAITS)
28.	User_Interface_Control_ Supervisor	UI	determines an application service request pending, calls . . .
29.	MCDS_Functions_Processor	UI	determines a new display pending, request a search for the DFT and returns control to . . .
30.	User_Interface_Control_ Supervisor	UI	performs a search for the DFT and calls . . .
31.	New_Display_Processor	UI	checks if FCWs are in the DEU, if necessary, issues an SVC to transmit background FCWs to DEU (see Steps #34-36), sets one-time update flag for Cyclic_Display Processor, and returns control to . . .
32.	User_Interface_Control_ Supervisor	UI	returns to WAIT state
33.	Cyclic_Display_Processor	UI	finds the one-time update flag, builds FCWs for all variable parameters, issues SVC to transmit variable FCWs to DEU
34.	SVC_Handler	FCOS	accepts and routes I/O request to . . .
35.	I/O_SVC_Service_Processor	FCOS	builds I/O control block and calls . . .

"PROGRAM OVERLAY (OPS)" (Cont'd)

#	Software Processor	Sub-system	Action
36.	IOP_Dispatcher	FCOS	starts the requested I/O operation
37.	DEU Buffers	DEU	at GPC direction, FCWs are inserted into the refresh cycle.

"New Display (PRO)"

Time Line

#	Software Processor	Sub-system	Action
1.	DEU software	DEU	assembles keyboard message and transmit it when commanded by GPC
2.	I/O_Completion_Processor	FCOS	locates and sets associated event and calls . . .
3.	Process_Dispatcher	FCOS	passes control to . . .
4.	MCDS_Input_Processor	UI	receives keyboard message, sets routing flag, calls . . .
5.	MCDS_Message_Processor	UI	adds data to DIT table, sets event (SVC)* for . . .
6.	User_Interface_Control_ Supervisor	UI	verifies message validity, stores message in MACT table, sets key-board message event (SVC)* for . . .
7.	Display_Presentation_ and_Control	UI	returns control to . . .
8.	Application Control Segment	APP	executes application code to process PRO message, executes CHANGE statement which set the 'new display' flag and calls . . .
9.	Display_Presentation_ and_Control	UI	determines the 'new block' request, and returns control to . . .
10.	Block cleanup segment of Application Control Segment	APP	executes application block clean-up code, and calls . . .
11.	Application_Moding_ and_Sequencor	UI	determines block number and returns control to . . .
12.	New Block within the Application Control Segment	APP	executes the application code, and executes DISPLAY statement which calls . . .

"New Display (PRO)" (cont.)

#	Software	Sub-system	Action
13.	Display_Presentation_and_Control	UI	recognizes new display request, sets flag in MACT table for processing by Application Control Interface, sets event for . . . (and WAITS)
14.	User_Interface_Control_Supervisor	UI	determines an application service request pending, calls . . .
15.	MCDS_Functions_Processor	UI	determines a new display pending, request a search for the DFT and returns control to . . .
16.	User_Interface_Control_Supervisor	UI	performs a search for the DFT, if not found issues a Mass Memory read request (SVC) and returns to the WAIT state
17.	SVC_Handler	FCOS	accepts and routes I/O request to . . .
18.	I/O_SVC_Service_Processor	FCOS	builds I/O control block and calls . . .
19.	IOP_Dispatcher	FCOS	starts I/O operation
20.	I/O_Completion_Processor	FCOS	locates and sets associated event and calls . . .
21.	Process_Dispatcher	FCOS	passes control to . . .
22.	User_Interface_Control_Supervisor	UI	determines it is an I/O completion, and calls . . .
23.	New_Display_Processor	UI	checks if FCWs are in the DEU, if necessary issues an SVC to transmit background FCWs to DEU (see Steps #26-28), sets one-time update flag for Cyclic_Display_Processor and returns control to . . .

"New Display (PRO)" (con'd)

#	Software	Sub-system	Action
24.	User_Interface_Control Supervisor	UI	returns to WAIT state
25.	Cyclic_Display_Processor	UI	finds the one-time update flag, builds FCWs for all variable parameters, issues SVC to transmit variable FCWs to DEU
26.	SVC_Handler	FCOS	accepts and routes I/O request to . . .
27.	I/O_SVC_Service_Processor	FCOS	builds I/O control block and calls . . .
28.	IOP_Dispatcher	FCOS	starts the requested I/O operation
29.	DEU Buffers	DEU	at GPC direction, FCWs are inserted into the refresh cycle.

Subsystem abbreviations:

DEU - Display Electronics Unit

FCOS - Flight Computer Operating System (FDS Vol 2 P1)

UI - User Interface (FDS Vol 2 P2)

APP - Applications Software (Control Segment)

SC - System Control (FDS Vol 2 P3)

APPENDIX H

CONTROL SEGMENT GRAMMAR

H.1 INTRODUCTION

There is a specific design requirement, that all applications software control logic be modularized in the software system end items. This requirement is given in order to conform to the principles of top down and structrured programming and to allow a software system design that reduces the impact of a relatively high number of Level "B" and "C" requirements changes.

All Operational Sequences (OPS) and Specialist functions (SPEC) conform to the constraints of and, in fact, are control segments. To adequately define control segments, their capabilities and functions, a grammar has been developed which defines the limitations and constraints of control segments. All OPS and SPEC in flight programs therefore conform to this grammar.

Control segments can be depicted as processing functions associated with particular display formats within the sequence. A series of blocks are defined to represent a series of displays and their associated processing. These blocks represent the total capabilities and limitations of the software that are used to build the control segment. Therefore, each block is capable of implementing any OPS or SPEC requirement that may be defined as associated with a display block.

Operational Sequences may be designed to be entered at multiple points. From an entry point a string of blocks any be sequentially executed and are referred to as a mode within an OPS. On the other hand, since a Specialist function may be entered only at the topmost point of its logic,it consists of only one mode, the SPEC itself.

Control segment blocks are composed of:

A block identification statement with an optional automatic advancement capability based on an event.

An optional initialization statement in the first block of a mode.

An action segment which includes software functions such as scheduling, calling, signaling, terminating, etc.

A display initiation and response function which includes initiating a display and then waiting for a response.

An action segment which may also include a branch to other control segment blocks.

An optional termination or cleanup segment which includes software to cancel, terminate, or otherwise cause the cessation of processing associated with the block.

Control segment modes consist of:

a. A mode identification statement with an optional automatic advancement capability based on an event.

b. All control segment blocks which comprise the mode.

c. An optional termination or cleanup segment which includes software to cause the termination of all processing associated with the mode.

Operational Sequences (OPS) and Specialist functions (SPEC) similarly are composed of:

a. A unique identification statement with an optional automatic advancement capability based on an event.

b. An optional group of initialization statements for the OPS or SPEC.

c. All lower level modes or blocks which comprise the OPS or SPEC.

d. An optional termination or cleanup segment which causes the complete termination of processing associated with the OPS or SPEC.

H.2 CONTROL SEGMENT GRAMMAR

Figures H-1 and H-2 present the control segment grammar syntax for an OPS or a SPEC. The following items refer to the syntax of the macros depicted:

- Each name in capital letters is a keyword in the macro.
- Names in small letters refer to user supplied names or statements.
- All special characters are part of the syntax and appear in the grammar usage with the exception of the braces, i.e., "{ }". These are used only to indicate an option of an argument selection.

As previously mentioned, there is a capability to invoke an automatic advancement feature on the OPS (or SPEC), mode or block level which transfers control to OPS 0-00 or the next sequential respective mode or block following the execution of all nominal cleanup segments. Different events may be specified for the three levels of automatic advancement.

Initialization of the OPS or SPEC is furnished by applications-provided control segment statements preceding the OPS or SPEC identification statement. These statements are executed once upon first entry into this high level segment. A mode initialization is permitted in the first control segment statement following the block identification. It should be noted that the DO...END group is allowed to bracket this initialization. However, this initialization is procedurally restricted to the first block of each mode of an OPS.

The DISPLAY statement within each control block initiates the display and then waits for a response. In some cases a block may not have a display format associated

```
program-name:  PROGRAM;
               INCLUDE macro-library-name;
               [control-segment-statements]
               OPS (operational-sequence-number, {ADVANCE_ON event | NO_AUTO_ADVANCE}) ;
                  MODE (mode-name, {ADVANCE_ON event | NO_AUTO_ADVANCE});
                  [INIT_] BLOCK (block-name, {ADVANCE_ON event | NO_AUTO_ADVANCE})
                  [one-initialization-control-segment-statement] ;
                  [control-segment-statements]
                  DISPLAY ({display-number | 0}) ;
                  [control-segment-statements]
                  CHANGE ({display-number | 0}) ;
                  [control-segment-statements]
                  [BLOCK_CLEAN_UP (block-name);
                  control-segment-statements]
                  BLOCK_END (block-name, {CLEAN_UP | NO_CLEAN_UP}) ;
                  [BLOCK (block-name, {ADVANCE_ON event | NO_AUTO_ADVANCE});
                  BLOCK_END (block-name, {CLEAN_UP | NO_CLEAN_UP});]
                        [additional blocks]
                  [MODE_CLEAN_UP (mode-name);
                  control-segment-statements]
                  MODE_END (mode-name, {CLEAN_UP_MODE | NO_CLEAN_UP_MODE}) ;
                        [additional modes]
                  [OPS_CLEAN_UP;
                  control-segment-statements]
                     OPS_END ({CLEAN_UP_OPS | NO_CLEAN_UP_OPS});
                     CLOSE program-name;
```

Figure H-1. Control Segment Grammar For OPS

```
program-name:  PROGRAM;
               INCLUDE macro-library-name;
               [control-segment-statements]
               SPEC ( specialist-number, { ADVANCE_ON event } );
                                         { NO_AUTO_ADVANCE  }
                 BLOCK ( block-name, { ADVANCE_ON event } )   ;
                                     { NO_AUTO_ADVANCE  }
                  [control-segment-statements]
                 DISPLAY ( ( { display-number } ) )   ;
                                { 0           }
                  [control-segment-statements]
                 CHANGE ( ( { display-number } ) );
                                { 0           }
                  [control-segment-statements]
                [BLOCK_CLEAN_UP (block-name);]
                [control-segment-statements  ]
                BLOCK_END ( block-name, { CLEAN_UP    } )   ;
                                       { NO_CLEAN_UP }
                          .
                          .
                          .
                       [additional blocks]
               [SPEC_CLEAN_UP;                  ]
               [control-segment-statements      ]
                SPEC_END ( { CLEAN_UP_SPEC    } )   ;
                           { NO_CLEAN_UP_SPEC }
                CLOSE program-name;
```

Figure H-2. Control Segment Grammar For SPEC

with it. This condition is indicated by the "DISPLAY (0)" option which does not disturb the display currently on the CRT.

One additional statement is required within each control block, the CHANGE macro statement. This provides the applications programmer with a means of setting an indicator directing the transfer of control to a new block. Before the transfer is made, the cleanup segment of the block, if provided, is always executed and, if appropriate, the cleanup segment associated with the mode is also executed in turn. The CHANGE statement may point to any block within the current mode of an OPS or within the SPEC or to the first block of any other mode of the OPS. An illegal mode or block specification will cause termination of the requesting control segment. The "CHANGE (0)" form of the statement is a request to advance to the next control block in ascending numerical sequence.

The following legal control segment statements, as currently defined, are available to the applications programmer when writing the control segments.

Call - statement

Schedule - statement

Signal - statement

Set/Reset - statements

Terminate - statement

Cancel - statement

If - statement

Assignment - statement (controlled by programming standards)

Do - group (controlled by programming standards)

Include - statement (controlled by programming standards

The grammar is a set of macros (replace statements) which have been defined by User Interface to provide the application programmer with a common language to generate control segments. With the grammar, the user structures his control segment into OPS sequences or SPEC functions and provides the necessary information through the input parameters for User Interface control. The grammar also permits the crew to maintain control over the application software through external inputs to the MCDS or other defined input sources.

The following is a list of the grammar macros with a description and a definition of what each provides.

- OPS (Operational-Sequence-number , { ADVANCE_ON event / NO_AUTO_ADVANCE }) ;

  OPS is used to define a control segment as an operational sequence. The first parameter, operational sequence number, is used as an ID to UI for control interface communications. The second parameter requests UI to advance the OPS (that is, terminate the program gracefully) when this event becomes true. NO_AUTO_ADVANCE means just that, no OPS advancement based on any event. "DS" is a statement label generated for use by the macro.

- SPEC (specialist number , { ADVANCE_ON event / NO_AUTO_ADVANCE }) ;

  SPEC is used to define a control segment as a specialist function. The first parameter is the Specialist Function number used as a control ID by UI. For further definition see the OPS macro above. "DS" is a statement label generated for the macro's internal use.

- MODE (mode name , { ADVANCE_ON event / NO_AUTO_ADVANCE }) ;

  MODE specifies the beginning of a new mode within the OPS. The first parameter is a unique name within this compilation assigned to this mode. The second parameter specifies whether the mode should be advanced to the next sequential mode based on some given event or not. This macro is used with the OPS only.

- BLOCK (block name , { ADVANCE_ON event / NO_AUTO_ADVANCE }) ;

  BLOCK specifies the beginning of a new block within the current MODE (or SPEC). The first parameter is a unique name within the compilation assigned to this block. The second parameter specifies whether the block should be advanced to the next sequential block based on the given event or not.

- INIT_BLOCK (Block name , {ADVANCE_ON event / NO_AUTO_ADVANCE})

  ----------executable statement;

  The INIT_BLOCK is a BLOCK which also provides for the execution of a single statement upon first entry but not upon subsequent return. The INIT_BLOCK is optional and is used only as the first block within the MODE. The single executable statement (which may be a DO; ----END; group) is available as initialization for the MODE if required.

- DISPLAY ({ display number / 0 }) ;

  DISPLAY is used within the confines of a BLOCK to request that the requested format be displayed on the CRT and updated at its specified rate. In addition, a WAIT on the events specified at the OPS (or SPEC), MODE, and BLOCK levels along with a WAIT on the Keyboard (user input). The parameter is the display number being requested. If it is zero, the display is not changed. It is only at this point (the WAIT) that external inputs are recognized.

- BLOCK_CLEAN_UP (block-name)

  BLOCK_CLEAN_UP specifies that the code following this statement is provided by the application programmer to gracefully terminate the specified (current) block. The block-name is the same as that within the BLOCK definition above. This macro is optional.

- BLOCK_END (block-name, {CLEAN_UP / NO_CLEAN_UP}) ;

  BLOCK_END specifies the close of the block identified by "block-name". The second parameter specifies (1) with CLEAN_UP, there is an applications provided clean up section (See BLOCK_CLEAN_UP above) and (2) with NO_CLEAN_UP, there is no applications clean up provided.

- MODE_CLEAN_UP (mode-name);

  MODE_CLEAN_UP specifies that the code following this statement is provided by the application programmer to gracefully terminate the specified (current) mode. The mode name is the same as that within the MODE definition above. This macro is optional.

- MODE_END (mode-name , {CLEAN_UP_MODE / NO_CLEAN_UP_MODE}) ;

  MODE_END specifies the close of the mode identified by "mode-name". The second parameter specifies that applications mode clean up is or is not provided.

- OPS_CLEAN_UP;

  OPS_CLEAN_UP specifies that the code following this statement is provided by the application programmer to gracefully terminate this operational sequence. This macro is optional.

- OPS_END ({CLEAN_UP_OPS / NO_CLEAN_UP_OPS}) ;

  OPS_END specifies that this is the close of this OPS. The parameter specifies whether an application clean up section for the OPS is or is not provided.

- SPEC_CLEAN_UP;

  SPEC_CLEAN_UP specifies that the code following this statement is provided by the application programmer to gracefully terminate this specialist function. This macro is optional.

- SPEC_END ({CLEAN_UP_SPEC / NO_CLEAN_UP_SPEC}) ;

  SPEC_END specifies that this is the close of this specialist function. The parameter specifies whether an application clean up section is or is not provided.

- CHANGE ({Display-number / 0}) ;

  CHANGE specifies an exit from the block. The parameter specifies the block (display number) to which transfer is to be made. If this parameter is zero, transfer is to the next sequential block; if non-zero, transfer is to the mode and block indicated. If the requested mode is different from the current mode, the block must be the first block of the new mode. It should be noted that the CHANGE is not a direct transfer and any code after this statement and prior to BLOCK_END (or BLOCK_CLEAN_UP) will be executed.

In addition to the grammar macros there is available a set of assistance macros designed to allow for simplified references to the MCDS keyboard inputs. These macros are short descriptive terms which may be used to reference the longer HAL names, which, in turn, reference compool data within structures. For example, "KEY" is a replace macro referencing the compool location where UI stores the MCDS code for the key that is to be passed to the control segment. The compool name for "KEY" is "CZ1V_D_KEY_ID $ (D_IND)". These assistance macros are provided to maintain the user's independence of keyboard changes which will be reflected through the macro. The assistance macros are defined as follows:

- KEY

  KEY is an integer single value giving the ID of the MCDS key passed to UI. KEY may be used in conjunction with other macros to determine the identity of a particular entry. (e.g., IF KEY=PRO THEN...) The following defines the macros to be used to test KEY.

  - PRO

    PRO gives the ID for the PRO key as received from the MCDS keyboard.

  - EXEC

    EXEC gives the ID for the EXEC key as received from the MCDS keyboard.

  - ITEM_ENTER

    ITEM_ENTER gives the ID for the ITEM key when data is entered with the ITEM number.

  - ITEM_EXEC

    ITEM_EXEC gives an ID for the ITEM key entry when that keyboard entry has an EXEC key termination.

  - RESUME

    RESUME gives the ID for the RESUME key as received from the MCDS keyboard. (It may be tested only in the SPEC clean up segment for the SPEC or a BLOCK within a SPEC.)

- For the ITEM (See ITEM_ENTER, or ITEM_EXEC above) there is information in addition to the KEY which may be passed. The item number may be tested by "IF ITEM_NO=1 THEN ...". The following lists the assistance macros which may be used in conjunction with the ITEM entry.

  - ITEM_NO

    ITEM_NO gives access to the integer single item number entered with an ITEM key entry. The ITEM_NO is valid for both ITEM key terminations - EXEC or ENTER.

  - ITEM_S

    ITEM_S gives access to the scalar value as a double precision data item entered with an ITEM_ENTER keyboard entry and associated with the current item number.

  - ITEM_I

    ITEM_I gives access to the integer value as a double precision data item entered with an ITEM_ENTER keyboard entry and associated with the current item number.

  - ITEM_O

    ITEM_O gives access to a 32-bit binary field containing any octal item entered with an ITEM_ENTER keyboard entry and associated with the current item number.

- Another group of assistance macros is provided to give the application an indication of current and up coming status. The new OPS number may determined for example by "IF D_NEW_OPS_NUMBER=1 THEN...". The following is a list of these macros.

  - D_OPS_NUMBER

    D_OPS_NUMBER gives the integer single value of the current OPS within the Miscellaneous Application Control Table (MACT).

  - D_MODE_NUMBER

    D_MODE_NUMBER gives the integer single value of the current MODE within the MACT.

  - D_SPEC_NUMBER

    D_SPEC__NUMBER gives the integer single value of the current SPEC within the MACT.

- D_BLOCK_NUMBER

  D_BLOCK_NUMBER gives the integer single value of the current BLOCK within the MACT.

- D_NEW_OPS_NUMBER

  D_NEW_OPS_NUMBER gives the integer single value of the OPS for which a current OPS is being cancelled. This is valid only in the clean up segment.

- D_NEW_MODE_NUMBER

  D_NEW_MODE_NUMBER gives the integer single value of the new mode being requested by the MCDS input for which a current mode is being cancelled. This is valid only in the clean up segment.

- D_DEU_NUMBER

  D_DEU_NUMBER gives the integer single value of the DEU from which a keyboard entry to the control segment has been made.

The grammar macros and all communications with the control segment is through the Miscellaneous Applications Control Table (MACT) which is a structure with several copies. One of these copies is assigned at control segment initiation to the control segment. The index is provided via a variable, D_IND,defined within the grammar macro set. It is initialized for the control segment when the OPS or SPEC macro is executed. Therefore, no macro using D_IND should be referenced prior to execution of the OPS or SPEC macro. If use of this index outside of the control segment is required, it must be provided by application coding procedures.

Table H-1 provides a summary of the grammar key words.

Table H-1 Grammar Key Words

ADVANCE_ON	ITEM_EXEC
BLOCK	ITEM_I
BLOCK_CLEAN_UP	ITEM_NO
BLOCK_END	ITEM_O
CHANGE	ITEM_S
CLEAN_UP	KEY
CLEAN_UP_MODE	MODE
CLEAN_UP_OPS	MODE_CLEAN_UP
CLEAN_UP_SPEC	MODE_END
D_BLOCK_NUMBER	NO_AUTO_ADVANCE
D_DEU_NUMBER	NO_CLEAN_UP
D_IND	NO_CLEAN_UP_MODE
DISPLAY	NO_CLEAN_UP_OPS
D_MODE_NUMBER	NO_CLEAN_UP_SPEC
D_NEW_MODE_NUMBER	OPS
N_NEW_OPS_NUMBER	OPS_CLEAN_UP
D_OPS_NUMBER	OPS_END
DS	PRO
D_SPEC_NUMBER	RESUME
EXEC	SPEC
INIT_BLOCK	SPEC_CLEAN_UP
ITEM_ENTER	SPEC_END

H.3 KEYBOARD MESSAGE PROCESSING

Associated with each display format is a subset of the keyboard messages which are valid user commands during the time the format is displayed at his position. Most of these messages are processed by User Interface programs and are invisible to the application control logic. However, some messages are, by their nature, processed by the applications program. This processing occurs within the control block following the DISPLAY statement, but preceding either the block cleanup segment or the BLOCK END statement.

Table H-2 presents a summary of all available keyboard messages.

Table H-3 is a list of all valid keyboard syntax messages transmitted to the GPC by the DEU.

For simplified access to the keyboard messages and included data parameter, the control segment grammar contains a set of macros for use by the applications programmer (see Appendix H.2). Their use is optional. These macros support the ITEM, PRO, and EXEC messages. The data exchanged between User Interface and the Control Segment is placed in a communications area, the common COMPOOL.

Table H-2. Keyboard Message Summary

Message ID	Name	Processed By
OPS	Operational Sequence	UI
SPEC	Specialist Function	UI
DISPLAY	Display Function	UI
RESUME	Resume	UI
ITEM	Item	Application
PRO	Proceed	Application
EXEC	Execute	Application
ACK	Acknowledge	UI
MSG RESET	Message Reset	UI
FAULT	Fault Summary	UI
CMPTR/CRT	Computer/CRT Bus Configuration	System Control
CMPTR/BUS	Computer/Bus Configuration	System Control

Table H-3. Keyboard Message Syntax

1. OPS ABC PRO (Note 1)
2. SPEC BCD PRO
3. DISPLAY BCD PRO
4. ITEM (A) B +/-XX.XX +/-XXXX +/- ---- E (Note 2)
5. ITEM (A) B EXEC
6. RESUME
7. FAULT
8. ACK
9. MSG RESET
10. PRO
11. EXEC
12. CMPTR/CRT AB EXEC
13. CMPTR/BUS AB EXEC

Note 1. The integers, 0 to 9, are indicated by the letters A, B, C, D and X.

Note 2. E represents the ENTER key.

PART 3

SYSTEM CONTROL

SDDS

System Control

Table of Contents

Table of Contents (cont'd)

1. INTRODUCTION

This document presents the System Control detail design descriptions of modules to be used for the Approach and Landing Test (ALT). The design presented reflects requirements as specified in the Computer Program Development Specification Level A (CPDS), Book 1 (Software), dated July 23, 1976 and all approved Change Requests.

The remainder of the INTRODUCTION, Section 1, discusses the documents purpose and Section 2 explains the format used to describe each module.

The PROGRAM DESCRIPTIONS Section, Section 3, contains the ALT detailed design for each module in System Control.

Appendices contain additional information concerning the System Control design:

Appendix A contains the System Control requirements - Detailed Design Cross Reference.

Appendix B contains the System Control resource allocation summary.

Appendix C contains the System Control module list.

Appendix D contains a table of error conditions and the default system action to be performed by System Control modules.

Appendix E contains the Data Descriptors Table.

Appendix F contains a list of OPS-dependent modules.

Appendix G contains a time line illustrating some of the functional capabilities of System Control.

A detailed knowledge of the HAL/S Realtime Statements and a general knowledge of the AP-101 flight computer are assumed. The following list of documents contain information regarding these topics:

- HAL/S Language Specification
- Interface Control Document: HAL/FCOS
  IBM File #6246156A
- Advanced System/4 Pi Model AP-101 Computer Software Systems Manual,
  IBM File #62280045
- User Interface, Release 4, User's Guide

1.1 PURPOSE

The Detail Design Specification provides explicit module descriptions of the implementation of the System Control requirements. The detailed breakout of the modules, the detailed logic flows and the detailed data descriptions provide the final level of design description preceding the actual program listing.

1.2 SCOPE

System Control is the software that supports system software functions related to initialization, reconfiguration, and other tasks necessary for system operation that are not specifically functions of the application software. It operates at the request of the system user to accomplish its functions. The services of other processes in the System Services segment of the processing hierarchy are utilized in accomplishing this process. It is divided into three major areas:

System Initialization provides the execution of system software to place the GPC into a state in which normal operation may begin. The configuration may be a single GPC or a multi-GPC set.

System Reconfiguration performs the system services of coordinating changes in the makeup of the DPS. Included are changes in the GPC set, their main memories, and IOP channels as well as supporting the display of configuration status and the inputs of the user to change the status. The operational sequence for system idling as a major function and as an entire GPC is included in this set of system services. User input requests for reconfiguration are performed by using other services of system software when possible.

System Specialist Functions are the set of specialist functions which provide system services and information during any operational sequence and from any user position regardless of the major function switch setting. Because of their commonality across major functions they are normally permanently resident in memory.

2. FORMAT EXPLANATION

The individual detail design description of each module consists of a Module Writeup which contains interfaces with other modules, a process description and any limitations the processing may have. The input and output to the module is presented in the Module Data Table and Module I/O Table. Finally, the detailed design is presented in the form of a structured control flow in the Module Control Flow.

2.1 MODULE IDENTIFIERS

Several methods for referring to a program have been defined. The program is identified by its HAL/Assembler name, English_Equivalent_Name or its three digit ID.

2.1.1 English_Equivalent_Names

An English_Equivalent_Name is provided for each HAL/S or assembly language program. Each name conforms to the following standards.

a. It shall be uniquely defined.

b. It shall not contain other than accepted English abbreviations, acronyms defined in the Level A CPDS, or the acronyms Appendix to the SDDS.

c. It shall be concise.

d. It shall be indicative of its use.

e. Underscore connectors shall be used between words within the English name. Each new word will begin with a capital letter.

Example:

1. Error_Logging_Routine

2. LDB_Message_Router

Note that the same standards apply to the "ITEM" name of I/O parameters (see Appendix E).

2.1.2 Module Identification (ID) Codes

All programs, procedures, and segments (where segments are convenient modularizations of complex control flows) are identified by a numerical ID of the format XXX.XX. These codes are utilized in the Source/Destination section of the Data Table (see Section 2.3) and in the appropriate parts of the Control Flow (see Section 2.5).

The module ID's conform to the following standards:

a. A unique three digit code shall be invented for each Program and Procedure. The numbers are assigned in the following way:

FCOS	100-199	Process Management
	200-299	I/O Management
	300-399	Configuration Management
UI	400-699	
SC	700-999	

b. Segment ID codes shall consist of the same three digit prefix as the module it is a part of, followed by a decimal and a two digit number (e.g., 112.01). Segments are numbered consecutively even if they are referenced only by another segment.

c. All ID codes are used consistently throughout the entire document.

2.2 MODULE WRITEUPS

The title at the beginning of each section consists of the section number, followed by the module English_Equivalent_Name, the module HAL/S or assembly language name and the module ID.

Example:

X.X.X.X. English_Equivalent_Name (MODULE1) (XXX)

Each module writeup is divided into the parts described below.

Function - This paragraph provides a very general description of the functions performed by the module.

a. Control Interface - This paragraph identifies all known users of the module and its form of invocation. Priority and rate parameters are specified as symbolic parameters defined in Software Awareness Memo (SAM) 10.

Control Interface
1. Form of Invocation
2. Known Users

Example:

1. SCHEDULE DMI_MCDS_IN PRIORITY (PRIO_DMI)
   REPEAT EVERY TIME_DMI;

2. (a) CALLed by (ID) English_Equivalent_Name
   (HAL/S_NAME)

   (b) SCHEDULEd by (ID) English_Equivalent_Name_2
   (HAL/S NAME2)

(c) Event English_Equivalent_Name (HAL/S NAME) set by (ID) English_Equivalent_Name_3 (HAL/S_NAME3)

b. Input - In most cases this is a reference to the module data table. However, if special interfaces need to be described they are described here. If the module is a control segment (an Operational Sequence or Specialist Function) the special grammar macros must be used. They are listed here. For a reference to a table "See Table X.X.X.-1" is used.

c. Process Description - This is a narrative which functionally describes the processing performed by the module as depicted in its Control Flow. If data parameters are referenced, they are referenced by their English_Equivalent_Names. It includes a reference to the Control Flow Figure number of the form: "The control flow for this module is shown in Figure X.X-X".

d. Outputs - In most cases this is a reference to the module Data Table. However, if special interfaces need to be described, they are described here. For a reference to the table "See Table X.X.X.X-1" is used.

e. Module References - This section identifies all other modules which this module references. Each module is identified by a line consisting of the three digit ID, English_Equivalent_Name and HAL/S or Assembly language name. Modules are CALLed or SCHEDULEd. It also includes the FCOS service routine name of FCOS Supervisor Calls that are invoked.

Example:

1. (001) Reset_Event_Processor (FPMRESET) is CALLed.
2. (002) Lights_and_Alarms_Processing (DLA_LIGHT_ALARM_PROC) is CALLed.

f. Module Attributes

This section specifies the HAL compilation type of the module, whether it is a program, external or internal procedure, function, or task. A program is the highest level unit of compilation and can only be invoked by a SCHEDULE. An external or internal procedure is a subroutine outside or inside the program scope, respectively, and can only be invoked by a CALL. A task is a block of code which is invoked by a SCHEDULE. A function is an operation invoked by the appearance of the name in an expression.

g. Template References - Specifies all templates referenced in the module. Each template is identified by an English_Equivalent_Name and a HAL/S name.

1. English_Name_1 (HALS_NAME_1
2. English_Name_2 (HALS_NAME_2)

h. Error Handling

This section describes the error handling techniques employed by this module. For each error detected by the module an explanation of the error and the action taken by the module is provided. If the module outputs and error message via the Annunciation Function the error is briefly discussed and the error message number (see Appendix D) is enclosed in parenthesis.

i. Constraints and Assumptions - This section describes the programming limitations to which this module is subject and delineates the assumptions which are reflected in the design of this module.

j. Detailed Implementation - This section is used to provide a text extension of the logic statements which are usually enclosed within a processing block of a Control Flow. These logic statements may be in HAL/S form, but assembler language code is not allowed. The process block contains a reference number which relates to a number in this section of the module writeup.

2.3 MODULE DATA TABLE

All data input and output to a module or the hardware and internal parameters necessary for the understanding of a processing event (i.e., all data referenced in the control flow, narrative, or detailed implementation section) are itemized in a tabular format (see Figure 2.3-1). Each item in the table is explained below.

1 Name - The module English_Equivalent_Name along with the HAL/S or assembler name in parenthesis.

Example:

MCDS_Input_Processor (DMI_MCDS_IN)
Configuration_Management_SVC_Service_Routine (FCMSVC)

2 Item Number - Items are sequentially numbered in the general order of their appearance in the associated control flow.

3 Item - The English_Equivalent_Name assigned to the data in the Data Descriptors Table (see Appendix E).

4 Description ID - The ID assigned to the data in the Data Descriptors Table (see Appendix E).

5 Activity Type - This column denotes the usage of the parameter. Codes are:

I - An external memory location referenced by the module.
O - An external memory location changed by the module.
L - Local variable
Z - COMPOOL constant
T - Temporary Variable (Variables used only within a HAL DO group)

Any number of activity types that apply are specified in this column.

6 Source - A listing of the three digit ID's of all modules updating the parameter or hardware source of the parameter. If more than four modules update the parameter, a reference to the Data Descriptors Appendix is inserted instead of the list. The three digit ID's are listed in numerical order.

7 Destination - A listing of the three digit ID's of all modules referencing the parameter or the hardware source to which the parameter is being written. If more than four modules reference the parameter, a reference to the Data Descriptors Appendix is inserted instead of the list. The three digit ID's are listed in numerical order.

8 HAL/S - The HAL name is used in coding. (Include the HAL equate name in parenthesis where applicable). If no name is assigned the column is blank.

9 MML - The Master Measurements List number in the requirements source. If not specified the column is left blank.

10 D - An 'X' is placed in the column if the parameter is available to be downlisted.

11 C - An 'X' is placed in the column if the parameter is available to be displayed on a CRT.

2.4 MODULE I/O TABLE

The module writeup contains a table specifying the data passed to FCOS in each I/O request. The format of the table and explanation of the parameters is found in Appendix G in Part 1 (FCOS) which contains a table of the I/O request definitions for each device and OP code. If the I/O is of the pre-initialized type the table is not included with the module writeup. Reference to this table or FCOS Appendix G is included in the control flow where the I/O SVC is issued.

2.5 MODULE CONTROL FLOW

2.5.1 Flow Charts

System Software modules are documented by Structured Control Flows. Where possible/practical, the structure of the flow follows the structure of the program listing. Each page of flowcharting presents a complete picture of a primary processing objective with subsequent expansions of detail appropriately referenced. The expansion is accomplished either through a segment that is presented in detail on later pages of the control flow or through a reference to the Detailed Implementation section of the module writeup. The first page of each flow contains one elliptical block specifying "Enter" and a block specifying "Return". Logic flow is generally from top to bottom and left to right thus no arrowheads are required on connecting lines. See Figure 2.5-1 for the Control Flow legend.

The initial page of the control flow is titled with the same title as the module writeup; that is English_Equivalent_Name and HAL/S or assembler name.

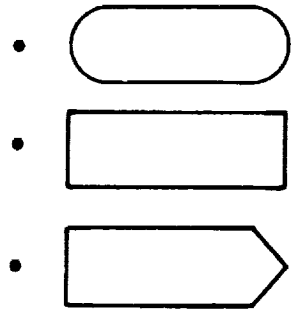
Example:

Figure X.X.X.X - English_Equivalent_Name (HAL_NAME)

Subsequent pages of a flow have the same title as the initial page without the HAL/S or assembler name. Additionally, they have a subtitle consisting of an English_Equivalent_Name for the segment and the ID defined in the process box which references the segment.

Example:

Figure X.X.X.X-2 English_Equivalent_Name
English_Equivalent_Segment_Name (XXX.1)

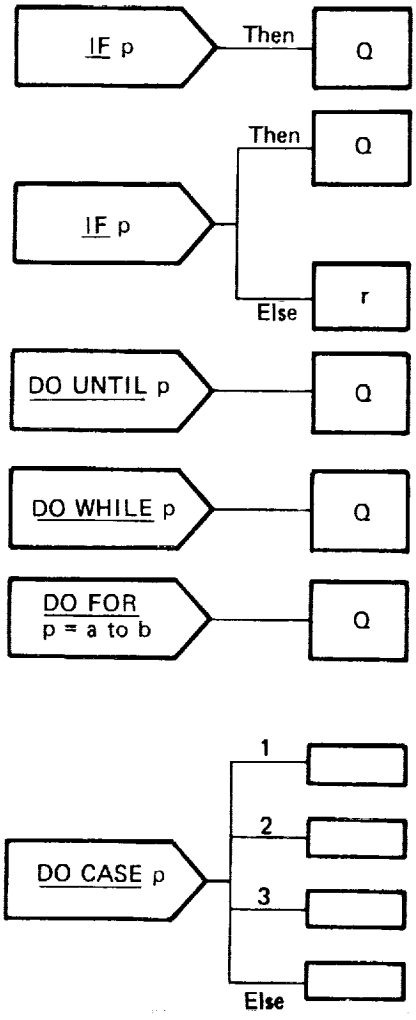


Terminal - identifies Enter/RETURN point (such as HAL/S executable units of compilation and internal procedures and functions)

Mathematical statement/process control block which sometimes references another flow for a lower level of detail

Decision statement block

Documentation Examples



If p true then do Q, and return in-line*. (The word THEN is required.)

If p true then do Q and return in-line*, else do r, and return in-line*.

Do Q until p true, then return in-line*.
Note: The decision on p is made after doing Q.

Do Q while p true, then return in-line*.
Note: The decision on p is made before doing Q.

Do Q while p increments from a to b, and return in-line*.

Do one of specified processes based on value of p, and return in-line.* NOTE: Some identification of each case choice is given on the line leading into box, such as an index number, or the word ELSE (for the branch taken if the case index is outside its valid range).

* - Return in-line means return to the statement immediately below the decision. If there is no such statement, retrace path to the last previous decision and repeat this logic.

Figure 2.5-1. Control Flow Legend

3. SYSTEM CONTROL PROGRAM DESCRIPTIONS

The System Control Function performs system services in a special category of data processing which is neither a function of FCOS nor a function of User Interface. These functions include the initialization of the system following the loading of a GPC via IPL or other means, support for OPS 0-00 functions, system reconfiguration control, and the set of Specialist functions which are system oriented rather than directly related to the major functions of Systems Management, Guidance, Navigation, and Control, or Payload Management.

The following figure presents the functional design of System Initialization, System Reconfiguration, and System Specialist Functions. Figure 3-1 contains the functional overview hierarchy diagram for System Control.

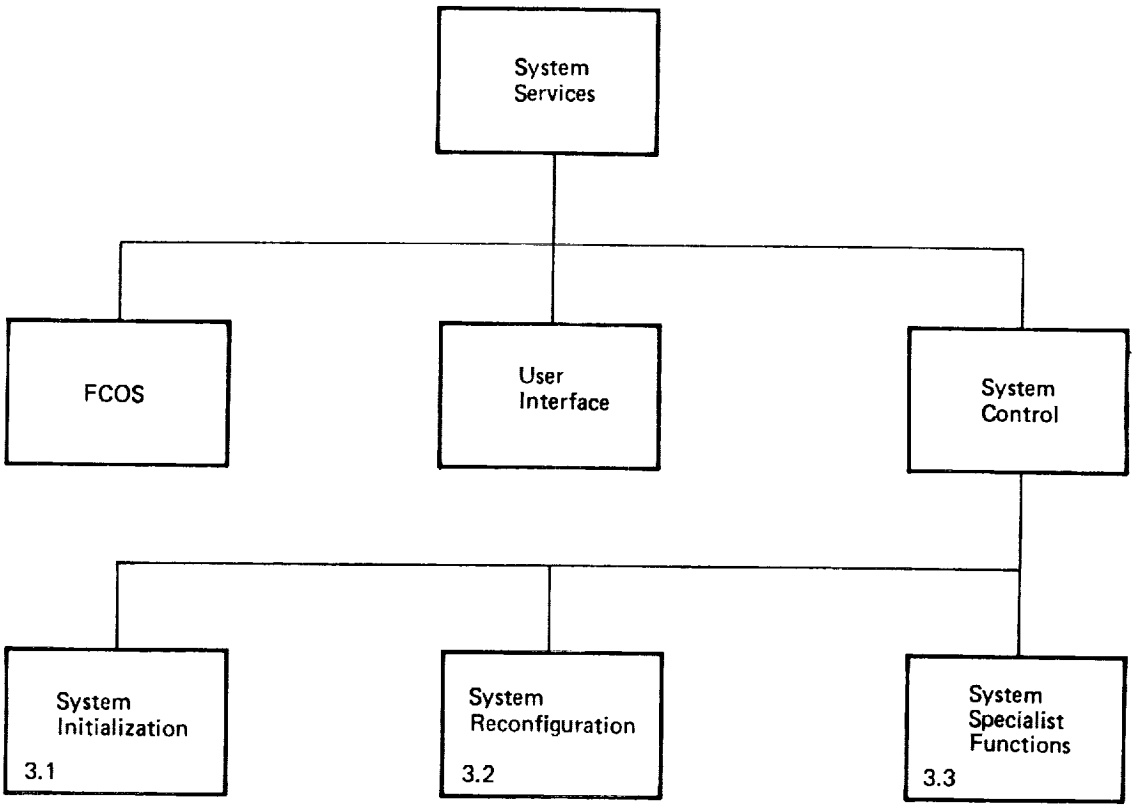


Figure 3-1. Functional Overview Hierarchy Diagram

3.1 SYSTEM INITIALIZATION

The System Initialization software initiates the execution of User Interface and System Control software and places the GPC into a state from which normal operation may begin. This is accomplished through a series of subfunctions designed to start only one GPC at a time. In the case where a redundant set of GPC's is to be initialized, all subsequent GPC's following the first are placed in a secondary mode and await a command by the first GPC to load the program overlay and initiate the first Operational Sequence (OPS). All GPC's activated are synchronized at a common sync point. In the case of a simplex configuration initialization, the GPC is synchronized with any other currently executing GPC set at the common sync point. Figure 3.1-1 shows the hierarchial diagram for System Initialization.

- Initial Start Sequence provides the interface with FCOS to initialize GPC operations. It distinguishes the first from the secondary GPC's and executes the startup algorithm accordingly. If successful, the READY talkback is turned on following the initiation of User Interface, System Interface Processing and the OPS 0-00 control segment.

- System_Interface_Processor (AIE_SIP) provides the control for system-wide processes which are required to operate at the same time but are not necessarily members of the redundant set. These GPC's are members of the common set. The processes are phased to execute at periods of the minor cycle when critical applications processes are not active.

- DEU_Loader (AIG_DEU_LOADER) provides the capability to load a DEU from mass memory when initiated by user action at specified times. The loading sequence requires a series of data transfers to the DEU and while the loading sequence is in progress DEU polling and message processing are under this module's control.

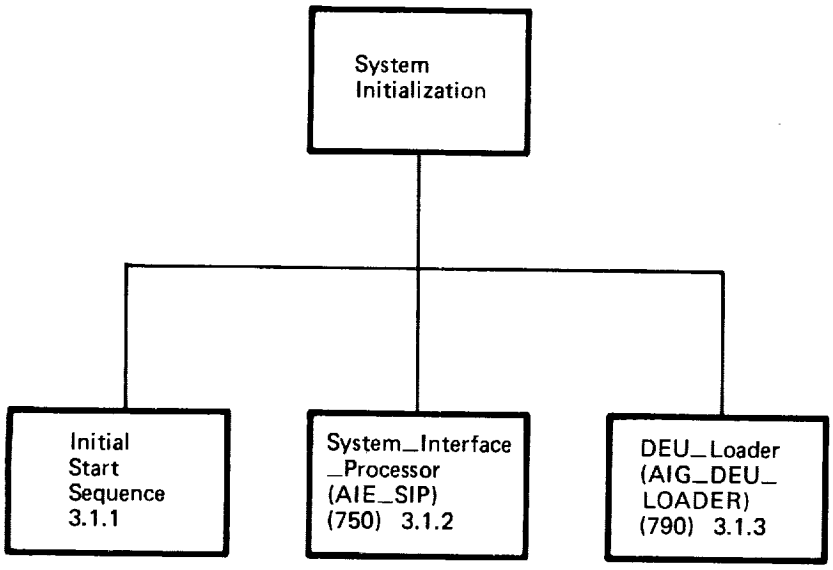


Figure 3.1–1. System Initialization Hierarchy

3.1.1 Initial Start Sequence

The Initial Start Sequence software is responsible for initializing GPC operations in either a simplex or redundant set. Figure 3.1.1-1 shows the hierarchial diagram of the Initial Start Sequence.

- GPC_Locator (AIB_GPC_LOCATOR) initiates the GPC execution following FCOS initialization by identifying the active GPC set, configuring buses appropriately, and setting initial conditions for subsequent processing.

- GPC_Startup (AIR_GPC_STARTUP) invokes the necessary processes for GPC initialization, initiates the Idle Operational Sequence and concludes the software IPL or power up sequence by turning on the READY talkback indicator.

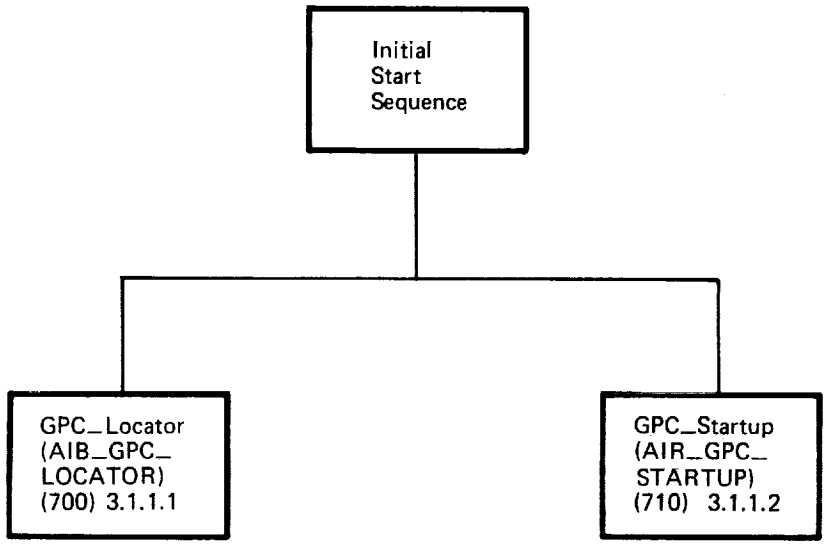


Figure 3.1.1–1. Initial Start Sequence Hierarchy

3.1.1.1 GPC_Locator (AIB_GPC_LOCATOR) (700)

To be provided.

3.1.1.2 GPC_Startup (AIR_GPC_STARTUP)(710)

The functions of this procedure are to invoke the necessary processes (in User Interface) for GPC initialization, initiate GPC OPS 0-00, and conclude the normal startup sequence by turning on the GPC ready indicator.

a. Control Interface

1. CALL AIR_GPC_STARTUP;
2. CALLed by (700) GPC_Locator (AIB_GPC_LOCATOR)

b. Input - See Table 3.1.1.2-1

c. Process Description - This procedure first tests the Downlist_Formatter_Enabled_Flag. If it = 1 (enabled) then a CALL to (660) Downlist_Formatter (DCD_DOWNLIST) is issued. Next, a CALL to (815) Force_OPS_0 (AOP_FORCE_OPS_0) is issued to cause (810) Idle_Operational_Sequence (ARB_IDLE_OPS) to be scheduled in all major functions. Then the appropriate UI and System Control Programs are scheduled:

(500) User_Interface_Control_Supervisor (DMC_SUPER)
(400) MCDS_Input_Processor (DMI_MCDS_IN)
(620) Cyclic_Display_Processor (DCI#CYC)
(800) GPC_Switch_Monitor (ARA_GPC_SWITCH)

A test is made to determine if this GPC is already in RUN mode which means a transition from HALT to RUN was made. If the Discrete_Bit_Run_Mode indicates that the GPC mode is RUN, then ERROR message OS010 is annunciated. Finally a discrete out SVC request is made to turn on the GPC Ready Talkback and then control is returned to the calling program. The control flow for this module is shown in Figure 3.1.1.2-1.

d. Ouputs - See Table 3.1.1.2-1

e. Module References -

1. (660) Downlist_Formatter_1 (DCDDWL1) is CALLed.
2. (815) Force_OPS_0 (AOP_FORCE_OPS_0) is CALLed.
3. (500) User_Interface_Control_Supervisor (DMC_SUPER) is SCHEDULED.
4. (400) MCDS_Input_Processor (DMI_MCDS_IN) is SCHEDULED.
5. (620) Cyclic_Display_Processor (DCICYC) is SCHEDULED.
6. (800) GPC_Switch_Monitor (ARA_GPC_SWITCH) is SCHEDULED.
7. (675) Annunciation_Macro_Interface (DMA_MAC) is CALLed.
8. (340) Miscellaneous_CM_Request_Processor (FCMSVC) is CALLed.
9. (101) Process_Scheduler (FPMSCHED) is called.
10. (665) Downlist_Formatter_2 (DCDDWL2) is CALLed.

f. Module Attributes - External Procedure

g. Template References -

1. User_Interface_Control_Supervisor (DMC_SUPER)
2. MCDS_Input_Processor (DMI_MCDS_IN)
3. GPC_Switch_Monitor (ARA_GPC_SWITCH)
4. Force_OPS_0 (AOP_FORCE_OPS_0)
5. UI_FCOS_Shared_Compool (CZ2_COMMON)
6. UI_Section_Of_Common_Compool (CZ1_COMMON)
7. Downlist_Formatter (DCD_DOWNLIST)
8. FCOS_Compool (FCMCOM)
9. Annunciation_Compool (CDL_ANNUN)
10. Annunciation_Macro_Interface (DMA_MAC)

h. Error Handling - a test is made to determine if the Discrete_Bit_Run_Mode shows that the self GPC is in the RUN mode (rather than STANDBY), and if it is then error message OS010 (see Appendix D) is annunciated, indicating a mode switch transition from HALT to RUN.

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.1.1.2-1

NAME GPC_Startup (AIR_GPC_STARTUP_)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	Downlist_Formatter_Enabled_Flag	I010.16		560,700 820,880	710 750	CZ2B_STATUS			
2	Time_of_SIP	J080		700	495,700, 710,830	CZ1V_A_TSIP			
3	Discrete_Bit_Run_Mode	&005.03				CZ2B_DIA1			

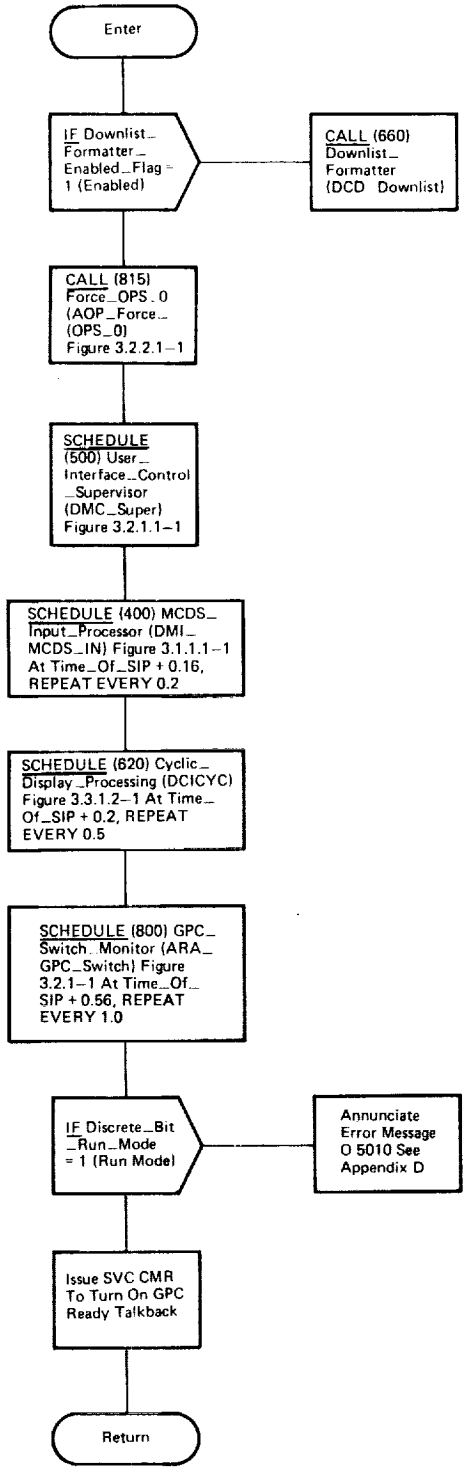


Figure 3.1.1.2-1. GPC_Startup (Air_GPC_Startup)

3.1.2 System_Interface_Processor (AIE_SIP) (750)

This module provides the control of system wide processes which are required to operate at the same time and in coordinated manner. It operates as the highest priority process to permit multiple GPC to communicate in an efficient manner and participate in GPC and MTU redundancy management processing.

a. Control Interface

1. SCHEDULE AIE_SIP(PRIO-AIE)

2. SCHEDULEd by (710) GPC_Startup (AIR_GPC_STARTUP)

b. Inputs - See Table 3.1.2-1.

c. Process Description - First, a check is made of New_GPC_Flags to see if a new GPC is trying to enter the common set. If any GPC bit is set, an ICC message request bit is set to inform all other GPCs currently in the common set. Otherwise a Lights_and_Alarms_Counter is decremented that will be tested later in the module to control the frequency for testing a Lights and Alarms request event. Also, a test is made of fault message requests bits to see if the Fault_Message_Scan routine must be called.

Secondly, the module calls the ICC_Message Collector to close out ICC message handling for the current SIP cycle. Then, FCOS flags are scanned to determine if GPC status messages must be sent to other GPC in the common set.

Finally, three values are tested to determine if other processing must be invoked.

1. If Lights_and_Alarms_Processor_Event is set and time to test- Issue Call to Lights_and_Alarm_Processing.

2. If Downlist is enabled - Call Downlist_Formatter

3. If enabled and MTU read complete - Issue SVC for MTU Redundancy Management.

The control flow for this module is presented in Figures 3.1.2-1 and 3.1.2-2.

d. Outputs - See Table 3.1.2-1.

e. Module References

1. (480) ICC_Message_Collector (DIM_ICC_COLLECTOR) is CALLed.
2. (485) ICC_Message_Collector_For_SIP (DIN_ICC_SIPCOLL) is CALLed.
3. (490) ICC_Message_Router (DME_ICC_ROUT) is CALLed.
4. (660) Downlist_Formatter_1 (DCDDWL2) is CALLed.
5. (670) Fault_Message_Scan (DMR_FMS) is CALLed.
6. (690) Lights_and_Alarm_Processor (DLS_LIGHT_ALARM_PROC) is CALLed.
7. (665) Downlist_Formatter_2 (DCDDWL2) is CALLed.

f. Module Attributes - PROGRAM

g. Template References

1. UI_Section_of_Common_Compool (CZ1_COMMON)
2. UI_FCOS_Shared_Compool (CZ2_COMMON)
3. FCOS_Compool (FCMCOM)
4. Annunciation_Compool (CDL_ANNUN)

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.1.2-1

NAME System_Interface_Processor (AIE_SIP)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	New_GPC_Flag	I440	I,O	700 750	750	CZ2B_NEW_GPC			
2	ICC_Status_Flags	I320	I,O	See APP. E	See APP. E	CZ2B_ICC_FLAG			
3	Lights_and_Alarms_Counter	I670	I,O	700 750	750	CZ2V_LAP_CNT			
4	Fault_Message_Bit	T110	I,O	675 750	670 750	CDL_MSG_BIT			
5	FMPT_HDR_Bits	T070	I	See APP. E	See APP. E	CDL_FMPT_HDR			
6	Sum_Word	I610.01	O	750	375	CZ2V_SUM_WD			
7	Sum_Word	J110	I	GPP 750	750	CZ1B_G_SUM_WD			
8	Load_Status	X068	L	750	750	AIEB_STAT			
9	Word_Index	X072	T	750	750	J			
10	Half_Index	X071	T	750	750	I			
11	Temp_Test_Flags	X070	L	750	750	AIEB_TST			
12	Bit_Index	X073	T	750	750	K			
13	ICC_MSG_CNT	X069	L	750	750	AIEV_MSG_CNT			
14	ICC_Buffer_Pointer	I430	I	480,485 490	480, 485 750	CZ2V_ICC_BUF_POINT			
15	ICC_MSG_Buffer	X067	L	750	750	AIEB_ICC_MSG			
16	ICC_MSG_Header	X077	L	750	100 750	AIEV_ICC_HDR			
17	ICC_BUF_Index	X074	L	750	750	INDX			
18	GPC_ID	#001	I	300	See APP. E	TFCMID			
19	Common_Set_Mask	I010.12	I	See APP. E	See APP. E	CZ2B_CS	See App. E	X	
20	ICC_REQUEST	X076	L	750	100	AIEV_ICC_SN			

DATA TABLE 3.1.2-1 (Cont'd.)

NAME System_Interface_Processor (AIE_SIP)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
21	ICC_Request_Buffer	X076.16	L	750	100	AIEV_ICC_SN.DLOC			
22	ICC_Buffer	I610	I,O	490,750, 800,830	See App. E	CZ2V_ICC_BUF			
23	ICC_Request_WD_Count	X076.15	L	750	100	AIEV_ICC_SN.DWDCT			
24	ICC_Buffer_Pointer	I430	I	480,485 490	480 485,750	CZ2V_ICC_BUF_POINT			
25	Light_Alarm_Processor_Event	T060	I	670 680,685	750,690	CDL_LAP_EVT			
26	LAP_Decrement_Valve	T040.08	I		690 750	CDL_LAP_DEC			
27	GST_Status_Flags	I010.14	O	See App. E	See App. E	CZ2B_STATUS		X	X
28	TMP_Enable_Indicator	#002.9	I	101 720	101 750	TFCMFLAG			
29	TMP_MTU_I/O_Completion_Event	#014	I	750	750	TFCMTUEV			
30	MTU_RM_SVC	X075	L	750	100	AIEV_MTU_SN			

Table 3.1.2-2
SYSTEM_INTERFACE_PROCESSOR

DEVICE	DEVICE ID	OP CODE	SVC NO.	SYNC TYPE	COMFAULT STATUS	ICC TYPE	TIME TAG	WAIT	PROT.	STATUS	INPUT OUT-PUT	MAJ. FUNC. ID	I/O PRIOR-ITY	WORD COUNT	BUFFER	EVENT	BUS NOMINAL
SSIPICC	3	N/A	24	CS	NC	YES	NO	YES	YES	YES	N/A	0	255	8,16, 32,64, or 128	C22V_ICC		GPC_ID

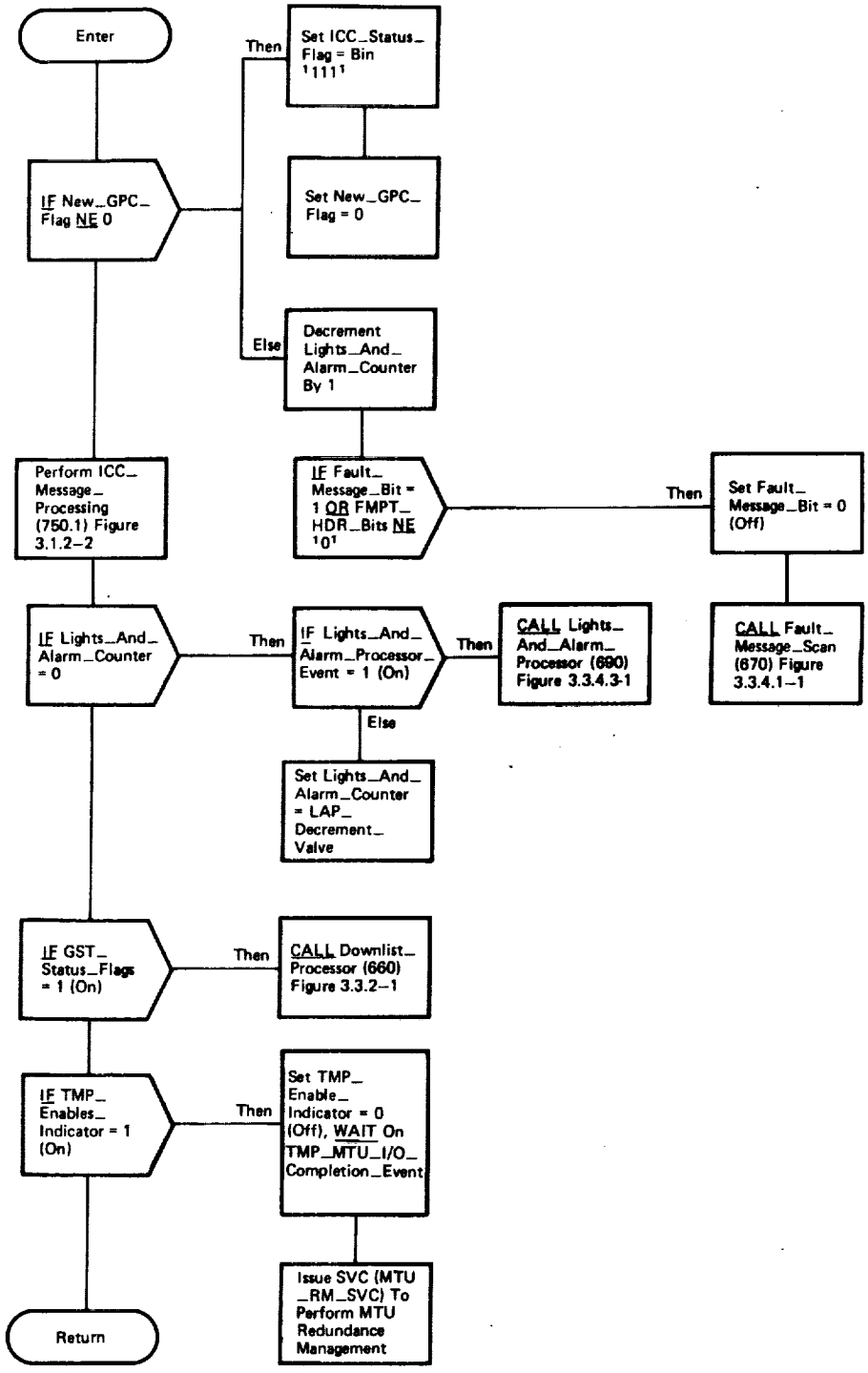


Figure 3.1.2–1. System_Interface_Processor (AIE_SIP)

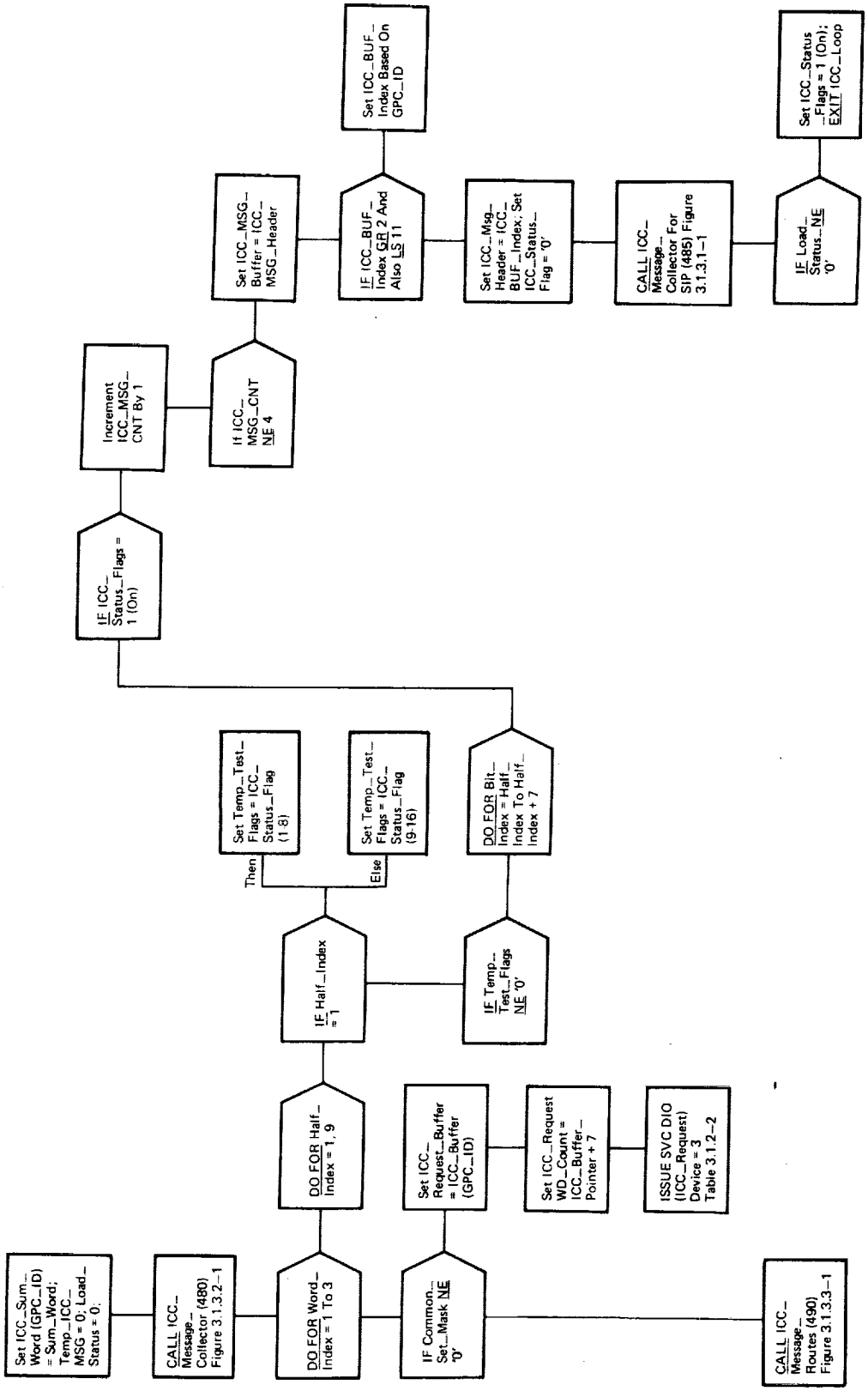


Figure 3.1.2-2. System_Interface_Processor ICC_Message_Processing (750.1)

3.1.3 DEU_Loader (AIG_DEU_LOADER) (790)

AIGDEULO provides the capability to load the DEU when the GPC is in OPS 0-00 following an IPL or is operating major function SM 8.

a. Control Interface -

1. SCHEDULE AIG_DEU_LOADER PRIORITY (PRIO_AIG)
2. SCHEDULED by (405) MCDS_Message_Processor (DMM_MCDS_PROCESS)

b. Input - See Table 3.1.3-1

c. Process Description - The DEU_Loader determines which DEU is to be loaded from the DEU_LOAD_TABLE. A DEU load is retrieved from mass memory and buffered in DEU_MEMORY_FILL_BUFFER. Blocks are transferred to the DEU via "memory fill" messages. A delay of 18 ms between each block transferred is impossed so that the DEU may transfer the block to its proper location. The last block transferred to the DEU is loaded into low core (location 2, word count 250_{10}) with location 29_{10} containing the DEU ID code. Following this fill and a delay of 300 milliseconds, a MODE status and BITE status request are issued to determine if the IPL was successful. An error detected at any point in the sequence results in termination of the process and the issuance of an error message. The user may try the IPL again at his descretion. The control flow for this module is presented in figure 3.1.3-3.

d. Output - See Table 3.1.3-1.

e. Module References -

(200) I/O_SVC_Service_Processor (FIOSVC) is invoked via DIO macro.

f. Module Attributes - Program

g. Template Reference -

1. UI_SECTION_OF_COMMON_COMPOOL (CZ1_COMMON)
2. UI_GENERAL_COMPOOL (CDM_UI_COMPOOL)
3. DEU_MASS_MEMORY_READ_COMPOOL (CAB_COMPOOL)
4. ANNUNCIATION_MACRO_INTERFACE (DMA_MAC)

h. Error Handling - The following error checks are performed:

1. Read Mass Memory, Memory Fill, Mode Status, and Bite Status SVC's - FCOS return status is checked for I/O errors which may result from the execution of the SVC.

2. Mode Status Flags - Same as stated below under assumptions.

3. Bite Status Flags - Same as stated below under assumptions.

If any of the above error conditions occur, the CRT_STATUS_FLAG2 (I/O suppress) will not be reset to inhibit display updates and a system error message will be output (I0025).

i. Constraints and Assumptions -

1. Positive indication of good IPL will be determined as follows:

a. Mode Status Flags

- DEU_BITE_STATUS_AVAILABLE = 1 (enabled)
- DEU_INITIALIZATION_REQUIRED = 0 (disabled)

b. BITE Status Software Register Flags

- DEU_INITIALIZATION_PERFORMED = 1 (enabled)
- DEU_CHECKSUM_ERROR = 0 (disabled)

c. BITE Status Hardware Register 1 Flags

- DEU_IPL_PERFORMED = 1 (enabled)
- DEU_IPL_ERROR = 0 (disabled)
- DEU_IPL_CIRCUIT_ERROR = 0 (disabled)

2. The CRT_STATUS_FLAG2 (I/O supress) will not be reset to enable display updates if the IPL of the DEU fails.

DATA TABLE 3.1.3-1

NAME DEU_LOADER (AIG_DEU_LOADER)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	DEU_LOAD_TABLE	J068	Z	405 790	790	CZ1B_D_DLT			
2	DEU_MESSAGE_HEADER	J060.02	Z	See APP.	E	CZ1B_D_DIT_MSG_HEADR	See APP. E		
3	DEU_BITE_STATUS_MESSAGE	J070.21	Z	405 790	405 790	CZ1V_D_BITE_STAT			
4	DEU_POLL_FLAGS	J062	Z	See APP.	E	CZ1B_DEU_POLL_FLAG			
5	CRT_STATUS_FLAGS	B010.40	Z	See APP.	E	CDMB_MAT_CRT_STAT			
6	DEU_MEMORY_FILL_BUFFER	D260	Z	790	790	CABB_DEU_LOAD			
7	DEU_LOAD_ERROR_FLAG	X001	L	790	790	AIGB_IO_ERR_FLG			

TABLE 3.1.3-2

DEVICE	DEVICE ID	OP CODE	SVC NO.	SYNC TYPE	COMFAULT STATUS	ICC TYPE	TIME TAG	WAIT	PROT.	STATUS	INPUT OUT-PUT	MAJ. FUNC. ID	I/O PRIOR-ITY	WORD COUNT	BUFFER	EVENT	BUS NOMINAL
MMU	10	3	24	RS	No	No	No	Yes	Yes	Yes	I	N/A	160	Vari-able	DEU_MEMORY FILL_BUFFR		6
DEU #1	5	1	24	RS	No	No	No	Yes	Yes	Yes	O	N/A	160	Vari-able	FIODBF1P		6
		2	24	RS	No	No	No	Yes	Yes	Yes	O	N/A	160	1	FIODBF1P		6
		5	24	RS	No	No	No	Yes	Yes	Yes	O	N/A	160	42	FIODBF1P		6
DEU #2	6	1	24	RS	No	No	No	Yes	Yes	Yes	O	N/A	160	Vari-able	FIODBF2P		7
		2	24	RS	No	No	No	Yes	Yes	Yes	O	N/A	160	1	FIODBF2P		7
		5	24	RS	No	No	No	Yes	Yes	Yes	O	N/A	160	42	FIODBF2P		7
DEU #3	7	1	24	RS	No	No	No	Yes	Yes	Yes	O	N/A	160	Vari-able	FIODBF3P		8
		2	24	RS	No	No	No	Yes	Yes	Yes	O	N/A	160	1	FIODBF3P		8
		5	24	RS	No	No	No	Yes	Yes	Yes	O	N/A	160	42	FIODBF3P		8

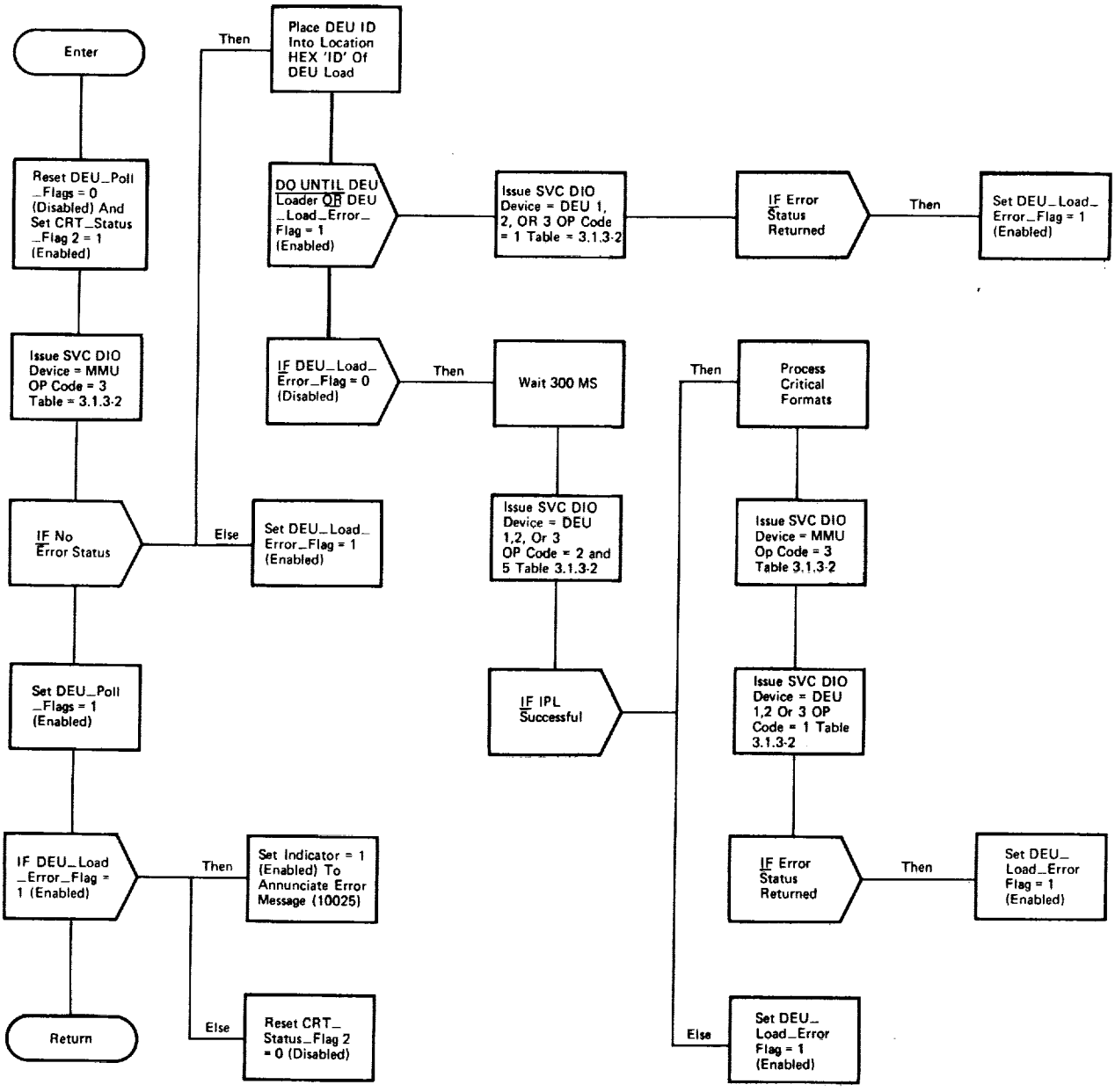


Figure 3.1.3-3. DEU_Loader (AIG_DEU_LOADER)

3.2.1 GPC Switch_Monitor(ARA_GPC_SWITCH) (800)

To be provided.

3.2 SYSTEM RECONFIGURATION

The System Reconfiguration software coordinates changes in the makeup of the DPS. Included are changes in the GPC set, their main memories, and IOP channels as well as supporting the display of configuration status and the inputs of the user to change the status. The operational sequence for system idling as a major function or as an entire GPC is included in this set of system services. It also provides the capability to monitor the GPC mode switch, the BFCS engage discrete and other selected discretes. Figure 3.2-1 shows the hierarchy diagram for System Reconfiguration.

- GPC_Switch_Monitor (ARA_GPC_SWITCH) monitors the GPC mode switch, the BFCS engage switch, and the I/O terminate A and B switch.

- Idle_Operational_Sequence (ARB_IDLE_OPS) is a software control segment which provides all the functions and tasks needed to satisfy DPS operations in the absence of any other operational sequence.

- DPS Reconfiguration software provides the decision logic on when and how to use the tools provided by DPS Configuration Management (FCOS) in changing the DPS configuration. The capabilities result in reconfigured GPC's, main memory reconfiguration, changes to the GPC Reconfiguration Table, and changes to the bus configuration and its table.

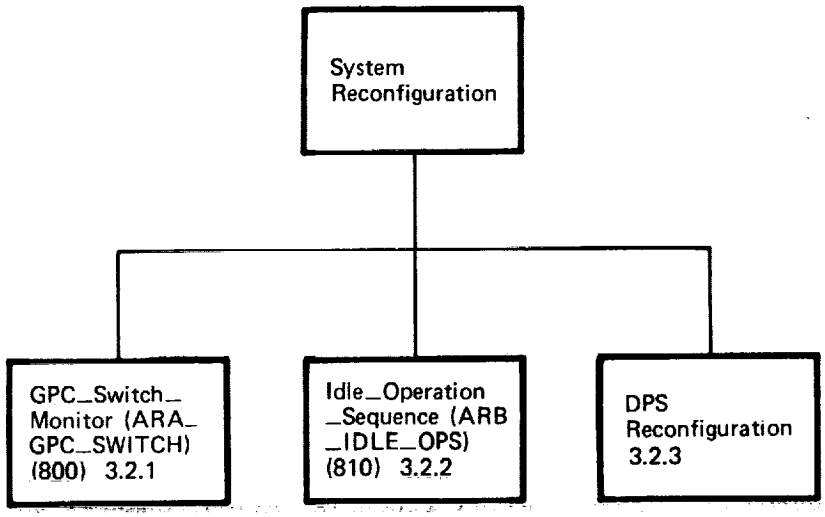


Figure 3.2–1. System Reconfiguration Hierarchy

3.2.2 Idle_Operational_Sequence (ARB_IDLE_OPS) (810)

ARB_IDLE_OPS consists of the the functions and tasks needed to satisfy DPS operations in the absence of any other operational sequence. It provides the user with displayed information and MCDS keyboard entry items which are not available as part of the operational sequences associated with the major functions (except as SPEC 0-00 or MODE 00).

a. Control Interface - IN

1. SCHEDULE ARB_IDLE_OPS PRIORITY (PRIO_ARB);
2. SCHEDULEd by (560) Sequence_Request_Processor (DMC_SEQ_REQ_PROC).

b. Input -

The following User Interface grammar macros supply input data to this module: PRO, ITEM

For a detailed description of these macros refer to appendix H.

Additional imput data descriptions are presented in Table 3.2.2-1.

c. Process Description - Upon entering the control segment the [illegible]nel_Display bit is set = 1 (enabled). This bit is used in conjunction with display support for the format. Following the DISPLAY grammar macro a test is made to see if the keyboard entry passed was a "PRO". If it was not a "PRO" keystroke then the DPS_Configuration_Item_Processor is called, otherwise processing continues.

The first test made when a "PRO" keystroke is detected is to see that the DPS Configuration Monitor was requested as a Specialist function (SPEC 0-00). If that is not the case the error message "Pro Not Compatable with Current Format or Step In Control Segment" (DU015) is issued.

Next a test is made to see if the variable Reconfiguration_Error is equal 101. If it is not equal 101 processing by the control segment is terminated normally via the BLOCK END,MODE END, and OPS END grammar macros.

If Reconfiguration_Error is equal 101, then the Reconfiguration_Table_Index is tested for a non-zero value. If the value is now-zero, then the GPC_Reconfiguration_Complete flag is set = 1 (enabled) and the MACT_Service_Request_Ready_Event is Set. In addition the ICC_Current_Load indicator is set =1 (enabled). Next the Memory_Configuration_Number for the current GPC_ID is stored in the ARB_A_Array_Member variable.

A "DO FOR" loop is set up to test the REC_GPC_OVL_ERR_MSK and the REC_GPC_OVL_TRY_MSK. Whenever any of the bits 1-4 are set = 1 (enabled) the variable ARB_Overlay_Counter is incremented by one (1). The variable ARB_Error_Counter is incremented by one (1) whenever any of the bits 6-9 are found to be = 1 (enabled).

Upon completion of the "DO FOR" loop, a test is made to see if the variables ARB_Overlay_Counter and ARB_Error_Counter are equal. If they are equal, the Simplex_Only bit is interrogated. If the bit is set = 1 (enable), the error message "Pro After MMU Error in Simplex OPS Transition" (OT008) is isssued. Should the bit be set = 0 (disabled) the error message "Pro After MMU Error In An RS OPS Transition when no GPC Received The Overlay" (OT009).

If the two counters are not equal, the REC_GPC_OVL_ERR_MSK is tested. Every bit that is set=1 (enabled) causes a corresponding bit in the GPC_Set word to be set = 0 (disabled). When this process is completed the Reconfiguration_Error variable is set to zero and the Application_Service_Request_Event is set = 1 (enabled). The control segment processing is then terminated normally. The control flow for this module is presented in Figure 3.2.2-1.

d. Outputs - See Table 3.2.2-1

e. Module References -

1. (860) DPS_Configuration_ITEM_Processor (ARF_DPS_CONFIG_ITEM) is CALLed.
2. (675) Annunciation_Macro_Interface (DMA_MAC) is CALLed.
3. (540) Display_Presentation_And_Control (DIS_PLAY) is CALLed.
4. (550) Application_Moding_And_Sequencor is CALLed.

f. Module Attributes - Program

g. Template References -

1.	UI_Section_of_Common_Compool	(CZ1_COMMON)
2.	UI_FCOS_Shared_Compool	(CZ2_COMMON)
3.	DPS_Configuration_ITEM_Processor	(ARF_DPS_CONFIG_ITEM)
4.	FCOS_Compool	(FCMCOM)
5.	Display_Presentation_and_Control	(DIS_PLAY)
6.	Application_Macro_Interface	(DNX_BMS)
7.	Annunciation_Moding_and_Sequencing	(DMA_MAC)
8.	UI_General_Compool	(CDM_UI_COMPOOL)
9.	Annunciation_Compool	(CDL_ANNUN)

h. Error Handling -

The following error messages are output via the Annunciation Function:

1. Pro after MMU error in simplex OPS Transition (OT008)

2. Pro after MMU error in RS OPS Transition when no GPC received the overlay (OT009)

3. Pro not compatible with current format or step in control segment (DU015).

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.2.2-i

NAME Idle_Operational_Sequences (ARB_IDLE_OPS)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	MF_LINE1_DISPLAY	I040.05	O	810		CZ2B_DPS_STATUS			
2	Reconfiguration_Status	I540	I	560,810 820,880	560,810 820,880	CZ2V_REC_XERR			
3	Reconfiguration_Table_Index	I490	I	560 870	560,810 820,880	CZ2V_REC_GRT_INDEX			
4	GPC_Reconfiguration_Completion	J040.13	O	See APP. E	505,540 810	CZ1B_D_FLAGS			
5	MACT_Service_Request_Ready_Events	J160	O	See App. E.	505 810	CZ1E_D_MACT_EVTS			
6	ICC_CURRENT_Load	I320.06	O	820,880	560 810	CZ2B_ICC_FLAG			
7	GPC_ID	#001	I	300	See App. E	TFCMID			
8	ARB_A_Array_Member	X002	T	810	810	ARB_A			
9	ARB_I	X003	T	810	810	ARB_I			
10	Reconfiguration_Error_Mask	I550.01	I	880		CZ2B_REC_OVL_STATUS			
11	ARB_Overlay_Counter	X004	T	810	810	ARB_OVL_CNT			
12	ARB_Error_Counter	X005	T	810	810	ARB_ERR_CNT			
13	Simplex_Only	I210.01	I	810 850	810 850	CZ2B_GRT_GPC_SET			
14	GPC_Set	I210	O	810 860	560,810 820,860	CZ2B_GRT_GPC_SET			
15	Application_Service_Request_Event	B050	O	540,820, 870,880	660, 810, 820	CDME_APP_SERVICE			
16	Reconfiguration_Attempt_Mask	I550.02	O	820	810	CZ2B_REC_OVL_STATUS			

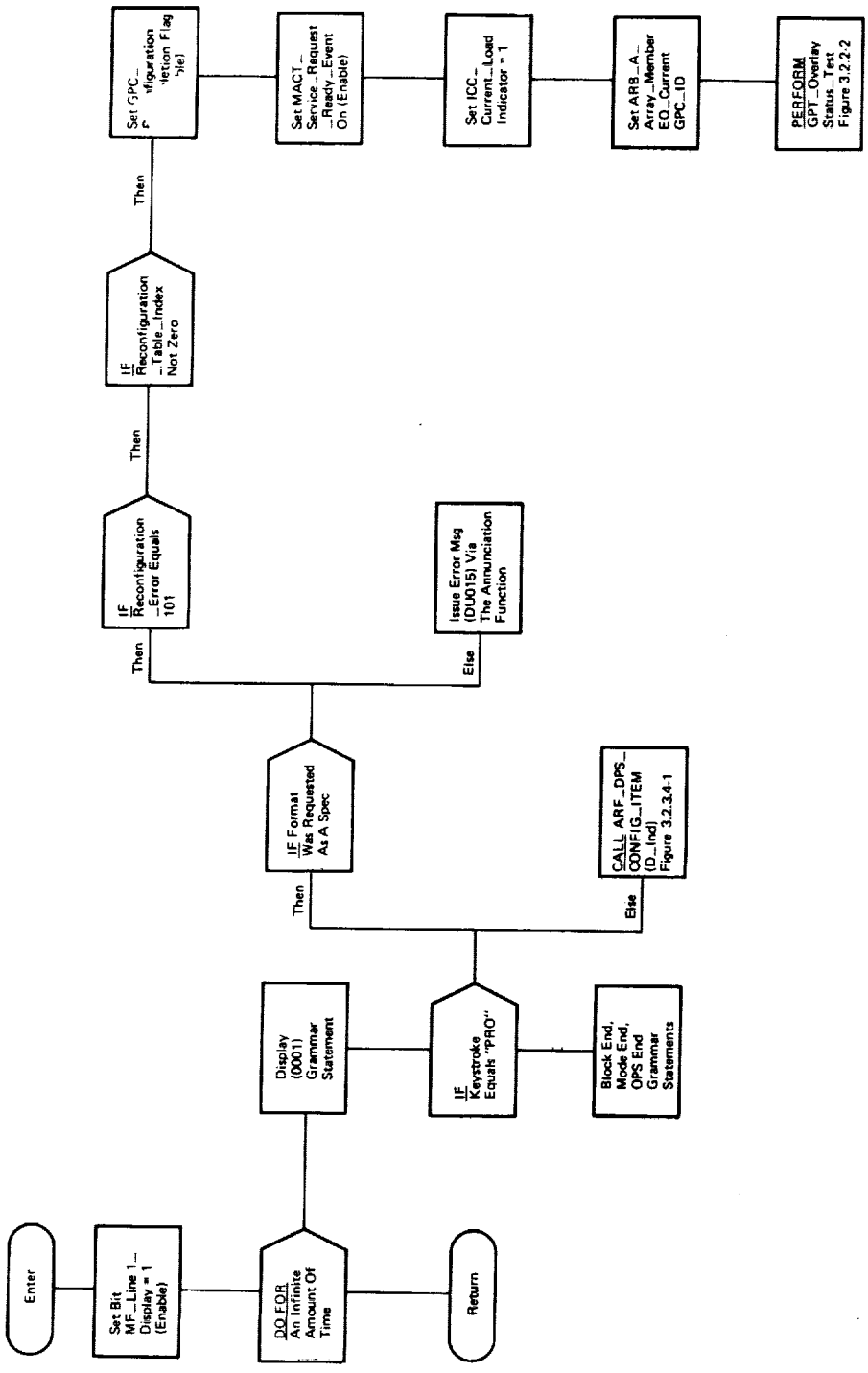


Figure 3.2.2-1. Idle_Operational_Sequence (ARB_Idle_OPS)

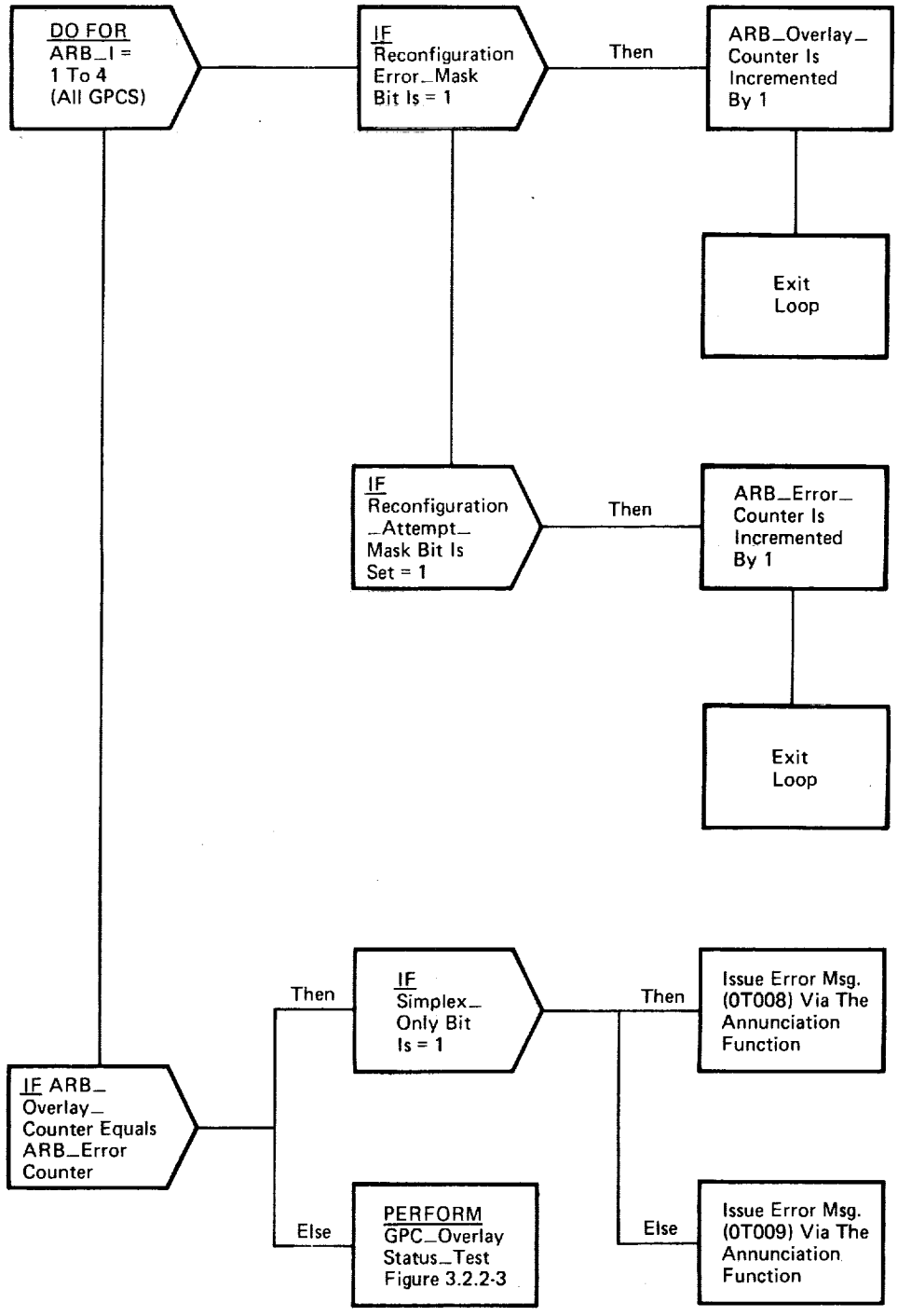


Figure 3.2.2-2. Idle_Operational_Sequence GRT_Overlay_Status_Test (810.1)

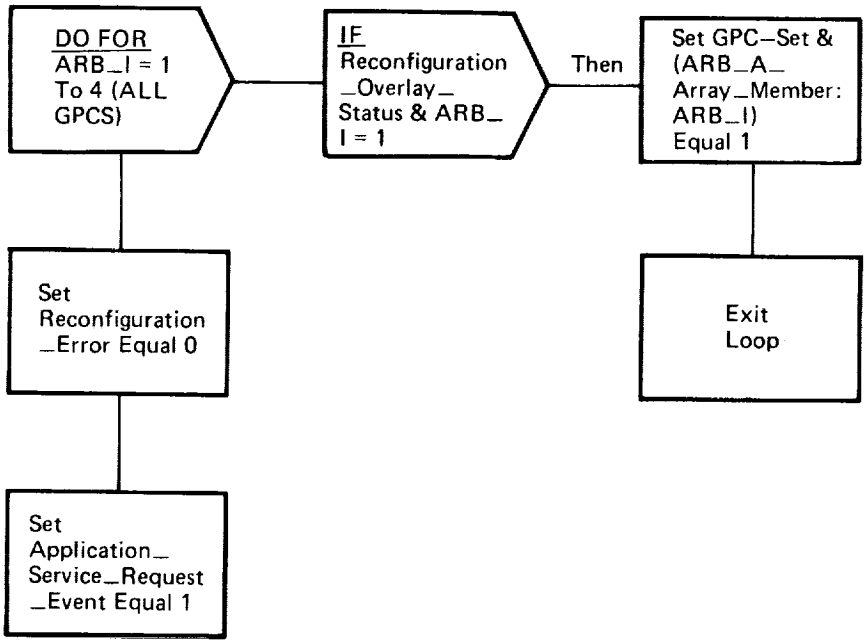


Figure 3.2.2-3. Idle Operational_Sequence GPC_Overlay_Status_Test (810.2)

3.2.2.1 Force_OPS_0 (AOP_FORCE_OPS_0)(815)

The function of this module is to cause the self GPC to be brought into GPC OPS 0-00 (i.e., all major functions in OPS 0-00).

a. Control Interface -

1. CALL AOP_FØRCE_ØPS_0
2. (a) CALLed by (400) MCDS_Input_Processor (DMI_MCDS_IN)
   (b) CALLed by (710) GPC_Startup (AIR_GPC_STARTUP)

b. Input - See Table 3.2.2.1-1

c. Process Description - This procedure first loops to find a DEU_Input_Table which is abailable (i.e., not currently being used by this GPC) to have an OPS 0-00 keyboard message placed in it. This availability is determined by testing the IØP_Transmitter_State and IØP_Receiver_State which indicates if a DEU bus is in command, listen, or disable mode. If the DEU bus is in command or listen mode, then the DEU_Poll_Flag must also be on for the DEU_Input_Table to be available. If the DEU bus is disabled, then the DEU_Input_Table is available, since that DEU is not being polled. If either of these availability states is met then the Buffer_Available_Status is set to 1 and the search for an available buffer is terminated. If the Buffer_Available_Status = 1, then the DEU_Poll_Flag is disabled and OPS_0_ message is stored in the DEU_Input_Table specified by DEU_Number. The appropriate DEU_Message_Ready_Flag is set = 1 and the Keyboard_Message_Ready_Event is set for (500) User_Interface_Central_Supervisor (DMC_SUPER). Also the FORCE_OPS_OICC message is built containing GPC_Prime_ID, Redundant_Set_Mask, and Current_OPS. Finally the ICC_Force_0 ICC flag is set = 1.

d. Outputs - See Table 3.2.2.1-1

e. Module References - (170) Set_Event_Processor (FPMSET) is CALLed.

f. Module Attributes - External Procedure

g. Template References -

1. UI_FCOS_Shared_Compool (CZ2_CØMMØN)
2. UI_Section_Of_Common_Compool (CZ1_CØMMØN)
3. FCOS_Compool (FCMCØM)

h. Error Handling - None

i. Constraints and Assumptions - It is assumed that, since this module is called by the MCDS_Input_Processor, at least one DEU_Input_Table buffer will always be available. Patches to the DEU_Poll_Flags to stop polling of two, DEU's could render this assumption false.

j. Detailed Implementation - None

DATA TABLE 3.2.2.1-1

NAME FORCE_OPS_0 (AOP_FORCE_OPS_0)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	IOP_Transmitter_State	I010.06	I	830,355 351	See APP.E	CZ2B_ACT_XMITR1	See APP E	X	
2	IOP_Receiver_State	I010.08	I	335,355	815,830 440,400	CZ2B ACT RECVR			
3	DEU_Poll_Flags	J062	I,0	See APP.	E	CZ1B_D_DEU_POLL_FLAG			
4	DEU_Input_Table	J060	0	See APP.	E	CZ1V_D_DIT			
5	ICC_Force_0	I320.16	0	815	815	CZ2B_ICC_FLAG			
6	Force_OPS_0_ICC	I350	0	815	485	CZ2B_ICC_FRC_0			
7	GPC_ID	#001	I	300	See APP E	TFCMID			
8	GPC_Prime_ID	I050	I	700,351 880,820	See APP. E	CZ2V_GPC_P			
9	Redundant_Set_Mask	I010.13	I	See APP. E	See APP. E	CZ2B_RS	See APP E	X	
10	Current_OPS	I010.02	I	880,560 820	See APP. E	CZ2V_CURRENT_OPS			
11	Keyboard_Message_Ready_Event	J020	0	405,710 800,830	815 500	CZ1E_D_MCDS_EVENT			
12	DEU_Message_Ready_Flags	J050	0	405,505 815,830	500 505	CZ1B_D_DIT_MSG_READY			
13	Buffer_Available_Status	X078	I,0	815	815	AOPV_STAT			
14	DEU_Number	X079	I,0	815	815	AOPV_DEU			
15	OPS_0_Message	X080	L	815	815	AOPK_MSG_HDR			
16	Number_of_DEUS	J004	I		See APP. E	CZ1K_D_NUMBER_DEUS			

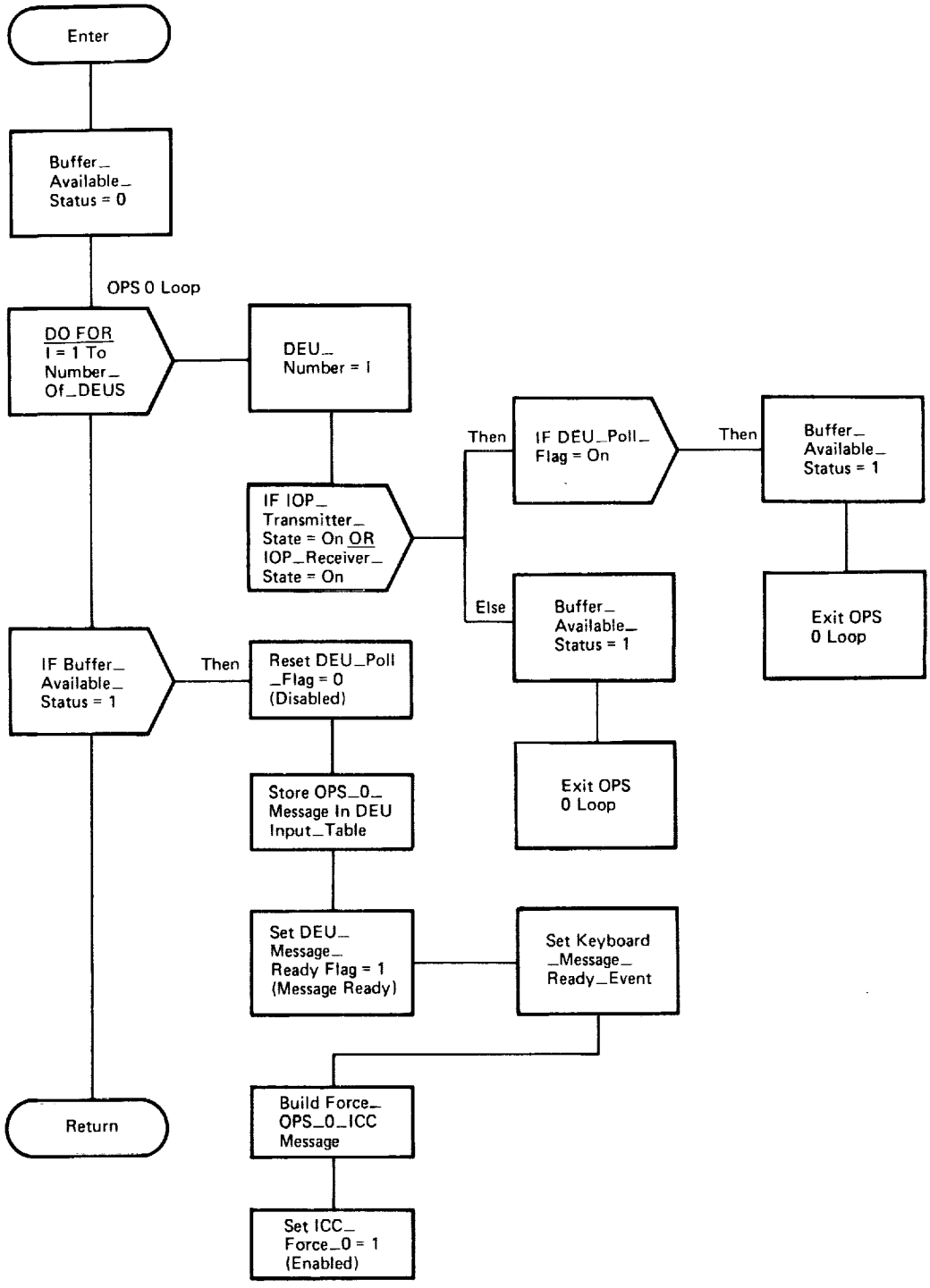


Figure 3.2.2.1-1. Force_OPS_0 (AOP_Force_OPS_0)

3.2.3 DPS Reconfiguration

The DPS Reconfiguration software is responsible for the reconfiguration of GPC sets, main memory reconfiguration, changes to the GPC Reconfiguration Table, and changes to bus configuration and its associated table. Figure 3.2.3-1 presents the hierarchial diagram of DPS Reconfiguration.

a. GPC_Reconfiguration provides the capability to establish a new redundant set and to overlay portions of GPC memory when an OPS transition is requested. It initiates the requested OPS after the loading of the overlay is completed and LDB processing is initiated, if required.

b. Bus_Configuration_Change provides the control logic for configuring the C-BUS transmitters and receivers. Bus changes are required as a result of user commands or from processing associated with OPS transitions. Changing bus commands from GPC to GPC is performed at common sync time.

c. GPC_Reconfiguration_Table_Change accepts and checks user inputs and updates the GPC Reconfiguration Table. These changes are not implemented immediately, but are executed at the next OPS transition.

d. DPS_Configuration_ITEM_Processor processes the ITEM requests made by the Idle OPS/SPEC control segment.

e. GPC_Reconfiguration_Message_Handler provides the control logic to receive and process ICC messages that are transmitted during the formation of a new redundant set which includes GPC's not included in the original set. The Secondary_GPC_Reconfiguration module is invoked upon receipt of the proper message type.

f. Secondary_GPC_Reconfiguration provides the capability to initiate an OPS transition requiring the introduction of one or more GPC's to the set of GPC's servicing a major function. The newly formed redundant set is synchronized and loaded with the proper overlay for the requested OPS.

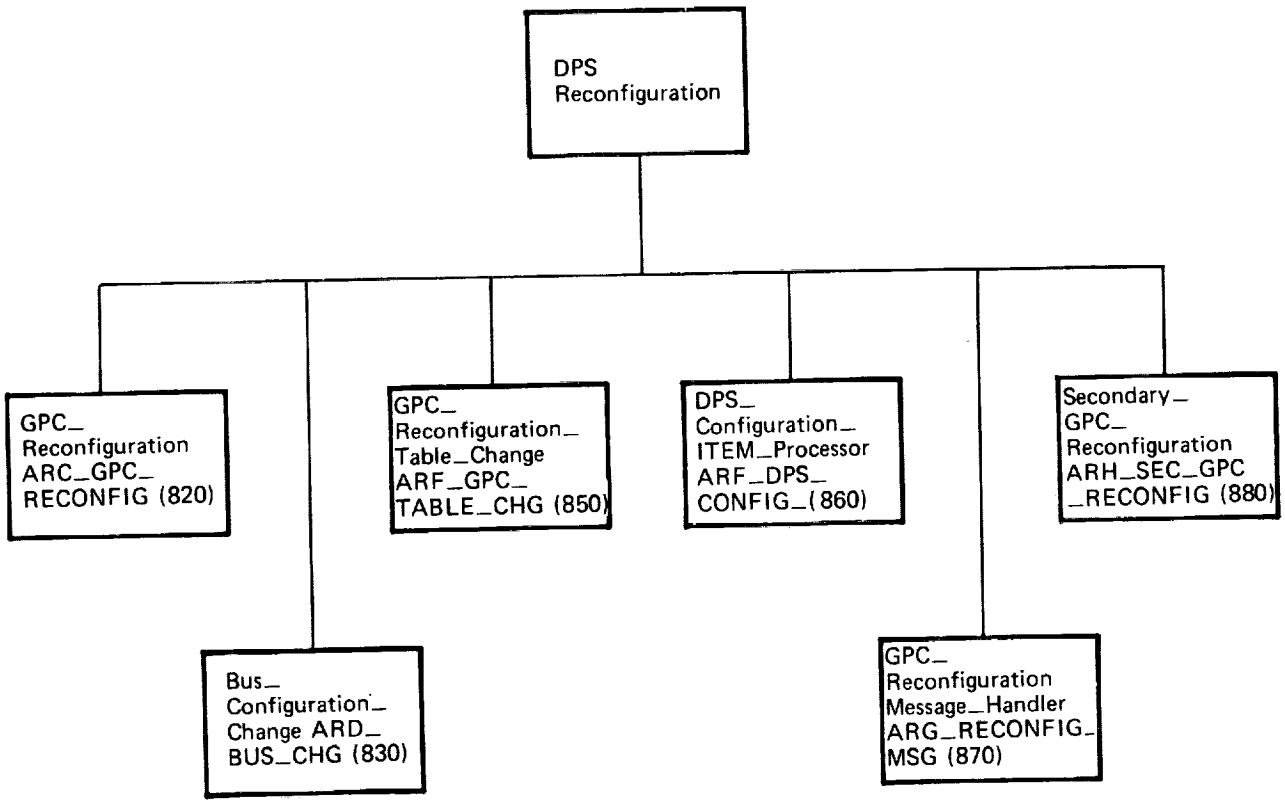


Figure 3.2.3–1. DPS Reconfiguration Hierarchy

3.2.3.1 GPC_Reconfiguration (ARC_GPC_RECONFIG) (820)

This module provides the capability to service a request to overlay main memory with a new memory configuration and provides the necessary processing controls for forming redundant sets of GPC's. Following overlay, it initiates and controls data initialization of the new GPC's and causes the initiation of the new OPS.

a. Control Interface -

1. SCHEDULE ARC_GPC_RECONFIG PRIORITY (PRIO_ARC)

2. a. SCHEDULED by (560) Sequence_Request_Processor (DMC_SEQ_REQ_PROC)

   b. Event Proceed_With_Reconfiguration (CZ2E_REC_PROCEED) set by (870) GPC_Reconfiguration_Message_Handler (ARG_RECONFIG_MSG)

b. Input -

See Table 3.2.3.1-1.

c. Process Description -

This function provides the capability to service a request to overlay main memory and to adjust the set of GPCs making up the redundant set (RS) for the data processing operations of a major function. An overlay is required for some OPS transitions, but not for all. The basis for making the decision to overlay or not, or to reconfigure GPCs (form a new RS) or not is the GPC_Reconfiguration Table and the GPC Set as a function of memory configuration. Three situations require main memory overlay: (1) an OPS is requested which is not contained in the current memory configuration (if it is in the current memory configuration, the User_Interface_Control_Supervisor proceeds to initiate the new OPS without invoking this module), (2) an OPS is requested which requires formation of a new redundant set consisting of GPCs not in the original set, (3) crew-set parameters force an overlay upon OPS request regardless of current memory contents. In each situation, a new memory configuration is brought into main memory and the User_Interface_Control_Supervisor proceeds to initiate the new OPS.

The User_Interface_Control_Supervisor is the principal processor of OPS requests and invokes this module whenever it is determined that any of the three above named conditions hole. The GPC controlling the user input unit becomes the controller GPC for coordinating the reconfiguration process. Other GPCs in the process may be either of two types: (1) a GPC may be in (a member of) the original RS and has "heard" the OPS request, (2) a GPC may be performing a dissimilar process and has

not heard the OPS request but will be required to participate in forming the new RS. The GPCs which do hear the OPS request may or may not become members of the new RS. They will execute different logic paths as a function of their impending state with respect to the new RS and end up as an operating member of the RS or not accordingly. The GPCs which do not hear the OPS request but which are needed in the new RS for the requested OPS become participants in the reconfiguration when they receive an ICC message from the controller GPC informing them of the impending changes in DPS configuration.

This function is segmented among and operates in both the controller (and others which heard the OPS request) and the secondary GPCs. Refer to Sections 3.2.3.5 and 3.2.3.6 for functional descriptions of the processes operating in the secondary GPCs to support reconfiguration.

The possible states for transition from one set of GPC's to another set of GPC's for a major function have been analyzed as follows: Three conditions (and their reciprocals) may characterize the new set of GPC's as a function of the original set. These are: self-GPC is included (or not) in the new set; other GPC's from the original set are included (or not) in the new set; other GPC's not from the original set are included (or not) in the new set. Therefore, the possible states are, reduced to seven, shown in Table 3.2.3.1-2. The design for this module is based on these seven possible states to assure that all cases have been provided for by the design. The control flow for this module is shown in Figures 3.2.3.1-1 through 7. This function is encoded with a main module and 4 internal procedures. Their control interfaces and control interfaces to external procedures are illustrated in Figure 3.2.3.1-8. The logic required and shown in Figure 3.2.3.1-1 through 7 does not require separate paths of logic for each of the seven states shown in Table 3.2.3.1-2. For example, the logic paths for 101 and 111 are the same as are the paths for 010 and 011.

When reconfiguration is required and the original set is a redundant set, each GPC in the redundant set is aware of the message requesting the OPS transition and proceeds to service it with respect to its own participation in the new configuration as specified by the GPC_Reconfiguration_Table and GPC_Set. When GPC's other than those of the original set are to be included in the new set, they are designated as secondary GPC's during the reconfiguration process.

d. Output -

See Table 3.2.3.1-1.

e. Module References -

1. (480) ICC_Message_Collector (DIM_ICC_COLLECTOR) is CALLed.
2. (440) LDB_I/O_Processor (DGI_LDB_IO) is CANCELed.
3. (660) Downlist_Formatter_1 (DCDDWL1) is CALLed.
4. (675) Annunciation_Macro_Interface (DMA_MAC) is CALLed.
5. (390) Sync_Mask_Build_Routine (FCMSMASK) is CALLed.
6. (340) Miscellaneous_CM_Request_Processor (FCMSVC) is CALLed.
7. (320) Overlay_Processor (FCMPOVLY) is CALLed.
8. (104) Wait_Processor (FPMWAIT) is CALLed.
9. (144) Time/Date_Application_Requests_Processor (FPMTMHAL) is CALLed
10. (170) Set_Event_Processor (FPMSET) is CALLed.
11. (171) Reset_Event_Processor (FPMRESET) is CALLed.
12. (665) Downlist_Formatter_2(DCDDWL2) is CALLed.

f. Module Attributes -

Program

g. Template References -

1. UI_Section_Of_Common_Compool (CZ1_COMMON)
2. UI_FCOS_Shared_Compool (CZ1_COMMON)
3. FCOS_Compool (FCMCOM)
4. Annunciation_Compool (CDL_ANNUN)
5. UI_General_Compool (CDM_UI_COMPOOL)
6. ICC_Message_Collector (DIM_ICC_COLLECTOR)
7. LDB_I/O_Processor (DGI_LDB_IO)
8. Downlist_Formatter (DCD_DOWNLIST)
9. Annunciation_Macro_Interface (DMA_MAC)

h. Error Handling -

1. Any request requiring the addition of a GPC not in OPS 0-00 to a RS causes rejection of the request and the output of an error message (OT007)

2. If any GPC in the required RS receives notification of an error from (320) Overlay_Processor (FCMPOVLY), SPEC 00 is initiated at the requesting DEU.

i. Constraints and Assumptions -

1. The formation of a new RS must be initiated via an MCDS commanded by a GPC in the current RS or, if no current RS exists, in any simplex GPC.

2. Only the formation of a new RS given all GPCs in OPS 0-00, or transition from OPS to OPS requiring only memory overlay and retaining the old RS is explicitly supported.

j. Detailed Implementation -

1. Message type 2 contains data about requested OPS and informs all GPCs that GPC(s) outside the original RS are exclusively to become members(s) of the new RS.

2. Message type 1 contains data about requested OPS and informs all GPCs that GPC(s) outside the original RS are to join with them in forming the new RS.

3. Waiting for receipt of the message sent serves to synchronize the execution of the original RS with the GPC(s) which will become member(s) of the RS.

4. Message type 3 informs new members of RS to proceed with memory overlay.

5. New RS is established: Subsequent RS synch points operate with this new RS mask.

6. LDB I/O is resumed by Bus_Change.

7. DEU Polling was disabled by Sequence_Request_Processor.

8. Either overlay for Major Function base or OPS is tested.

9. Downlist_Formatter was disabled by Sequence_Request_Processor

10. Elapsed time = current time - T_ZERO.

11. Next point in time = Current time + 4 seconds - .004 seconds
-(elapsed time - integer part of elapsed.

- .004 seconds is bias to account for downlist I/O time.
- 4 seconds forces the time to the point following the next immediate time point.

- Parenthetical expression represents fractional portion of time into the current 2 sec downlist frame.

- CLOCKTIME is uniform in all GPCs although less accurate than RUNTIME. Its accuracy is sufficient to maintain sync in all GPCs.

12. MACT_Service_Request_Flag is set = 1 (enabled), MACT_Service_Request_Ready_Events is set, and Application_Service_Request_Event is set.

13. Reference GPC_Reconfiguration_Message_Handler for description of messages.

DATA TABLE 3.2.3.1-1

NAME GPC_Reconfiguration (ARC_GPC_RECONFIG)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	Reconfiguration_DEU_Source	I500	I	560	560 820	CZ2V_REC_DEU			
2	Reconfiguration_Table_Index	I490	I	560 870	560,810 820,880	CZ2V_REC_GRT_INDEX			
3	GPC_ID	#001	I	300	See App. E	TFCMID			
4	Reconfiguration_Overlay_Status	I550	O	820	810	CZ2B_REC_OVL_STATUS			
5	Memory_Source	I040.02	I	860 ILD	560,820 880	CZ2B_DPS_STATUS		X	X
6	Overlay_In_Progress_Flag	I010.17	O	820,870 880		CZ2B_STATUS			
7	ICC_GST_Status	I320.09	O	See APP. E	142	CZ2B_ICC_FLAG			
8	Reconfiguration_Status	I540	I/O	560,810 820,880	560,810 820,880	CZ2B_REC_XERR			
9	Redundant_Set_Mask	I010.13	I/O	See APP.	E	CZ2B_RS	See APP. E	X	
10	Discrete_In_Register_A	I610.02	I	800	See APP. E	CZ2B_DIA1 /CZ2B_DIA2			
11	Secondary/Required/Current_RS_Mask	X082		820	820	ARC_RS_MASK			
12	GPC_Set	I210	I	810 860	560, 810 820, 860	CZ2B_GRT_GPC_SET			
13	Major_Function_Overlay	I620.03	I		820 880	CZ2V_GRT_MFOVL			
14	Major_Function_Overlay_Number	I010.04	I/O	820 880	820 880	CZ2V_MF_OVLY			
15	Program_Overlay	620.02	I		820 880	CZ2V_GRT_POVL			
16	Program_Overlay_Number	I010.03	I/O	820 880	820 880	CZ2V_PROG_OVLY			
17	Current_OPS	I010.02	I/O	560,820 880	See APP. E	CZ2V_CURRENT_OPS			
18	Proceed_With_Reconfiguration	I470	I/O	820, 870 880	820 880	CZ2E_REC_PROCEED			
19	MACT_Service_Request_Flags	J040.01	O	See APP. E	505,540 810,820	CZ1B_D_FLAGS			
20	MACT_Service_Request_Ready_Events	J160	O	See APP. E	505 810	CZ1E_D_MACT_EVTS			

DATA TABLE 3.2.3.1-1 (Cont'd.)

NAME GPC_Reconfiguration (ARC_GPC_RECONFIG)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
21	Application_Service_Request_Event	B050	O	540,820, 870,880	660 810,820	CDME_APP_SERVICE			
22	GSE_Poll_State	I300.02	O	495,820 830, 870	830	CZ2B_STATUS_FLAG			
23	DEU_Update_Rate	B010.20	O	500,620 820,880	620 625	CDMV_MAT_DEU_RATE			
24	MCDS_Input_Busy	I680	I	400	See APP. E	CZ2V_MCDS_IN			
25	DEU_Poll_Flags	J062	O	See APP.	E	CZ1B_DEU_POLL_FLAG			
26	CRT_Status_Flags	B010.40	O	See APP.	E	CDMB_MAT_CRT_STAT			
27	ICC_Current_Load	I320.06	O	820 880	560 810	CZ2B_ICC_FLAG			
28	Reconfiguration_OPS_Request	I520	I	560 820	560 820,880	CZ2V_REC_OPS			
29	Reconfiguration_Major_Function	I510	I	560	560, 820,880	CZ2V_REC_MF			
30	Overlay_Complete_Event	X083	O	820	820	ARC_OVL_EVT			
31	Memory_Configuration_Number	I010.05	O	405, 820, 880	See APP. E	CZ2V_MC	V91U1546C,V91U1547C V91U1548C,V91U1549C	X	X
32	BITE_Update_Flag	I460	O	820 880	660 665,910	CZ2V_BITE_UPD_VAL			
33	Flight_Critical_BTU_BITE	I100.01	O	820 880, 920	485, 660, 665	CZ2B_BTU_BITE_FC		X	X
34	Payload_BTU_BITE	I100.10	O	See APP. E		CZ2B_BTU_BITE_PL		X	X
35	PCMMU_BTU_BITE	I100.13	O	820 880	920	CZ2B_BTU_BITE_PCM	V72M8261P	X	X
36	MMU1_BTU_BITE	I100.14	O	280,282 820,880	282	CZ2B_BTU_BITE_MM1		X	X
37	MMU2_BTU_BITE	I100.15	O	280, 820,880	282	CZ2B_BTU_BITE_MM2		X	X
38	MTU1_BTU_BITE	I100.16	O	149,361 820,880	660 665	CZ2B_BTU_BITE_MTU1			
39	MTU2_BTU_BITE	I100.17	O	149,361 820,880		CZ2B_BTU_BITE_MTU2			
40	MTU3_BTU_BITE	I100.18	O	149,361 820,880		CZ2B_BTU_BITE_MTU3			

DATA TABLE 3.2.3.1-1 (Cont'd.)

NAME GPC_Reconfiguration (ARC_GPC_RECONFIG)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
41	DEU_BTU_BITE	I100.19	O	405 820, 880	660 665	CZ2B_BTU_BITE_DEU		X	X
42	ICC_BTU_BITE	I320.13	O	820 880		CZ2B_ICC_FLAG			
43	T_Zero	I020	I	700	485,700 820,880	CZ2V_TZERO			
44	Downlist_Formatter_Enabled_Flag	I010.16	O	560,700, 820,880	710 750	CZ2B_STATUS			
45	CAM_Status_Votes	I010.18	O	375 820,880	620	CZ2B_STATUS			X
46	CAM_Counters	I240	O	375 820,880	485 660,665	CZ2V_CAM_CNTRS			
47	ICC_CAM_GPC1-5	I320.17-21	O	820 880		CZ2B_ICC_FLAG			
48	GPC_Reconfiguration_Completion	J040.13	O	505,820 870,880	505 560	CZ1B_D_FLAGS			
49	ICC_Message	X084	L	820		ARC_ICC_MSG123			
50	Reconfiguration_Mode_Request	I530	O	560 820	560 820	CZ2V_REC_MODE			

Table 3.2.3.1-2. Possible GPC Transitions

Self* Included	Others From Original Set Included	Others Not From Original Set Included	Example** of Transition	Explanation
0	0	1	1 2 3 → 4	Not self or any other from original set.
0	1	0	1 2 3 4 → 1 2	Not self but others from original set.
0	1	1	1 2 → 2 3	Not self but others from original set and others not from original set.
1	0	0	1 2 3 4 → 1	Self only from original set.
1	0	1	1 → 1 2 3 4	Self and others not from original set.
1	1	0	1 2 3 4 → 1 2	Self and others from original set.
1	1	1	1 2 → 1 2 3	Self and others from original set and others not from original set.

** Note: Underscored GPC is "Self"
* Note: 0 = False; 1 = True

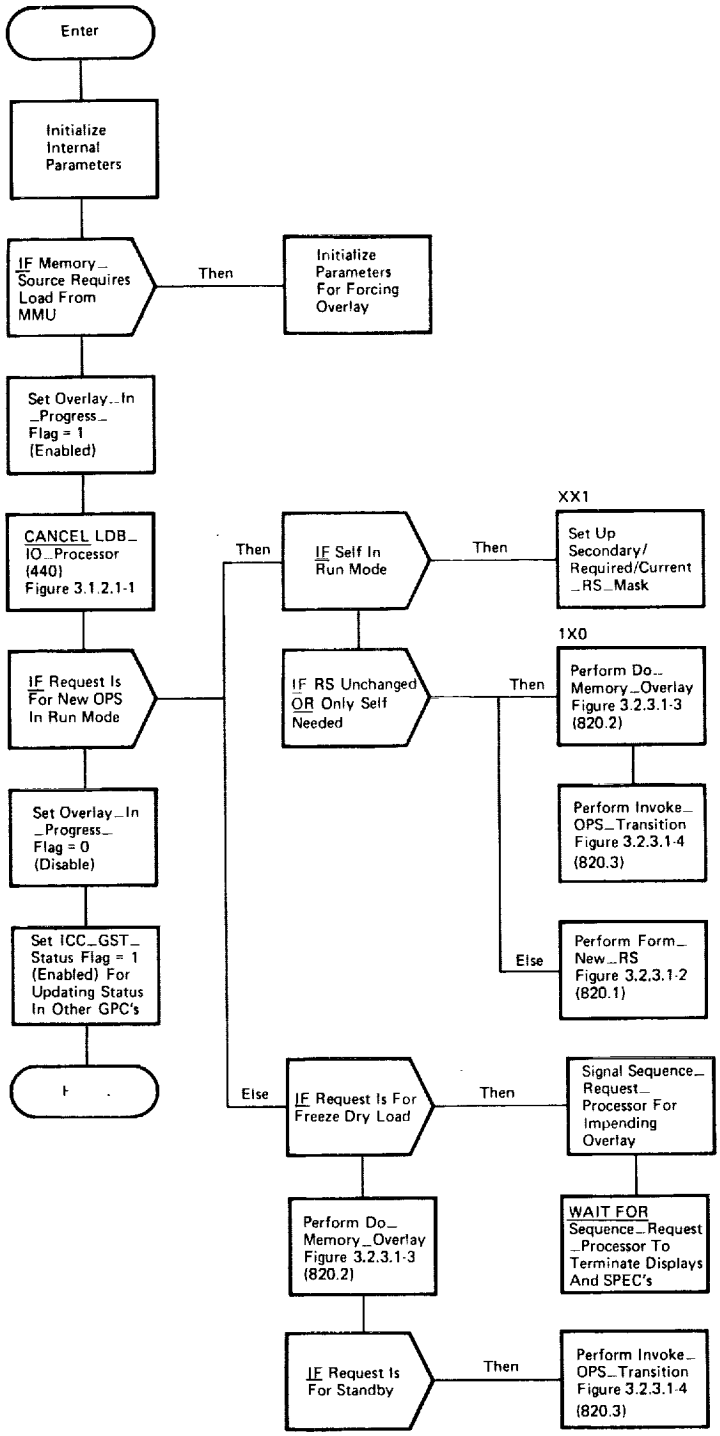


Figure 3.2.3.1-1. GPC_Reconfiguration (ARC_GPC_RECONFIG)

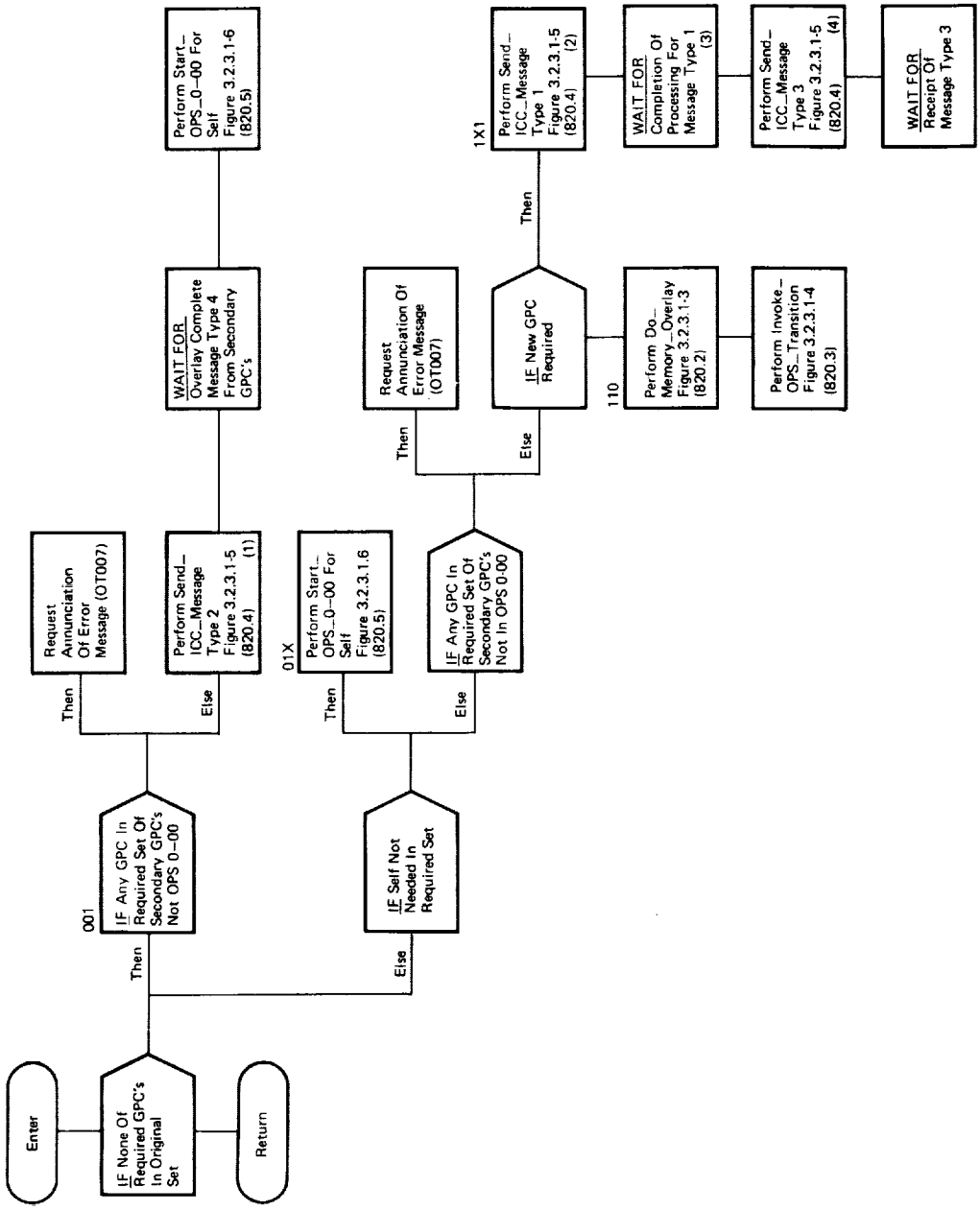


Figure 3.2.3.1-2. GPC_Reconfiguration Form_New_RS (820.1)

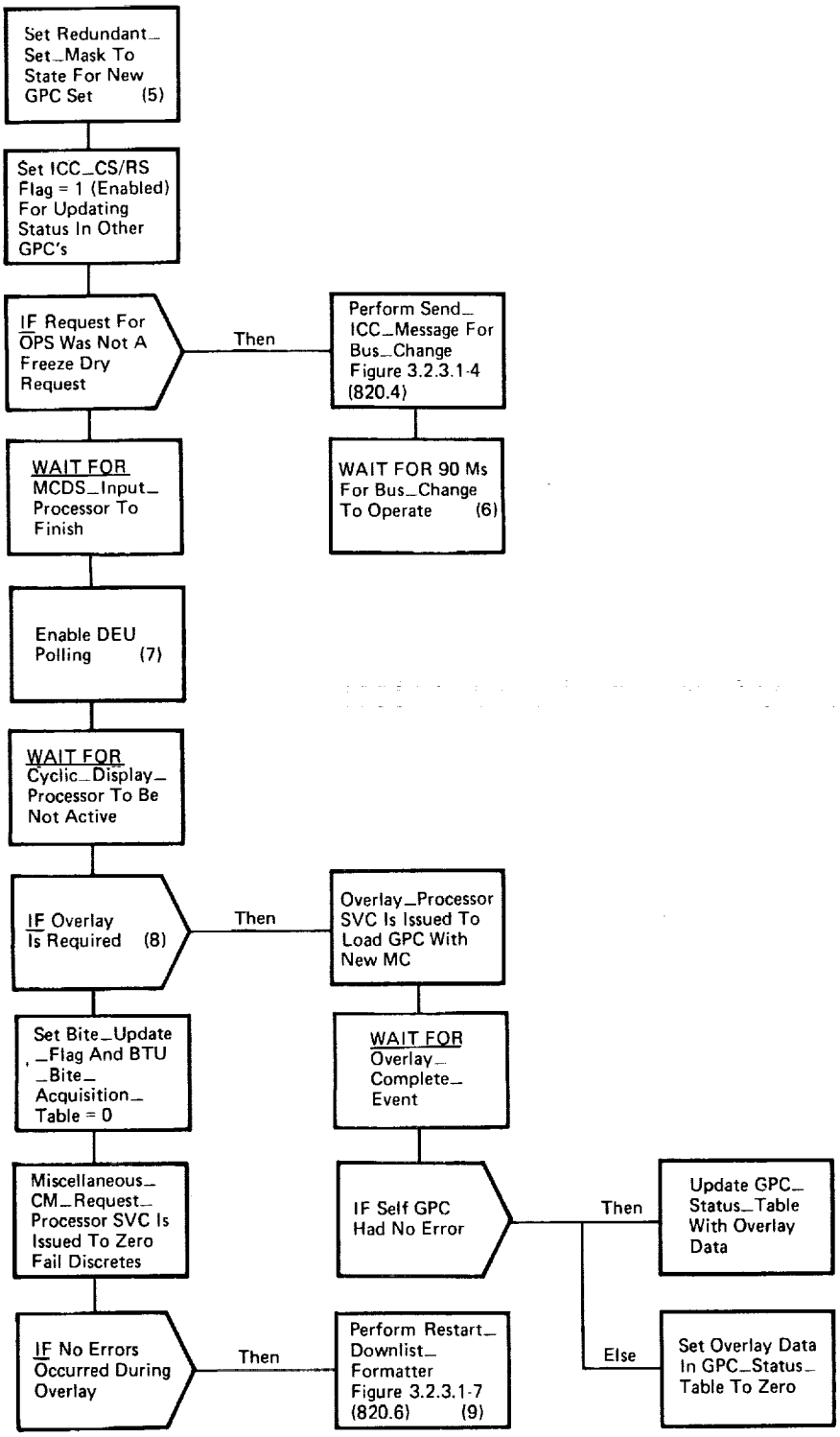


Figure 3.2.3.1-3. GPC_Reconfiguration Do_Memory Overlay (820.2)

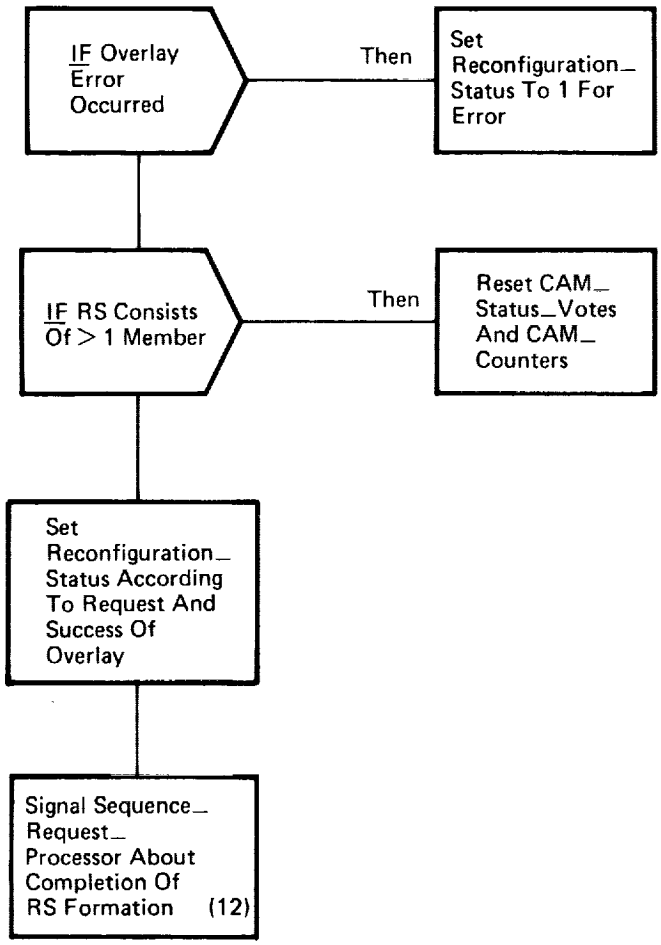


Figure 3.2.3.1-4. GPC_Reconfiguration Invoke_OPS_Transition (820.3)

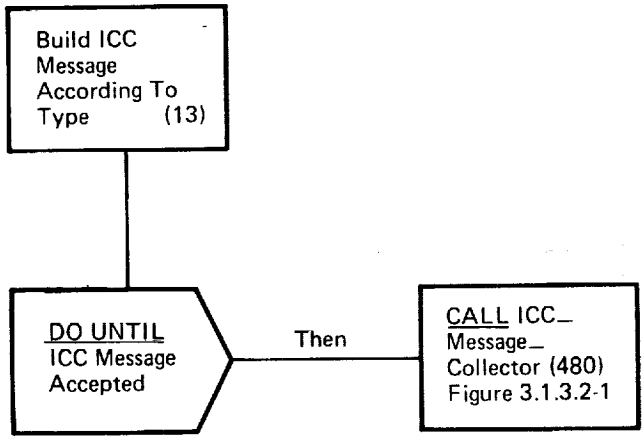


Figure 3.2.3.1-5. GPC_Reconfiguration Send_ICC_Message (820.4)

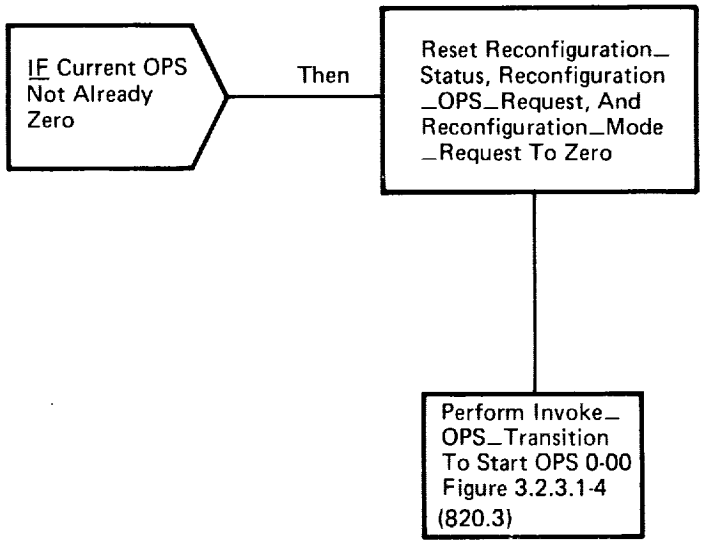


Figure 3.2.3.1-6. GPC_Reconfiguration Start_OPS_Zero (820.5)

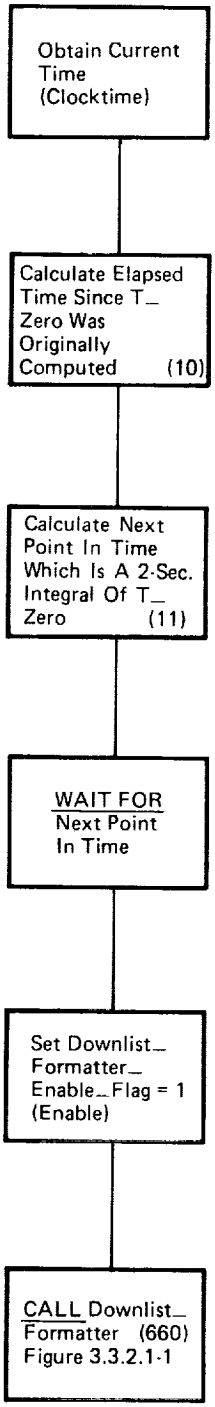


Figure 3.2.3.1-7. GPC_Reconfiguration Restart_Downlist_Formatter (820.6)

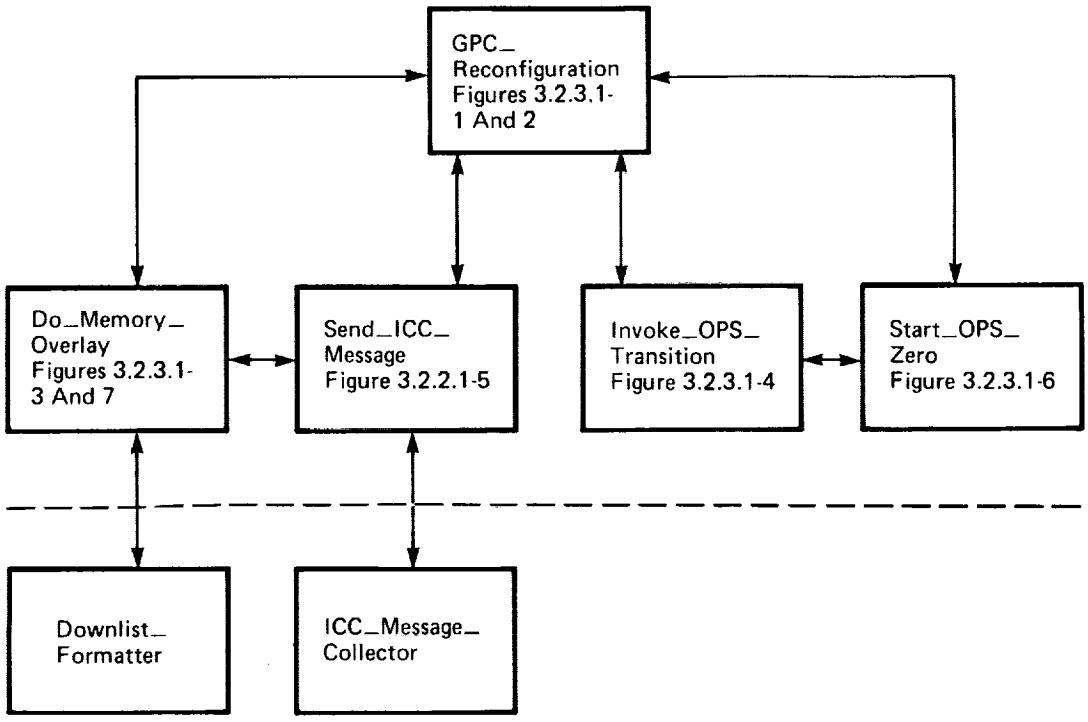


Figure 3.2.3.1-8. GPC_Reconfiguration Control Interfaces

3.2.3.2 Bus_Configuration_Change (ARD_BUS_CHG) (830)

Bus_Configuration_Change provides a method of transferring bus command from GPC to GPC. Bus_Configuration_Change is invoked as a result of crew inputs such as the CMPTR/BUS and CMPTR/CRT keys or from processing associated with OPS transitions, fail to sync, or changes in the GPC mode switch.

a. Control Interface -

1. CALL ARD_BUS_CHG (ICC_Buffer_Number, ICC_Array_Index)

2. CALLed by (490) ICC_Message_Router (DME_ICC_ROUT)

b. Input - Bus configuration requests are passed to Bus_Configuration_Change via the ICC_Message_Router. The messages are placed in the ICC_Buffer by the programs requesting the configuration changes. The content of the bus configuration messages and the originating program for each are as follows:

1. Major function switch change

Message content:

DEU number - identifies the DEU receiving the major function switch change

MF code - identifies the new major function switch setting

Message originator:

MCDS_Message_Processor

2. CMPTR/CRT key input

Message content:

CMPTR code - GPC ID of new commander from crew input

CRT code - DEU ID from crew input

DEU number - identifies the DEU receiving the inputs

Message originator:

User_Interface_Control_Supervisor

3. CMPTR/BUS key input

   Message content:

   CMPTR code - string ID for moding or zero from crew input

   Bus code - mode ID or zero from crew input

   DEU number - identifies the DEU receiving the input

   Message originator:

   User_Interface_Control_Supervisor

4. OPS initiate request

   Message content:

   GPC set - GPC's processing the major function involved in the OPS initiation or transition

   Original GPC set - GPC's formerly processing the major function if a new redundant set was formed

   MF code - identifies the major function initiated or involved in the OPS transition

   Memory overlay number - identifies the memory overlay involved in the OPS transition

   Message originator: Sequence_Request_Processor

5. Forced OPS 0-00 request

   Message content:

   GPC ID - identifies the GPC going to GPC OPS 0-00

   Prime GPC ID - identifies the prime GPC at the time of the OPS 0-00 request

   Redundant set mask - the redundant set mask of the GPC going to GPC OPS 0-00

   MF OPS numbers - identifies the current OPS selected for each major function supported in the GPC

Message originator:

Force_OPS_0

6. Fail to sync request

Message content:

Failed GPC set - GPC's which failed to sync

Prime GPC ID - identifies the prime GPC at the time of the fail to sync

Message originator: The Sync_Fail_Processor puts the message data in Fail_to_Sync_GPC and Prime_at_Fail_to_Sync. The data is moved into the ICC_Message_Buffer by the System_Interface_Processor.

ICC messages are described in Appendix F of User Interface. Additional inputs to Bus_Configuration_Change are defined in Table 3.2.3.2-1.

c. Process Description - The Bus_Handover_Cooperation procedure of Bus_Configuration_Change validity checks bus assignments to ensure that the Current_Bus_Commander and the New_Bus_Commander are cooperating in the bus handover. Successful common set sync is assumed to be necessary and sufficient authority for the Current_Bus_Commander and the New_Bus_Commander to respectively release and assume command of a bus. However, if the Current_Bus_Commander has failed sync with 000 sync codes, the New_Bus_Commander assumes that condition as sufficient authority to proceed with takeover of the bus.

Bus configuration is accomplished by issuing the RECONFIG SVC. Each GPC in the common set participates in the bus assignment by issuing the SVC with the Bus_Request_Type set based on the GPC's relationship to the New_Bus_Commander. If the GPC is the New_Bus_Commander, the Bus_Request_Type will be set to ASGN_BUS. If the GPC is in a redundant set with the New_Bus_Commander, the Bus_Request_Type is REL_BUS or LSTN_BUS depending on whether the GPC is the Current_Bus_Commander. If the GPC is not in a redundant set with the New_Bus_Commander, Bus_Request_Type is set to DIS_BUS.

Processing in Bus_Configuration_Change is divided into six major areas:

1. Major function Switch Change - When a major function change is detected on a DEU, Bus_Configuration_Change determines the New_Bus_Commander of the DEU based on the following hierarchy:

a. If a previous CMPTR/CRT request had assigned the DEU to a GPC in GPC OPS 0, the Assigned_OPS_0_GPC retains command.

b. If the GPC specified in the Computer_CRT_Assignment_Table supports the new major function of the DEU, that GPC is assigned the DEU.

c. If the new major function of the DEU is supported in a redundant set, GPC_Prime_ID becomes commander.

d. If the new major function of the DEU is supported in a simplex GPC in run mode, the Run_GPC becomes commander.

e. If the new major function of the DEU is supported in a simplex GPC in standby mode, the Standby_GPC becomes commander.

f. If the new major function is not supported in any GPC, the lowest ID GPC in GPC OPS 0 (Zero_GPC) becomes commander.

g. If none of the above conditions are satisfied, the Current_Bus_Commander retains command.

2. CMPTR/CRT key request - The New_Bus_Commander of the CRT is assigned from the input CMPTR code. The Computer_CRT_Assignment_Table is updated unless the New_Bus_Commander is in GPC OPS 0. If the New_Bus_Commander does not support the major function of the CRT and the New_Bus_Commander is not in GPC OPS 0, an error message is issued. If the New_Bus_Commander is in GPC OPS 0, the Assigned OPS_0_GPC is set equal to New_Bus_Commander.

3. CMPTR/BUS key request - If the Input_Computer and Bus_Number are zero, the following processing occurs:

a. The SVC DO_MSK_MGMT is issued to reset all bus and data path masks.

b. The SVC BCEMOD is issued to restore all BCE chains.

c. If the Input_Computer is not in GPC OPS 0, the RECONFIG SVC is issued with Bus_Request_Type set to NOM_CONF. LDB_IO_Processor is SCHEDULED if the GSE_Poll_Flag = 1 and GSE_Poll_State = 0 and Self_ID and GPC_Prime_ID are in the same redundant set.

If the Input_Computer and Bus_Number are non-zero, the RECONFIG SVC is issued to mode string 5, where Bus_Number will specify mode 6 or 7.

4. OPS initiate request - The SVC DO_MSK_MGMT is issued to reset all bus and data path masks. The SVC BDEMOD is issued to reset all BCE chains. The SVC RECONFIG is issued with Bus_Request_Type set to NOM_CONF to distribute the flight critical strings according to the Nominal_String_Assignment_Table. The set SVC is issued for DEU_Shared_Event to initiate MCDS_Input_Processor to force a major function change on all DeU's.

   Each GPC in the redundant set being formed issued the RECONFIG SVC for each DEU being commanded by a member of the redundant set to ensure that all members of the set process subsequent DEU inputs.

   The LDB_IO_Processor is SCHEDULEd if the GSE_Poll_Flag=1 and GPC_Prime_ID GPC and Self_ID GPC are in the new redundant set.

5. Fail to sync request - faile to sync bus reassignment occurs only if all of the following conditions are satisfied:

   a. The GPC failed sync with 000 discretes

   b. The failing GPC was simplex

   c. All GPC's are in GPC OPS 0

   d. The GPC that failed was the lowest ID GPC in the common set.

   If the conditions are satisfied, the next lowest ID GPC is assigned flight critical strings 1-5 and the PL buses. The SVC BECMOD is issued to restore all BCE chains. The SET SVC is issued for DEU_Shared_Event to initiate MCDS_Input_Processor to force a major function change on all DEU's.

6. Force OPS 0 request - If the New_0_GPC is simplex and is the lowest ID GPC and all GPC's are in GPC OPS 0, the following processing occurs:

   a. Flight critical strings 1-5 and the PL buses are assigned to the New_0_GPC.

   b. The SVC BCEMOD is issued to restore all BCE chains.

   c. The set SVC is issued for DEU_Shared_Event to initiate MCDS_Input_Processor to force a major function change on all DEU's.

d. Output - See Table 3.2.3.2-1.

e. Module References -

1. (170) Set_Event_Processor (FPMSET) is CALLed.
2. (330) BCE_Element_Bypass_Processor (FCMBCEMD) is CALLed.
3. (350) Bus_Reconfiguration_Pre-Processor (FCMBMAN) is CALLed.
4. (355) Bus/Data_Path_Mask_Manager (FCMBMASK) is CALLed.
5. (440) LDB_IO_Processor (DGI_LDB_IO) is SCHEDULEd.
6. (675) Annunciation_Macro_Interface (DMA_MAC) is CALLed.

f. Module Attributes - External Procedure

g. Template References -

1. UI_FCOS_Shared_Compool (CZ2_COMMON)
2. FCOS_Compool (FCMCOM)
3. UI_Section_of_Common_Compool (CZ1_COMMON)
4. Annunciation_Macro_Interface (DMA_MAC)
5. Annunciation_Compool (CDL_ANNUN)
6. UI_General_Compool (CDM_UI_COMPOOL)
7. LDB_I/O_Processor (DGI_LDB_IO)

h. Error Handling -

1. If the Current_Bus_Commander or the New_Bus_Commander is not cooperating in the bus handover, the bus configuration request is rejected and an error message is issued (OS021-OS030, OS033-OS035).
2. If a CMPTR/CRT key request assigns a DEU to a GPC that does not support the major function of that DEU and the GPC is not in GPC OPS 0, the request is processed but an error message is issued (SC001).

3. If a CMPTR/BUS key request for moding of string 5 is received from a DEU not being commanded or listened to by the commander of string 5, the request is rejected and an error message is issued (SC002).

i. Constraints and Assumptions - A bus handover request is considered valid if the current commander and the new commander of the bus successfully complete common set sync. However, if the current commander fails to sync with discretes 000, the bus handover request is also considered valid.

j. Detailed Implementation -

1. If GPC_to_Command-DEU for Input_MF not = 0 and Current_OPS for Input_MF in GPC_Status_Table for GPC_to_Command_DEU not = 0.

2. Hardware bus numbers for DEU 1,2 and 3 are 6,7 and 8, respectively.

3. The Bus_Handover_Cooperation internal procedure expects three input parameters: Current_Bus_Commander, New_Bus_Commander, and Bus_Number/Bus_Number_for_Handover.

4. If Common_Set_Mask of self GPC_ID=0 and self GPC_ID in Fail_to_Sync_GPC.

5. If New_Bus_Commander = 0 or
   New_Bus_Commander = self GPC_ID or
   New_Bus_Commander in Common_Set_Mask of self GPC_ID

6. If Discrete_Bit_GPC_Q_Sync_Code_1 = 0 and
   Discrete_Bit_GPC_Q_Sync_Code_2 = 0 and
   Discrete_Bit_GPC_Q_Sync_Code_3 = 0

   where Q is determined as follows:

   Q = Current_Bus_Commander ID - self GPC_ID
   if Q LT 0, then Q=Q+5

7. Bus_Number = 0 for flight critical strings and = hardware bus number for DEU or PL buses. The Bus_Number is used to clear the IOP_Transmitter_State in self GPC_ID's copy of the failed Current_Bus_Commander's GPC_Status_Table. This information cannot be updated by the failed GPC and must be cleared or two GPC's will indicate command of the bus. For flight critical string assignments, the Current_Bus_Commander is indicated in the Current_String_Assignment_Table and will be updated by the RECONFIG SVC processing.

8. If Current_OPS for all major functions not = 0 and
   Memory_Configuration_Number not = 0

9. If Current_Bus_Commander = 0 or
   IOP_Receiver_State = 1 for Message_DEU in GPC_Status_Table of
   Current_Bus_Commander

10. The input GPC set is defined in bits 1-5 of the second word of the input ICC message and referenced from the ICC message rather than a local variable:

    CZ2B_ICC_MSG_BUF $(B;I+1:1 to 5)

11. If IOP_Transmitter_State = 1 for Bus_Number in GPC_Status_Table for loop GPC.

DATA TABLE 3.2.3.2-1

NAME Bus_Configuration_Change (ARD_BUS_CHG)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	Input_DEU	X041	I Ø	830	830	ARDVL_CRT			
2	Input_MF	X042	I Ø	830	830	ARDVL_MF			
3	Commander_Set	X043	I Ø	830	830	ARDVL_CMDR_SET			
4	Assigned_OPS_0_GPC	X044	I Ø	830	830	ARDVL_CRT_FLG			
5	New_Bus_Commander	X045	I Ø	830	830	ARDVL_NEW_CMDR			
6	GPC_to_Command_DEU	I150.01	I Ø	830	830	CZ2V_CCAT			
7	Run_GPC	X046	I Ø	830	830	ARDVL_RUN			
8	Standby_GPC	X047	I Ø	830	830	ARDVL_STBY			
9	Zero_GPC	X048	I Ø	830	830	ARDVL_ZERO			
10	GPC_ID	#001	I	300	See APP. E	TFCMID			
11	Common_Set_Mask	I010.12	I	See Appendix E		CZ2B_CS	V91M2087P V91M2121P	X	
							V91M2155P V91M2200P		
12	Current_OPS	I010.02	I	560,820, 880	See APP. E	CZ2V_CURRENT_OPS			
13	Redundant_Set_Mask	I010.13	I	See Appendix E		CZ2B_RS	V91M2070P V91M2104P	X	
							V91M2138P V91M2172P		
14	GPC_Prime_ID	I050	I	351, 700, 880,820	See APP. E	CZ2V_GPC_P			
15	Discrete_Bit_Run_Mode	&005.03	I	See Appendix E		CZ2B_DIAL			
16	MF_Flag	X049	I Ø	830	830	ARDVL_MF_FLG			
17	Handover_Status	X050	I Ø	830	830	STAT			
18	Input_Cmptr	X051	I Ø	830	830	ARDVL_CMPTR			

DATA TABLE 3.2.3.2-1 (Cont'd.)

NAME Bus_Configuration_Change (ARD_BUS_CHG)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
19	Bus_Number	X052	I Ø	830	830	ARDVL_BUS			
20	Message_DEU	X053	I Ø	830	830	ARDVL_DEU_NO			
21	Bus_Parms_Variable_Data_1	X054	Ø	830	350	BUS_PARMS.VAR_DATA1			
22	Memory_Configuration_Number	I010.05	I	405,820, 880	See APP. E	CZ2V_MC	V91U1546C V91U1547C	X	X
							V91U1548C V91U1549C		
23	GSE_Poll_Flag	I040.01	I	860,910	830	CZ2B_DPS_STATUS		X	X
24	GSE_Poll_State	I300.02	I Ø	495, 820 830, 870	830	CZ2B_STATUS_FLAG			
25	Current_Bus_Commander	X055	I Ø	830	830	ARDVL_CURR_CMDR			
26	Current_String_Assignment_Table	I080	I	351, 368	350, 392 830	CZ2B_STRNG_ASSIGN			X
27	Bus_Number_for_Handover	X056	I Ø	830	830	ARDVL_B_NO			
28	Bus_Request_Type	X057	Ø	830	350	BUS_PARMS.TYPE_IND			
29	Bus_Parms_Variable_Data_2	X058	Ø	830	350	BUS_PARMS_VAR_DATA2			
30	Input_Overlay_Number	X059	I Ø	830	830	ARDVL_MEM			
31	DEU_Shared_Event	I690	Ø	See App. E	400	CZ2E_SHRD_EVT			
32	MAT_Major_Function_Setting	B010.10	Ø	505,520	505,560, 600,620	CDMV_MAT_MF			
33	Current_MF	I270	I	405	405, 485 830	CZ2V_DEU_MF			
34	Fail_Sync_Set	X060	I Ø	830	830	ARDVL_FAIL_SET			
35	New_O_GPC	X061	I Ø	830	830	ARDVL_GPC			
36	Fail_to_Sync_GPC	I330	I	368	485, 830 950	CZ2B_FAIL_SYNC_ SET			

DATA TABLE 3.2.3.2-1 (Cont'd.)

NAME Bus_Configuration_Change (ARD_BUS_CHG)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
37	IOP_Transmitter_State	I010.06	I Ø	351, 355 830	See App. E	CZ2B_ACT_XMITR1	V91M8706P V91M8740P	X	
							V91M8774P V91M8808P		
38	IOP_Receiver_State	I010.08	I	335, 355	400,440 815,830	CZ2B_ACT_RECVR			
39	Bus_Masks	I010.09	I	355	See App. E	CZ2B_BUS_MASK1	V91M7922P V91M7957P	X	X
							V91M8011P V91M8055P		
40	DEU_Message_Ready_Flags	J050	Ø	405, 505 815, 830	500, 505	CZ1B_D_DIT_ MSG_READY			
41	Keyboard_Message_Ready_Event	J020	Ø	405, 710 800, 830	500, 815	CZ1E_D_MCDS_EVENT			
42	Keyboard_Message	J060.20	Ø	See Appendix E		CZ1V_D_DIT_KYBD_ MSG			
43	Lowest_O_GPC	X062	I Ø	830	830	N			
44	FC_String	X063	I Ø	830	830	STRING			
45	Bus_Mask_Variable_Data	X064	Ø	830	355	BUS_MASK_PARMS_ VAR_DATA			
46	ICC_Buffer_Number	X065	I	490	830	B			
47	ICC_Array_Index	X066	I	490	830	I			
48	ICC_Buffer	I610	I	490,750, 800,830	See APP. E	CZ2V_ICC_BUF			
49	Computer_CRT_Assignment_Table	I150	I Ø	830	830	CZ2V_CCATS			
50	GPC_Status_Table	I010	I Ø	830	830	CZ2V_GST			
51	Discrete_Bit_GPC_N+1 to 4_Sync_Code_1 to 3	&005.14-25	I	See Appendix E		CZ2B_DIA2			
52	IOP_Transmitter_State	I010.07	I Ø	355, 830	440,660, 665,830	CZ2B_ACT_XMITR2	V91M8723P V91M8757P	X	
							V91M8791P V91M8822P		

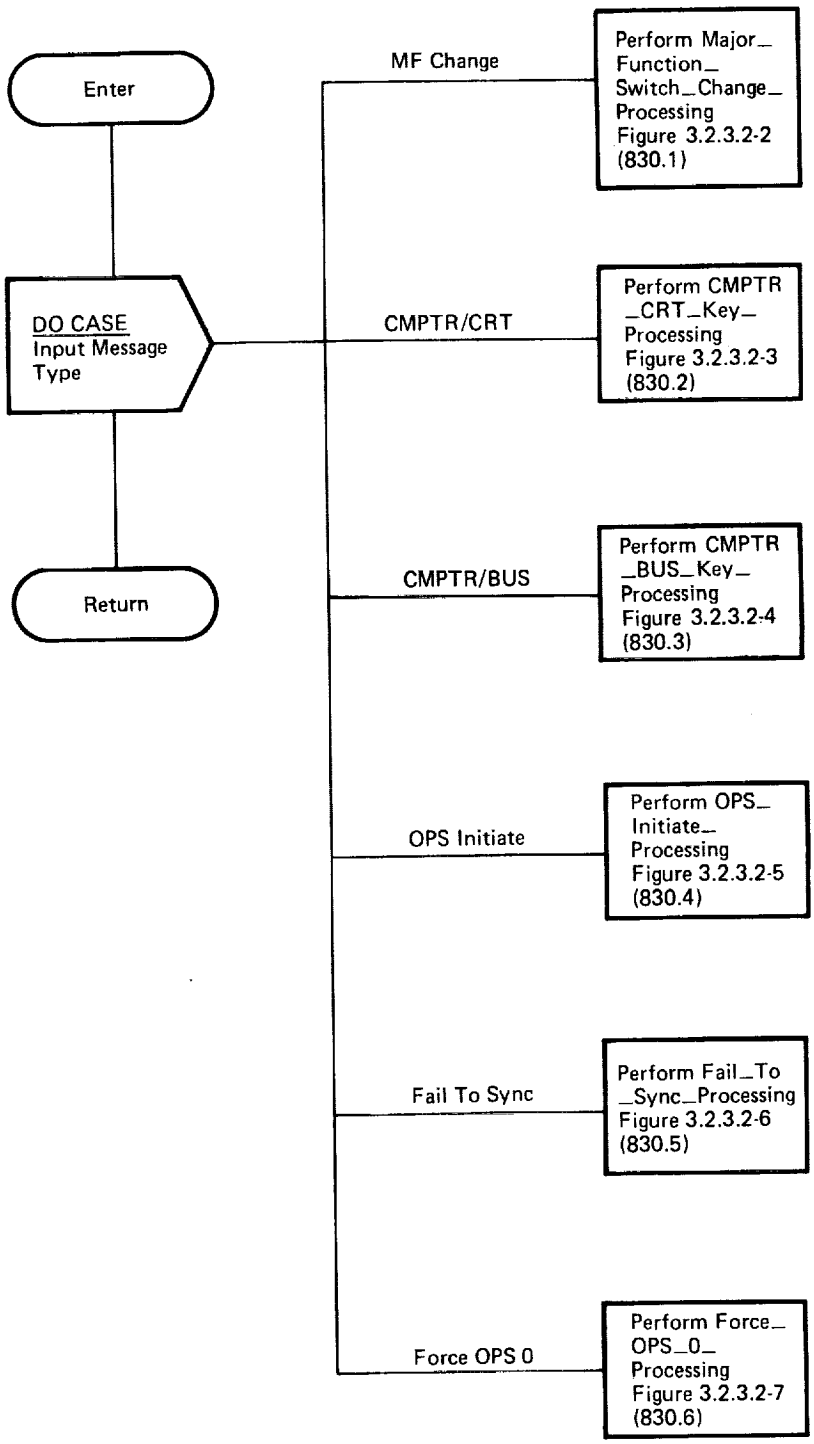


Figure 3.2.3.2–1. Bus_Configuration_Change (ARD_BUS_CHG)

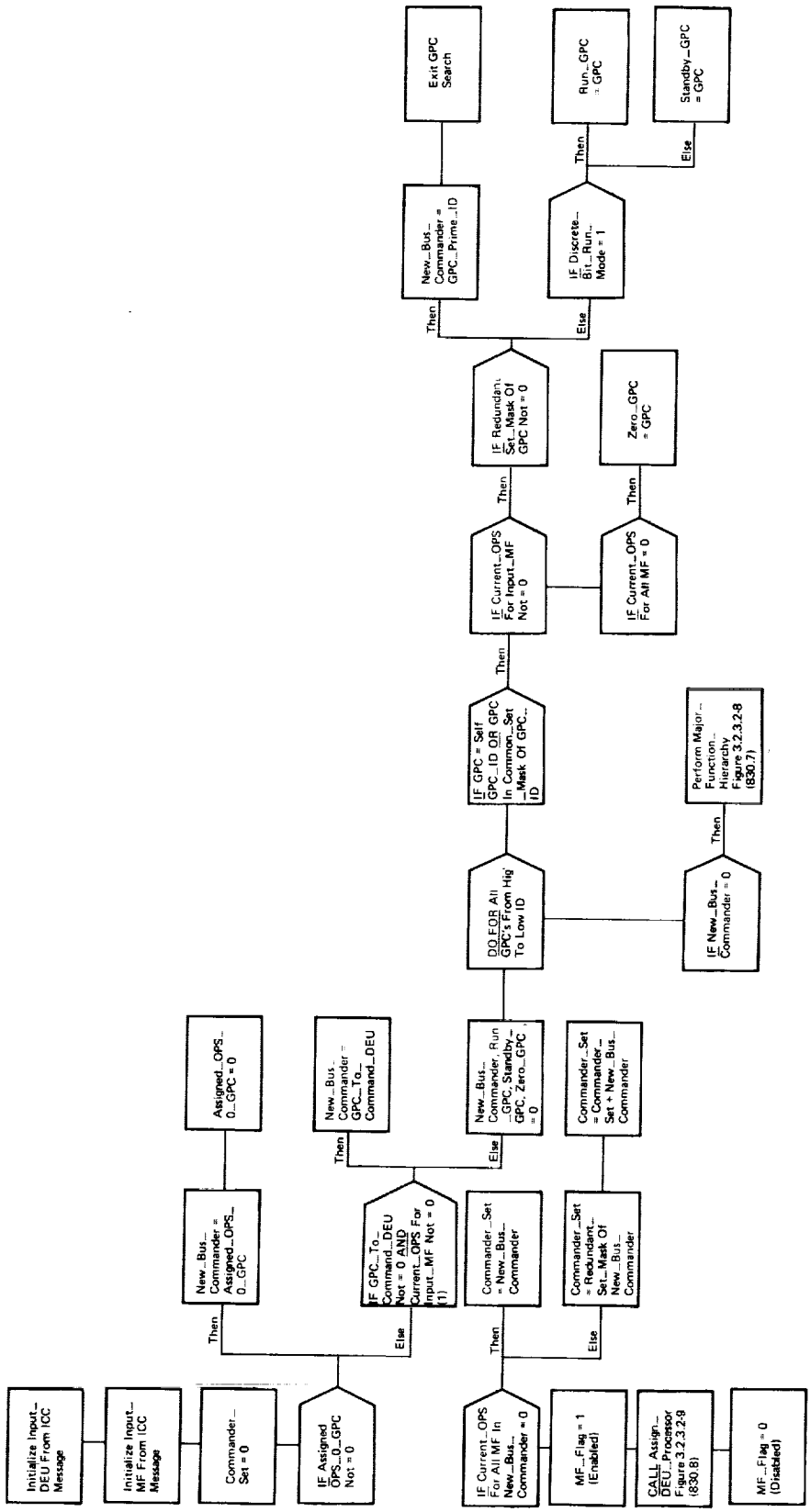


Figure 3.2.3.2-2. Bus_Configuration_Change Major_Function_Switch_Change_Processing (830.1)

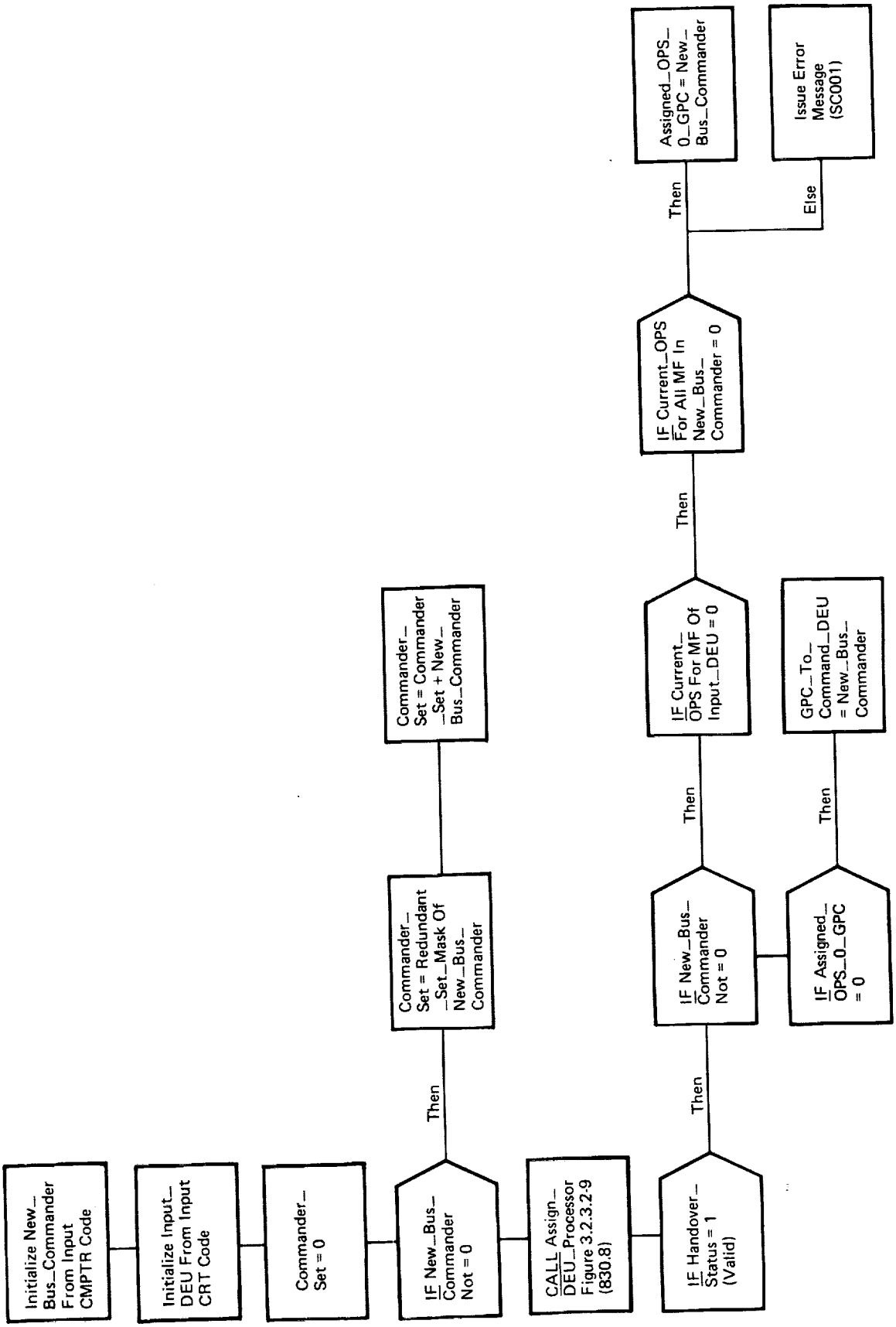


Figure 3.2.3.2-3. Bus_Configuration_Change CMPTR_CRT_Key_Processing (830.2)

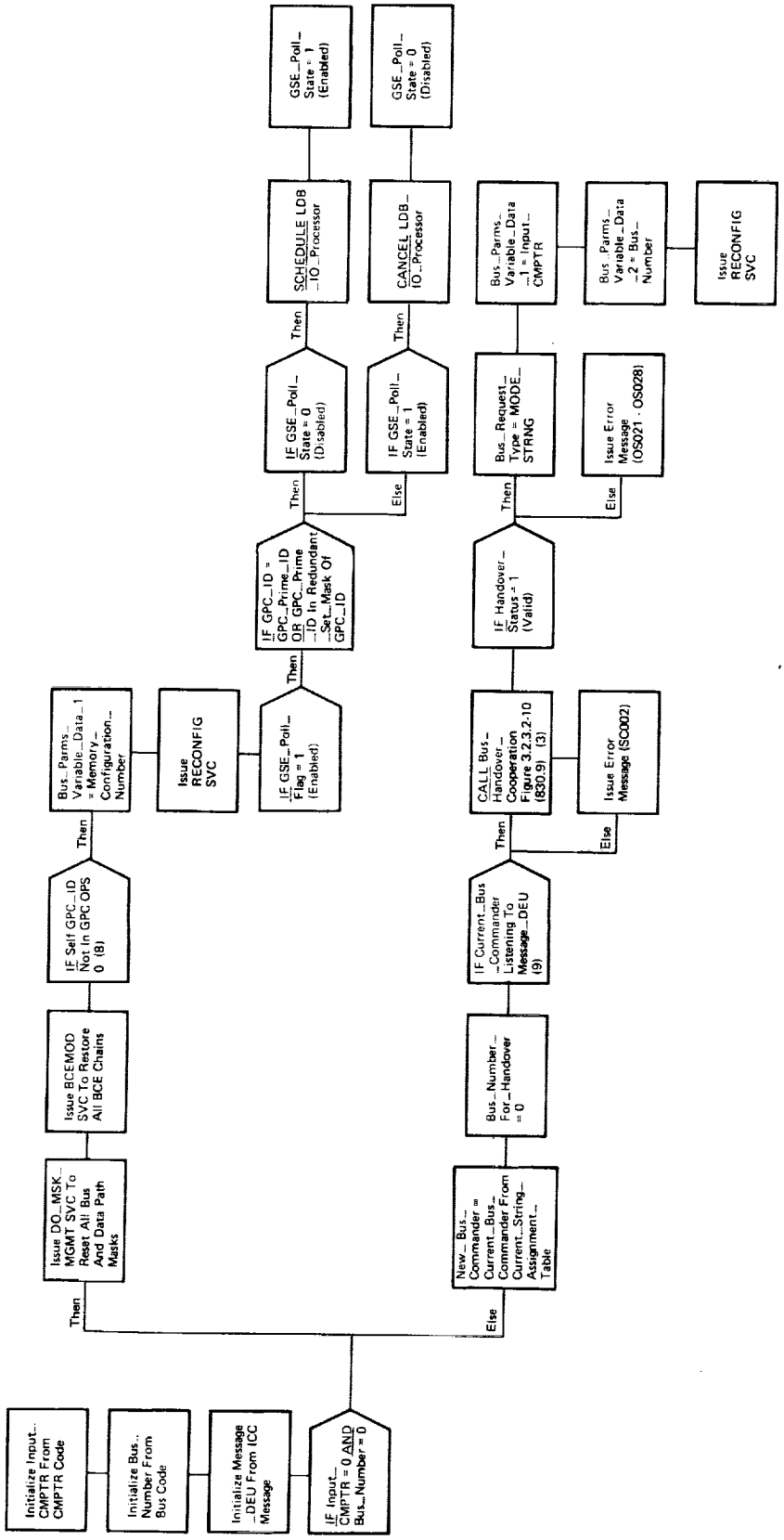


Figure 3.2.3.2-4. Bus_Configuration_Change CMPTR_BUS_Key_Processing (830.3)

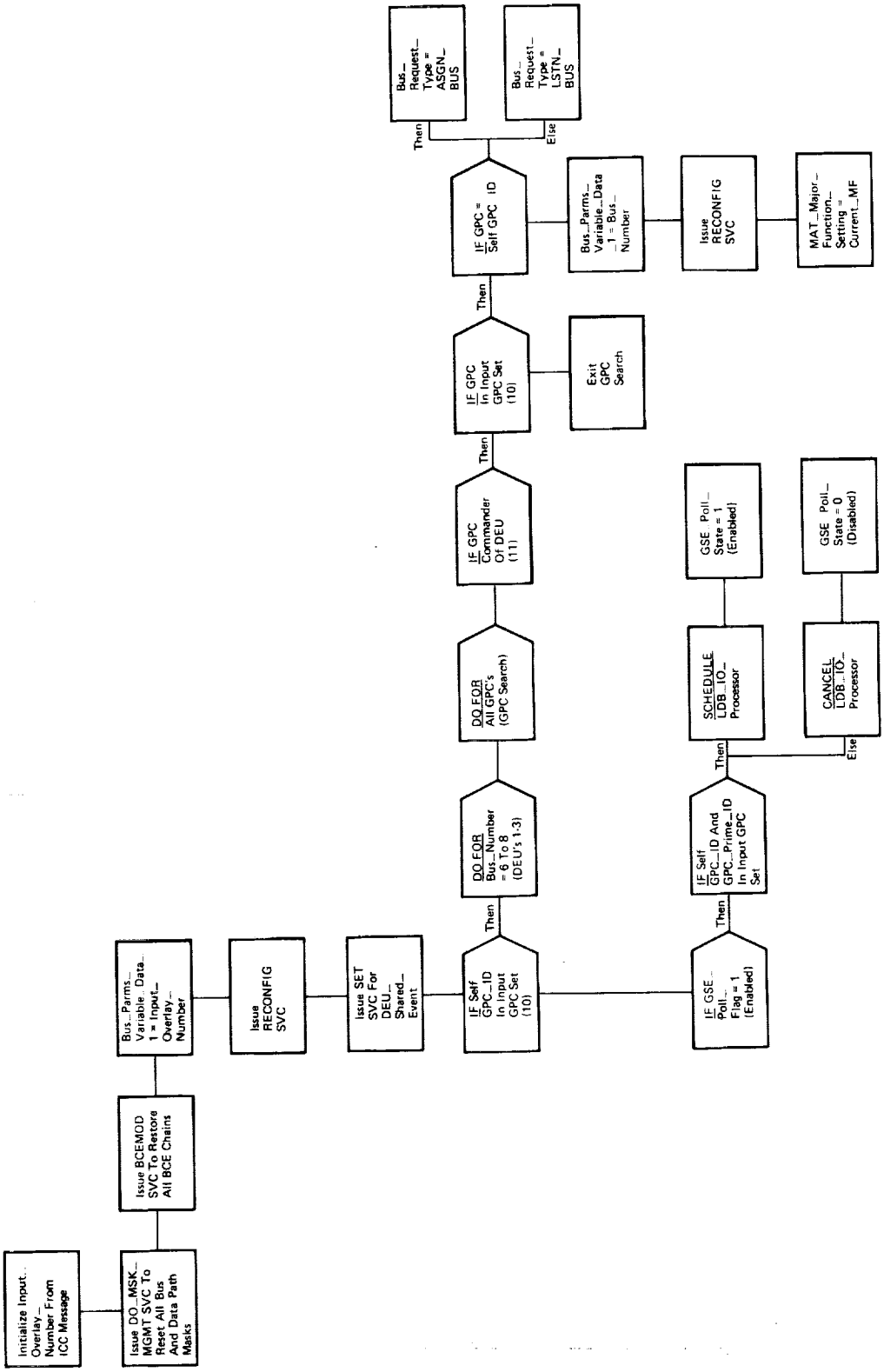


Figure 3.2.3.2-5. Bus_Configuration_Change OPS_Initiate_Processing (830.4)

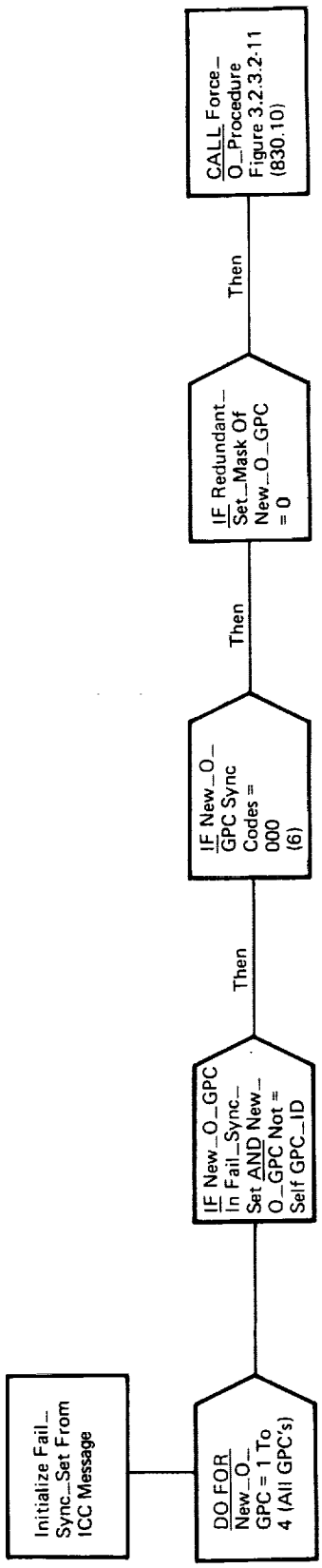


Figure 3.2.3.2–6. Bus_Configuration_Change Fail_To_Sync_Processing (830.5)

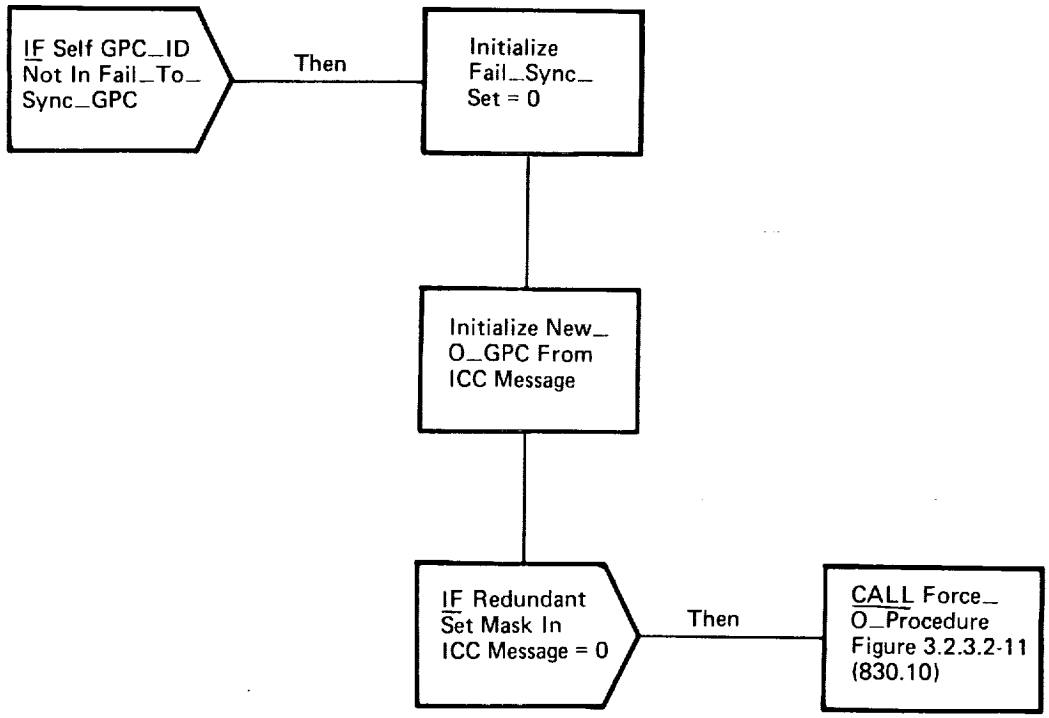


Figure 3.2.3.2–7. Bus_Configuration_Change Force_OPS_O_Processing (830.6)

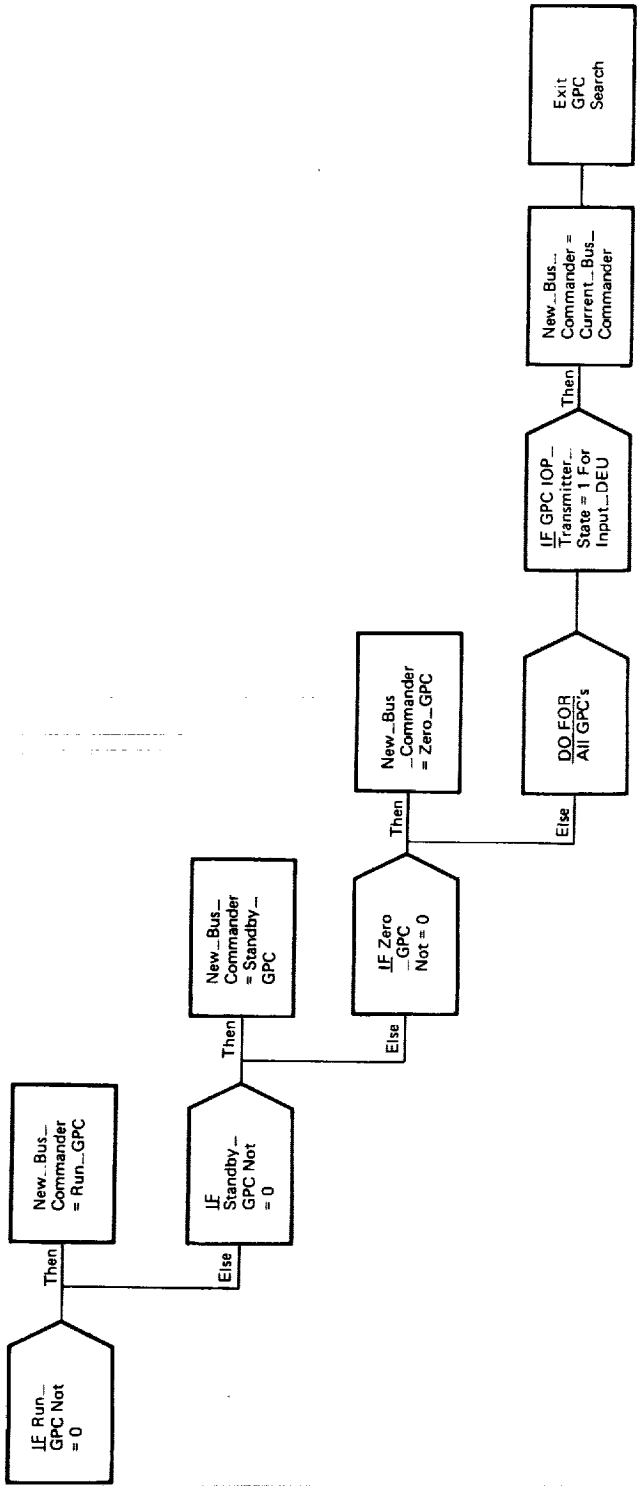


Figure 3.2.3.2–8. Bus_Configuration_Change Major_Function_Hierarchy (830.7)

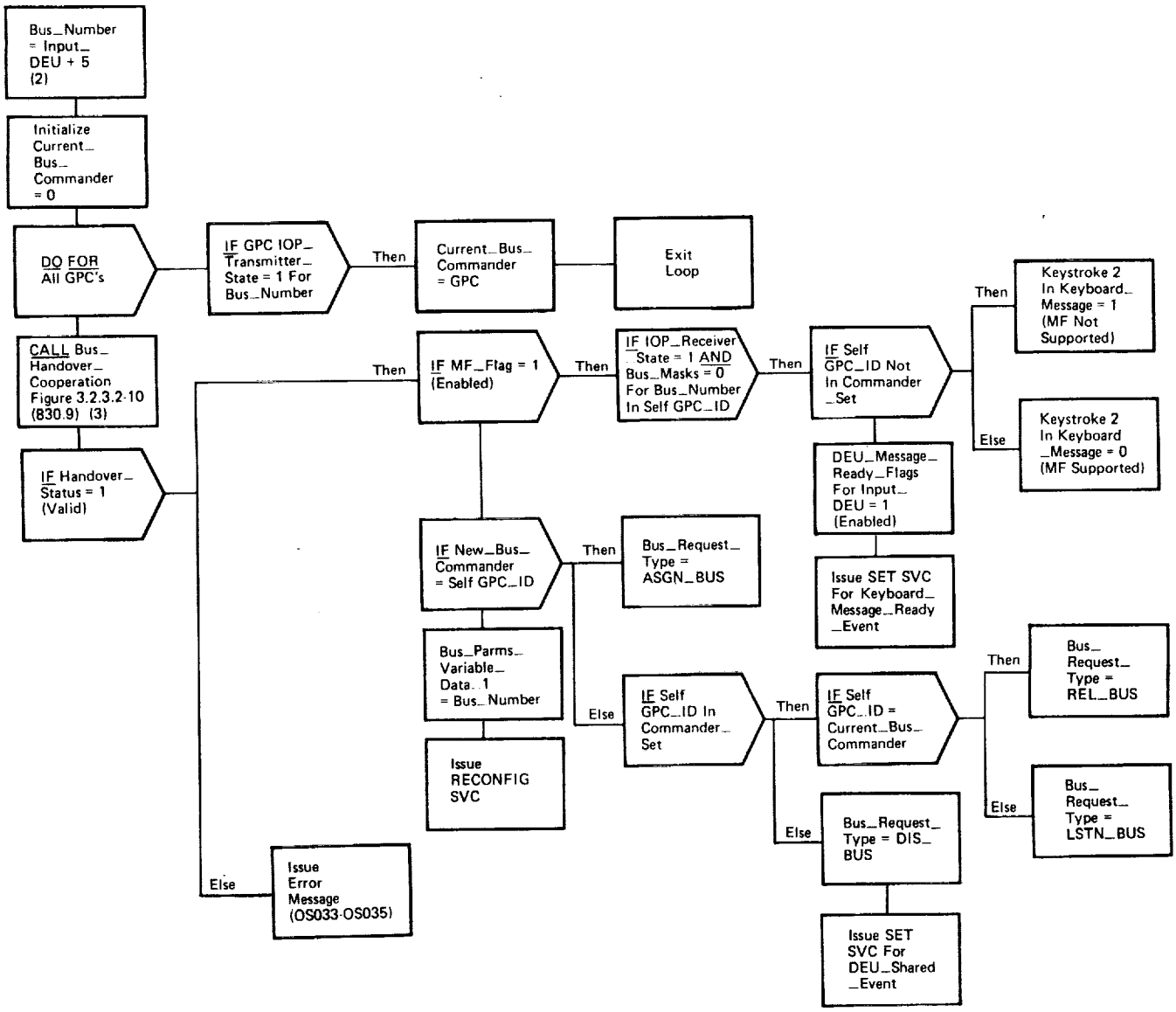


Figure 3.2.3.2–9. Bus_Configuration_Change Assign_DEU_Processor (830.8)

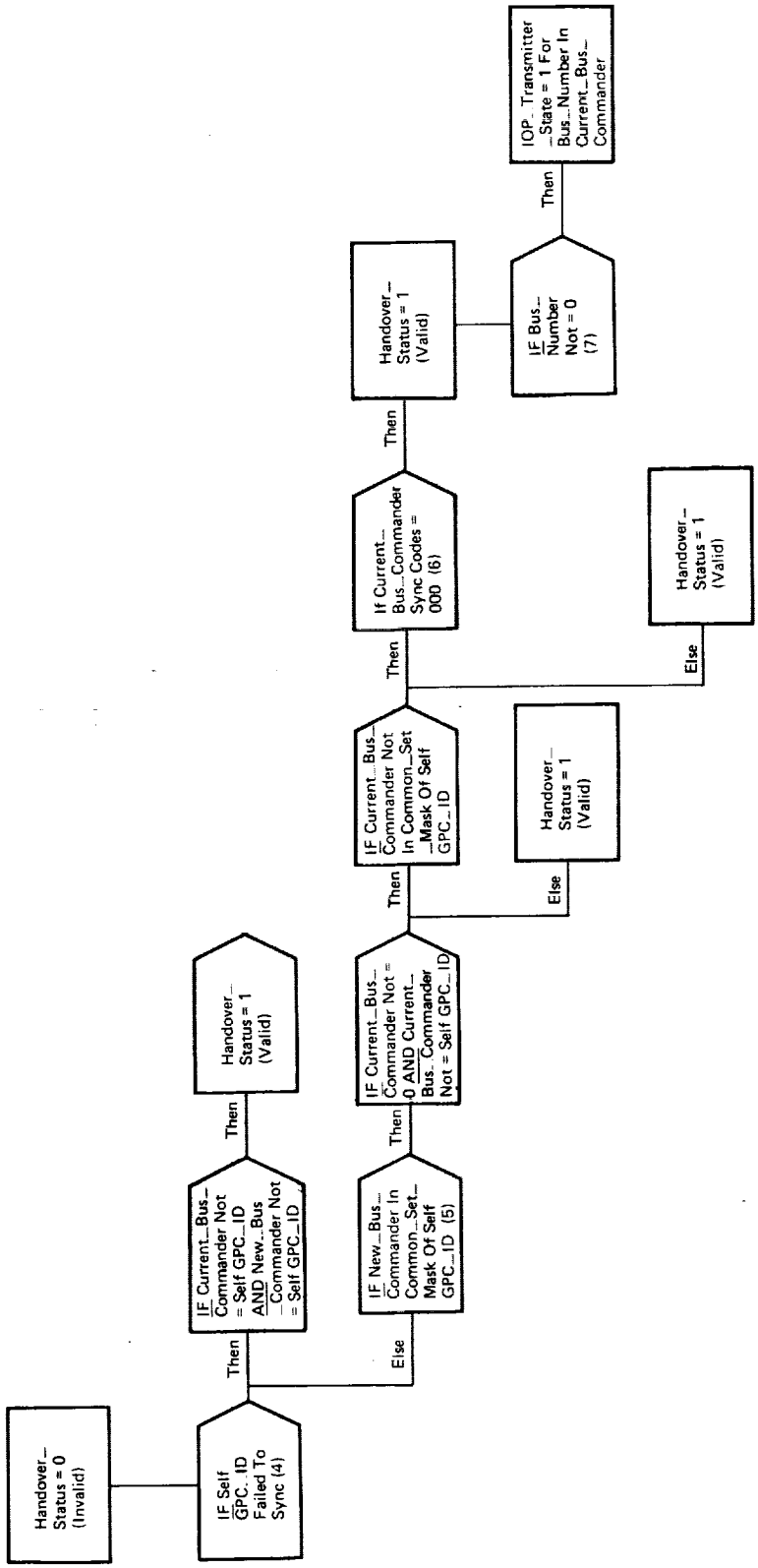


Figure 3.2.3.2–10. Bus_Configuration_Change Bus_Handover_Cooperation (830.9)

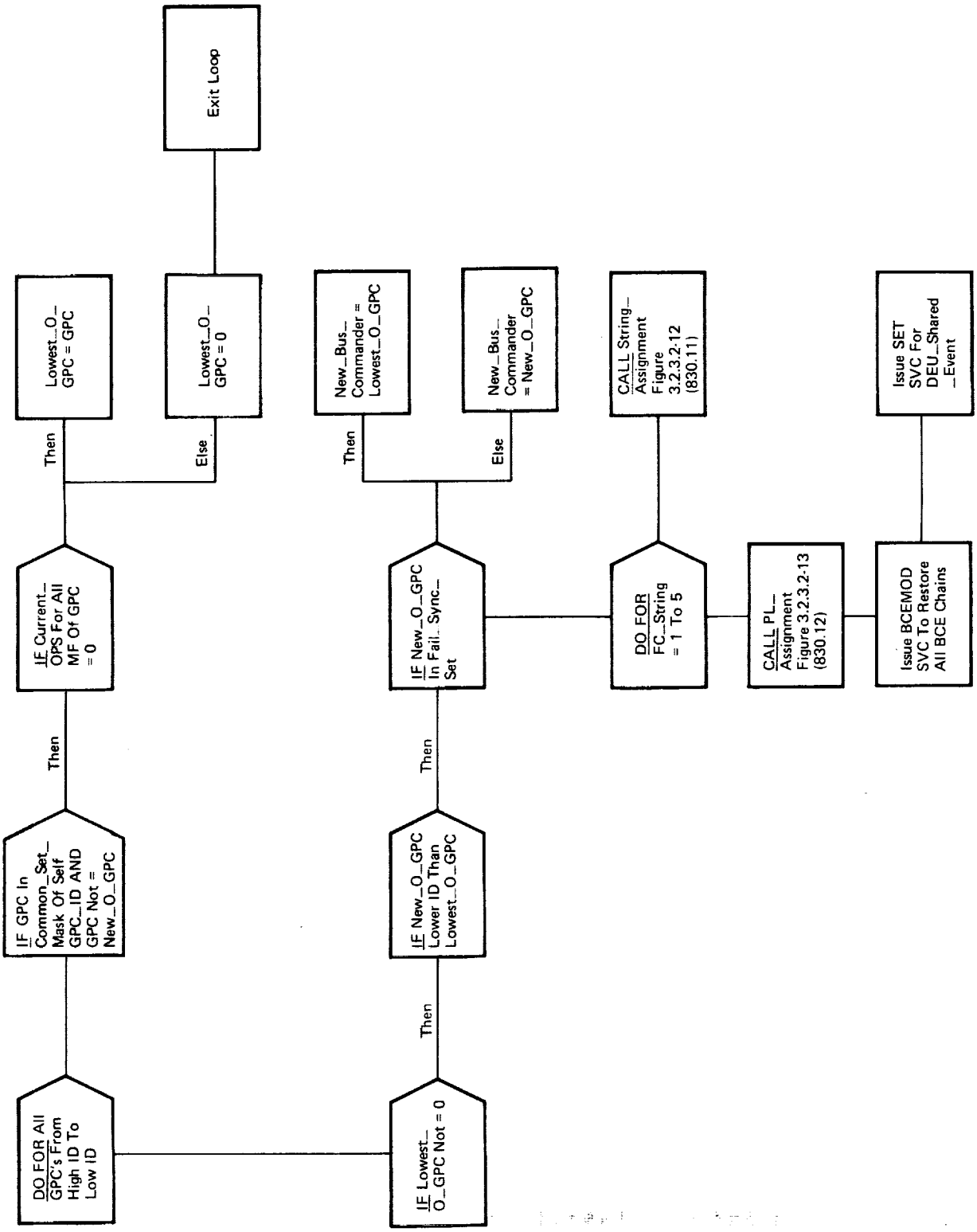


Figure 3.2.3.2–11. Bus_Configuration_Change Force_O_Procedure (830.10)

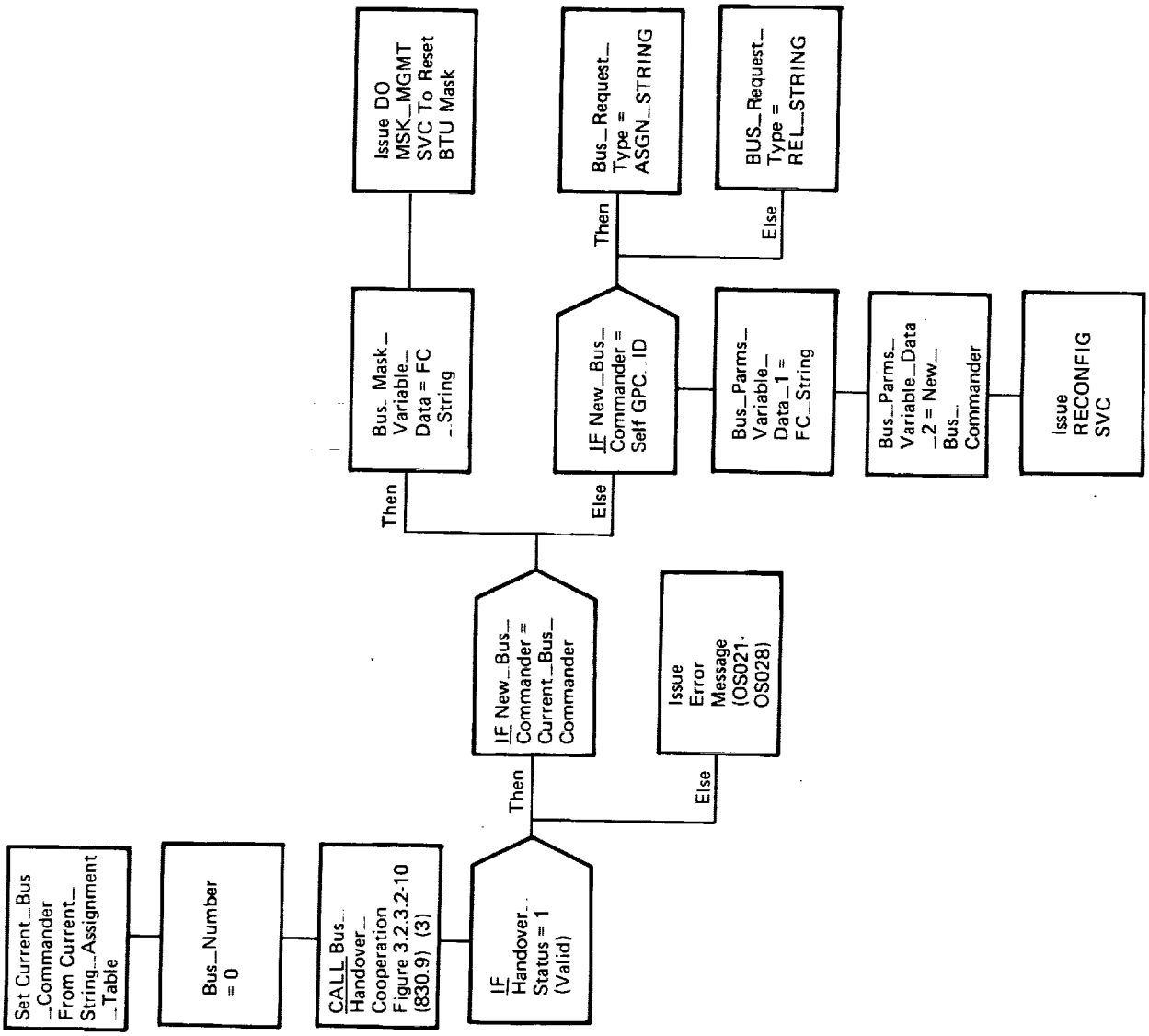


Figure 3.2.3.2–12. Bus_Configuration_Change String_Assignment (830.11)

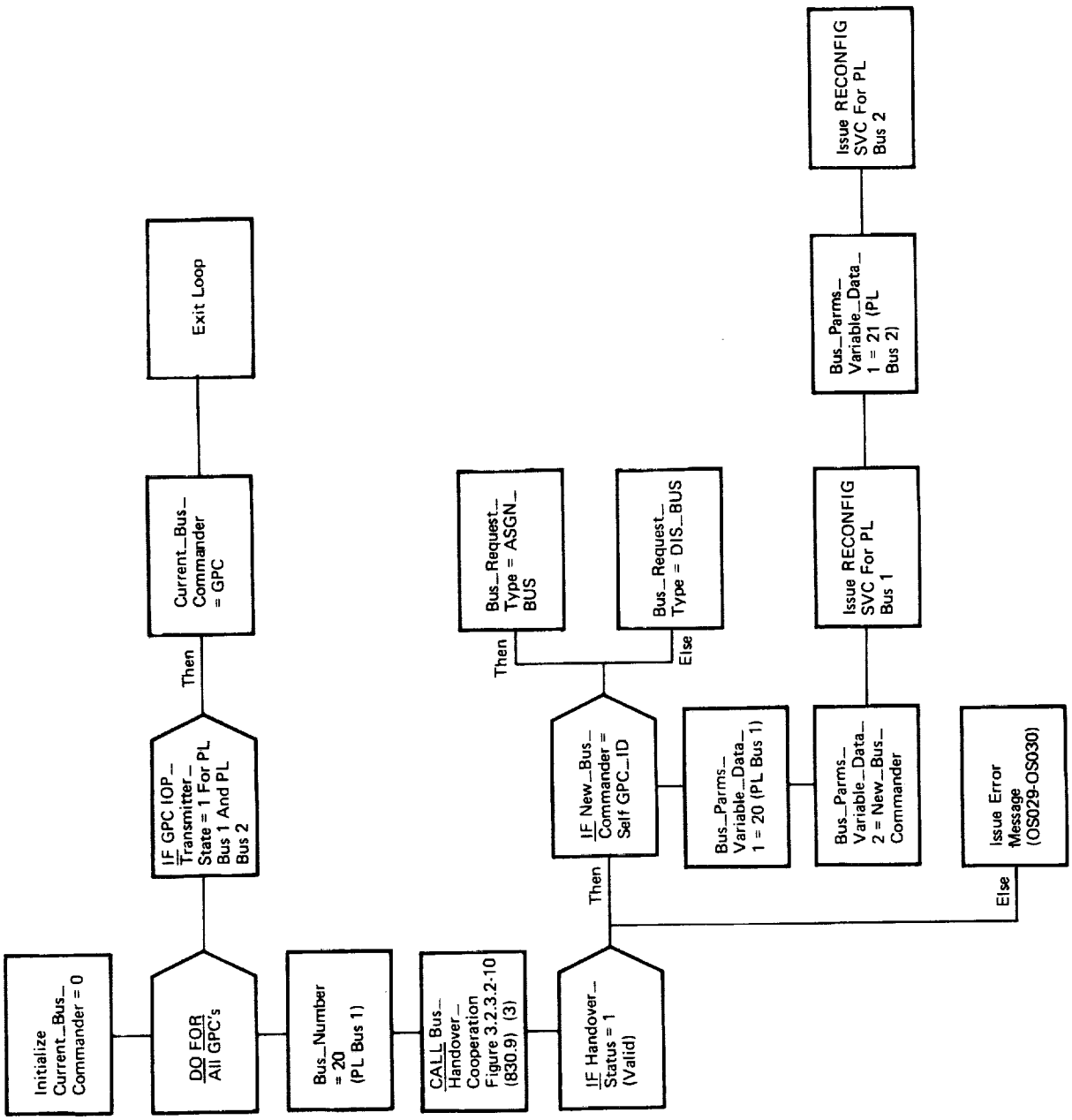


Figure 3.2.3.2–13. Bus_Configuration_Change PL_Assignment (830.12)

3.2.3.3 GPC_Reconfiguration_Table_Change (ARF_GPC_TABLE_CHG) (850)

ARF_GPC_TABLE_CHG Processes user inputs requesting GPC reassignments and updates to the GPC set and string assignment table.

a. Control Interface -

1. CALL ARF_GPC_TABLE_CHG (D-IND);
2. CALLed by: (860) DPS_Configuration_ITEM_Processor (ARF_DPS_CONFIG_ITEM)

b. Input - The following User Interface grammar macro supplies input data to this module:

ITEM_NO

For a detailed description of this macro refer to appendix H.

Additional input data descriptions are presented in Table 3.2.3.3-1.

c. Process Description - Upon entry to this procedure a test is made to see if the item number selected was 7. If that was the item request then logic to support the memory configuration request is executed.

If the item input was other than 7 then a test is made to see if a memory configuration request has been made previously. If no memory configuration request has been made then control returns to the calling procedures.

If a memory configuration request has been made than a test is made to see if the item input was to request a change in the GPC set assignments or to request updates to the GPC/STRING assignment table. The appropriate logic is then executed to support the item input.

d. Outputs - See Table 3.2.3.3-1.

e. Module References - None

f. Module Attributes - Internal Procedure

g. Template References -

1. UI_FCOS_Shared_Compool (CZ2_COMMON)
2. UI_Section_of_Common_Compool (CZ1_COMMON)
3. Annunciation_Macro_Interface (DMA_MAC)
4. Annunciation_Compool (CDL-ANNUN)

h. Error Handling -

1. If a request to change the GPC set for a specific Memory Configuration number which previously included only one (1) GPC to multiple GPC's is made, an error via the annunciation function will be output (OT006).

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.2.3.3-1

NAME GPC_Reconfiguration_Table_Change (ARF_GPC_TABLE_CHG)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	ARFV_GRT_OPS_Copy	X006	L	850	850	ARFV_GRT_OPS_ STRUCT			
2	ARFV_Mem_Config_NO	X007	L	850	850	ARFV_MC_NUM			
3	ARFB_Mem_Config_Found	X008	L	850	850	ARFB_MC_FOUND			
4	Memory_Configuration	I620.01	I		820, 860 880	CZ2V_GRT_MC			
5	Memory_Configuration_NBR_For_GPC/String_Change	I160	Ø	860, 850	485, 860	CZ2V_MC_REQ	V91X1592X		X
6	Major_Function_Per_Memory_Configuration	I170	Ø	850, 860		CZ2V_MF_MC			X
7	OPS_Per_Memory_Configuration	I180	Ø	850, 860		CZ2V_OPS_MC	V91X1594X		X
8	GPC_Per_Memory_Configuration	I190	Ø	850, 860		CZ2B_GPC_MC			X
9	String_Per_Memory_Configuration	I200	Ø	850, 860		CZ2B_STRNG_MC			X
10	ICC_GPC_Reconfiguration_ETC	I320.34	Ø	850	850	CZ2B_ICC_FLAG			
11	ICC_NOMINAL_STRING	I320.14	Ø	850	850	CZ2B_ICC_FLAG			
12	ARFV_Item_Cntr	X009	L	850	850	ARFV_ITEM_COUNTER			
13	ARFV_Loop	X010	T	850	850	ARFV_LOOP			
14	ARFB_Item_Num	X011	L	850	850	ARFB_ITEM_NUM			
15	SIMPLEX_ONLY	I210.01	I Ø	810, 850	810, 850	CZ2B_GRT_GPC_SET			
16	ARFB_Item_Error	X012	L	850	850	ARFB_ITEM_ERROR			
17	ARFV_Odd_Num_Tab	X013	L	850	850	ARFV_ITEM_NUM_ODD			
18	ARFV_Strng_Num	X014	L	850	850	ARFV_ITEM_STRG_NUM			
19	Nominal_String_Assignment_Table	I110	Ø	850, 860	351, 485 860	CZ2B_NOM_STRNG		X	X

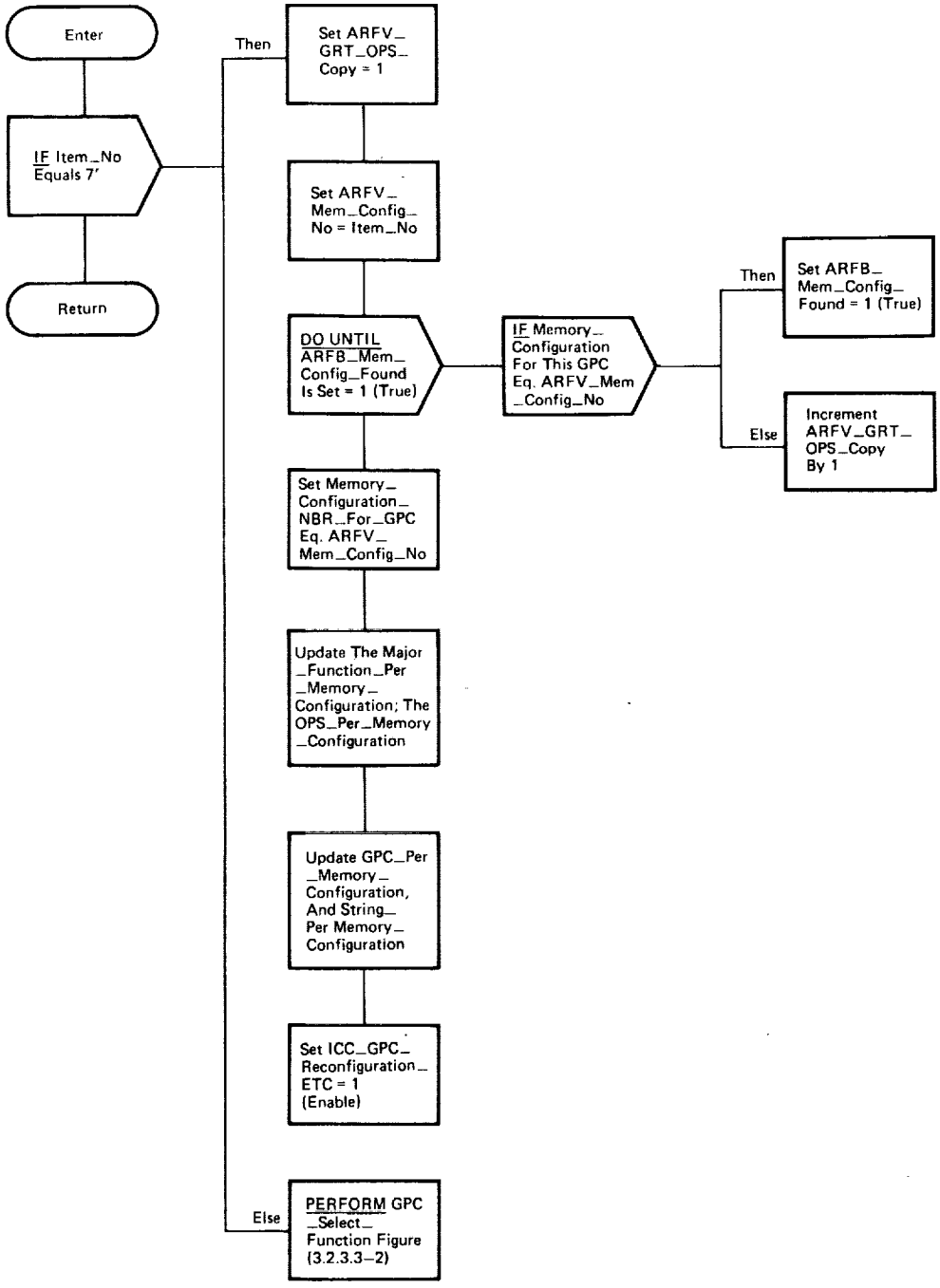


Figure 3.2.3.3-1. GPC_Reconfiguration_Table_Change (ARF_GPC_Table_CHG)

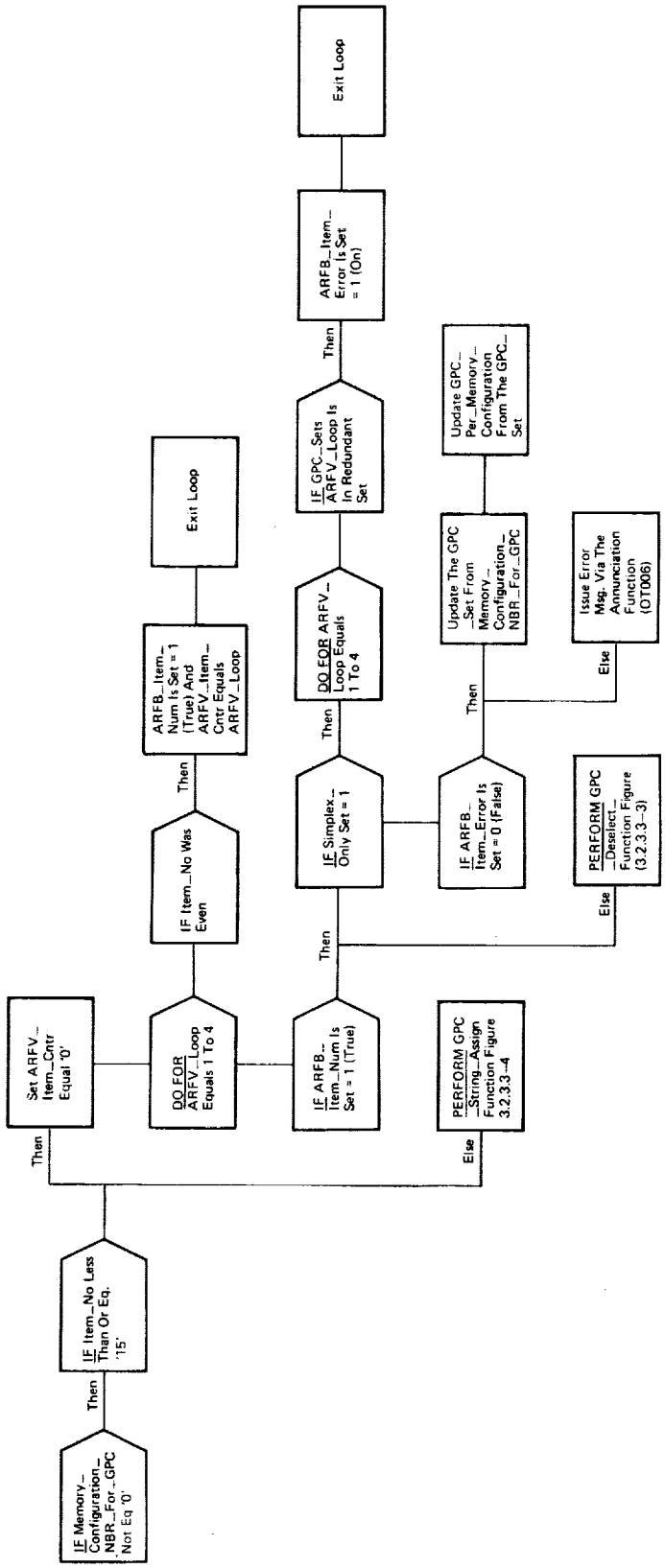


Figure 3.2.3.3-2. GPC_Reconfiguration_Table_Change GPC_Select_Function (850.1)

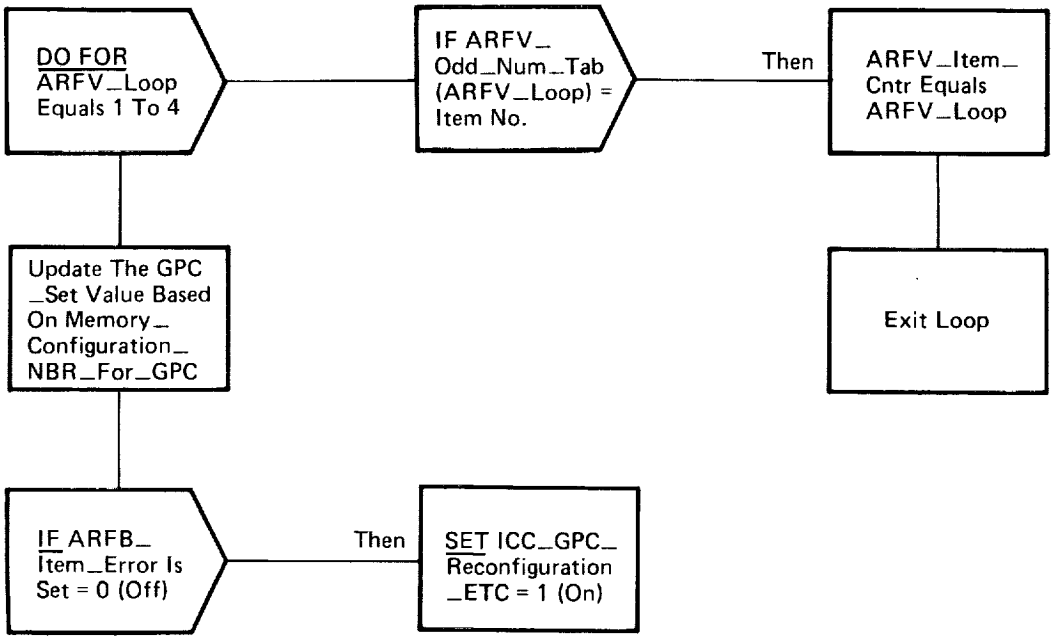


Figure 3.2.3.3-3. GPC_Reconfiguration_Table_Change GPC_Deselect_Function (850.2)

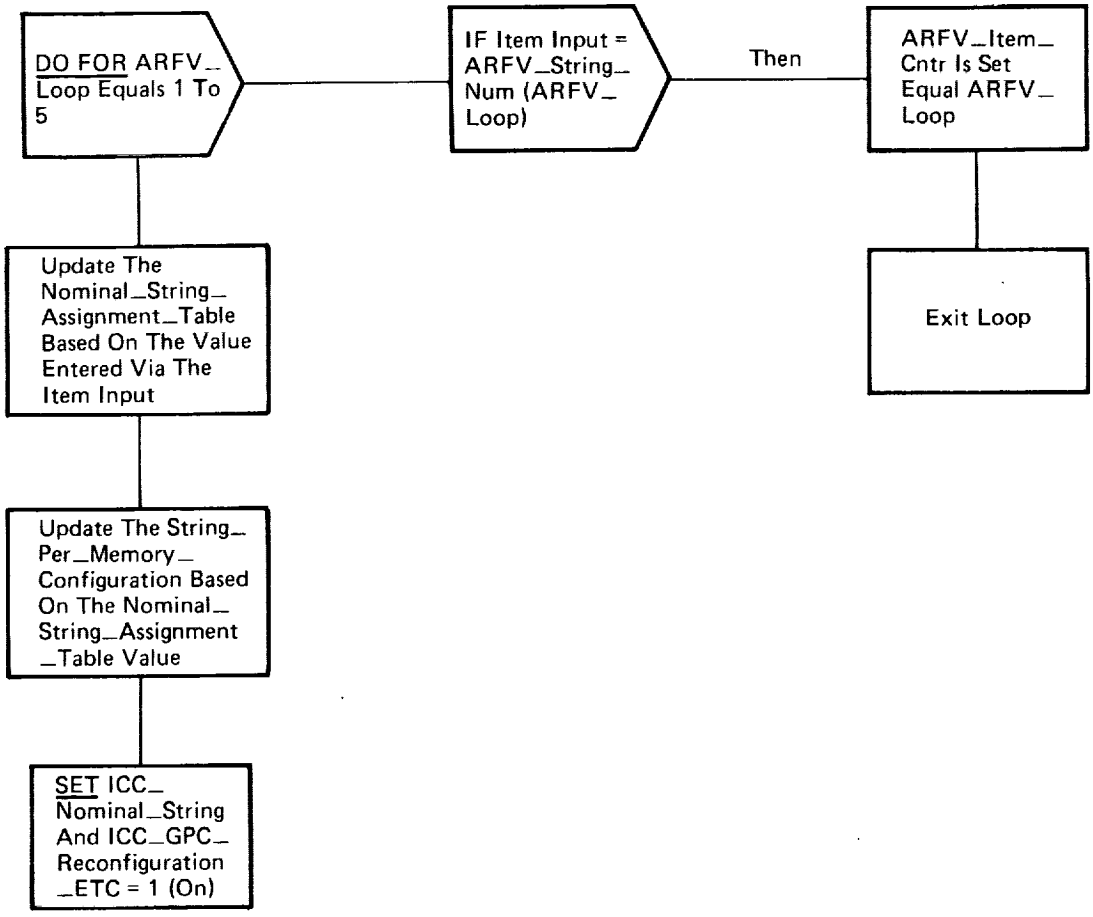


Figure 3.2.3.3-4. GPC_Reconfiguration_Table_Change GPC_String_Assign_Function (850.3)

3.2.3.4 DPS_Configuration_ITEM_Processor (ARF_DPS_CONFIG_ITEM) (860)

ARF_DPS_CONFIG_ITEM services the Item requests made by OPS 0-00, SPEC 0-00, and Mode 00.

a. Control Interface -

1. CALL ARF_DPS_CONFIG_ITEM (D_IND);
2. CALLed by: (810) Idle_Operational_Sequence (ARB_IDLE_OPS)

b. Input - The following User Interface grammar macro supplies input data to this module: ITEM_NO

For a detailed description of this Macro refer to appendix H.

Additional input data descriptions are presented in Table 3.2.3.4-1.

c. Process Description - Upon entering this procedure the ARF_Loop1_Flag is set = 0 (Disabled). Then, using a DO FOR Loop, a test is made to determine the Item entry made on the Configuration Monitor format.

When the item input is determined the ARF_Loop1_Flag is set = 1 (enabled) and the ARF_CASE_NO variable is set equal the ARF_DO_CASE_INDEX value.

A DO CASE processing scheme is then used to process the desired item. A correlation between item inputs and the DO CASE entries that do the processing is as follows:

ITEM #	DO CASE ENTRY
1	1
2	1
3	2
4	2
5	3
6	3
7-20	4
21	5
22	5

Once the appropriate processing has been done to satisfy the item input, control is returned to the IDLE_Operational_Sequence Control segment.

The control flow for this module is presented in Figure 3.2.3.4-1.

d. Outputs - See Table 3.2.3.4-1.

e. Module References -

1. (850) GPC_Reconfiguration_Table_Change (ARF_GPC_TABLE_CHG) is called.

2. (480) ICC_Message_Collector (DIM_ICC_COLLECTOR) is called.

f. Module Attributes - External Procedure

g. Template References -

1.	UI_Section_of_Common_Compool	(CZ1_COMMON)
2.	UI_FOCS_Shared_Compool	(CZ2_COMMON)
3.	ICC_Message_Collector	(DIM_ICC_COLLECTOR)
4.	Annunciation_Macro_Interface	(DMA_MAC)
5.	Annunciation_Compool	(CDL_ANNUN)

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.2.3.4-1

NAME DPS_Configuration_ITEM_Processor (ARF_DPS_CONFIG_ITEM)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	ARF_LOOP1_FLAG	X015	T	860	860	ARF_FLAG1			
2	ARF_CASE_NO	X016.	L	860	860	ARF_K			
3	ARF_DO_CASE_INDEX	X017.	L	860	860	ARF_DO_CASE_INDEX			
4	ARF_I	X018.	T	860	860	ARF_I			
5	ARF_ITEM_NO	X019.	T	860	860	ITEM_NO			
6	RM_VOTER_FLAG	I40.04	O	ILD 860		CZ2B_DPS_STATUS		X	X
7	ARF_ICC_MSG_TYPE	X020.	L	860	495	ARF_VT_ICC_MSG			
8	RM_WATCH_DOG_TIMER_FLAG	I40.03	O	ILD 860		CZ2B_DPS_STATUS	V91X1533X	X	X
9	ARF_C	X021.	T	860	860	ARF_C			
10	MASS_MEMORY/MAJOR_FUNCTION_ASSIGNMENT	I060	O	ILD 860	810	CZ2B_MM_MF			
11	MEMORY_SOURCE	I40.02	O	ILD 860	560 820,880	CZ2B_DPS_STATUS	V93X4371X	X	X

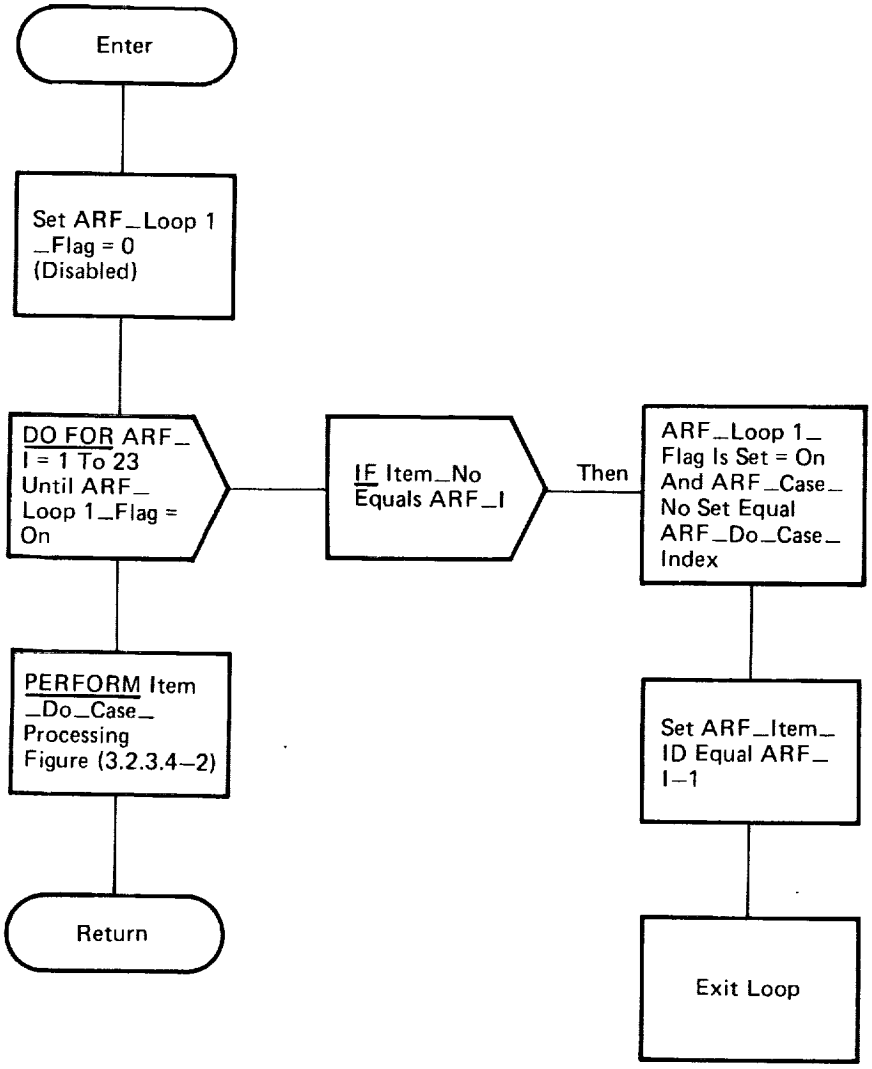


Figure 3.2.3.4-1. DPS_Configuration_Item_Processor (ARF_OPS_ CONFIG_ITEM)

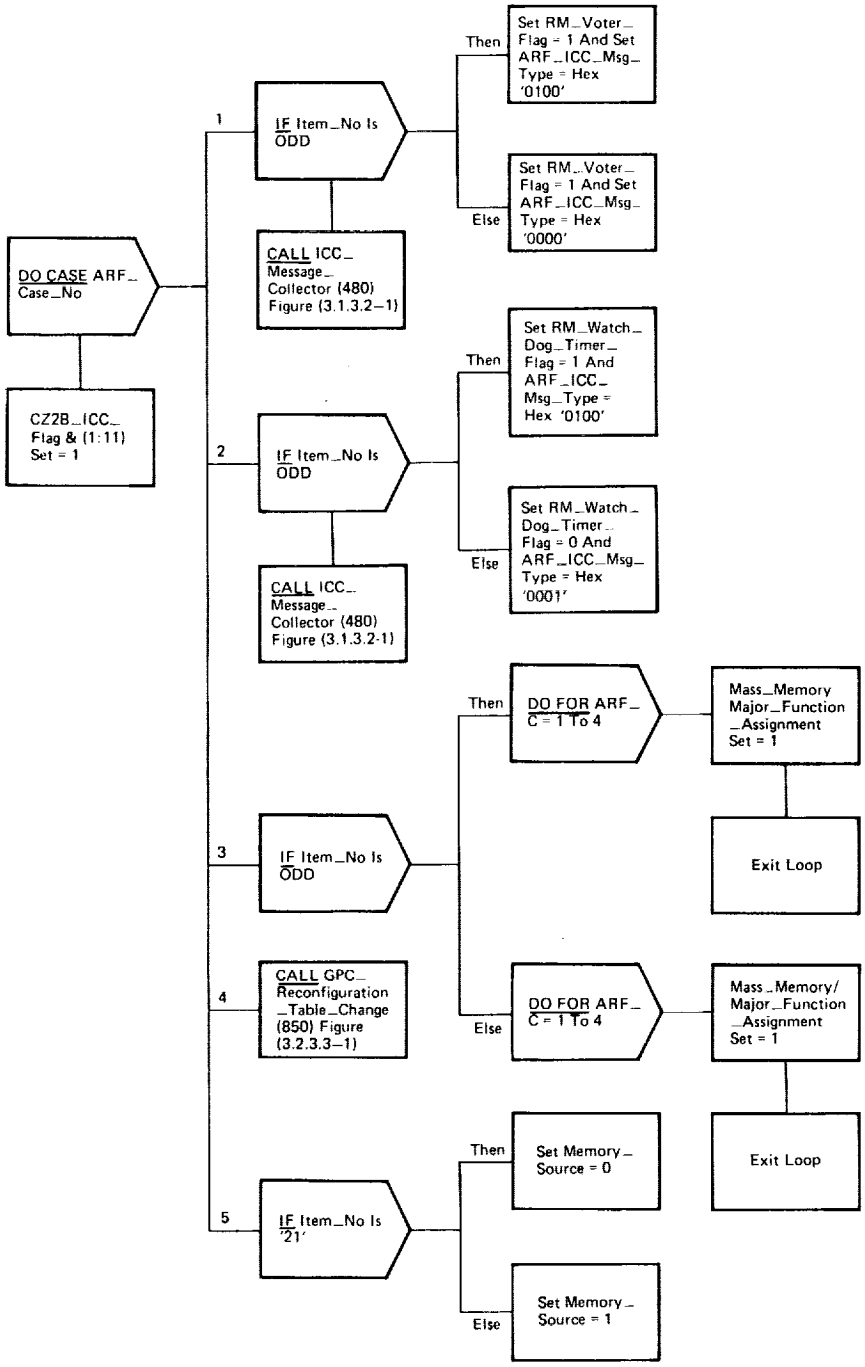


Figure 3.2.3.4-2. DPS_Configuration_Item_Processor Item_Do_Case_Processing (860.1)

3.2.3.5 GPC_Reconfiguration_Message_Handler (ARG_RECONFIG_MSG)(870)

This module provides the interface between the ICC_Message_Router(DME_ICC_ROUT) and (1) the GPC_Reconfiguration(ARC_GPC_RECONFIG) module and (2) the Secondary_GPC_Reconfiguration(ARH_SEC_GPC_RECONFIG) module.

a. Control Interface -

1. CALL ARG_RECONFIG_MSG(ARG_ICC_BUF, ARG_IMT)
2. CALLed by (490) ICC_Message Router(DME_ICC_ROUT)

b. Input - See Table 3.2.3.5-1. - Any of five message types are received by this module over the ICC via the ICC_Message_Router. Types 1, 2, and 3 are sent by GPC_Reconfiguration, type 4 is sent by the Secondary_GPC_Reconfiguration and type 6 is sent by Sequence_Request_Process.

- Type 1 - Reconfiguration overlay is about to take place. Some (perhaps all) of the GPCs in original RS are to be members of the new RS. The Prime GPC has command of the MMU bus while other GPCs will listen on the same MMU channel as Prime is commanding to receive the overlay. The message contains major function ID, OPS number being selected, Mode Number within the OPS, DEU number at which OPS request was made, mask identifying required GPCs, and the index to the applicable GRT data for the OPS.

- Type 2 - A new OPS is to initiated. None of the GPCs of the original RS are to be members of the new RS. The message contains same data as for Type 1.

- Type 3 - The overlay for the new OPS is to proceed now.

- Type 4 - Overlay in secondary GPCs has completed. Successful or unsuccessful status is included.

- Type 6 - Sequence_Request_Processor is finished preparing itself for the reconfiguration.

c. Process Description - This module is required in all GPCs to receive messages transmitted on the ICC whenever a new RS is formed to include GPCs not configured to "hear" the original request from the user. Its purpose is (1) to interface for a minimum of time with the ICC_Message_Router by receiving the message and allowing the ICC_Message_Router to be enabled to receive subsequent ICC messages, (2) to invoke the Secondary_GPC_Reconfiguration module, and (3) to set events to control and coordinate execution of both modules to assure proper operation of both in forming a new RS. The control flow for this module is shown in Figure 3.2.3.5-1. Message types 1 and 2 result in

the SCHEDULE of Secondary_GPC_Reconfiguration while message type 3 provides the primary and secondary GPC's with the signal to proceed with the overlay. Message type 4 is a status message from the secondaries indicating completion of overlay. Message type 4 requires secondaries to cancel any active SPECs and adjust the display buffer for OPS 0-00.

d. Output - See Table 3.2.3.5-2.

e. Module References -

(880) Secondary_GPC_Reconfiguration(ARH_SEC_GPC_RECONFIG) is SCHEDULED
(170) Set_Event_Processor(FPMSET) is CALLed.
(171) Reset_Event_Processor(FPMRESET) is CALLed.
(144) Time/Date_Application_Requests_Processor(FPMTMHAL) is CALLed.
(440) LDB_I/O_Processor (DGI_LDB_IO) is CANCELed.

f. Module Attributes - External Procedure

g. Template References -

1. UI_Section_Of_Common_Compool(CZ1_COMMON)
2. UI_FCOS_Shared_Compool(CZ2_COMMON)
3. FCOS_Compool(FCMCOM)
4. UI_General_Compool(CDM_UI_COMPOOL)

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation -

1. All secondary GPCs are allowed to return control back to Sequence_Request_Processor. On completion of that function, Sequence_Request_Processor issues message type 6 and this process counts each message as it comes in and only schedules Secondary_GPC_Reconfiguration after all have responded.

DATA TABLE 3.2.3.5-1

NAME GPC_Reconfiguration_Message_Handler (ARG_RECONFIG_MSG)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	Message_Type	L278	I/O	870	870	ARG_MSG_TYPE			
2	ICC_Messages	I610.18	I	480 485,490	See App. E	CZ2B_ICC_MSG_BUF			
3	Last_SIP_Time	I570	I	870,880	870	CZ2V_REC_TSIP			
4	Reconfiguration_Message_Type	I480	O	870	880	CZ2V_REC_MSG_TYPE			
5	Overlay_In_Progress_Flag	I010.17	O	820 870,880		CZ2B_STATUS			
6	ICC_GST_Status	I320.09	O	See App. E	142	CZ2B_ICC_FLAG			
7	Reconfiguration_Table_Index	I490	O	560, 870	560,810 820,880	CZ2Y_REC_GNT_INDEX			
8	Reconfiguration_Redundant_Set	I560	O	870	880	CZ2B_REC_RS			
9	GSE_Poll_State	I300.02	O	495, 820 830, 870	830	CZ2B_STATUS_FLAG			
10	GPC_Secondary_Reconfiguration_Request	J040.14	O	505, 820 870	505	CZ1B_D_FLAGS			
11	MACT_Service_Request_Ready_Events	J160	O	See App. E	505, 810	CZ1E_D_MACT_EVTS			
12	Application_Service_Request_Event	B050	O	540, 820 870	660 810, 820	CDME_APP_SERVICE			
13	Proceed_With_Reconfiguration	I470	O	820, 870,880	820 880	CZ2E_REC_PROCEED			
14	GPC_ID	#001	I	300	App. E	TFCMID			

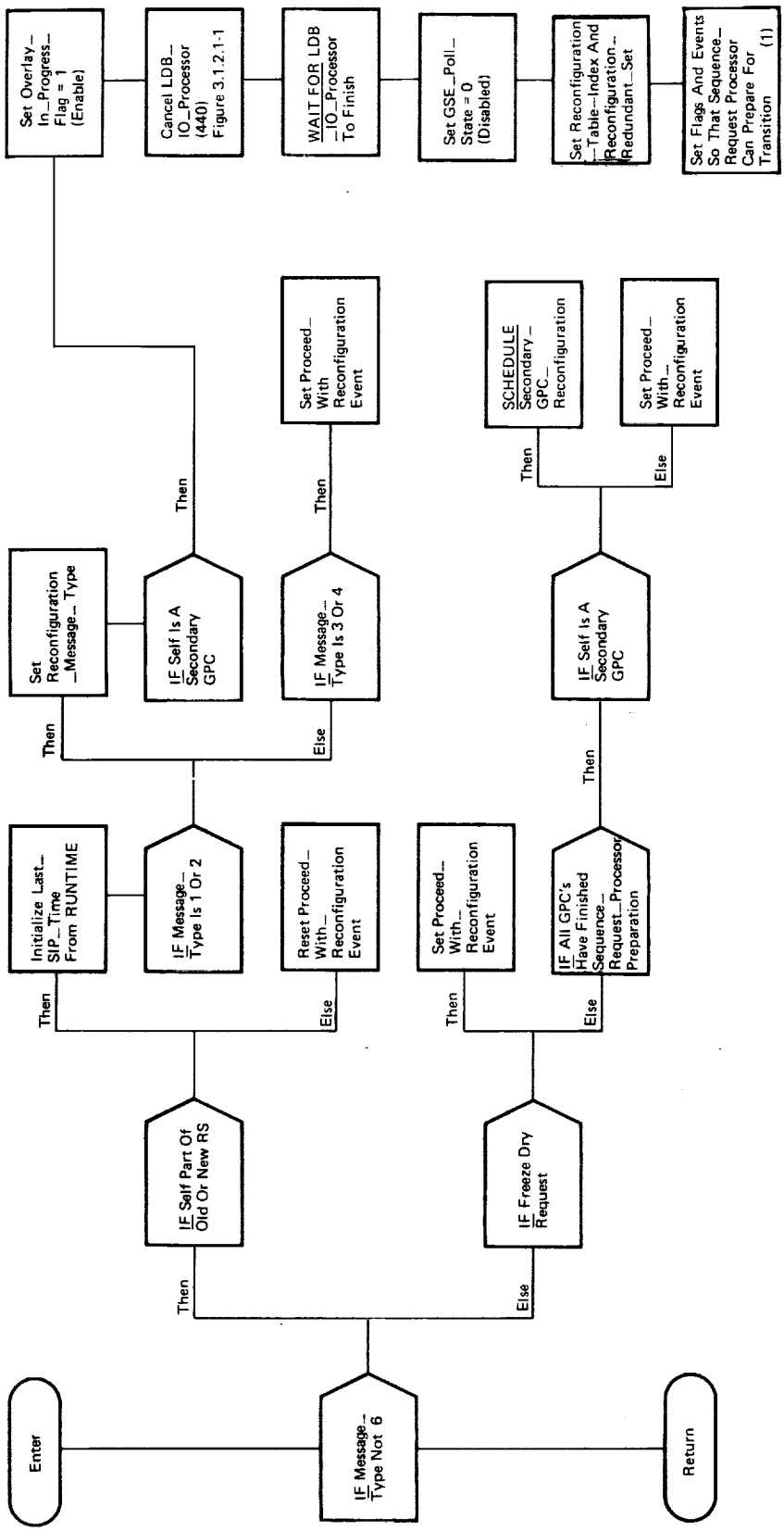


Figure 3.2.3.5-1. GPC_Reconfiguration_Message_Handler (ARG_RECONFIG_MSG.)

3.2.3.6 Secondary_GPC_Reconfiguration (ARH_SEC_GPC_RECONFIG)(880)

This module operates in GPCs about to become members of the RS. It responds to data passed to it by the old RS to prepare the GPC for loading and starting the new requested OPS.

a. Control Interface -

1. SCHEDULE ARH_SEC_GPC_RECONFIG PRIORITY (PRIO_ARH)
2. a. SCHEDULED by (870) GPC_Reconfiguration_Message_Handler (ARG_SEC_GPC_RECONFIG).
   b. Event Proceed_With_Reconfiguration (CZ2E_REC_PROCEED) set by (870) GPC_Reconfiguration_Message_Handler (ARG_RECONFIG_MSG).

b. Input - See Table 3.2.3.6-1.

c. Process Description - This module processes information supplied to it for the purpose of reconfiguring the GPC set. Secondary means that this GPC is about to become a member of the GPC set servicing a major function but is not now in that role. It operates in conjunction with the GPC_Reconfiguration module via messages sent and received over the ICC. Messages received are serviced by the GPC_Reconfiguration_Message_Handler. The secondary GPCs may form a redundant set by themselves or with the GPC(s) "hearing" the original OPS request.

The "hearing" GPC sends a message to the new GPCs to alert them to their role as secondary GPCs. Depending on memory and current GPC configuration, messages are passed between the original GPCs and the secondaries as they proceed through the steps of overlay and OPS initiation to assure coordinated formation of the RS. The functional control flow for this module is shown in Figure 3.2.3.6-1.

d. Output - See Table 3.2.3.6-2.

e. Module References -

1. (480) ICC_Message_Collector(DIM_ICC_COLLECTOR) is CALLed.
2. (660) Downlist_Formatter(DCDDWL1) is CALLed.
3. (440) LDB_I/O_Processor(DGI_LDB_IO) is CANCELed.
4. (170) Set_Event_Processor(FPMSET) is CALLed.
5. (171) Reset_Event_Processor(FPMRESET) is CALLed.
6. (104) Wait_Processor(FPMWAIT) is CALLed.
7. (144) Time/Date_Application_Requests_Processor(FPMTMHAL) is CALLed.
8. (340) Miscellaneous_CM_Request_Processor(FCMSVC) is CALLed.
9. (390) Sync_Mask_Build_Routine (FCMSMASK) is CALLed.
10. (320) Overlay_Processor(FCMPOVLY) is CALLed.
11. (665) Downlist_Formatter_2 (DCDDWL2) is CALLed.

f. Module Attributes - Program

g. Template References -

1. UI_Section_Of_Common_Compool(CZ1_COMMON)
2. UI_FCOS_Shared_Compool(CZ2_COMMON)
3. FCOS_Compool(FCMCOM)
4. UI_General_Compool(CDM_UI_COMPOOL)

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.2.3.6-1

NAME Secondary_GPC_Reconfiguration (ARH_SEC_GPC_RECONFIG)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	Reconfiguration_Table_Index	I490	I	870 560	880, 820 810, 560	CZ2V_REC_GRT_INDEX			
2	GPC_ID	#001	I	300	See App. E	TFCMID			
3	Reconfiguration_Redundant_Set	I560	I	870	880	CZ2B_REC_RS			
4	Reconfiguration_Overlay_Status	I550	O	820	810	CZ2B_REC_OVL_STATUS			
5	Memory_Source	I040.02	I	860	820 880, 560	CZ2B_DPS_STATUS	V93X4371X	X	X
6	Major_Function_Overlay	I620.03	I		820, 880	CZ2V_GRT_MFOVL			
7	Major_Function_Overlay_Number	I010.04	I/O	820, 880	820, 880	CZ2V_MF_OVLY			
8	Program_Overlay	I620.02	I		820, 880	CZ2V_GRT_POVL			
9	Program_Overlay_Number	I010.03	I/O	820, 880	820, 880	CZ2V_PROG_OVLY			
10	Reconfiguration_Message_Type	I480	I	870	880	CZ2V_REC_MSG_TYPE			
11	Proceed_With_Reconfiguration	I470	I	870, 820 880	820 880	CZ2E_REC_PROCEED			
12	BITE_Update_Flag	I460	O	820 880	910, 665 660	CZ2V_BITE_UPD_VAL			
13	Flight_Critical_BTU_BITE	I100.01	O	820, 920 880	665, 660 485	CZ2B_BTU_BITE_FC		X	X
14	Payload_BTU_BITE	I100.10	O	See App. E		CZ2B_BTU_BITE_PL		X	X
15	PCMMU_BTU_BITE	I100.13	O	820, 880	920	CZ2B_BTU_BITE_PCM	V72M8261P	X	X
16	MMU1_BTU_BITE	I100.14	O	280, 820 880, 282	282	CZ2B_BTU_BITE_MM1		X	X
17	MMU2_BTU_BITE	I100.15	O	280, 820 880	282	CZ2B_BTU_BITE_MM2		X	X
18	MTU1_BTU_BITE	I100.16	O	361, 149 820, 880	660 665	CZ2B_BTU_BITE_MTU1			
19	MTU2_BTU_BITE	I100.17	O	361, 820 149, 880		CZ2B_BTU_BITE_MTU2			
20	MTU3_BTU_BITE	I100.18	O	361, 149 820, 880		CZ2B_BTU_BITE_MTU3			

DATA TABLE 3.2.3.6-1 (Cont'd.)

NAME Secondary_GPC_Reconfiguration (ARH_SEC_GPC_RECONFIG)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
21	DEU_BTU_BITE	I100.19	O	405, 820 880	660 665	CZ2B_BTU_BITE_DEU	See App. E	X	X
22	ICC_BTU_BITE	I320.13	O	820, 880		CZ2B_ICC_FLAG			
23	CAM_Counters	I240	O	375, 820 880	485, 660 665	CZ2V_CAM_CNTRS			
24	ICC_CAM_GPC1-5	I320.17 - .21	O	820 880		CZ2B_ICC_FLAG			
25	CAM_Status_Votes	I010.18	O	375, 820 880	620	CZ2B_STATUS	See App. E		X
26	ICC_GST_Status	I320.09	O	See App. E	142	CZ2B_ICC_FLAG			
27	T_Zero	I020	I	700	485, 700 820, 880	CZ2V_TZERO			
28	Downlist_Formatter_Enabled_Flag	I010.16	O	560, 700 820, 880	710, 750	CZ2B_STATUS			
29	Reconfiguration_Status	I540	I/O	560, 810 820, 880	560, 810 820, 880	CZ2V_REC_XERR			
30	GPC_Reconfiguration_Completion	J040.13	O	505, 820 870, 880	505 560	CZ1B_D_FLAGS			
31	MACT_Service_Request_Flags	J040.01	O	See App. E	505, 540 810, 820	CZ1B_D_FLAGS			
32	Application_Service_Request_Event	B050	O	540, 820 870, 880	660 810, 820	CDME_APP_SERVICE			
33	Overlay_In_Progress_Flag	I010.17	O	820, 870 880		CZ2B_STATUS			
34	Redundant_Set_Mask	I010.13	O	See App. E		CZ2B_RS	V91M2070P V91M2104P	X	
							V91M2138P V91M2172P		
35	ICC_CS/RS	I320.08	O	820, 880		CZ2B_ICC_FLAG			
36	Reconfiguration_Major_Function	I510	I	560	560, 820 880	CZ2V_REC_MF			
37	Memory_Configuration	I620.01	I		820, 860 880	CZ2V_GRT_MC			
38	DEU_Update_Rate	B010.20	O	500, 620 820, 880	620,625	CDMV_MAT_DEU_RATE			
39	MCDS_Input_Busy	I680	I	400	See App. E	CZ2V_MCDS_IN			

DATA TABLE 3.2.3.6-1 (Cont'd.)

NAME Secondary_GPC_Reconfiguration (ARH_SEC_GPC_RECONFIG)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
40	DEU_Poll_Flags	J062	O	See App. E		CZ1B_DEU_POLL_FLAG			
41	CRT_Status_Flags	B010.40	O	See App. E		CDMB_MAT_CRT_STAT			
42	Current_OPS	I010.02	O	560, 820 880	See App. E	CZ2V_CURRENT_OPS			
43	Reconfiguration_OPS_Request	I520	I	560 820	820, 560 880	CZ2V_REC_OPS			
44	ICC_Current_Load	I320.06	O	820, 880	560, 810	CZ2B_ICC_FLAG			
45	Memory_Configuration_Number	I010.05	O	405, 820 880	See App. E	CZ2V_MC	V91U1546C V91U1547C	X	X
							V91U1548C V91U1549C		
46	Last_SIP_Time	I570	O	870, 880	870	CZ2V_REC_TSIP			

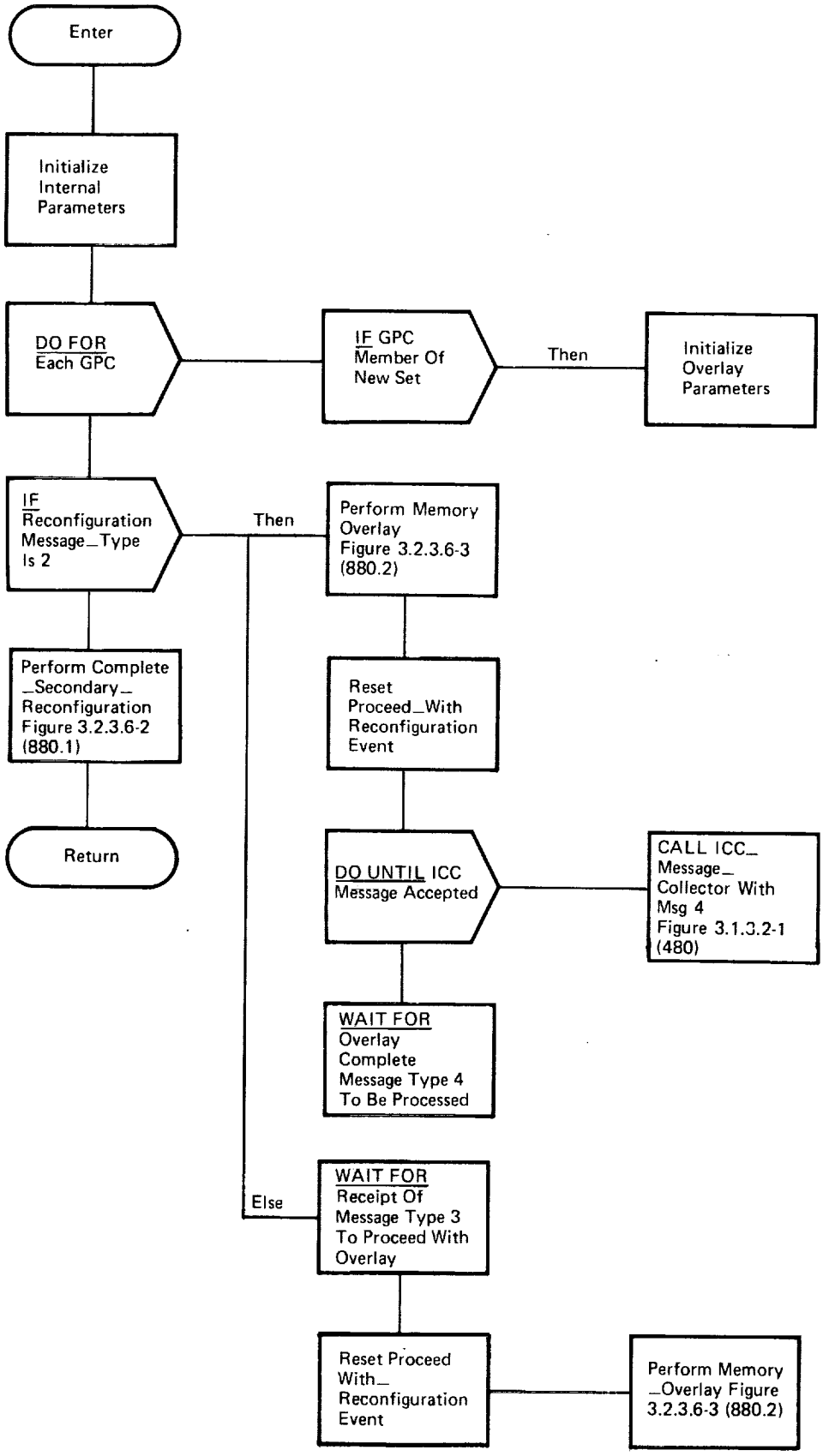


Figure 3.2.3.6-1. Secondary_GPC_Reconfiguration (ARH_SEC_GPC_REC)

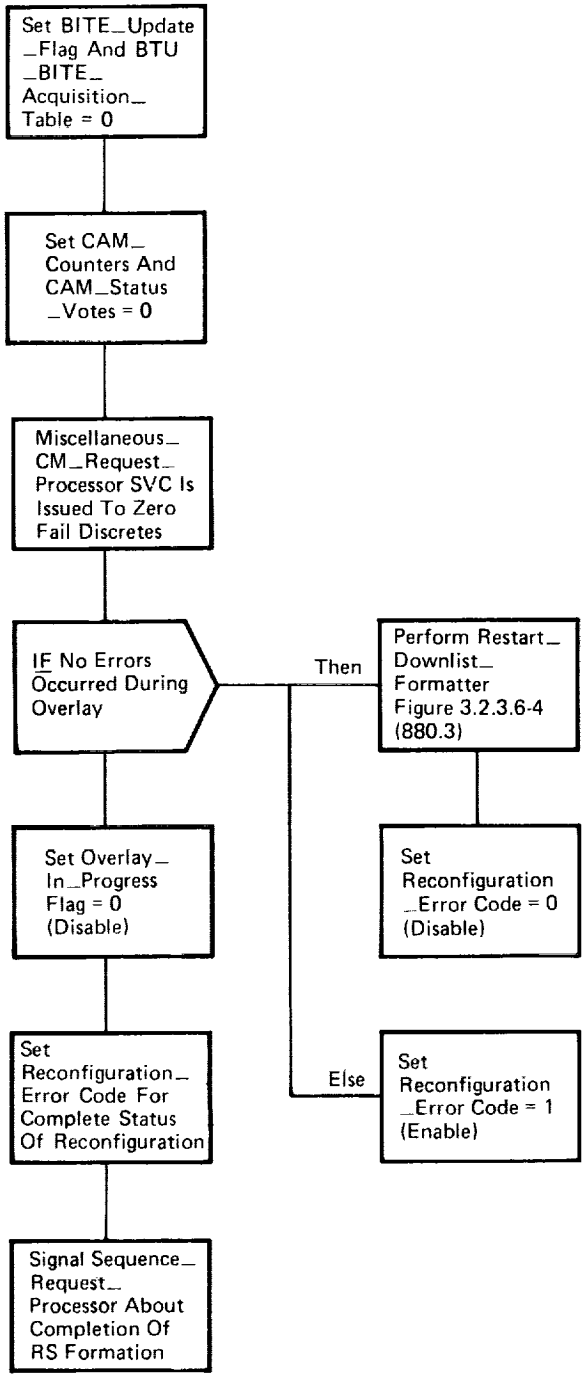


Figure 3.2.3.6-2. Secondary GPC_Reconfiguration Complete_Secondary_Reconfiguration (880.1)

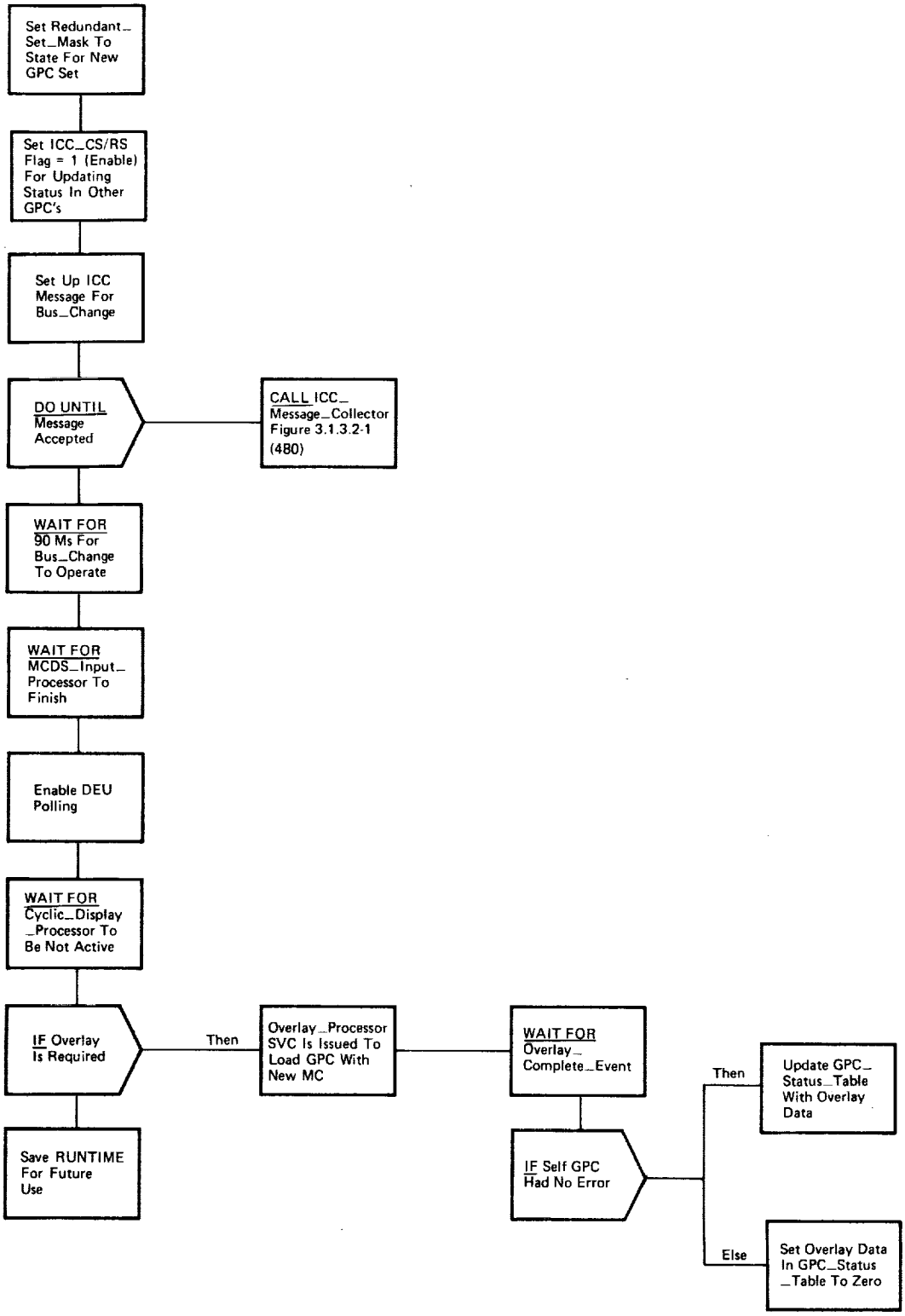


Figure 3.2.3.6-3. Secondary_GPC_Reconfiguration Memory_Overlay (880.2)

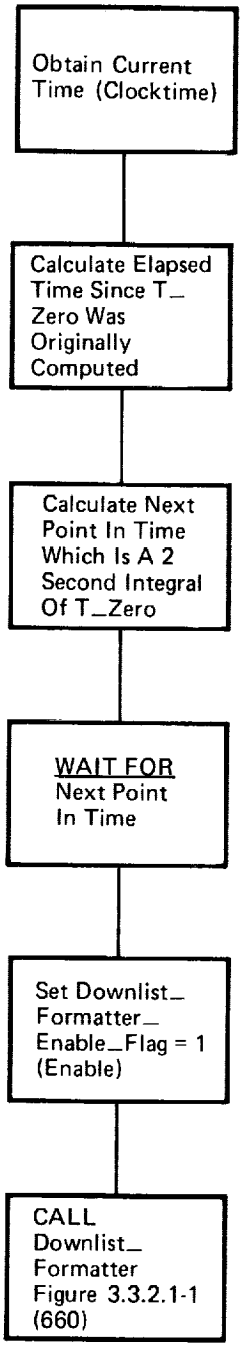


Figure 3.2.3.6-4. Secondary_GPC_Reconfiguration Restart_Downlist_Formatter (880.3)

3.3 System Specialist Functions

The System Specialist Functions are the set of specialist functions (SPEC) which are system oriented as compared to being directly related to a major function. These SPEC's are capable of being initiated from any user position regardless of the major function switch setting. They are permanently resident in memory with other System Services functions because of their commonality across major functions. Figure 3.3-1 presents the hierarchial diagram.

a. Read/Write_Specialist_Function (ASB_RD_WRT) provides a general GPC memory read and write capability. The associated display provides the means for the user to enter and read to or from either protected or unprotected GPC memory. Control of BTU Bite Acquisition and main memory dump are performed by this Specialist Function.

b. Time_Management_Specialist_Function (ASC_TIME_MGMT) provides control of the Mission Time (MT) function and updates to the Master Timing Unit (MTU) as well as monitoring the status of the time sources from the MTU accumulator and internal GPC prime clock.

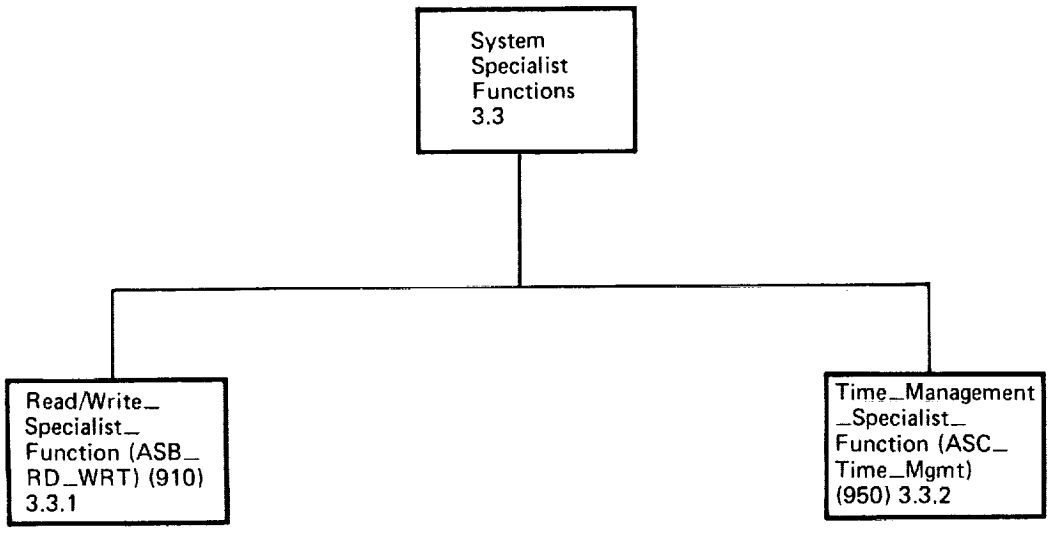


Figure 3.3-1. System Specialist Function Hierarchial Diagram

3.3.1 Read/Write_Specialist_Function (ASB-RD_WRT) (910)

ASB_RD_WRT consists of all the functions required to support the GPC memory read/write display.

a. Control Interface -

1. SCHEDULE ASB_RD_WRT PRIORITY (PRIO_ASB)
2. SCHEDULED by: (560) Sequence_Request_Processor (DMC_SEQ_REQ_PROC)

b. Input - The following User Interface grammar macros supply input data to this module:

ITEM_NO
ITEM_O
ITEM_I

For a detailed description of these macros refer to appendis H.

Additional input data descriptions are presented in Table 3.3.1-1.

c. Process Description - Upon initial entry into this program a test is made to determine the current mode of operation, DATA or CODE. The variables ADD_ID_Blanking_Bit, Desired_Data_Blanking_Bit, and Current_Data_Blanking_Bit are set = 1 (enabled) so that upon initial request of the format all data fields for "ADD ID", "DESIRED", and "CURRENT" are blanked out. The program Read/Write_Cyclic_Display_Update (ASH_RW_CYC_UPDATE) it then scheduled.

A DO FOR loop is used to determine which of the possible 22 item inputs was made. If the item input affects one of the six lines used for memory display then the variable ASB_LINE_NO is updated to reflect the line being addressed. The variable ASB_CASE_NO is also updated with the do case index value required to process the item input.

A DO CASE processing scheme is then used to process the desired item. A correlation between item inputs and the DO CASE entries that do the processing is as follows:

Item #	DO CASE ENTRY
5	1
7	1
9	1
11	1
13	1
15	1

ITEM #	DO CASE ENTRY
17&18	2
6	3
8	3
10	3
12	3
14	3
16	3
4	4
3	5
19	6
1&2	7
20	8
21	9
22	10

If a PRO keystroke, rather than an item request, was made, a CHANGE(0) grammar macro statement is executed, clean up processing for the Block and Spec is done via the grammar macro support and the Read/Write_Cyclic_Display_Update program is canceled.

Otherwise the logic flow returns to the DISPLAY grammar statement to wait for the next keyboard input.

The control flow for this module is presented in Figure 3.3.1-1.

d. Outputs - See Table 3.3.1-1.

e. Module References -

1. (915) Read/Write_Cyclic_Display_Update (ASH_RW_CYC_UPDATE) is SCHEDULED.

2. (480) ICC_Message_Collector (DIM_ICC_COLLECTOR) is CALLed.

3. (920) Bite_Execute_Item_Processor (ASI_RW_BITE_PROC)
4. (540) Display_Presentation_And_Control (DIS_PLAY) is CALLed.

f. Module Attributes - Program

g. Template References -

1. UI_Section_of_Common Compool (CZ1_COMMON)
2. UI_FCOS_Shared_Compool (CZ2_COMMON)
3. Bite_Execute_Item_Processor (ASI_RW_BITE_PROC)
4. Display_Presentation_and_Control (DIS_PLAY)
5. (550) Application_Moding_And_Sequences is CALLed.

5. Application_Moding_and_Sequencor (DNX_BMS)
6. Read/Write_Cyclic_Display_Update (ASH_RW_CYC_UPDATE)
7. Memory_Read/Write_Compool (CDJ_RW_DATA)
8. Annunciation_Macro_Interface (DMA_MAC)
9. Annunciation_Compool (CDL_ANNUN)
10. Downlist_Compool (CDW_DOWNLIST_COMPOOL)
11. ICC_Message_Collector (DIM_ICC_COLLECTOR)

h. Error Handling -

1. If a GPC location is input, that exceeds the limits of the core size, an error message will be output via the annunciation function (DU031).

2. Whenever there is an incompatability between the mode of operation, CODE or DATA, and the GPC address specified when attempting to write into storage an error message will be output via the annunciation function (DU023).

3. An attempt to update GPC memory without having specified a desired value will result in an error message via the annunciation function (DU022).

4. An attempt to update GPC memory without indicating the address ID will result in an error message via the annunciation function (DU025).

5. Attempting to perform a sequential read operation without having first indicated the beginning address ID will result in an error message via the annunciation function (DU021).

6. If the downlist function is enabled and either the dump starting address or the number of words has not been input an error message will be output via the annunciation function (DU032).

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.3.1-1

NAME Read/Write-Specialist-Function (ASB_RD_WRT)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	ASB_LINE_NO	X022	T	910	910	ASB_I			
2	ASB_CASE_NO	X023	T	910	910	ASB_K			
3	READ_WRITE_END_OF_LOOP_INDICATOR	W090	I/O	910	910	CDJV_FLAG_1			
4	ASB_ITEM_O	X024	T	910	910	ASB_ITEM_O			
5	GPC_ADDRESS_ID_SAVE_AREA	W040.	I/O	910,915	910 915	CDJB_R_W_LOCID			
6	ASB_MEM_INDEX	X025	T	910	910	ASB_MEM_INDEX			
7	ADD_ID_VALIDITY_BIT	W010.2	I/O	910	910 915	CDJB_R_W_L			
8	GSE_POLL_FLAG	I040.01	O	860 910	830	CZ2B_DPS_STATUS		x	x
9	ASB_GSE_ICC_MSG	X026	O	910	495 910	ASB_GSE_ICC_MSG			
10	DESIRED_DATA_SAVE_AREA	W050.	I/O	910	910	CDJB_R_W_DSRDO			
11	DESIRED_DATA_VALIDITY_BIT	W020.2	I/O	910	910	CDJB_R_W_D			
12	ICC_DST_STATUS	I320.11	O	351,910, 950		CZ2B_ICC_FLAG			
13	BITE_UPDATE_FLAG	I460	I	820 880	660,665 910	CZ2V_BITE_UPD_VAL			
14	ASB_LOOP_CNT	X027	T	910	910	ASB_J			
15	DESIRED_VALUE_PRESENT_IND	W070.4	I/O	910	910	CDJB_R_W_EXC_STAT			
16	LOC_ID_PRESENT_IND	W070.3	I/O	910	910	CDJB_R_W_EXC_STAT			
17	DESIRED_DATA_BLANKING_BIT	W020.1	I/O	910	910	CDJB_R_W_D			
18	CURRENT_DATA_SAVE_AREA	W060.	O	910 915	910 915	CDJB_R_W_CURRO			
19	PROTECT_INTERRUPT	I300.01	O	325,910	910	CZ2B_STATUS_FLAG			
20	CODE_MODE_SELECT	W070.2	I/O		910	CDJB_R_W_EXC_STAT			

DATA TABLE 3.3.1-1 (Cont'd.)

NAME Read/Write-Specialist-Function (ASB_RD_WRT)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
21	DATA_MODE_SELECT	W070.1	I/O	910	910	CDJB_R_W_EXC_STAT			
22	ASB_LINE_CNT	X028	T	910	910	ASB_K			
23	ASB_MEM_LOC	X029	T	910	910	ASB_A			
24	ASB_MEM_MAX	X030	L	910	910	ASB_MEM_MAX			
25	ASB_LOC_THIS_LINE	X031	T	910	910	ASB_ITEM_C			
26	ADD_ID_BLANKING BIT	W010.1	I/O	910	910 915	CDJB_R_W_L			
27	CURRENT_DATA_BLANKING_BIT	W030.1	I/O	910 915	910 915	CDJB_R_W_C			
28	Main_Memory_Dump_Request	F040	O	660,665 910	915	CDWD_MAIN_MEM_ DMP_REQ			
29	Main_Memory_Dump_Length	F070	O	660,665 910	915	CDWD_MAIN_MEM_ DMP_LEN			
30	Main_Memory_Dump_Start_Address	F060	O	660,665 910	915	CDWD_MAIN_MEM_ DMP_PTR			

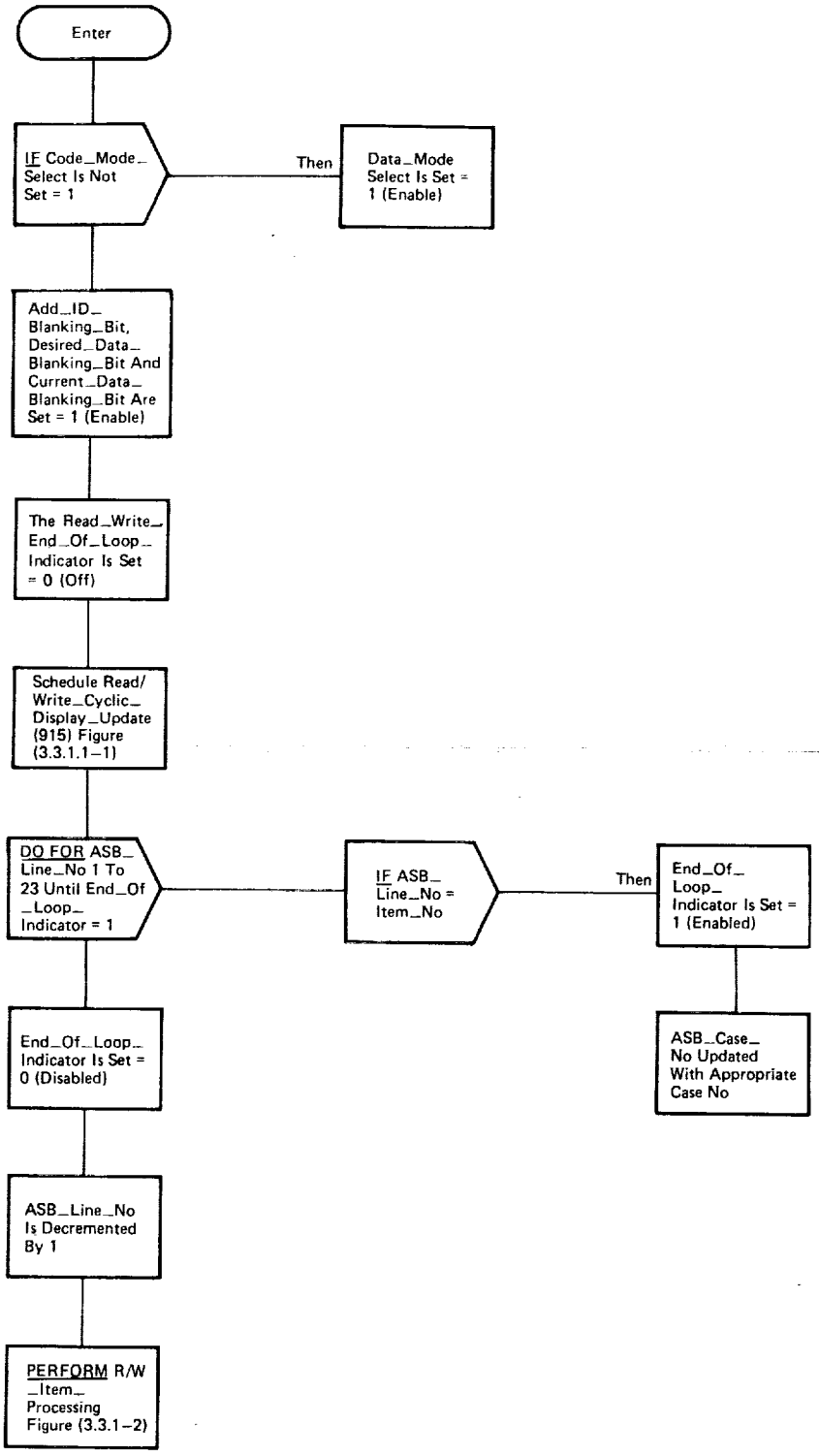


Figure 3.3.1-1. Read/Write_Specialist_Function (ASB_RD_WRT)

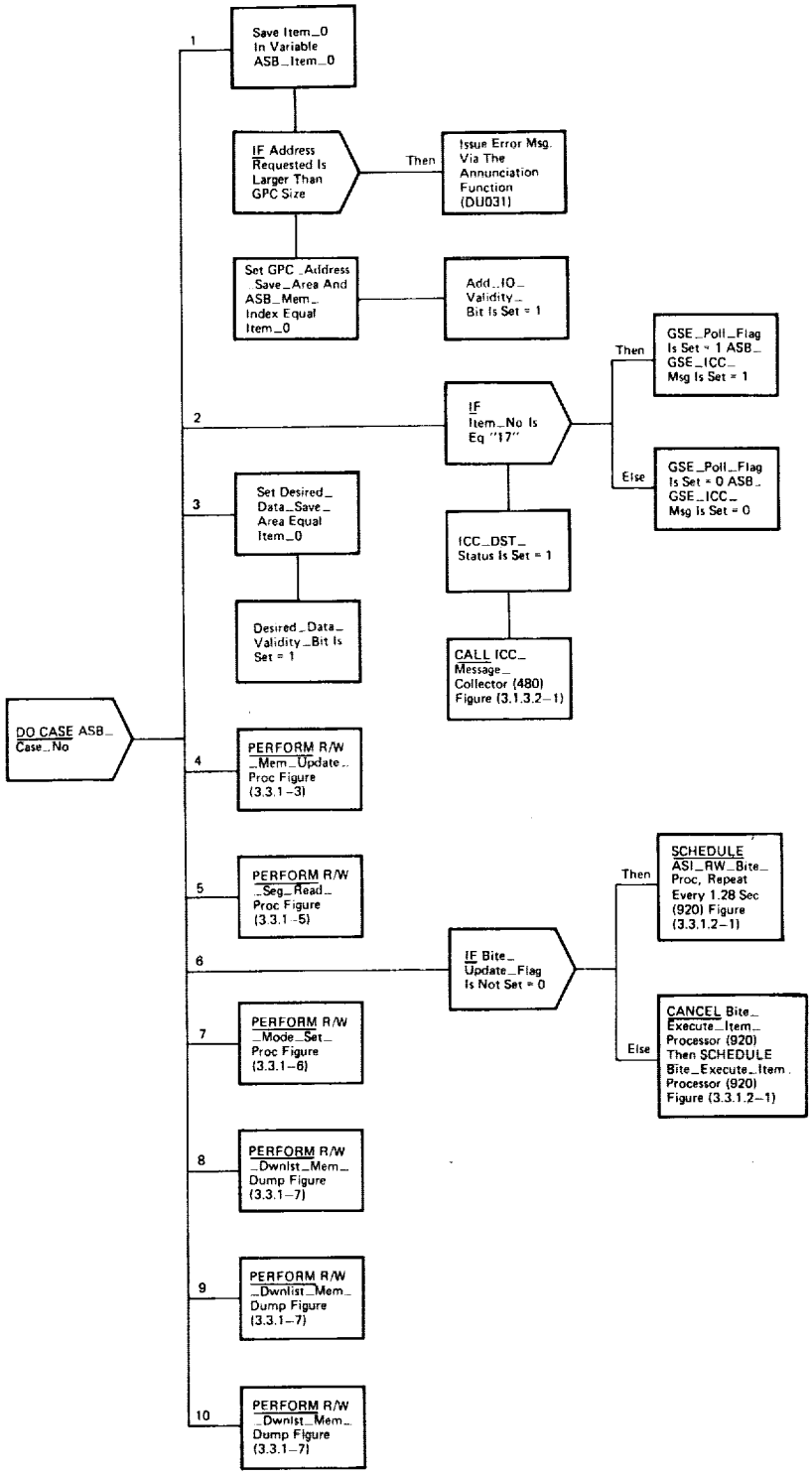


Figure 3.3.1-2. Read/Write_Specialist_Function (R/W_Item_Processing) (910.1)

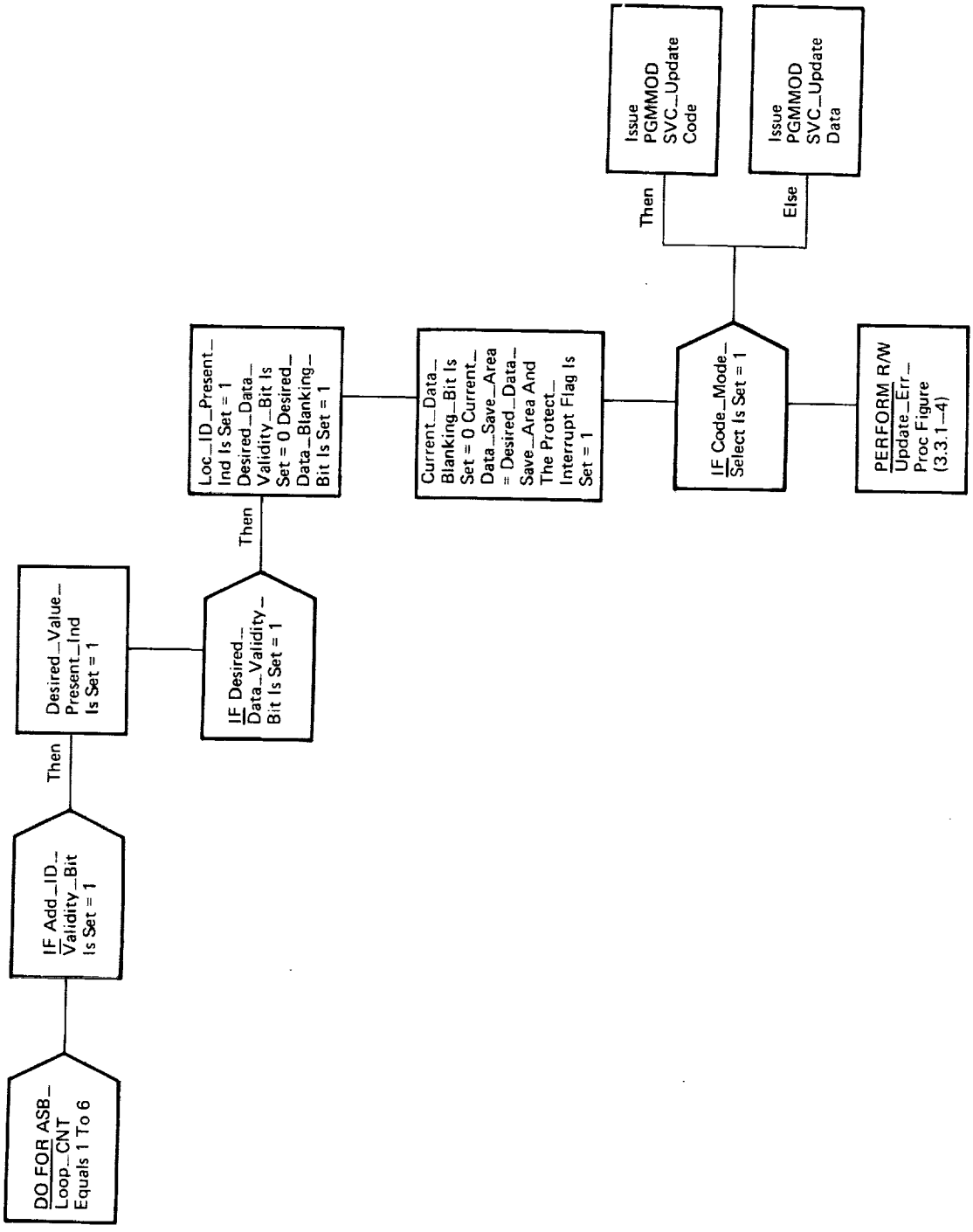


Figure 3.3.1-3. Read/Write_Specialist_Function (R/W_Mem_Update_Proc) (910.2)

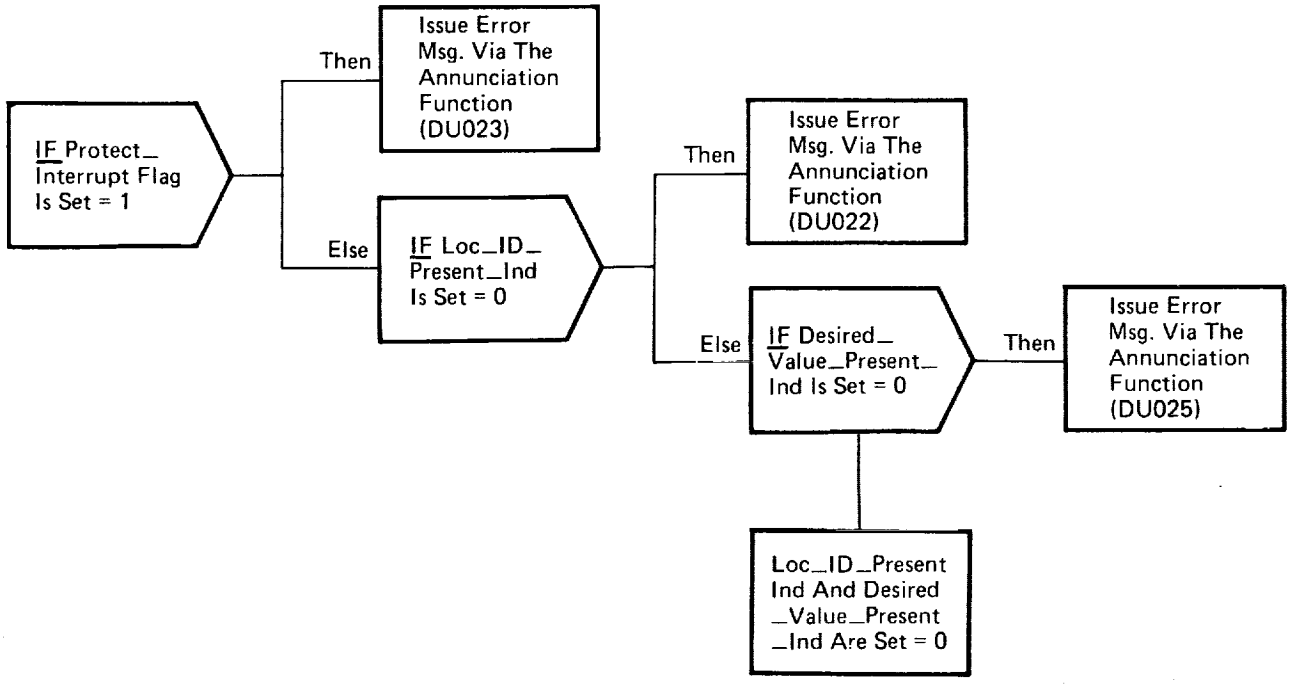


Figure 3.3.1-4. Read/Write_Specialist_Function (R/W_Update_ERR_Proc) (910.3)

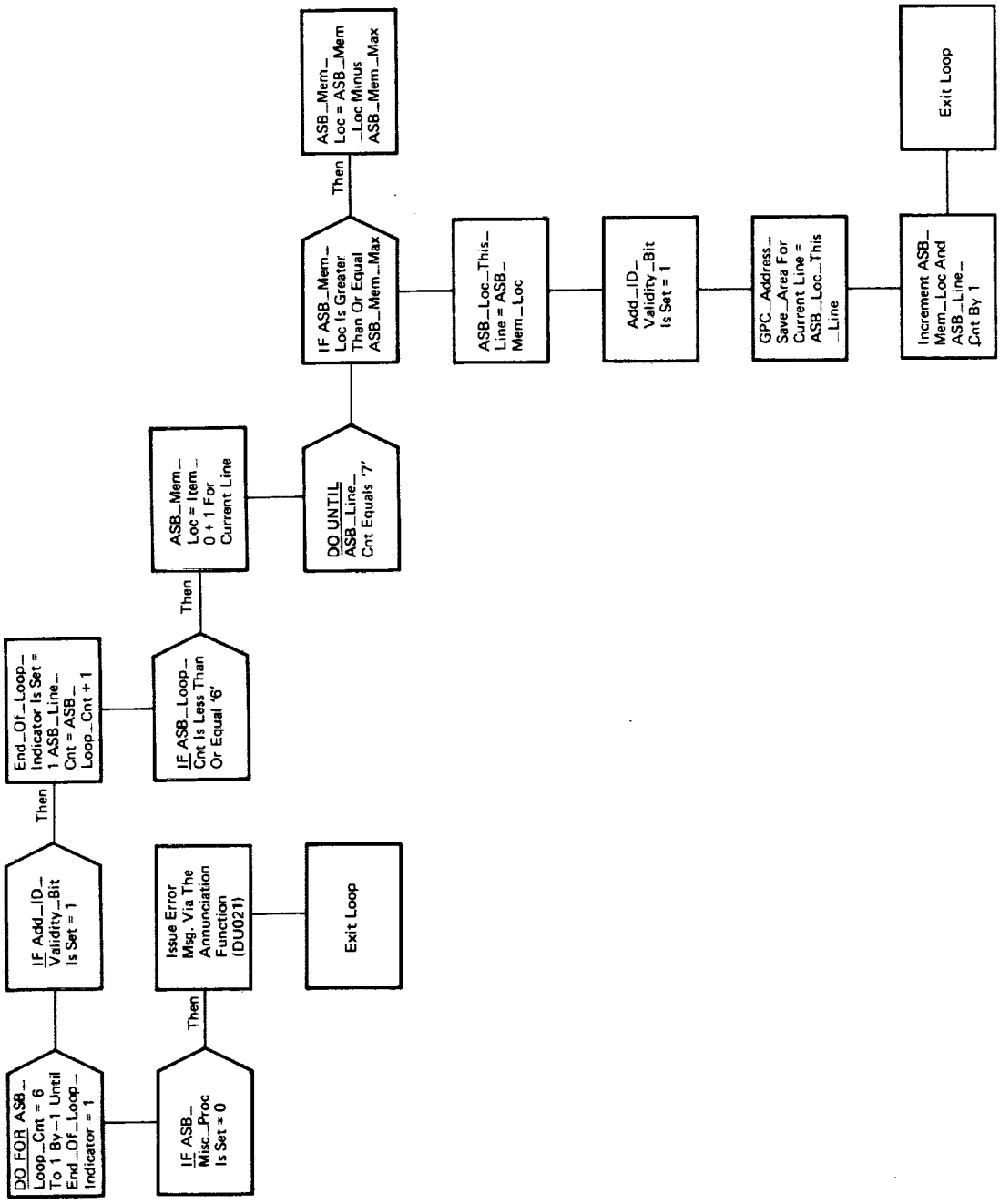


Figure 3.3.1-5. Read/Write_Specialist_Function (R/W_Seq_Read_Proc) (910.4)

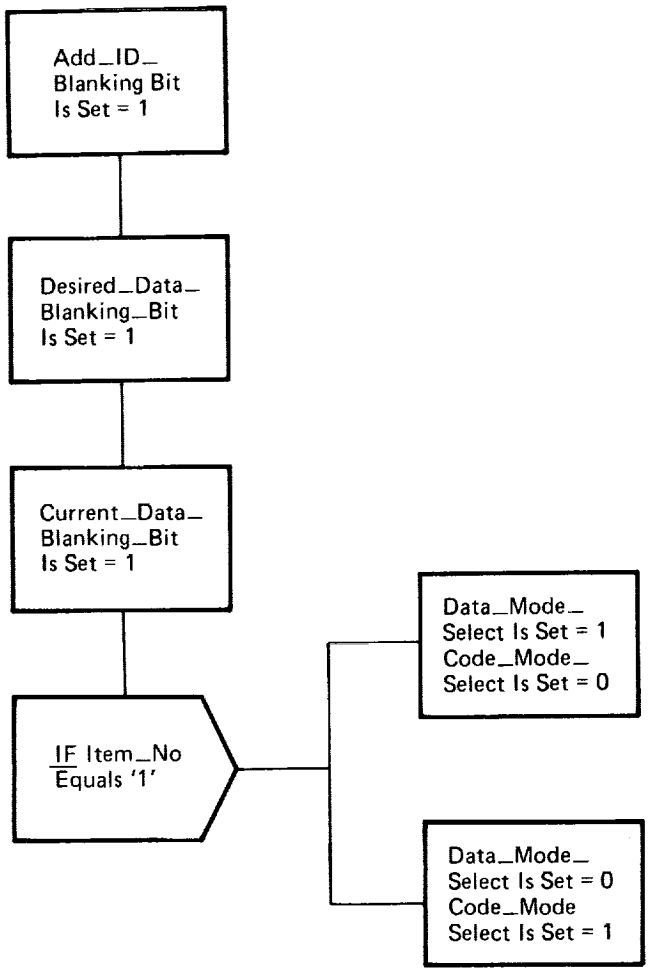


Figure 3.3.1-6. Read/Write_Specialist_Function (R/W_Mode_SEL_Proc) (910.5)

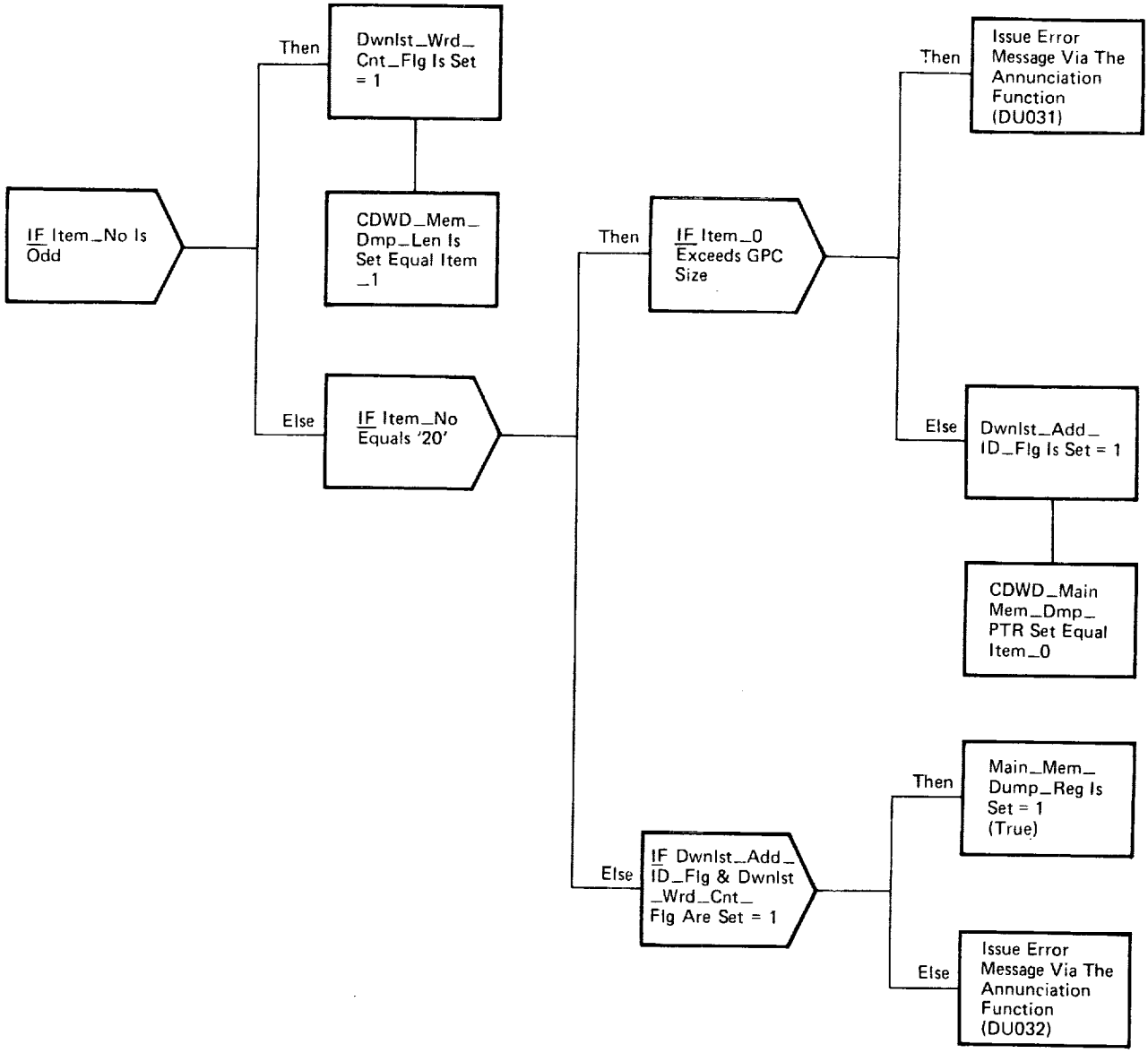


Figure 3.3.1-7. Read/Write_Specialist_Function (R/W_DwnIst_Mem_Dump) (910.6)

3.3.1.1 Read/Write_Cyclic_Display_Update (ASH_RW_CYC_UPDATE) (915)

ASH_RW_CYC_UPDATE contains the logic required to provide for the cyclic updating of the "CURRENT" data fields on the Memory Read/Write specialist function display. The cyclic rate used is once per second.

a. Control Interfaces -

1. SCHEDULE ASH_RW_CYC_UPDATE PRIORITY (PRIO_ASH),
   REPEAT EVERY 1.0;
2. SCHEDULED by: (910) Read/Write_Specialist_Function (ASB_RD_WRT)

b. Input - See Table 3.3.1.1-1

c. Process Description - Whenever this program is entered a DO FOR loop is executed to determine whether an address identification input has been entered for each line on the format.

If an address has been specified for the current line then the ASH_Read_Flag, indicating a read operation was requested, is set=1 (enabled). Then the GPC_Address_IO_Save_Area for the current line is stored in the variable ASH_Read_Start. Now that the necessary items in the FCOS program modification macro parameter list have been supplied the program modification SVC is executed. This results in the variable ASH_Word1_Data being updated from the DATA_Read_Out_Area.

Then the Current_Data_Save_Area for the line being processed is updated with the data from ASH_Word1_Data. To make that information available for display on the DEU the Current_Data_Blanking_Bit for this line is set = 0 (disabled).

If no address is supplied for a particular line no processing is done. The loop counter is incremented if the maximum value has not yet been reached and the loop executed again if needed.

The control flow for this module is presented in Figure 3.3.1.1-1.

d. Outputs - See Table 3.3.1.1-1.

e. Module References - None

f. Module Attributes - Program

g. Template References - Memory_Read/Write Compool (CDJ_RW_DATA)

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.3.1.1-1

NAME Read/Write_Cyclic_Display_Update

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	ASH_D	X032	T	915	915	ASH_D			
2	GPC_Address_ID_Status	W010	I	910 915	915 910	CDJB_R_W_L			
3	ASH_Read_Flag	X033	L	915	915	ASH_READ_FC. DFLGSVC			
4	ASH_Read_Strt	X034	L	915	915	ASH_READ_FC. DPMSTRT			
5	GPC_Address_ID_Save_Area	W040	I	910 915	915 910	CDJB_R_W_LOCID			
6	ASH_Word1_Data	X035	L	915	915	ASH_FW1_DATA			
7	Data_Read_Out_Area	W080	I	910, 915	915 910	CDJV_DATA_ AT_LOCID			
8	Current_Data_Save_Area	W060	O	910 915	915 910	CDJB_R_W_ CURRO			
9	Current_DATA_BLANKING_Bit	W030.1	O	910 915	910,915	CDJB_R_W_C			

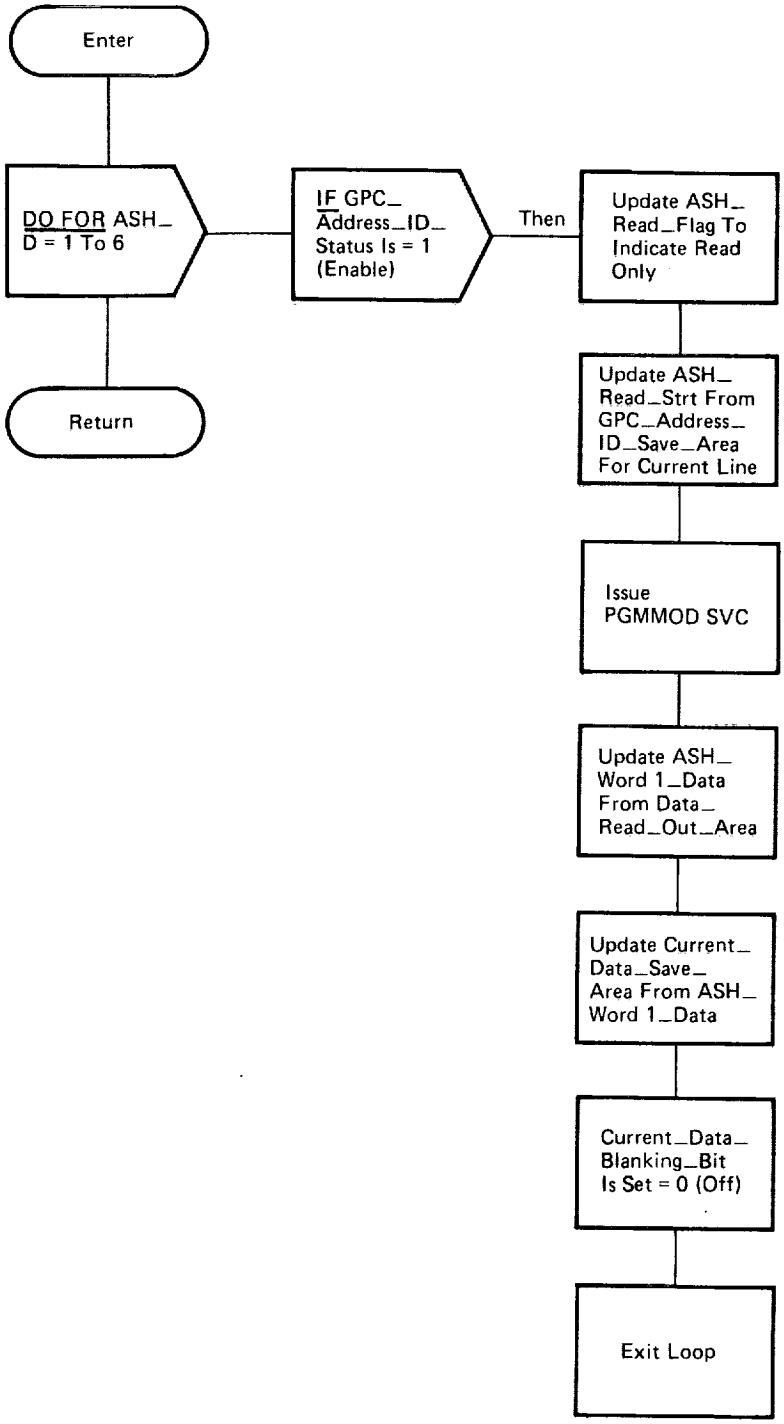


Figure 3.3.1.1-1. Read/Write_Cyclic_Display_Update (ASH_RW_CYC_Update)

3.3.1.2 Bit_Execute_Item_Processor (ASI_RW_BITE_PROC) (920)

ASI_RW_BITE_PROC is the module that initiates the reading and storing of the current BITE status data for the Flight Critical and Payload MDMS as well as the PCMMU.

a. Control Interface -

1. SCHEDULE ASI_RW_BITE_PROC PRIORITY (PRIO_ASI), REPEAT EVERY 1.28;

2. Scheduled by: (910) Read/Write_Specialist_Function (ASB_RD_WRT)

3. SCHEDULE ASI_RW_BITE_PROC PRIORITY (PRIO_ASI)

4. Scheduled by: (910) Read/Write_Specialist_Function (ASB_RD_WRT)

b. Input - See Table 3.3.1.2-1.

c. Process Description - Whenever this module is executed a series of FCOS macros which read and store the flight forward and flight aft BITE data, the Payload MDMS and the PMU BITE status are issued. In addition the ICC_GST flag is set = 1 (enable) so all GPC's would be aware of the functions execution.

d. Outputs - See Table 3.3.1.2-1.

e. Module References - None

f. Module Attributes - Program

g. Template References -

1. UI_FCOS_Shared_Compool (CZ2_COMMON)

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.3.1.2-1

NAME Bite_Execute_Item_Processor (ASI_RW_BITE_PROC)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	ICC_GST	I320.03	0	920	920	CZ2B_ICC_FLAG			

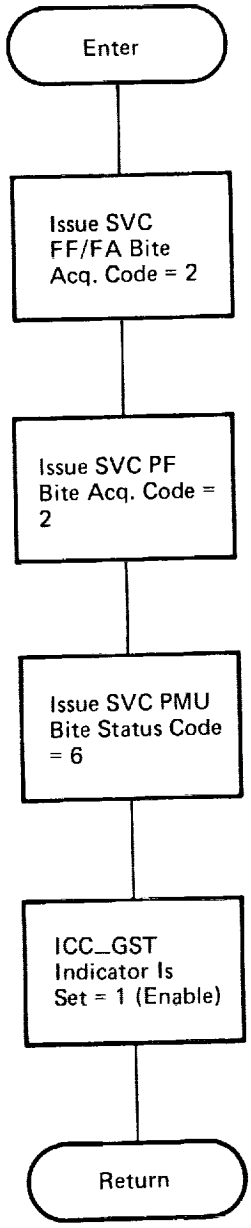


Figure 3.3.1.2-1. Bite_Execute_Item_Processor (ASI_RW_BITE)

3.3.2 Time_Management_Specialist_Function (ASC_TIME_MGMT) (950)

The Time_Management_Specialist_Function provides control of the GPC time management processor and updates to the master timing unit (MTU) MET/GMT accumulators and GPC MET/GMT. In addition, provisions are made for monitoring the MTU oscillator and accumulator strings, the GPC prime time, and time management processor source selection.

a. Control Interface -

1. SCHEDULE ASC_TIME_MGMT PRIORITY (PRIO_ASC)

2. SCHEDULEd by (560) Sequence_Request_Processor (DMC_SEQ_REQ_PROC)

b. Input - The following data is input to the Time_Management_Specialist_Function in the form of control segment grammar macros:

ITEM_NO - input ITEM number

ITEM_I - integer data associated with ITEM key

ITEM_S - scalar data associated with ITEM key

D_DEU_NUMBER - number of DEU displaying the time management display.

The grammar macros are discussed in Appendix H of User Interface.

Additional inputs to this program are described in Table 3.3.2-1.

c. Process Description - Control segment initialization is performed by invoking the SPEC and BLOCK grammar macros. Time_Mgmt_Status_Flags and delta input save areas are initialized. The Time_Management_Display_Update_Cyclic_Processor and the Time_Management_MTU_Update_Processor are SCHEDULEd and the time management display is requested via the DISPLAY grammar macro. The Time_Management_Specialist_Function then waits for keyboard inputs.

The following processing occurs for ITEM inputs:

1. ITEM's 1-4: the input data values are saved in the Time_Mgmt_Delta_Input_Values, the MET_Delta_Values_Saved flag is set, and the applicable Time_Mgmt_Blanking_Flags are reset.

2. ITEM's 5-8: the input data values are saved in the Time_Mgmt_Delta_Input_Values, the GMT_Delta_Values_Saved flag is set, and the applicable Time_Mgmt_Blanking_Flags are reset.

3. ITEM 9: if the Update_In_Process flag = 1, no processing occurs. If the Update_In_Process flag = 0 and the GMT_Delta_Values_Saved flag = 1 or the MET_Delta_Values_Saved flag = 1, the following occurs:

   a. GMT_Delta_Values_Saved flag = 1: the GMT_Delta_ Value is calculated from the Time_Mgmt_GMT_Delta_Values and the Time_Mgmt_GMT_Seconds_Delta_Value. If only one GPC is active and in OPS 0, the Request_Initial_GMT_Update flag is set = 1. If more than one GPC is active or a single GPC is active but not in OPS 0, the GMT_Delta_Value is checked for less than 15 milliseconds. If the GMT_Delta_Value is in limits, the Request_Normal_GMT_Update flag is set = 1. If it is out of limits, an error message is issued.

   b. MET_Delta_Values_Saved flag = 1: the MET_Delta_Value is calculated from the Time_Mgmt_MET_Delta_Values and the Time_Mgmt_MET_Seconds_Delta_Value. The Request_MET_Update flag is set = 1.

   The Update_In_Process flag is set = 1 and the SVC SET is issued for the MTU_Update_Event to activate Time_Management_MTU_Update_Processor.

4. ITEM 10: if the Update_In_Process flag = 1, no processing occurs. If the Update_In_Process flag = 0, the MTU_Request_Type_Indicator is set to MASTER_RESET, the Update_In_Process flag is set = 1, and the SVC set is issued for the MTU_Update_Processor.

5. ITEM 11: if the Update_In_Process flag = 1, no processing occurs. If the Update_In_Process flag = 0, the MTU_Request_Type_Indicator is set to RESET_MET, the Update_In_Process flag is set = 1, and the SVC SET is issued for the MTU_Update_Event to activate Time_Management_MTU_Update_Processor.

6. ITEM's 12-15: the Force_Selected_Time_Source is set based on the input ITEM number and the ICC_DST_Status flag is set = 1.

Control segment termination is performed using the BLOCK_END, SPEC_CLEAN_UP, and SPEC_END grammar macros. The Time_Management_Display_Update_Cyclic_Processor is CANCELed. The Terminate_Time_Mgmt_Spec flag is set = 1 and if the Update_In_Process flag = 0, the SVC SET is issued for the MTU_Update_Event to activate Time_Management_MTU_Update_Processor. A CANCEL is then issued for Time_Management_MTU_Update_Processor.

d. Outputs - See Table 3.3.2-1.

e. Module References -

1. (170) Set_Event_Processor (FPMSET) is CALLed.
2. (171) Reset_Event_Processor (FPMRESET) is CALLed.
3. (955) Time_Management_Display_Update_Cyclic_Processor (ASG_CYCLIC_UPDATE) is SCHEDULEd.
4. (975) Time_Management_MTU_Update_Processor (ASJ_MTU_UPDATE) is SCHEDULEd.
5. (675) Annunciation_Macro_Interface (DMA_MAC is CALLed)
6. (550) Application_Moding_And_Sequencing (DNX_BMS) is CALLed.
7. (540) Display_Presentation_And_Control (DIS_PLAY) is CALLed.

f. Module Attributes - Program

g. Template References -

1. UI_Section_Of_Common_Compool (CZ1-COMMON)
2. UI_FCOS_Shared_Compool (CZ2_COMMON)
3. Time_Management_Compool (CAA_SC_COMPOOL)
4. FCOS_COMPOOL (FCMCOM)
5. Annunciation_Compool (CDL_ANNUN)
6. Application_Moding_And_Sequencing (DNX_BMS)
7. Display_Presentation_And_Control (DIS_PLAY)
8. Annunciation_Macro_Interface (DMA_MAC)

9. Time_Management_Display_Update_Processor (ASG_CYCLIC_UPDATE)

10. Time_Management_MTU_Update_Processor (ASJ_MTU_UPDATE)

h. Error Handling -

1. If ITEM's 9, 10, or 11 are input before a previously entered ITEM 9, 10, or 11 request has completed, the new request is ignored.

2. If ITEM 9 is entered without previously entering ITEM's 1-8, an error message is issued (TM003).

3. If a normal GMT update of more than 15 milliseconds is requested, an error message is issued (TM002).

i. Constraints and Assumptions - None

j. Detailed Implementation -

1. CAAB_TM_STAT_FLG = HEX '8000'
GMT_Display_Blank_Status = 1, all other flags in flag word = 0.

2. GMT_Delta_Value is the sum of the ITEM 5-8 inputs for days, hours, minutes and seconds in seconds.

```
GMT_Delta_Value = Time_Mgmt_GMT_Delta_Values for days X 86400
                + "      "   "   "    "     for hours X 3600
                + "      "   "   "    "     for minutes X 60
                + Time_MGMT_GMT_Seconds_Delta_Values
```

3. IF Common_Set_Mask for self GPC_ID = 0 AND Current_OPS for all major functions in self GPC_ID = 0 AND Fail_to_Sync_GPC for self GPC_ID = 0, the request is considered an initial GMT update.

4. MET_Delta_Value is the sum of the ITEM 1-4 inputs for days, hours, minutes and seconds in seconds.

```
MET_Delta_Value = Time_Mgmt_MET-Delta_Values for days X 86400
                + "      "   "   "    "     for hours X 3600
                + "      "   "   "    "     for minutes X 60
                + Time_Mgmt_MET_Seconds_Delta_Values
```

DATA TABLE 3.3.2-1

NAME Time_Management_Specialist_Function (ASC_TIME_MGMT)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	Time_Mgmt_Status_Flags	H040	I O	950,955 975	950 975	CAAB_TM_STAT_FLG			
2	Time_Mgmt_Delta_Input_Values	H030	I O	950	950	CAAV_TM_DELTA_DISP			
3	MET_Delta_Values_Saved	H040.30	I O	950	950	CAAB_TM_STAT_FLG			
4	Time_Mgmt_Blanking_Flags	H050	O	950		CAAB_TM_BLNK_FLG			
5	GMT_Delta_Values_Saved	H040.25	I O	950	950	CAAB_TM_STAT_FLG			
6	Update_In_Process	H040.35	I O	950 975	950	CAAB_TM_STAT_FLG			
7	GMT_Delta_Value	H090	O	950	975	CAAV_DELT_GMT			
8	Time-Mgmt_GMT_Delta_Values	H030.1	I O	950	950	CAAV_TM_GMT_DELT	V91W1070C V91W1071C		X
							V91W1072C		
9	Time_Mgmt_GMT_Seconds_Delta_Value	H030.2	I O	950	950	CAAV_TM_GMT_SEC	V91W1073C		X
10	Request_Initial_GMT_Update	H040.55	O	950 975	975	CAAB_TM_STAT_FLG			
11	Request_Normal_GMT_Update	H040.50	O	950 975	975	CAAB_TM_STAT_FLG			
12	MET_Delta_Value	H100	O	950	975	CAAV_DELT_MET			
13	Time_Mgmt_MET_Delta_Values	H030.3	I O	950	950	CAAV_TM_MET_DELT	V91W1080C V91W1081C		
							V91W1082C		X
14	Time_Mgmt_MET_Seconds_Delta_Value	H030.4	I O	950	950	CAAV_TM_MET_SEC	V91W1083C		X
15	Request_MET_Update	H040.45	O	950 975	975	CAAB_TM_STAT_FLG			
16	MTU_Update_Event	H070	O	148,950 975		CAAE_MTU_UPDATE			
17	MTU_Request_Type_Indicator	H080.1	O	950 975	148	CAAV_MTU_PARMS. TYPE_IND			
18	Force_Selected_Time_Source	I040.07	O	149 950	149	CZ2B_DPS_STATUS			

DATA TABLE 3.3.2-1 (Cont'd.)

NAME Time_Management_Specialist_Function (ASC_TIME_MGMT)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
19	ICC_DST_Status	I320.11	O	351,910 950		CZ2B_ICC_FLAG			
20	Terminate_Time_Mgmt_Spec	H040.40	O	950	975	CAAB_TM_STAT_FLG			
21	GMT_Display_Blank_Status	H040.10	O	950 955		CAAB_TM_STAT_FLG			x
22	Common_Set_Mask	I010.12	I	See App.	E	CZ2B_CS	See App. E	x	
23	GPC_ID	#001	I	300	See App. E	TFCMID			
24	Current_OPS	I010.02	I	560,820 880	See App. E	CZ2V_CURRENT_OPS			
25	Fail_to_Sync_GPC	I330	I	368	485, 830 950	CZ2B_FAIL_SYNC_SET			
26	Time_Source_Data	X039	L		950	ASCBL_SOURCE			
27	Error_Message_Flag	X040	O	950	950	ASCBL_FLAG			

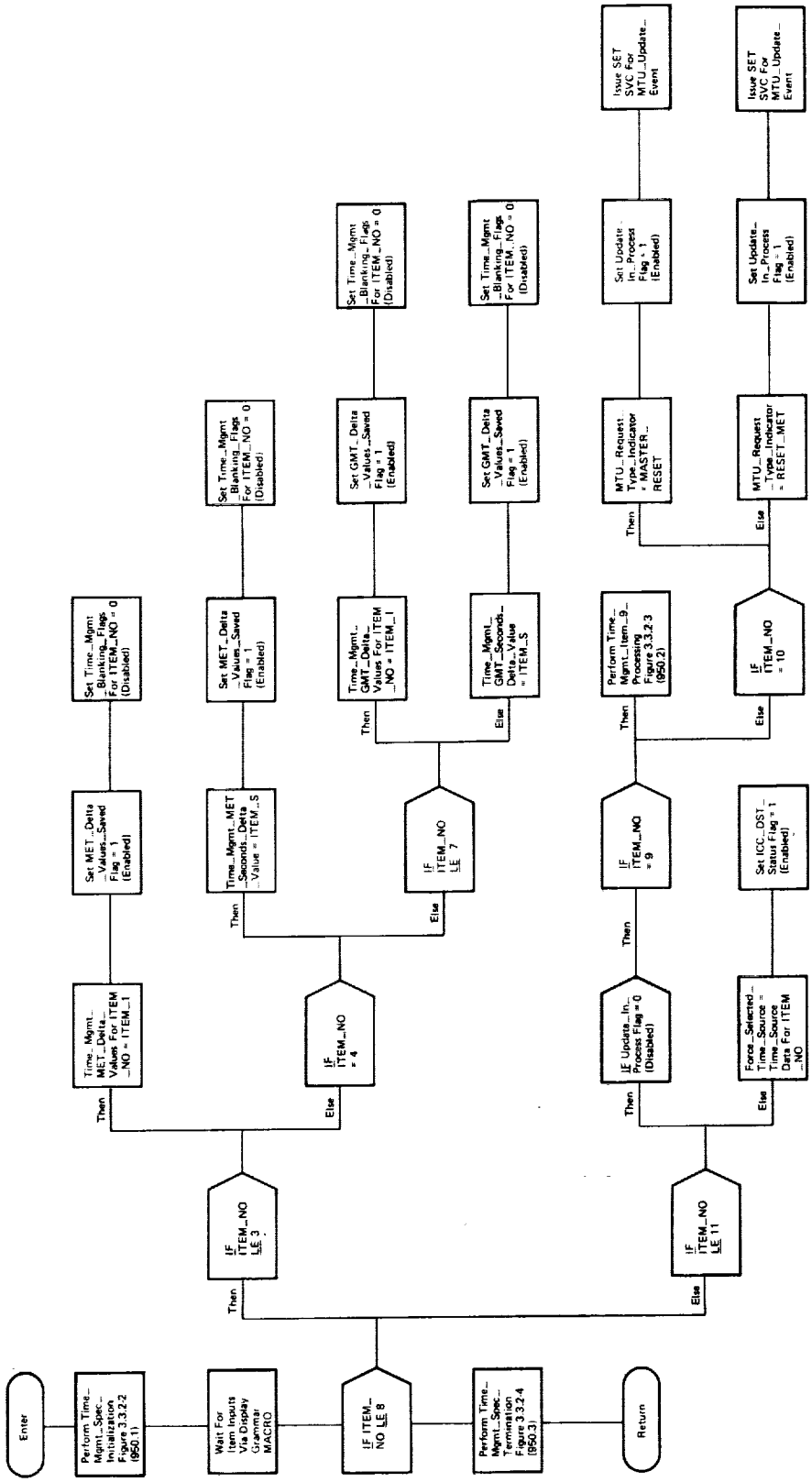


Figure 3.3.2.2. Time_Management_Specialist_Function Time_Mgmt_Spec_Initialization (950.1)

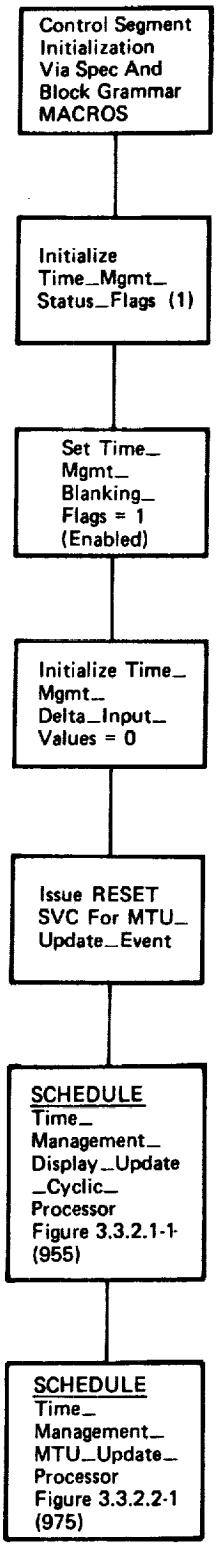


Figure 3.3.2-2. Time_Management_Specialist_Function Time_Mgmt_Spec_Initialization (950.1)

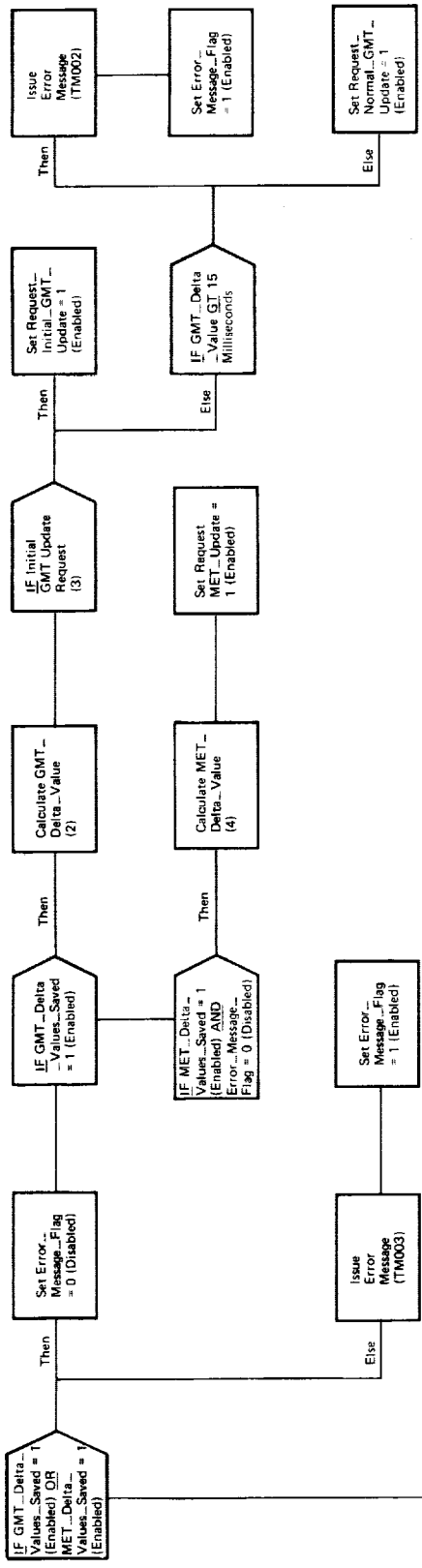


Figure 3.3.2-3. Time_Management_Specialist_Function
Time_Mgmt_Item_9_Processing (950.2)

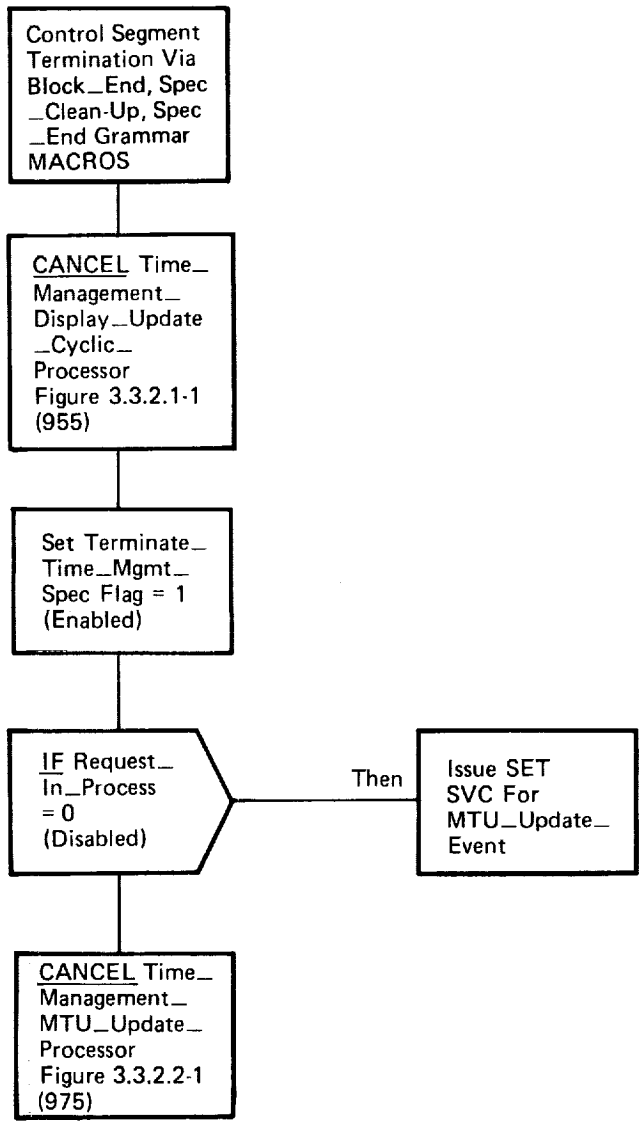


Figure 3.3.2-4. Time_Management_Specialist_Function Time_Mgmt_Spec_Termination (950.3)

3.3.2.1 Time_Management_Display_Update_Cyclic_Processor (ASG_CYCLIC_UPDATE) (955)

ASG_CYCLIC_UPDATE Processes data for the time management display.

a. Control Interface -

1. SCHEDULE ASG_CYCLIC_UPDATE PRIORITY (PRIO_ASG), REPEAT EVERY TIME_ASG

2. SCHEDULED by (950) Time-Management_Specialist_Function (ASC_TIME_MGMT)

b. Input - See Table 3.3.2.1-1

c. Processor Description - The MTU_1_Bite_Word is checked for the status of the temperature and amplitude of the primary and secondary oscillators. If the primary oscillator is out of limits, the MTU_Primary_Oscillator_Status flag is set; if the status is good, the flag is reset. If the secondary oscillator is out of limits, the MTU_Secondary_Oscillator_Status flag is set; if the status is good, the flag is reset.

The SVC MTU_FMT_CONV is issued for each MTU time value (MTU_1_Buffer, MTU_2_Buffer, MTU_3_Buffer) to convert the data to displayable format. The converted values are saved in Time_Mgmt_GMT_Display_Values for display. The milliseconds portions of the time values are extracted and saved in Time-Mgmt_GMT_Milliseconds_Display_Values.

The SVC FXFLCVT is issued to obtain the display format of the Internal_Time_Microseconds for GPC_Prime_ID. The converted values and milliseconds portion are saved as described for the MTU time values.

The GMT_Display_Blank_Status is reset to indicate valid time values are saved.

d. Output - See Table 3.3.2.1-1

e. Module References -

1. (150) Time_Conversion_SVC_Servicer (FPMTMCVT)

f. Module Attributes - Program

g. Template References -

1. UI_FCOS_Shared_Compool (CZ2_COMMON)

2. FCOS_Compool (FCMCOM)

3. Time_Management_Compool (CAA_SC_Compool)

h. Error Handling - None

i. Constraints and Assumptions - None

j. Detailed Implementation - None

DATA TABLE 3.3.2.1-1

NAME Time_Management_Display_Update_Cyclic_Processor (ASG_CYCLIC_UPDATE)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	MTU_1_Bite_Word	#017.3	I	149	149 955	TFCMM1BW			
2	Oscillator_1_Amplitude	#017.32	I	149	955	TFCMM1BW			
3	Oscillator_1_Temperature	#017.35	I	149	955	TFCMM1BW			
4	MTU_Primary_Oscillator_Status	H040.15	O	955		CAAB_TM_STAT_FLG			X
5	Oscillator_2_Amplitude	#017.33	I	149	955	TFCMM1BW			
6	Oscillator_2_Temperature	#017.36	I	149	955	TFCMM1BW			
7	MTU_Secondary_Oscillator_Status	H040.20	O	955		CAAB_TM_STAT_FLG			X
8	MTU_1_Buffer	#017.2	I	267	955	TFCMMTU1			
9	MTU_2_Buffer	#018.2	I	267	955	TFCMMTU2			
10	MTU_3_Buffer	#019.2	I	267	955	TFCMMTU3			
11	Time_Mgmt_GMT_Display_Values	H010	O	955		CAAV_TM_GMT_DISP	V91W1040C V91W1042C V91W1044C V91W1046C		X
12	Time_Mgmt_GMT_Milliseconds_Display_Values	H020	O	955		CAAV_TM_MLS_DISP	V91W1040C V91W1042C V91W1044C V91W1046C		X
13	Internal_Time_Microseconds	I610.10	I			CZ2B_GPC_MT			
14	GPC_Prime_ID	I050	I	351, 700 820, 880	See App. E	CZ2V_GPC_P			
15	GMT_Display_Blank_Status	H040.10	O	950 955		CAAB_TM_STAT_FLG			X

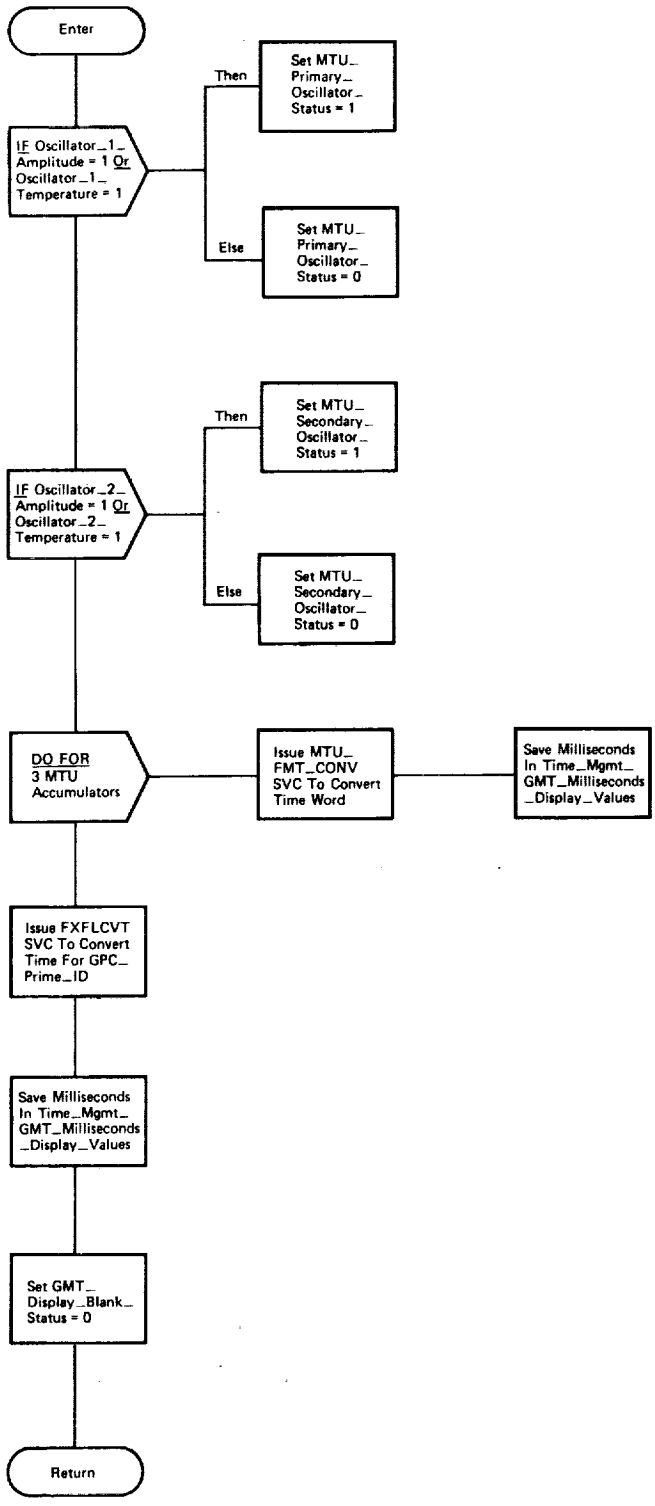


Figure 3.3.2.1-1. Time_Management_Display_Update_Cyclic_Processor (ASG_CYCLIC_UPDATE)

3.3.2.2 Time_Management_MTU_Update_Processor (ASJ_MTU_UPDATE) (975)

ASJ_MTU_UPDATE processes requests from the Time_Management_Specialist_Function for MTU updates.

a. Control Interface -

1. SCHEDULE ASJ_MTU_UPDATE PRIORITY (PRIO_ASJ)

2. SCHEDULEd by (950) Time_Management_Specialist_Function (ASC_TIME_MGMT)

b. Input - See Table 3.3.2.2-1.

c. Process Description - The Time_Management_MTU_Update_Processor receives requests for MTU updates through the setting of the MTU_Update_Event. The Time_Management_Specialist_Function sets this event whenever a GMT or MET update is required, when an accumulator sync or MET reset request is received, or when the specialist function is terminated. Flags in the Time_Mgmt_Status_Flags indicate the type of request. If the Terminate_Time_Mgmt_Spec flag = 1, the Time_Management_MTU_Update_Processor terminates by executing the CLOSE SVC.

If the Request_MET_Update flag = 1 or the Request_Normal_GMT_Update flag = 1 or the Request_Initial_GMT_Update flag = 1, the MTU_Update_Parameter_List is built. The MTU_Request_Type_Indicator, the MTU_Update_Coincident_Time, the MTU_Update_Time and the MTU_Update_Delta_Time are initialized in the parameter list. The UPDATE_MTU SVC is issued, followed by a WAIT and RESET for the MTU_Update_Event.

If the Request_MET_Update flag = 0 and the Request_Normal_GMT_Update flag = 0, and the Request_Initial_GMT_Update flag = 0, the MTU_Request_Type_Indicator in the MTU_Update_Parameter_List has already been initialized to MASTER_RESET or RESET_MET by the Time_Management_Specialist_Function. The UPDATE_MTU SVC is issued, followed by a WAIT and RESET for the MTU_Update_Event.

If the Terminate_Time_Mgmt_Spec flag = 0, the Request_MET_Update flag, the Request_Normal_GMT_Update flag, the Request_Initial_GMT_Update flag, and the Update_In_Process flags are all set = 0. The Time_Management_MTU_Update_Processor then WAITs on the MTU_Update_Event for another request.

d. Outputs - See Table 3.3.2.2-1.

e. Module References -

1. (104) Wait_Processor (FPMWAIT) is CALLed.

2. (171) Reset_Event_Processor (FPMRESET) is CALLed.

3. (148) MTU_Update_Processor (FPMUPMTU) is CALLed.

4. (144) Time/Date_Application_Requests_Processor (FPMTMHAL) is CALLed.

f. Module Attributes - Program

g. Template References -

1. UI_FCOS_Shared_Compool (CZ2_COMMON)

2. Time_Management_Compool (CAA_SC_COMPOOL)

h. Error Handling - None

i. Constraints and Assumptions - When calculating the coincidence time for GMT and MET updates, the current time is rounded up to the next minute and one minute is added. Therefore, the time delay for a requested update may be up to two minutes.

j. Detailed Implementation -

1. Coincident MET time is obtained as follows:

   a. Current MET in minutes -
   Coincident_Total = Current_MET/60

   b. Round to next integral minute
   Coincident_Time_In_Minutes = CEILING (Coincident_Total)

   c. Add one minute time lag allowance
   Coincident_Time_In_Minutes = Coincident_Time_In_Minutes + 1

   d. MTU_Update_Coincident_Time = REMAINDER (Coincident_Time_In_ Minutes, 60)

2. Coincident_Total = Coincident_Time_In_Minutes 60
MTU_Update_Time = CLOCKTIME + (Coincident_Total - Current_MET)

3. Coincident GMT time is obtained as follows:

   a. Current GMT in Minutes -
   Coincident_Total = CLOCKTIME/60

b. Round to next integral minute
Coincident_Time_In_Minutes = CEILING (Coincident_Total)

c. Add one minute for time lag allowance
Coincident_Time_In_Minutes = Coincident_Time_In_Minutes + 1

d. MTU_Update_Coincident_Time = REMAINDER (Coincident_Time_In_Minutes, 60)

4. Coincident_Total = Coincident_Time_In_Minutes 60
MTU_Update_Time = Coincident_Total + GMT_Delta_Value

DATA TABLE 3.3.2.2-1

NAME Time_Management_MTU_Update_Processor (ASJ_MTU_UPDATE)

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	MTU_Update_Event	H070	O	148 950 975		CAAE_MTU_UPDATE			
2	Time_Mgmt_Status_Flags	H040	I O	950 975	950 975	CAAB_TM_STAT_FLG			
3	Terminate_Time_Mgmt_Spec	H040.40	I	950	975	CAAB_TM_STAT_FLG			
4	Request_MET_Update	H040.45	I O	950 975	975	CAAB_TM_STAT_FLG			
5	Request_Normal_GMT_Update	H040.50	I O	950 975	975	CAAB_TM_STAT_FLG			
6	Request_Initial_GMT_Update	H040.55	I O	950 975	975	CAAB_TM_STAT_FLG			
7	MTU_Update_Parameter_List	H080	O	950 975	148	CAAV_MTU_PARMS			
8	MTU_Request_Type_Indicator	H080.10	O	950 975	148	CAAV_MTU_PARMS. TYPE_IND			
9	MTU_Update_Coincident_Time	H080.20	O	975	148	CAAV_MTU_PARMS. COINC_TIME			
10	MTU_Update_Time	H080.30	O	975	148	CAAV_MTU_PARMS_ UPDAT_TIME			
11	MTU_Update_Delta_Time	H080.40	O	975	148	CAAV_MTU_PARMS. MET_DELTA			
12	Update_In_Process	H040.35	O	950,975	950	CAAB_TM_STAT_FLG			
13	Coincident_Total	X036	I,O	975	975	ASJVL_CTIME			
14	Current_MET	X037	I,O	975	975	ASJVL_MET_TIME			
15	Coincident_Time_In_Minutes	X038	I,O	975	975	ASJVL_CO_TIME			
16	MET_Delta_Value	H100	I	950	975	CAAV_DELT_MET			
17	GMT_Delta_Value	H090	I	950	975	CAAV_DELT_GMT			

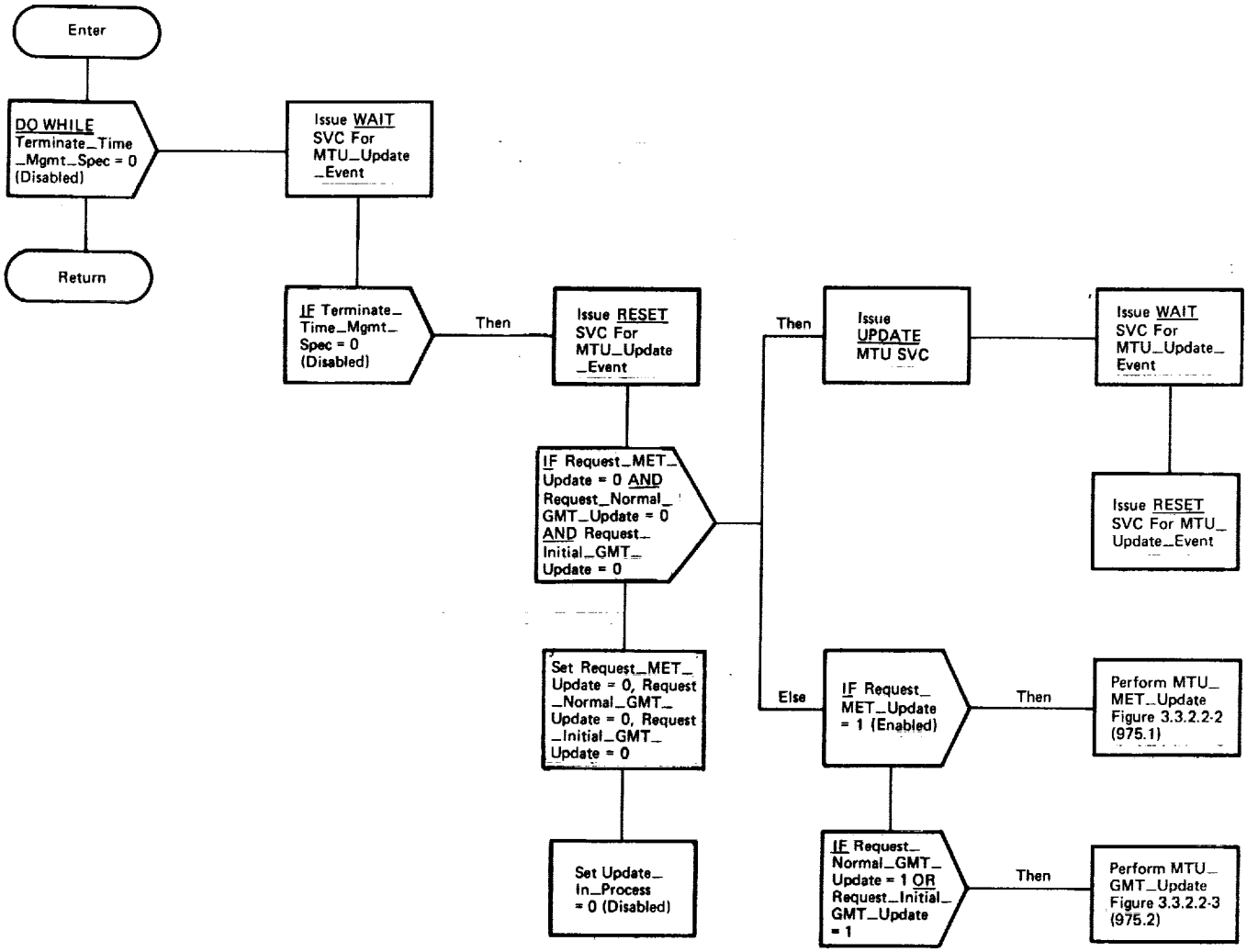


Figure 3.3.2.2-1 Time_Management_MTU_Update_Processor (ASJ_MTU_Update)

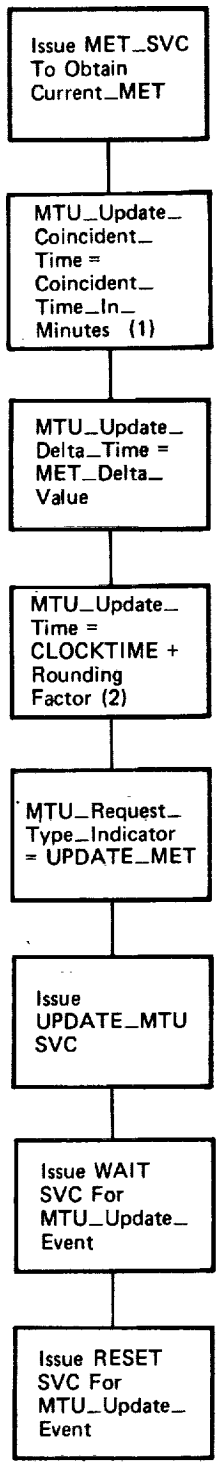


Figure 3.3.2.2-2. Time_Management_MTU_Update_Processor MTU_Met_Update (975.1)

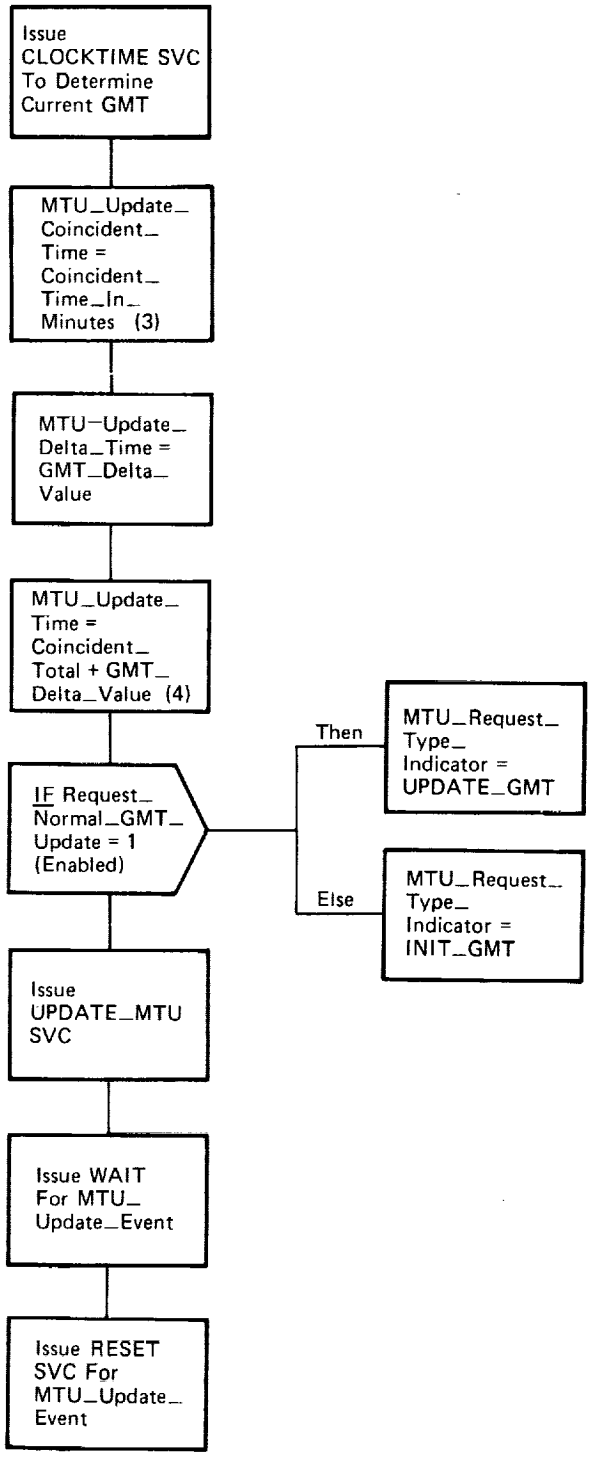


Figure 3.3.2.2-3. Time_Management_MTU_Update_Processor MTU_GMT_Update (975.2)

3.4 CRT DISPLAYS

The CRT Displays section describes the contents of displays that are maintained by System Control rather than by the application areas. These displays are associated with the Operational Sequence and Specialist Functions that are documented under System Control.

The description of each display provides the display title, a functional description, a detailed explanation of the input parameters, a detailed narrative covering the contents of the display and a picture of the display.

The System Control CRT displays consist of the following:

a. DPS Configuration Monitor display which is initiated by the Idle Operational Sequence (Sec. 3.2.2).

b. Memory Read/Write display which is invoked by the Read/Write Specialist Function (Sec. 3.3.1).

c. Time Management display which is supported by the Time Management Specialist Function (Sec. 3.3.2).

3.4.1 DPS Configuration Monitor

This display provides a crew interface to review and change the configuration of the DPS subsystem. This interface complements the use of the CMPTR/BUS key and the CMPTR/CRT key. The display is supported by system software and is associated with OPS 000 and SPEC 000. The data inputs are considered system parameters. These parameters are common across all GPC's and shall be identical regardless of the GPC driving the display. Likewise, input values to the display shall be distributed among all active GPC's.

a. Inputs - See Table 3.4.1 - 1

b. Process Description - The display is basically organized in three major sections. The left one-third provides information of the current operational status/configuration of all GPC's, controls for GPC redundancy management, and controls for mass status of the GPC RM and I/O data paths. The right one-third provides review/change capability for the nominal memory configuration table and controls for memory source selection.

At the upper left, the operational status and utilization of the four GPC's are presented. This information includes MODE switch status, redundant/simplex SET status, memory configuration (MC) loaded, major functions (MF)active, and CPU duty cycle. These outputs are provided for each GPC. The MODE output is a text representation of the switch position. The possible values include RN,SB, and HT indicating run, standby, and halt, respectively. The SET output is a text indication that the GPC is operating simplex or as a member of a redundant set. In addition, the SET output indicates a failure to maintain common sync. The possible values are RD, SC, or FS indicating redundant, simplex, and fail to common sync. Should a redundant GPC fail its redundant sync, it will be forced to simplex. This can be distinguished from a normal simplex GPC by noting the MC. The MC output is the numeric value of the resident memory configuration. The legal values are 0 to 4 where 0 indicates only system software is loaded. The MF output is an alphanumeric indication of the active major functions and associated OPS. There are two outputs for MF. The top line always displays the major function which initiates the current memory configuration. If only one major function is contained, the lower output field shall be blank. The possible values are GX for GNC, SX for SM, and PX for PAYLOAD where X is the OPS number (0-9). The CPU output is the current numerical value of the GPC duty cycle displayed in percent. The value is an average over a finite time interval. Whenever the value exceeds 95 percent, a status indicator (↓) shall be driven to the right of the value. Inhibited in all GPC's, then an asterisk would appear to the right of Item 2 and Item 4. Whenever the opposite condition is selected, the asterisk will toggle to the executed item number. In the above configuration, if Item 1 was executed, the asterisk would blank beside Item 2 and appear beside Item 1.

Beneath the GPC RM controls, the assignment of mass memory units is provided. To the left are controls for assigning a mass memory unit (MM1, MM2). Only one mass memory may be assigned at a time. The selected unit will be indicated by an asterisk beside the corresponding item number. Whenever the opposite unit is selected, the asterisk will toggle to the executed item number. Below the symbols MM1 and MM2 are the status of the corresponding unit. The output is a text indication that the unit is ready for use or currently busy. The legal values are RDY for ready and BSY for busy.

The center column of the display provides status of the GPC RM and GPC DATA PATH configuration/status. The upper portion presents the software version of the CAM. The matrix is presented in the layout as the hardware panel. The voting GPC's are listed vertically (VOTE) and the failed GPC's horizontally (FAIL). The output fields shall be blank as long as there are no failed votes. A fail vote shall be indicated by an F character in the appropriate row/column. Each GPC may issue a failed vote of another GPC or of itself. However, if a GPC forms a failed vote for all other GPC;s in the redundant set, then it shall issue a failed vote only on itself. All failed indications shall be reset upon formation of a new redundant set. Beneath the CAM, the configuration/status is presented for the DATA PATHS on the payload buses (PL 1,2), launch data buses (LB 1,2), DEU buses (DK 1,2,3), and mass memory buses (MM 1,2). For each of the listed data paths, the GPC in command is indicated by an asterisk in the corresponding column. Only one GPC may be in command of a given bus. The selection of the commanding GPC for the payload buses is controlled by software hierarchy. The selection of the commanding GPC for the launch data buses is as described previously. The selection of the commanding GPC for the DEU buses is controlled by the CMPTR/CRT key. For each data path, a status indicator (↓) is provided for each GPC. This indicator appears beside the output fields for the GPC in command. The indicator shall be driven by each GPC whenever a BTU Failure is detected on the corresponding data path. The mass memory and DEU buses have a single BTU per bus; thus a data path failure is indicative of a failure of the BTU. The launch data buses and the payload buses have multiple BTU's per bus. Data path failures on these buses require additional information to determine the BTU affected. The fail indicators shall be reset upon any OPS transition requiring memory reconfiguration or any bus reassignment. Beneath the DATA PATH matrix, the GPC/string (STRG) assignment/status is presented. For each of the five strings, the GPC in command is indicated by number above the STRG number. The GPC assignment can be made by the CMPTR/BUS key. Below the STRG numbers, the configuration mode and port status of the flight critical and payload MDM's are presented. The flight critical MDM's are below STRG 1-4. STRG 1 FWDP reflects FF1, and STRG 1 AFTP reflects FA1. STRG 2-4 similarly reflect FF 2-4 and FA 2-4. The payload MDM's are below STRG 5, PL 1 beside FWDP/S, and PL 2 beside AFTP/S. The configuration mode is an indication of which MDM ports are being utilized. The ports are labeled P and S for the primary and secondary ports. For each string, the active ports are indicated by an asterisk in the corresponding row. For those ports which are not selected, the output field shall be blank. Beside each output field for the port selection, a status indicator shall be provided to indicate <u>any GPC</u> detection of data path failures to the corresponding port. The indicator shall be a (↓). The

status indicator shall be driven independently of the active port selection. For example, if the primary port for the forward MDM fails, a (↓) will appear in the appropriate column for FWDP. When the configuration mode is changed to use secondary ports, the asterisk will move to the FWDS and AFTS rows, but the (↓) indicator will remain in the FWDP row. The status indicators shall only be reset by a STRG reassignment or an OPS transition requiring memory reconfiguration.

The right column of the display contains the controls for memory reconfiguration. The top section (MEMORY) is used to review and change a portion of the nominal configuration table. The table contents, which may be changed, are the target GPC selection and GPC/STRG assignment for each available memory configuration. The memory configuration is identified by the data entry for MC, Item 32. Upon this entry, the OPS, GPC SEL/DES, and STRG/GPC output fields are driven. The OPS output is an indication of which OPS request inititates the loading of the identified MC. The GPC's selected to receive the identified MC are indicated by an asterisk in the SEL (select)column. Those not selected as targets will have an asterisk in the DES (deselect) column. The GPC target selection can be modified by executing the appropriate control inputs. For example, if GPC 4 fails and is currently a target for the identified MC, it could be deselected by executing Item 40. The asterisk would blank next to Item 39 and appear beside Item 40. The deselection would be effective upon the next request of the OPS displayed to the right of the MC.

Below the GPC SEL/DES are the STRG/GPC assignments. These are likewise dependent on the identified MC. For each STRG, the GPC number is displayed to indicate the GPC that will assume command of that string. The legal values are 0 to 4. To change the desired assignment, the new GPC value is entered with the corresponding item number. For example, if STRG 4 was currently to be assigned to GPC 4 and GPC failed, then STRG 4 could be reassigned to GPC 1 by entering Item 19 with a data value of 1. The change would be effective upon the OPS displayed to the right of MC.

Below the STRG/GPC assignments is the MEMORY SOURCE section. The MEMORY SOURCE provides the control inputs to select the source for memory reconfiguration. The two options are MM(Mass memory) and AUTO (GPC if available, mass memory otherwise). The AUTO shall be the nominal selection; however, there may be occasions where it is desirable to force use of mass memory. The active mode shall be indicated by an asterisk beside the corresponding item number. Only one may be active at a time. The active mode can be changed by executing the appropriate item number.

There is only one fault message directly associated with the use of this display. The single message is ILLEGAL ENTRY. There are, however, many fault messages which will be initiated by conditions which are reflected on the display. These messages will aid the interpretation of the DPS configuration/status.

This display is shown in Figure 3.4.1 - 1.

DATA TABLE 3.4.1-1

NAME DPS_Configuration_Monitor

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	Duty_Cycle_High_Flag	I010.15	Z	142	620,660 665	CZ2B_STATUS		X	X
2	CAM_Status_Votes	I010.18	Z	375 820, 880	620		See App.E		X
3	Vote_Fail	I400	Z			CZ2K_VOTE_FAIL			
4	Discrete_In_Register_A	I610.02	Z	800	See App. E	CZ2B_DIA1/CZ2B_DIA2			
5	CS_And_RS_Masks_For_DPS_CM_Page	I140	Z	390		CZ2B_SET			
6	Memory_Configuration_Number	I010.05	Z	405,820, 880	See App. E	CZ2V_MC	V91U1546C V91U1547C	X	X
							V91U1548C V91U1549C		
7	DPS_Status_Flags	I040	Z	700,660, 910,950	See App. E	CZ2B_DPS_STATUS			
8	Major_Function_Display_Code	I010.20	Z	800	620,485	CZ2V_MF_DPS			X
9	Ops_Number_Display_Code	I010.21	Z	800	620,660, 665	CZ2B_OPS_DPS	See App. E	X	X
10	GPC_Duty_Cycle	I010.01	Z	142,800	See	CZ2V_DUTY_CYCLE	V91Q1586C V91Q1587C	X	X
					App. E		V91Q1588C V91Q1589C		
11	Down_Arrow	I390	Z			CZ2K_DWN_ARROW			
12	LDB_On	I370	Z			CZ2K_LDB_ON			
13	LDB_OFF	I380	Z			CZ2K_LDB_OFF			X
14	GPC_Per_Memory_Configuration	I190	Z	850,860		CZ2B_GPC_MC			X
15	Redundancy_Managed_Discretes	I310	Z		800	CZ2B_DIA_RM1			
16	IOP_Transmitter_State	I010.07	Z	355,830	440,660, 665,830	CZ2B_ACT_XMITR2	V91M8723P V91M8757P	X	
							V91M8791P V91M8822P		
17	Data-Path_Controller_FC_PL_LDB	I010.22	Z	244,355	485,620	CZ2B_DP_DISP			X

DATA TABLE 3.4.1-1 (Cont'd)

NAME DPS_Configuration_Monitor

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
18	IOP_Transmitter_State	I010.06	Z	351, 355, 830	See App. E	CZ2B_ACT_XMITR1	V91M8706P V91M8740P V91M8774P V91M8808P	X	
19	Data_Path_Controller_DK_GSE	I010.23	Z	800	620,800	CZ2B_DK_GSE_DP			X
20	Current_String_Assignment_Table	I080	Z	351, 368	350 392,830	CZ2B_STRNG_ASSIGN			X
21	String_Moding_Assignment_Table	I090	Z	351,ILD		CZ2B_MODE	V91U4110C	X	
22	Memory_Configuration_Nbr_for_GPC/String_Change	I160	Z	850,860	485, 860	CZ2V_MC_RFQ	V91X1592X		X
23	Major_Function_Per_Memory_Configuration	I170	Z	850,860		CZ2V_MF_MC			X
24	OPS_Per_Memory_Configuration	I180	Z	850,860		CZ2V_OPS_MC	V91X1594X		X
25	String_Per_Memory_Configuration	I200	Z	850,860		CZ2B_STRNG_MC			X

```
XXXX/0001/XXXX DPS CONFIG MONITOR  X DDD/HH/MM:SS

GPC    1  2  3  4        CAM   FAIL
MODE  XX XX XX XX         GPC 1 2 3 4        MEMORY
SET   XX XX XX XX         V 1 X X X X      7 MCX  OPS XX
MC     X  X  X  X         O 2 X X X X     GPC SEL DES
MF    XX XX XX XX         T 3 X X X X       1   8X  9X
      XX XX XX XX         E 4 X X X X       2  10X 11X
CPU   XXSXXSXXSXXS      DATA PATHS          3  12X 13X
      VOTER   TIMER       PL1 XSXSXSXS      4  14X 15X
GPC ENA INH ENA INH       PL2 XSXSXSXS
     1X  2X  3X  4X       LB1 XSXSXSXS    STRG    GPC
   MM1 MM2                LB2 XSXSXSXS     1     16 X
   XXX XXX                DK1 XSXSXSXS     2     17 X
    5X  6X                DK2 XSXSXSXS     3     18 X
                          DK3 XSXSXSXS     4     19 X
                          MM1 XSXSXSXS     5     20 X
                          MM2 XSXSXSXS
                           GPC  X X X X X  MEMORY SOURCE
                          STRG  1 2 3 4 5    MM    21X
                          FWDP XSXSXSXSXS    AUTO  22X
                             S           XS
                          AFTP XSXSXSXSXS
                             S           XS

                                                (XX)
```

Figure 3.4.1-1. DPS Configuration Monitor Display

3.4.2 Memory Read/Write

This display provides a crew interface to review and change the value contained in any specified GPC memory location. This display is primarily intended to address data (unprotected) sections of memory. However, a special operating mode will be provided to address code (protected) memory locations. The general use of the display is to first read the present value of a given location. The read function requires crew data inputs to identify memory location. The value may then be changed by a two-step load function. The desired new value is displayed separately as a crew data input. The new value is verified and then transferred to memory by a crew control input.

a. Inputs - Input items are specified in Table 3.4.2-1.

b. Process Description - The display is basically organized with crew data inputs located on the left two-thirds, display outputs on the right one-third, and crew control inputs along the bottom. Each row of the display contains the same components and should be treated on an individual basis. From left to right, the components of each row are ADD ID, DESIRED, and CURRENT. The description of each component is as follows.

1. ADD ID. A six-sigit octal number identifying the memory location being addressed. For quantities stored in full words (i.e., 32 bits), both the most and least significant half addresses must be entered.

2. DESIRED. The new value desired for the identified location. All changes to memory will be done in an ARM/FIRE manner. This requires that the new value be entered and displayed for verification, and then sent to memory for storage. The value being addressed must be entered in half-word octal by using six characters and no sign. The field will blank upon execution of the control input WRITE.

3. CURRENT. The present value stored in the identified location. The displayed value will be of the format defined in 2. Octal quantities will be output as six characters for half words. The six-character half word will be displayed right-adjusted. This field shall be dynamic and updated at a rate of one per second. The value will be initially displayed when the ADD ID is entered.

The display will be initialized with all data fields blank except MODE DATA and GSE POLL. The MODE shall be DATA for the unprotected locations and CODE for the protected locations. The display will be initialized with MODE reflecting DATA. The operating mode can be changed by the control inputs MODE SELECT (Items 1 and 2). Upon execution of either input, the active mode will also be indicated by an asterisk to the right of the corresponding item number, and all data fields will blank. When the CODE mode is selected, the CODE text shall flash. The input

values will be sent to only those GPC's which contain the major function selected on the initiating CRT (in OPS-0 to the GPC which commands the initiating CRT).

The display or review the present value of a quantity in memory, the appropriate ADD ID is made by item entry. The ADD ID may be entered in any row by using the corresponding item number (5, 7, 9 ... 15). When the ADD ID is entered, the present value is displayed in the CURRENT field of the same row. The display of locations in octal shall be on a half-word basis. If it is necessary to change the displayed value, the new value is entered in the DESIRED field of the row. After the DESIRED entry is verified, the memory load is initiated by executing the control input WRITE (Item 4). The DESIRED field will blank, and the CURRENT field will reflect the new value stored.

The capability will be provided to address sequential locations with-out entering data inputs from each individual location. The ADD ID input for the first or starting location is entered. Upon execution of the control input SEQUENTIAL READ (Item 3), the following sequential locations are output in the rows below. The ADD ID and CURRENT fields will be displayed with the value. If multiple locations are displayed when the SEQUENTIAL READ is executed, the last entered ADD ID is assumed to be the starting address. The number of rows or locations displayed can be regulated by selection of the row in which the starting ADD ID is entered. The sequential locations are sequential by quantity, not necessarily by number, The sequential capability is for readout only and will have no application to the load function.

The controls for the selection and initiation of main memory dumps are provided by ADD (Item 20), WDS (Item 21), and DUMP (Item 22). The data input ADD specifies the starting address of the dump. The data input WDS allows the selection of the number of sequential locations to be included in the dump. The control input DUMP initiates the downlink process. The ADD is loaded in octal while the WDS is loaded in decimal. Execution of DUMP without data entries for ADD and WDS will be rejected.

The controls for the GSE POLL function are located in the lower left part of the display. The controls allow turning the POLL on or off vis Items 17 and 18. The active condition is indicated by an asterisk beside the corresponding item.

There shall be one error message available: ILLEGAL ENTRY. This error message shall be driven when one of the input constraints described above are violated. Example violations are as follows:

* Decimal digit entered for ADD ID
* SEQUENTIAL READ executed with no ADD ID entered

* WRITE executed with no DESIRED entered
* ADD ID incompatible with MODE selected
* WRITE executed with DESIRED entered but no corresponding ADD ID

BITE READ will perform a "snapshot" or "cyclic" read of FC MDM, PL MDM, and PCMMU BITE when executed. Each GPC acquires the BITE for BTU's it is commanding and makes the BITE available for contiguous storage of all BITE's in each GPC. The "snapshot" or cyclical mode is controlled by changing the logic setting of a single parameter through general memory read/write. Subsequent execution of Item 19 will act on the parameter logic state. It is initialized at each new memory configuration to the "snapshot" mode.

This display is shown in Figure 3.4.2-1.

DATA TABLE 3.4.2-1

NAME Memory Read/Write Display

NO.	ITEM		ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	GPC_ADDRESS_ID_SAVE_AREA	(ADD ID No. 1)	W040	I	910,915	910,915	CDJB_R_W_LOCID			
2	GPC_ADDRESS_SAVE_AREA	(ADD ID No. 2)	W040	I	910,915	910,915	CDJB_R_W_LOCID	V91X4301X		X
3	GPC_ADDRESS_SAVE_AREA	(ADD ID No. 3)	W040	I	910,915	910,915	CDJB_R_W_LOCID	V91X4302X		X
4	GPC_ADDRESS_SAVE_AREA	(ADD ID No. 4)	W040	I	910,915	910,915	CDJB_R_W_LOCID	V91X4303X		X
5	GPC_ADDRESS_SAVE_AREA	(ADD ID No. 5)	W040	I	910,915	910,915	CDJB_R_W_LOCID	V91X4304X		X
6	GPC_ADDRESS_SAVE_AREA	(ADD ID No. 6)	W040	I	910,915	910,915	CDJB_R_W_LOCID	V91X4305X		X
7	DESIRED_DATA_SAVE_AREA	(DESIRED VALUE NO. 1)	W050	I	910	910	CDJB_R_W_DSRDO	V93X4340X		X
8	DESIRED_DATA_SAVE_AREA	(DESIRED_VALUE NO. 2)	W050	I	910	910	CDJB_R_W_DSRDO	V93X4341X		X
9	DESIRED_DATA_SAVE_AREA	(DESIRED_VALUE_NO. 3)	W050	I	910	910	CDJB_R_W_DSRDO	V93X4342X		X
10	DESIRED_DATA_SAVE_AREA	(DESIRED_VALUE_NO. 4)	W050	I	910	910	CDJB_R_W_DSRDO	V93X4343X		X
11	DESIRED_DATA_SAVE_AREA	(DESIRED_VALUE_NO. 5)	W050	I	910	910	CDJB_R_W_DSRDO	V93X4344X		X
12	DESIRED_DATA_SAVE_AREA	(DESIRED_VALUE_NO. 6)	W050	I	910	910	CDJB_R_W_DSRDO	V93X4345X		X
13	CURRENT_DATA_SAVE_AREA	(CURRENT_VALUE NO. 1)	W060	O	910,915	910,915	CDJB_R_W_CURRO	V91X1160X		X
14	CURRENT_DATA_SAVE_AREA	(CURRENT_VALUE NO. 2)	W060	O	910,915	910,915	CDJB_R_W_CURRO	V91X1161X		X
15	CURRENT_DATA_SAVE_AREA	(CURRENT_VALUE NO. 3)	W060	O	910,915	910,915	CDJB_R_W_CURRO	V91X1162X		X
16	CURRENT_DATA_SAVE_AREA	(CURRENT_VALUE NO. 4)	W060	O	910,915	910,915	CDJB_R_W_CURRO	V91X1163X		X
17	CURRENT_DATA_SAVE_AREA	(CURRENT_VALUE No. 5)	W060	O	910,915	910,915	CDJB_R_W_CURRO	V91X1164X		X
18	CURRENT_DATA_SAVE_AREA	(CURRENT_VALUE No. 6)	W060	O	910,915	910,915	CDJB_R_W_CURRO	V91X1165X		X
19	DATA_MODE_SELECT		W070.1	I	910	910	CDJB_R_W_EXC_STAT			
20	CODE_MODE_SELECT		W070.2	I	910	910	CDJB_R_W_EXC_STAT			

DATA TABLE 3.4.2-1 (cont'd)

NAME Memory Read/Write Display

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
21	GSE_POLL_FLAG	I040.01	I	860,910	830	CX2B_DPS_STATUS		X	X
22	DWNLIST_ADD_ID_FLG	W070.5	I	910	910	CDJB_R_W_EXC_STAT	V93X4375X		X
23	DWNLIST_WRD_CNT_FLG	W070.6	I	910	910	CDJB_R_W_EXC_STAT	V93X4382X		X

```
XXXX/0041/XXXX  MEMORY READ/WRITE  X DDD/HH:MM:SS

MODE
  DATA 1X     SEQ ID 3
  XXXX 2X      WRITE 4

   ADD ID      DESIRED      CURRENT
 5 XXXXXX    6 XXXXXX       XXXXXX
 7 XXXXXX    8 XXXXXX       XXXXXX
 9 XXXXXX   10 XXXXXX       XXXXXX
11 XXXXXX   12 XXXXXX       XXXXXX
13 XXXXXX   14 XXXXXX       XXXXXX
15 XXXXXX   16 XXXXXX       XXXXXX

GSE POLL                          MEMORY DUMP
   ON 17X    BITE READ  19          20 ADD ID XXXXXX
  OFF 18X                           21 WDS XXXXX
                                    DUMP 22

                                                (XX)
```

Figure 3.4.2-1.

3.4.3 Time Management

The time management display provides a method of monitoring the MTU oscillators and accumulator strings, the GPC prime time, and the time source selected for each GPC. Additionally, the display can be used to update the MTU MET/GMT accumulators and GPC MET/GMT or to force a particular time source to be selected.

a. Inputs - See Table 3.4.3-1.

b. Process Description - The display is formatted with MTU oscillator measurements in the upper left and time source measurements in the upper center. The controls for time source selection are in the upper right portion of the display. The middle left and lower right of the display contain the MTU MET/GMT controls and the time source selection measurements.

1. The MTU oscillators are monitored by three sets of measurements as defined in the following paragraphs.

   (a) OSC ACT are discretes intended to show which MTU oscillator has been selected, automatically by the MTU or manually by the crew, via a dedicated switch, to drive the MTU accumulator strings. An asterisk is displayed opposite the particular MTU oscillator, PRI, or SEC selected. A blank will indicate that an oscillator is not selected for accumulator driving.

   (b) OSC STAT is a bilevel measurement that represents the status of the MTU oscillator based on temperature and amplitude. The bilevel is part of the MTU BITE and is the OR of 4 bits (2 per oscillator) of the 16-bit MTU BITE word. When either the temperature or amplitude of an oscillator is out of limits, as determined by the internal MTU comparator logic, BITE is displayed opposite the particular oscillator. Under nominal operations, this bilevel will be blank.

   (c) COMP is a bilevel measurement that reports present status of the MTU oscillators based on comparison of the frequency of each oscillator. This bilevel, like OSC BITE, is part of the MTU BITE and is based on the state of one bit of the MTU BITE word. When the frequencies of the two MTU oscillators are within limits, YES will be displayed. NO will be displayed when the internal MTU comparator logic detects an out-of-limits condition.

2. Time source measurements:

   (a) Monitoring and status of the time sources from the MTU accumulator and internal GPC prime clock are accomplished by four GMT time words located in the upper center of the display.

These time words are the input sources of the GPC time management processor (TMP) and are controlled via the item numbers in the upper right of the display. Status (STAT) information is maintained on the three MTU accumulators and is discussed in paragraph (b) below. The results of the TMP filter for input source failures are displayed adjacent to each GMT time word with "↓" as an indication of the failure.

(b) The STAT measurements are bilevels which indicate the present status of the MTU accumulators 1, 2, and 3. This bilevel is derived from three bits of the MTU BITE word and, if set, will indicate that the accumulator(s) have failed to incorporate the update signals from the MTU oscillator. When this condition exists, BITE is displayed opposite the particular MTU accumulator.

3. Time source selection controls:

Items 12 through 15 are the controls used to manage the inputs to the GPC TMP. User input of ACUM 1, ACUM 2, ACUM 3, or GPC_{PT} shall force the TMP in each GPC, on a given cycle, to start selection with the time obtained from the requested source. If this source is found to be out of tolerance by any GPC, that GPC shall return to automatic source selection on the next cycle. For automatic source selection, the TMP in each GPC shall sequentially difference the four time sources with GPC myself (GPC_M) time until an absolute delta of less than a constant, K_1 is found, and select this source. If a selected source is an MTU accumulator source and if it fails the K_1 test in any GPC, then the other two MTU accumulator sources shall be tested before trying GPC_{PT}. As a result, TMP shall fault down to GPC_{PT} only when all three MTU sources fail the K_1 test on a single given cycle; however, once that occurs, the MTU sources shall no longer be examined until the TMP is reinitialized or a user input request via Items 12 through 14 is received to select an MTU source. If GPC_{PT} is selected and it fails the K_1 test, then GPC_M shall be used until TMP reinitialization or a user input request via Items 12 through 15 is received. Whenever a source selection is changed due to a failed condition of the selected source, that GPC shall generate an appropriate error message.

4. MTU MET/GMT controls:

(a) Items 1 through 9 are the controls used to update the MTU and GPC, GMT, and/or MET times. Items 1 through 4 allow the input of a delta time in days, hours, minutes, seconds, and milliseconds to the mission elapsed time of the MTU and GPC.

Upon execution of Item 9, time update, the values loaded in Items 1 through 4 will be used by the GPC to construct and initiate the necessary commands to the MTU to cause MET to be updated by the input delta time. In addition, each GPC shall adjust its MET clock by the requested update. Items 5 through 8 allow the input of a delta time in days, hours, minutes, seconds, and milliseconds to the GMT accumulators of the MTU and GPC. Upon execution of Item 9, time update, the values loaded in Items 5 through 8 will be used by the GPC to construct and initiate the necessary commands to the MTU to cause GMT to be updated by the input delta time. In addition, however, the value of the update shall be rejected if the request is for more than 15 milliseconds. Each GPC shall adjust its RUNTIME and processing by the requested update. By limiting the update to 15 milliseconds, no GPC will go inactive for more than 15 milliseconds. The single exception to an update of more than 15 milliseconds is during initial GPC power up (single GPC). During this process, GMT may be updated by days, hours, minutes, seconds, and milliseconds of a full year. If Items 1 through 4 and 5 through 8 (at least one item of each) are loaded and Item 9 is executed, both MET and GMT, as noted above, will be simultaneously updated.

(b) ACUM SYNC. Execution of Item 10 shall force all GMT time sources to be set equal to the source currently selected by the "Prime" GPC; that is, if the current source selected by the Prime GPC is an MTU, each GPC shall compute and attempt to command a zero delta MTU/GMT update via the bus containing that source. This results in all other sources being synchronized to the MTU selected by the Prime GPC.

(c) Item 11 is used to set the MET accumulators in the MTU and GPC MET to all zeros.

5. Time source selection measurements:

GPC SOURCE. These messages indicate which of the five available sources, ACUM 1, ACUM 2, ACUM 3, GPC_{PT}, and GPC_{M} (Internal), are presently selected for the TMP of all GPC's. Each of the four GPC slots shall contain one of the following: A1, A2, A3, PT, or IT.

DATA TABLE 3.4.3-1

NAME Time Management Display

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
1	Controlling_Oscillator	#017.31	I			TFCMM1BW			X
2	MTU_Primary_Oscillator_Status	H040.15	I	955		CAAB_TM_STAT_FLG	V91X1001X		X
3	MTU_Secondary_Oscillator_Status	H040.20	I	955		CAAB_TM_STAT_FLG	V91X1009X		X
4	Time_Mgmt_GMT_Display_Values	H010	I	955		CAAV_TM_GMT_DISP	V91W1040C V91W1042C		X
							V91W1044C V91W1046C		
5	Time_Mgmt_GMT_Milliseconds_Display_Values	H020	I	955		CAAV_TM_MLS_DISP	V91W1040C V91W1042C		X
							V01W1044C V01W1046C		
6	MTU_1_Status	#013.1	I	149		TFCMMASB	V93X4103X		X
7	MTU_2_Status	#013.2	I	149		TFCMMASB	V93X4105X		X
8	MTU_3_Status	#013.3	I	149		TFCMMASB	V93X4107X		X
9	GPC_Prime_Time_Status	#013.4	I	149		TFCMMASB	V93X4109X		X
10	Frequency_Divider_1	#017.37	I	149		TFCMM1BW			
11	Frequency_Divider_2	#017.38	I	149		TFCMM1BW			X
12	Frequency_Divider_3	#017.39	I	149		TFCMM1BW			X
13	Oscillator_Frequency_Difference	#017.34	I			TFCMM1BW			X
14	Time_Source	I010.19	I	149	620,660, 665	CZ2B_STATUS	V91Q1710C V91Q1711C	X	X
							V91Q1712C V91Q1713C		
15	Time_Mgmt_GMT_Delta_Values	H030.1	I	950	950	CAAV_TM_GMT_DELT	V91W1070C V91W1071C		X
							V91W1072C		
16	Time_Mgmt_GMT_Seconds_Delta_Value	H030.2	I	950	950	CAAV_TM_GMT_SEC	V91W1073C		X

DATA TABLE 3.4.3-1 (cont'd)

NAME Time Management Display

NO.	ITEM	ID	ACT	SOURCE	DESTI-NATION	HAL NAME	MML	D	C
17	Time_Mgmt_MET_Delta_Values	H030.3	I	950	950	CAAV_TM_MET_DELT	V91W1080C V91W1081C		X
							V91W1082C		
18	Time_Mgmt_MET_Seconds_Delta_Value	H030.4	I	950	950	CAAV_TM_MET_SEC	V91W1083C		X

```
XXXX/0021/XXXX    TIME MANAGEMENT    X DDD/HH:MM:SS

   MTU OSC     |          TIME SOURCES                    | SEL FL
    ACT STAT   |              GMT                  STAT   | FORCE
PRI X  XXXX    | ACUM 1  XXX/XX:XX:XX.XXXS         XXXX   |  12
SEC X  XXXX    | ACUM 2  XXX/XX:XX:XX.XXXS         XXXX   |  13
COMP XXX       | ACUM 3  XXX/XX:XX:XX.XXXS         XXXX   |  14
               | GPC PT  XXX/XX:XX:XX.XXXS                |  15

GPC      1  2  3  4
SOURCE  XX XX XX XX

                      DYS        HR         MN        SC/MS
MET DELTA    1 [±]XXX  2 [±]XX  3 [±]XX  4 [±]XX.XXX
GMT DELTA    5 [±]XXX  6 [±]XX  7 [±]XX  8 [±]XX.XXX
TIME UPDATE  9.

ACUM SYNC   10
MET RESET   11

                                                  (XX)
```

Figure 3.4.3-1. Time Management Display

CPDS - Detailed Design Cross Reference

APPENDIX A

SSDS/CPDS CROSS REFERENCE

SSDS	CPDS
3.1.1	4.6.4.1-B,C, 4.6.4.1.2-A
3.1.1.1	4.6.3.1.11, 4.6.3.3.1, 4.6.4.1, 4.6.4.1.2, 4.6.4.5.1, 4.6.4.5.2
3.1.1.2	4.6.4.1.2-D, 5.8.1.3.1, 5.8.2
3.1.2	4.6.3.3.5, 4.6.4.1, 4.6.4.2, 4.6.4.1.3, 4.3.5.7.1, 5.3.5.7.5, 5.8.2.5-D
3.1.3	4.6.2.3.1, 4.6.2.3.3-D, 4.6.4.1-A, 4.6.4.15, 5.3.2.1-B, 5.3.5.1.1, 5.3.5.8, 5.8.1.3.3-B
3.2.1	4.6.3.1.8-A-B, 4.6.3.3.2-C, 4.6.4.1.1, 5.5.4, 5.6.3, 5.8.1.1-B,C, 5.8.1.3.1-C,D, 5.8.2.5-E
3.2.3.1	4.6.2.2-C, 4.6.2.2.3, 4.6.2.2.5, 4.6.2.3.9, 4.6.3.1.8-B, 4.6.3.3.1-I,K,L, 4.6.4.1.4, 4.6.5, 5.3.3.7.5, 5.3.5.9, 5.8.2.5-A,B, 5.8.3, 5.9.4.1
3.2.3.2	4.6.3.1.8-A,B, 4.6.3.3.1-K, 4.6.4.3.1, 4.6.4.3.5-A, 4.6.4.6-7, 4.6.4.4, 4.6.4.5.2-A, 5.3.3.1.2-D, 5.3.3.7-9, 5.3.5.1.3, 5.8.2.1-F, 5.8.2.5, 5.8.3
3.2.3.3	4.6.2.2.5-F, 4.6.3.3.1-A, 5.8.2.1-B, 5.9.4.1
3.2.3.4	4.6.2.2.5, 4.6.2.3.11, 4.6.3.3.1, 4.6.4.3.5-A, B, C, 4.6.4.5.3, 5.3.5.1.2, 5.8.2, 5.8.2.3, 5.8.2.5-E.4
3.2.3.5	4.6.2.2.2-C, 4.6.2.2.3, 4.6.2.2.5-D,E,F, 4.6.3.3.1-I, 5.3.5.9, 5.3.5.9.2-7
3.2.3.6	4.6.2.2.2-C, 4.6.2.2.3, 4.6.2.2.5-D, E, F, 4.6.3.3.1-I, 5.3.5.9, 5.3.5.9.2-7
3.3.1	4.6.3.3.1-G, 4.6.6.2.1, 5.2-F, 5.9.3
3.3.2	4.6.2.4.3-C, 4.6.2.4.5-10, 4.6.6.2.3, 5.9.2
3.3.3	4.6.2.3-H, 5.9.5

Resource Allocation Summary

Appendix B

SC Resource Allocation Summary

Item	Main Memory Words	
System Initialization		700
Initial Start Sequence		
GPC Locator (AIB_GPC_LOCATOR)	371	
GPC Startup (AIR_GPC_STARTUP)	63	
System Interface Processor (AIE_SIP)	266	
DEU Loader (AIG_DEU_LOADER)	(*)	
System Reconfiguration		814
GPC Switch Monitor (ARA_GPC_SWITCH	392	
Idle Operational Sequence (ARB_IDLE_OPS	293	
Force OPS 0 (AOP_FORCE_OPS_0)	129	
DPS Reconfiguration		2337
GPC Reconfiguration (ARC_GPR_RECONFIG)	627	
Bus Configuration Change (ARD_BUS_CHG)	900	
GPC Reconfiguration Table Change (ARF_GPC_TABLE_CHG)	(*)	
DPS Configuration ITEM Processor (ARF_DPS_CONFIG_ITEM)	309	
GPC Reconfiguration Message Handler (ARG_RECONFIG_MSG)	118	
Secondary GPC Reconfiguration (ARH_SEC_GPC_RECONFIG)	383	
System Specialist Functions		1060
Idle Specialist Function (ARB_IDLE_OPS)	(*)	
Read/Write Specialist Function (ASB_RD_WRT)	444	
Read/Write Cyclic Display Update (ASH_RW_CYC_UPDATE)	47	
Bite Execute Item Processor (ASI_RW_BITE_PROC)	32	
Time Management Specialist Function (ASC_TIME_MGMT)	286	
Time Management Display Update Cyclic Processor (ASG_CYCLIC_UPDATE)	110	
Time Management MTU Update Processor (ASJ_MTU_UPDATE)	141	
DEU Self Test Specialist Function (ASE_DEU_ST)	(*)	
DEU Self Test Loader (ASF_DEU_ST_LOADER)	(*)	

Compool Data	Main Memory Words	
Compool Total		3008
UI-FCOS_SHARED_COMPOOL (CZ2_COMMON)	572	
SYSTEM_CONTROL_COMPOOL (CAA_SC_COMMON)	72	
READ/WRITE_DATA (CDJ_RW_DATA)	32	
DEU/MMU_READ_BUFFER (CAB_COMMON)	*(4353)	
SC DISPLAYS (CDBDFB)	282	
DISPLAY_BUFFERS (CDG02Y/CDH025)	2050	

* Size not included because item is not permanently resident.

APPENDIX C

SYSTEM CONTROL MODULE LIST

Module ID	Module Name	HAL/S Name	Section
700	GPC_Locator	AIB_GPC_LOCATOR	3.1.1.1
710	GPC_Startup	AIR_GPC_STARTUP	3.1.1.2
750	System_Interface_Processor	AIE_SIP	3.1.2
790	DEU_Loader	AIG_DEU_LOADER	3.1.3
800	GPC Switch Monitor	ARA_GPC_SWITCH	3.2.1
810	Idle_Operational_Sequence	ARB_IDLE_OPS	3.2.2
815	Force_OPS_0	AOP_FORCE_OPS_0	3.2.2.1
820	GPC_Reconfiguration	ARC_GPC_RECONFIG	3.2.3.1
830	Bus_Configuration_Change	ARD_BUS_CHG	3.2.3.2
850	GPC_Reconfiguration_Table_Change	ARF_GPC_TABLE_CHG	3.2.3.3
860	DPS_Configuration_ITEM_Processor	ARF_DPS_CONFIG_ITEM	3.2.3.4
870	GPC-Reconfiguration_Message_Handler	ARG_RECONFIG_MSG	3.2.3.5
880	Secondary_GPC_Reconfiguration	ARH_SEC_GPC_RECONFIG	3.2.3.6
910	Read/Write_Specialist_Function	ASB_RD_WRT	3.3.1
915	Read/Write_Cyclic_Display-Update	ASH_RW_CYC_UPDATE	3.3.1.1
920	Bite_Execute_Item_PRocessor	ASI_RW_BITE_PROC	3.3.1.2
950	Time_Management_Specialist_Function	ASC_TIME_MGMT	3.3.2
955	Time_Management_Display_Update_ Cyclic_Processor	ASG_CYCLIC_UPDATE	3.3.2.1
975	Time_Management_MTU_Update_Processor	ASJ_MTU_UPDATE	3.3.2.2

APPENDIX D

ERROR CONDITIONS

ALT SYSTEM SOFTWARE ERRORS

NO. BY TYPE	ERROR CONDITION	SOFTWARE RESPONSE	FMPT ID	TEXT MAJOR	TEXT MIN	CLASS
OT004	ALL TARGET GPCs NOT IN RUN MODE	OPS STARTED IN RUN GPC	G1	GPC	CONF	3
OT005	NO TARGET GPC IN RUN MODE	REQ REJ	G1	GPC	CONF	3
OT006	REQUEST REDUNDANT OPERATION SIMPLEX OPS	REQ REJ	K1	ILLEGAL ENTRY	MC	5
OT007	REQUEST MEMORY RECONFIGURATION WHEN GPC(S) NOT IN OPS 00 RUN	REQ REJ	K1	ILLEGAL ENTRY	MC	5
OT008	PRO AFTER MMU ERROR IN SIMPLEX OPS TRANSITION	REQ REJ	K9	ILLEGAL ENTRY	MMU	5
OT009	PRO AFTER MMU ERROR IN AN RS OPS TRANSACTION WHEN NO GPC RECEIVED THE OVERLAY	REQ REJ	K9	ILLEGAL ENTRY	MMU	5
TM002	NORMAL GMT UPDATE GREATER THAN 15 MILLISECONDS	INP REJ	K3	ILLEGAL ENTRY	LIM	5
TM003	TIME UPDATE NOT ENTERED WHEN UPDATE COMMAND INPUT	REQ REJ	K6	ILLEGAL ENTRY	DATA	5
I0025	I/O ERROR IN DEU 1 INIT	REQ REJ	I13	I/O ERROR	DEU1	3

NO.BY TYPE	ERROR CONDITION	SOFTWARE RESPONSE	FMPT	TEXT MID	MAJOR	MIN	CLASS
I0026	I/O ERROR IN DEU 2 INIT	REQ REJ	I14		I/O ERROR	DEU2	3
I0027	I/O ERROR IN DEU 3 INIT	REQ REJ	I15		I/O ERROR	DEU3	3
OS018	ATTEMP TO ASSIGN LDB TO NON-GPCp IN RS	REQ REJ	K5		ILLEGAL ENTRY	CONF	5
OS021	BUS HANDOVER FAIL/FC1	REQ REJ	H1		BUS HANDOVER	FC	3
OS022	BUS HANDOVER FAIL/FC2	REQ REJ	H1		BUS HANDOVER	FC	3
OS023	BUS HANDOVER FAIL/FC3	REQ REJ	H1		BUS HANDOVER	FC	3
OS024	BUS HANDOVER FAIL/FC4	REQ REJ	H1		BUS HANDOVER	FC	3
OS025	BUS HANDOVER FAIL/FC5	REQ REJ	H1		BUS HANDOVER	FC	3
OS026	BUS HANDOVER FAIL/FC6	REQ REJ	H1		BUS HANDOVER	FC	3
OS027	BUS HANDOVER FAIL/FC7	REQ REJ	H1		BUS HANDOVER	FC	3
OS028	BUS HANDOVER FAIL/FC8	REQ REJ	H1		BUS HANDOVER	FC	3
OS029	BUS HANDOVER FAIL/PL1	REQ REJ	H2		BUS HANDOVER	PL	3
OS030	BUS HANDOVER FAIL/PL2	REQ REJ	H2		BUS HANDOVER	PL	3

NO. BY TYPE	ERROR CONDITION	SOFTWARE RESPONSE	FMPT	TEXT MID	MAJOR	MIN	CLASS
OS033	BUS HANDOVER FAIL/DK1	REQ REJ	H4		BUS HANDOVER	DK	3
OS034	BUS HANDOVER FAIL/DK2	REQ REJ	H4		BUS HANDOVER	DK	3
OS035	BUS HANDOVER FAIL/DK3	REQ REJ	H4		BUS HANDOVER	DK	3
OS036	BUS HANDOVER FAIL/MM1	REQ REJ	H5		BUS HANDOVER	MMU	3
OS037	BUS HANDOVER FAIL/MM2	REQ REJ	H5		BUS HANDOVER	MMU	3
OS046	ICC BUFFER CANNOT ACCEPT MESSAGE	IGNORE	X1		OVERFLOW	ICC	3
DU021	SEQ READ REQ WITH NO LOC ID (R/W SPEC)	INP REJ	K6		ILLEGAL ENTRY	DATA	5
DU022	EXEC LOAD WITH NO DATA (R/W SPEC)	INP REJ	K6		ILLEGAL ENTRY	DATA	5
DU023	LOC ID INCOMPATIBLE WITH MODE (R/W SPEC)	INP REJ	K6		ILLEGAL ENTRY	DATA	5
DU025	EXEC LOAD WITH NO LOC ID (R/W SPEC)	REQ REJ	K6		ILLEGAL ENTRY	DATA	5
DU031	ADDRESS EXCEEDS MEMORY SIZE	REQ REJ	K6		ILLEGAL ENTRY	DATA	5
DU032	NO DUMP ADDRESS SPECIFIED	REQ REJ	K6		ILLEGAL ENTRY	DATA	5

NO. BY TYPE	ERROR CONDITION	SOFTWARE RESPONSE	FMPT	TEXT MID	MAJOR	MIN	CLASS
DU035	REQUEST FOR DEU SASTP WHEN GPC NOT IN SM9	REQ REJ	K6		ILLEGAL ENTRY	CONF	5
SC001	ATTEMPT TO ASSIGN A DEU TO A GPC WITH UNLIKE MF (CMPTR/ CRT KEY)	REQ REJ	K5		ILLEGAL ENTRY	CONF	5
SC002	ATTEMPT TO ASSIGN BUSES CONTROLLED BY A GPC THAT IS NOT LISTENING TO THE REQUESTING DEU	REQ REJ	K5		ILLEGAL ENTRY	CONF	5
SC003	ILLEGAL AB CODES OF CMPTR/BUS KEY	REQ REJ	K4		ILLEGAL ENTRY	SYN	5
SC005	ATTEMPT TO ASSIGN STRING 5 TO GPC NOT COMMANDING STRINGS 1, 2, OR 3	REQ REJ	K5		ILLEGAL ENTRY	CONF	5
SC009	CRT CODE = 0	REQ REJ	K6		ILLEGAL ENTRY	DATA	5

APPENDIX E

DATA DESCRIPTION TABLE

The Data Descriptors Appendix provides detailed information about each parameter. Parameters in the Data Descriptors can be cross referenced to parameters in the modular Data Tables via a parameter identification tag (ID). The definition of each field in the table follows.

1. ID - A parameter ID has been assigned to each data item in the format

   ZXXX[.XX]

   where:

   Z - Single character denoting the function of the parameter, chosen from one of the following:

   - A - Moding of Control Segments (See Vol II, P2)
   - D - Display Formats (See Vol II, P3)
   - B - Mass Memory (See Vol II, P2)
   - T - Annunciation (See Vol II, P2)
   - V - LDB (See Vol II, P2)
   - F - Downlist (See Vol II, P2)
   - W - Read/Write of Main Memory (See Vol II, P3)
   - H - Time Management (See Vol II, P3)
   - I - FCOS/CI Common Compool (See Vol II, P3)
   - J - Common (Miscellaneous) (See Vol II, P2)
   - L - User Interface Local Variable (See Vol II, P2)
   - X - System Control Local Variable (See Vol II, P3)
   - Y - FCOS I/O Management (See Vol II, P1)
   - @ - FCOS Process Management (See Vol II, P1)
   - O - FCOS Configuration Management (See Vol II, P1)
   - # - FCOS COMPOOL (See Vol II, P1)
   - Q - FCOS Control Blocks (See Vol II, P1)
   - S - FCOS SVC Parameter List (See Vol II, P1)
   - & - Hardware Parameters (See Vol II, P1)

   XXX - A unique three digit number assigned to each major division. (A major division is a structure, table, array, or a single parameter not contained in one of the previously mentioned items.

   [.XX]- Second level qualifier for elements of arrays, parts of structures or tables or the bits in a flag or discrete word.

2. ITEM - A unique meaningful English_Equivalent_Name for the data item.

3. HAL/Assembler Name - The HAL/S or Assembler Name used in coding. The column is blank if no name is assigned as is the case with bits in a flag word. (User Interface and System Control column name is "HAL NAME" FCOS column is "ASSEMBLER NAME").

4. Description - A concise statement of the nature of the parameter and its purpose.

5. Source - A listing of the three digit ID's of all modules updating the parameter and/or one of the following special codes:

   ILD - The parameter is of the Initial Load (I-LOAD) type (see Level A CPDS, Table 6-4).

   MRW - The parameter is available to memory read write (see Level A CPDS, Table 6-4).

6. Destination - A list of the three digit ID's of all modules referencing the parameter and/or one of the following special codes:

   DL - The parameter is available for downlink.
   CRT - The parameter is available for display.

7. Attributes - The attributes of the parameter are as follows:

   a. Data Type - The HAL/S declaration type or assembler language type

ST(n)	-	Structure with n copies
A(m,n),k	-	Array of m,n dimensions of HAL parameter type k
BS(n)	-	Bit string of length n
BT	-	Bit
C(n)	-	Character string of length n
SC	-	Scalar/Assembler Language Floating Point
I	-	Integer/Assembler Language Fixed Point Halfword
BO	-	Boolean
M(n,m)	-	Matrix of n,m dimensions
V(n)	-	Vector of n elements
E	-	Event
Y	-	Halfword Address Constant
AC	-	Fullword Address Constant
Z	-	Fullword Indirect Address Constant
X	-	Hexadecimal
F(n)	-	Fullword Fixed Point n Copies

NOTE: Double length of a parameter type is denoted by prefixing a 'D' to the above characters.

b. Initial Value - The initialization value if applicable;

INIT(value)

c. Units of Measure - The units of measure of the parameter if applicable:

UM(unit)

d. Other - Any of the attributes such as REPLACE, LOCU(h), etc.

8. MML - The MML number associated with the parameter. The column is blank if none is defined.

In part 1 (FCOS) bits in a word will be numbered starting with 0. In parts 2 and 3 (UI and SC) bits in a word will be numbered starting with 1.

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
D010	DFT_CONTROL_WORD 1		CONTAINS DISPLAY DEPENDENT IDENTIFICATION DATA BIT 1 NOT USED BIT 2 INDICATES RESIDENCY OF STATIC DATA 0= GPC RESIDENCE 1= DEU RESIDENCE BIT 3 NOT USED BITS 4-12 HEX VALUE INDICATING DFG VERSION USED TO GENERATE THE DISPLAY FORMAT TABLE (DFT) BITS 13-16 HEX VALUE INDICATING THE NUMBER OF DISPLAY NUMBERS ASSOCIATED WITH THE DISPLAY	DFG	520	BS(16)	
D010.1	GPC_DEU_RESIDENCE_INDICATOR						
D010.2	DISPLAY_NUMBER_ALIAS_VALUE						
D020	DFT_ICS_LENGTH		HEX VALUE INDICATING THE LENGTH OF THE DFT, IN 16-BIT WORDS, WHEN USING HAL/S-101 SOFTWARE SUPPORT	520 620	DFG	BS(16)	
D020.1	DFT_FSIM_LENGTH		HEX VALUE INDICATING THE LENGTH OF THE DFT, IN 16-BIT WORDS, WHEN USING THE HAL/S-360 SOFTWARE SUPPORT. NOTE: IOS D020 AND D020.1 ARE MUTUALLY EXCLUSIVE ENTRIES IN THE DFT	DFG	520	BS(16)	
D030	DFT_KVT_POINTER		HEX VALUE INDICATING THE NUMBER OF DFT ENTRIES FROM THE BEGINNING OF THE DFT TO THE KEYBOARD VERIFICATION TABLE (KVT)	DFG	520	BS(16)	
D040	DFT_FCW_POINTER		HEX VALUE INDICATING THE NUMBER OF DFT ENTRIES FROM THE BEGINNING OF THE DFT TO THE FIRST STATIC FORMAT CONTROL WORD (FCW)	DFG	600	BS(16)	
D050	DFT_DDT_POINTER		HEX VALUE INDICATING THE NUMBER OF DFT ENTRIES FROM THE BEGINNING OF THE DFT TO THE FIRST DYNAMIC DATA TABLE (DDT) ENTRY	DFG	600 620 635	BS(16)	
D060	DFT_DISPLAY_ID_TABLE		CONTAINS IDENTIFICATION DATA FOR THE DFT GENERATED. THERE IS 1 16-BIT ENTRY FOR EACH DFG HEADER COMMAND CODED	DFG	520	BS(16)	
D060.1	MAJOR_FUNCTION_CODE		BITS 1&2 INDICATE MAJOR FUNCTION AS FOLLOWS:				

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
			"00" SYSTEMS SERVICES "01" PAYLOAD MANAGEMENT "10" GNEC "11" SYSEM MANAGEMENT				
D060.2	DISPLAY_ID_NUMBER		BITS 3-16 CONTAIN THE DISPLAY NUMBER IN BINARY FORMAT				
D070	DFT_KVT_DATA		CONTAINS DATA DESCRIBING THE ITEM PROCESSING FOR THE DISPLAY BEING DEFINED	DFG	520	BS(16)	
D070.1	ITEM_COUNT_IND		BIT 1-8 HEX VALUE INDICATING THE NUMBER OF ITEM ENTRIES ON THE FORMAT				
D070.2	PRO_KEY_VALIDITY_BIT		BIT 9 WHEN ON INDICATES THE PRO KEYSTROKE IS ALLOWED ON THE FORMAT				
D070.3	EXEC_KEY_VALIDITY_BIT		BIT 10 WHEN ON INDICATES THE EXEC KEYSTROKE IS ALLOWED ON THE FORMAT BIT 11-13 NOT USED				
D070.4	ONE_TIME_ONLY_UPDATE_FLAG		BIT 14 WHEN ON INDICATES THE FORMAT IS TO BE UPDATED ONLY ONCE BIT 15-16 NOT USED				
D080	DFT_SCALAR_LIMIT_TABLE_POINTER		HEX VALUE INDICATING THE NUMBER OF DFT ENTRIES FROM THE BEGINNING OF THE DFT TO THE SCALAR LIMIT TABLE. VALUE IS ZERO WHEN NO LIMIT CHECKS MADE ON SCALAR ITEM INPUTS	DFG	510	BS(16)	
D090	DFT_NON_SCALAR_LIMIT_TABLE_POINTER		HEX VALUE INDICATING THE NUMBER OF DFT ENTRIES FROM THE BEGINNING OF THE DFT TO THE NON-SCALAR LIMIT TABLE.VALUE IS ZERO WHEN NO LIMIT CHECKS MADE ON NON-SCALAR ITEM INPUTS	DFG	510	BS(16)	
D100	DFT_ITEM_WORD 1		CONTAINS DATA DESCRIBING ITEM "N" WHERE "N" MAY BE FROM 1 TO 99 BIT 1 WHEN ON INDICATES ITEM ENTRY TERMINATES WITH THE PRO KEYSTROKE BIT 2 WHEN ON INDICATES ITEM ENTRY TERMINATES WITH THE EXEC KEYSTROKE BIT 3-5 INDICATE ITEM DATA ENTRY TYPE AS FOLLOWS:	DFG	510	BS(16)	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
			'101' EITHER SCALAR OR OCTAL '102' OCTAL '010' INTEGER '001' SCALAR BIT 6-12 INDICATE ENTRY WITHIN THE APPROPRIATE LIMIT TABLE BIT 13 WHEN ON INDICATES THE ENTIRE DISPLAY SHOULD BE UPDATED WHENEVER THIS ITEM IS EXECUTED BIT 14 WHEN ON INDICATES THE ENTIRE DISPLAY SHOULD BE UPDATED WHENEVER THIS ITEM IS ENTERED BITS 15-16 NOT USED				
D110	DFT_ITEM_WORD 2		CONTAINS THE FORMAT (NUMBER OF CHARACTERS) OF THE ITEM DATA ENTRY	DFG	510	BS(16)	
D120	DFT_SCALAR_LIMIT_TABLE		CONTAINS LOWER AND UPPER LIMIT VALUES USED TO VALIDATE PREVIOUSLY IDENTIFIED ITEM DATA ENTRIES. WHEN NO SCALAR ITEM ENTRIES REQUIRE LIMIT CHECKING THIS PORTION OF THE COMPOOL IS NOT GENERATED	DFG	510	BS(64)	
D130	DFT_NON_SCALAR_LIMIT_TABLE		CONTAINS LOWER AND UPPER LIMIT VALUES USED TO VALIDATE PREVIOUSLY IDENTIFIED ITEM DATA ENTRIES. WHEN NO NON-SCALAR ITEM ENTRIES REQUIRE LIMIT CHECKING THIS PORTION OF THE COMPOOL IS NOT GENERATED	DFG	510	BJ(32)	
D140	DFT_STATIC_FCWS		CONTAINS STATIC FORMAT CONTROL WORDS (FCWS) FOR PROCESSING BY THE DEU. THE NUMBER OF STATIC FCWS WILL VARY FROM DISPLAY TO DISPLAY	DFG	600	BS(16)	
D150	DFT_DYNAMIC_DATA_TABLE		CONTAINS A MIXTURE OF FCWS AND FCWS,WHICH WHEN PROCESSED BY THE CYCLIC DISPLAY PROCESSOR,REPRESENT THE DYNAMI PORTION OF THE DISPLAY. THE LENGTH OF THE DYNAMIC DATA TABLE (DDT) WILL VARY FROM DISPLAY TO DISPLAY	DFG	635 640	BS(16)	
D160	MULTIPLE_DISCRETE_ADDRESS_TABLE	ARBXCCCC	TABLES CONTAINING INFORMATION ABOUT ADDRESSES AND BITS USED IN MULTIPLE DISCRETE(MDT) PROCESSING A TABLE NAME IS FORMED AS FOLLOWS: "AR" -INDICATES MAJOR FUNCTION	DFG	635	ADDRESSES ARE DECLARED AS NAME VARIABLES BITS REFERENCED ARE	

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
			CORRELATION AS FOLLOWS: "D" SYSTEM SERVICES "G" GN&C "S" SYSTEM MANAGEMENT "P" PAYLOAD SYSTEMS "BB" -A NUMERIC VALUE FROM 00-99 USED TO PROVIDE UNIQUE NAMES FOR THE TABLES "X" -CODED AS "X" TO DENOTE THAT TABLE GENERATED IS AN "X" TABLE "CCCC"-USER SUPPLIED DATA THAT IDENTIFIES THE TABLE AND OR ITS FUNCTION			DECLARED AS BS(16)	
D170	MULTIPLE_DISCRETE_TEXT_TABLE	ABBYCCCC	TABLES CONTAINING THE TEXT TO BE DISPLAYED DEPENDING ON THE BIT SETTINGS PROCESSED IN THE MULTIPLE DISCRETE TEST (MDT) PROCESS. THE TABLE NAME IS DERIVED AS FOLLOWS: "A" -INDICATES MAJOR FUNCTION CORRELATION AS FOLLOWS "D"SYSTEMS SERVICES "G"GN&C "S"SYSTEM MANAGEMENT "P"PAYLOAD SYSTEMS "BB" -A NUMERIC VALVE FROM 00-99 USED TO PROVIDE UNIQUE NAMES FOR THE TABLES "Y" -CODED AS "Y" TO DENOTE THAT TABLE GENERATED IS A "Y" TABLE "CCCC"-USER SUPPLIED DATA THAT IDENTIFIES THE TABLE AND OR ITS FUNCTION	DFG	635	TEXT DATA IS REPRESENTED AS BS(16) WHICH PROVIDES AN INDEX INTO A TABLE CONTAINING ALL VALID DISPLAYABLE TEXT IN THE SYSTEM	
D200	BILEVEL_TEST_CMD	BLT	DFG COMMAND TO BE USED WHENEVER THE TESTING OF TWO(2) BITS IS REQUIRED BEFORE DYNAMIC DATA CAN BE DISPLAYED.	DFG	620 635	A(3),BS(16)	

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
D200.01	BLT_OP_CODE		BITS 1-6 INITIALIZED TO BIN '000001' BITS 7,8 NOT USED	DEG			
D200.02	BLT_BIT1_ID		BITS 9-12 CONTAINS THE FIRST BIT ID TO BE TESTED -1(0-15)	DEG			
D200.03	BLT_BIT2_ID		BITS 13-16 CONTAINS THE SECOND BIT ID TO BE TESTED-1(0-15)	DEG	635		
D200.04	BLT_ADR1_PTR		THE GPC LOCATION FOR THE COMPOOL VARIABLE WHERE THE FIRST BIT TO BE TESTED IS STORED	DEG	635		
D200.05	BLT_ADR2_PTR		THE GPC LOCATION FOR THE COMPOOL VARIABLE WHERE THE SECOND BIT TO BE TESTED IS STORED	DEG	635		
D201.00	FCW_REMOTE_CMD	FCWSR	DEG COMMAND TO BE USED WHENEVER THE USER WISHES TO REFERENCE CHARACTER FCWS THAT HAVE BEEN CREATED IN THE GPC AND ARE TO BE SHIPPED TO THE DEU	DEG	635	A(2),BS(16)	
D201.01	FCW_OP_CODE		BITS 1-6 INITIALIZED TO BIN '000010' BITS 7-15 NOT USED	DEG			
D201.02	FCW_TYPE		BIT 16 WHEN SET=1 THE USER WISHES TO REFERENCE THE FCW'S DIRECTLY WHEN SET=0 THE USER WISHES TO REFERENCE THE FCW'S INDIRECTLY	DEG			
D202.00	REMOTE_TEXT_CMD	RTC	DEG COMMAND THAT ALLOWS THE USER TO DISPLAY 1 OF 2 CHARACTER STRINGS,WITH A MAXIMUM LENGTH OF 4,DEPENDING ON THE SETTING OF A BIT IN COMPOOL	DEG	620 635	A(3),BS(16)	
D202.01	RTC_OP_CODE		BITS 1-6 INITIALIZED TO BIN '000011' BITS 7-8 NOT USED	DEG			
D202.02	RTC_CONTROL_BIT		BITS 9-12 CONTAINS THE CONTROL BIT ID TO BE TESTED -1(0-15).IF THE TESTED BIT IS ON THE FIRST HALF OF THE PAIRED CHARACTER TEXT IS	DEG	620 635		

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
			DISPLAYED,OTHERWISE THE SECOND HALF IS DISPLAYED				
D202.03	RTC_INTENSITY		BITS 13,14 00=NORMAL 01=DOUBLE,IF CONTROL BIT=1 10=DOUBLE,IF CONTROL BIT=0	DEC			
D202.04	RTC_TEXT_LENGTH		BITS 15,16 00=1 CHARACTER 01=2 CHARACTER 10=3 CHARACTER 11=4 CHARACTER	DEC			
D202.05	RTC_ADR1_PTR		THE GPC LOCATION OF THE COMPOOL VARIABLE CONTAINING THE CONTROL BIT	DEC	620 635		
D202.06	RTC_ADR2_PTR		THE GPC LOCATION OF THE CHARACTER STRINGS TO BE DISPLAYED	DEC			
D204	VARIABLE_PARAMETER_CMD	VPARM	DEC COMMAND USED BY THE USER TO DEFINE I/O CHARACTERISTICS OF VARIABLE PARAMETERS	DEC	620 635	A(3)B5(16)	
D204.01	VPARM_OP_CODE		BITS 1-6 INITIALIZED TO BIN '000101'	DEC	620		
D204.02			BITS 7-8 NOT USED	DEC	640		
D204.04	VPARM_INTERNAL_CHARACTERISTICS		BITS 9-12	DEC	620		
D204.05	VPARM_SPS		0001-SINGLE PRECISION SCALAR	DEC	620		
D204.06	VPARM_SPI		0010-SINGLE PRECION IN INTEGER	DEC	620		
D204.07	VPARM_SPI		0011-DOUBLE PRECISION INTEGER	DEC	620		
D204.08	VPARM_DPS		0100-DOUBLE PRECISION SCALAR	DEC	620		
D204.09	VPARM_BIT_16		0101-16 BIT INTEGER	DEC	620 640		
D204.10	VPARM_BIT_32		0110-32 BIT DATA	DEC	620 640		
D204.11	VPARM_OUTPUT_CHARACTERISTICS		BITS 13-16	DEC	620		
D204.12	VPARM_TIME		0001-TIME				
D204.13	VPARM_INTEGER		0010-INTEGER				

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
D204.14	VPARM_OCTAL		0011-OCTAL				
D204.15	VPARM_SM_CAL		0100- CALIBRATION-SN				
D204.16	VPARM_RAD_TO_DEG		0101- RADIANS TO DEGREES				
D204.17	VPARM_KILO_TO_FT		0110-KILOMETERS TO FEET				
D204.18	VPARM_METERS_TO_FEET		0111-METERS TO FEET				
D204.19	VPARM_MET_SEC_TO_KNOTS		1000-METERS /SEC TO KNOTS				
D204.20	VPARM_MET_NAT_MILES		1001-FEET TO NAUTICAL MILES				
D205.00	VPARM_FMT 1		BITS 1-4 INDICATES TOTAL NUMBER OF DIGITS IN FIELD TO BE OUTPUTED	DEG	635 640		
D205.01	VPARM_FMT2		BITS 5-8 INDICATES THE NUMBER OF DIGITS TO THE RIGHT OF THE DECIMAL POINT	DEG	640		
D205.02	VPARM_DOWNLIST_STATUS		BIT 9: 0=DOWNLIST DISABLED 1=DOWNLIST ENABLED				
D205.03	VPARM_ZEROES_STATUS		BIT 10 0=LEADING ZEROES SUPPRESSED 1=LEADING ZEROES DISPLAYED	DEG	620 640		
D205.04	VPARM_SIGN_STATUS		BIT 11-14:	DEG			
D205.05			BITS 15-16 NOT USED	DEG			
D205.06	VPARM_ADDR_PTR		THE GPC LOCATION OF THE VARIABLE TO BE DISPLAYED	DEG	640		
D211.00	REMOTE_CHARACTER	RCHAR	DEG COMMAND USED TO DISPLAY CHARACTER DATA REMOTELY	DEG	620 635	A(2), BS(15)	
D211.01	REMOTE_CHAR_OP_CODE		BITS 1-6 INITIALIZED TO BIN '001010'	DEG			
D211.02			BITS 7-8 NOT USED	DEG			
D211.03	REMOTE_CHARACTER_LENGTH		BITS 9-16 EQUAL NUMBER OF CHARACTERS TO BE DISPLAYED	DEG	640 635		
D211.04	REMOTE_CHARACTER_		BITS 1-16 ADDRESS OF REMOTE DATA	DEG	640		

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
	ADDR		TO BE DISPLAYED		635		
D213.00	STATUS_BYTE_CMD	SBC	DFG COMMAND USED TO DISPLAY STATUS INDICATORS AND/OR BLANK OUT A DYNAMICALLY UPDATED PARAMETER	620	620 635	A(2),BS(16)	
D213.01	STATUS_BYTE_OP_CODE		BITS 1-6 INITIALIZED TO BIN '001101'				
D213.02			BITS 7-16 NOT USED				
D213.03	STATUS_BYTE_ADDR		THE GPC LOCATION OF THE STATUS WORD TO BE PROCESSED				
D217.00	MULTIPLE_DISCRETE_TEST_CMD	MDT	DFG COMMAND USED WHEN MORE THAN TWO BITS MUST BE TESTED TO DETERMINE THE CORRECT DATA TO DISPLAY	DFG	620 635	A(3),BS(16)	
D217.01	MDT_OP_CODE		BITS 1-6 INITIALIZED TO BIN '010001' BITS 7-16 NOT USED	DFG			
D217.02	MDT_XTABLE_ADDR		THE GPC LOCATION OF THE XTABLE TO BE USED	DFG	620 635		
D217.03	MDT_YTABLE_ADDR		THE GPC LOCATION OF THE YTABLE TO BE USED	DFG	620 635		
D218.00	TEST_COMMAND	TEST	DFG COMMAND USED TO CONDITIONALLY EXECUTE OR BRANCH AROUND SELECTED DFG INPUT COMMANDS	DFG	620 635	A(3),BS(16)	
D218.01	TEST_OP_CODE		BITS 1-6 INITIALIZED TO BIN '010010'	DFG	620 635		
D218.02	TEST_CONTROL_BIT		BITS 13-16 CONTROL BIT VALUE-1	DFG	620 635		
D218.03	TEST_DDTS_TO_SKIP		BITS 1-16 NUMBER OF DDT ENTRIES TO SKIP	DFG	620 635		
D218.04	TEST_ADDR		THE GPC LOCATION OF THE CONTROL BIT TO BE TESTED	DFG	620 635		
D219.00	IMMEDIATE_DATA	IMMEDIATE	THIS DFG COMMAND ALLOWS A NUMBER OF PRE-BUILT FCWS TO BE DYNAMICALLY SENT TO THE SCREEN	DFG	620 635		
D219.01	IMMEDIATE_OP_CODE		BITS 1-6 INITIALIZED TO BIN '010100'				
D219.02	NUMBER_OF_FCWS		BITS 9-16 NUMBER OF FCWS THAT ARE		620		

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	OST	ATTRIBUTES	MML
			TO BE DYNAMICALLY DISPLAYED ON THE SCREEN		635		
D220.00	RATE_COMMAND	RATE	DFG COMMAND USED TO INDICATE THE UPDATE RATE FOR SUBSEQUENT DDT ENTRIES AND THE NUMBER OF DDTS IN THAT RATE GROUP	DFG	620 635	A(2),BS(16)	
D220.01	RATE_OP_CODE		BITS 1-6 INITIALIZED TO BIN '010101'	DFG			
D220.02	RATE_VALUE		BITS 13-16 RATE VALUE	DFG			
D220.03	RATE_FCW_COUNT		BITS 1-16 FCW COUNT FOR CURRENT RATE	DFG	620 635		
D221.00	BRANCH_COMMAND	BR	DFG COMMAND USED TO SKIP PAST A SPECIFIED NUMBER OF DDT ENTRIES	DFG	620 635	A(2),BS(16)	
D221.01	BRANCH_OP_CODE		BITS 1-6 INITIALIZED TO BIN '010110'. BITS 7-16 NOT USED	DFG			
D221.02	BRANCH_DDT_SKIP_COUNT		BITS 1-16 NUMBER OF DDT ENTRIES TO SKIP	DFG	620 635		
D222.00	ON_DEMAND_CMD	DEMAND	DFG COMMAND USED TO ALLOW USER A TECHNIQUE FOR SPECIFYING THAT A DISPLAY IS TO BE REDRIVEN	DFG	635	A(2),BS(16)	
D222.01	ON_DEMAND_OP_CODE		BITS 1-6 INITIALIZED TO BIN '010111'	DFG			
D222.02			BITS 7-12 NOT USED	DFG			
D222.03	ON_DEMAND_BIT_NUMBER		BITS 13-16 INDEX TO BIT TO BE TESTED	DFG	635		
D222.04	ON_DEMAND_BIT_ADDR		BIT 1-16 ADDRESS OF USERS BIT TO BE TESTED	DFG	620 635		
D250	DEU_MEMORY_FILL_STATUS	CABV_ERR_STAT	DEU MEMORY FILL TRANSMISSION STATUS	790	790	A(2),I	
D260	DEU_MEMORY_FILL_BUFFER	CABB_DEU_LOAD	BUFFER USED TO READ THE DEU CONTROL PROGRAM IN FROM MASS MEMORY	790	790	A(8704),BS(16)	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	OST	ATTRIBUTES	MML
H010	TIME_MGMT_GMT_DISPLAY_VALUES	CAAV_TM_GMT_DISP	GMT VALUES FOR MTU ACCUMULATORS 1-3 AND GPC PRIME TIME IN SECONDS	955	CRT	A(4),DSC,INIT(0)	V91W1040C V91W1042C V91W1044C V91W1046C
H020	TIME_MGMT_GMT_MILLISECONDS_DISPLAY_VALUES	CAAV_TM_MLS_DISP	GMT MILLISECOND VALUES FOR MTU ACCUMULATORS 1-3 AND GPC PRIME TIME	955	CRT	A(4),SC,INIT(0)	V91W1040C V91W1042C V91W1044C V91W1046C
H030	TIME_MGMT_DELTA_INPUT_VALUES	CAAV_TM_DELTA_DISP	GMT AND MET DELTA TIME INPUT VALUES	950	950	ST(1)	
H030.1	TIME_MGMT_GMT_DELTA_VALUES	CAAV_TM_GMT_DELT	GMT DELTA TIME INPUTS FOR DAYS, HOURS,MINUTES	950	950 CRT	A(3),DI	V91W1070C V91W1071C V91W1072C
H030.2	TIME_MGMT_GMT_SECONDS_DELTA_VALUE	CAAV_TM_GMT_SEC	GMT DELTA TIME INPUT FOR SECONDS AND MILLISECONDS	950	950 CRT	DSC	V91W1073C
H030.3	TIME_MGMT_MET_DELTA_VALUES	CAAV_TM_MET_DELT	MET DELTA TIME INPUTS FOR DAYS, HOURS,MINUTES	950	950 CRT	A(3),DI	V91W1080C V91W1081C V91W1082C
H030.4	TIME_MGMT_MET_SECONDS_DELTA_VALUE	CAAV_TM_MET_SEC	MET DELTA TIME INPUT FOR SECONDS AND MILLISECONDS	950	950 CRT	DSC	V91W1083C
H040	TIME_MGMT_STATUS_FLAGS	CAAB_TM_STAT_FLG	STATUS FLAGS FOR CONTROLLING TIME MANAGEMENT DISPLAY AND MTU UPDATES (BITS 2-6,16 NOT USED)	950 955 975	950 975	BS(16)	
H040.10	GMT_DISPLAY_BLANK_STATUS		BIT 1-DISPLAY GMT VALUES =1 DISPLAY BLANKS =0 DISPLAY GMT VALUES FOR MTU ACCUMULATORS AND GPC PRIME TIME	950 955	CRT		
H040.15	MTU_PRIMARY_OSCILLATOR_STATUS		BIT 7-PRIMARY OSCILLATOR STATUS =1 TEMPERATURE OR AMPLITUDE OF THE PRIMARY OSCILLATOR IS OUT OF LIMITS =0 STATUS IS IN LIMITS	955	CRT		V91X1001X
H040.20	MTU_SECONDARY_OSCILLATOR_STATUS		BIT 8-SECONDARY OSCILLATOR STATUS =1 TEMPERATURE OR AMPLITUDE OF THE SECONDARY OSCILLATOR IS OUT OF LIMITS =0 STATUS IS IN LIMITS	955	CRT		V91X1009X

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
H040.25	GMT_DELTA_VALUES_SAVED		BIT 9-GMT ITEM'S 5-8 ENTERED =1 GMT DELTA VALUES SAVED FOR ITEM 5,6,7 OR 8 =0 NO DELTA VALUES ENTERED	950	950		
H040.30	MET_DELTA_VALUES_SAVED		BIT 10-MET ITEM'S 1-4 ENTERED =1 MET DELTA VALUES SAVED FOR ITEM 1,2,3 OR 4 =0 NO DELTA VALUES ENTERED	950	950		
H040.35	UPDATE_IN_PROCESS		BIT 11-MTU UPDATE IN PROCESS =1 MTU UPDATE SVC ISSUED(OR SCHEDULED TO BE ISSUED) =0 NO UPDATE IN PROCESS	950 975	950		
H040.40	TERMINATE_TIME_MGMT_SPEC		BIT 12-TERMINATE PROCESSING FLAG =1 SIGNAL ASJ_MTU_UPDATE TO CLOSE DUE TO TERMINATION =0 NO TERMINATION REQUESTED	950	975		
H040.45	REQUEST_MET_UPDATE		BIT 13-MET UPDATE REQUESTED =1 ISSUE MTU SVC FOR MET UPDATE =0 NO MET UPDATE REQUESTED	950 975	975		
H040.50	REQUEST_NORMAL_GMT_UPDATE		BIT 14-NORMAL GMT UPDATE REQUESTED =1 ISSUE MTU SVC FOR NORMAL GMT UPDATE =0 NO GMT UPDATE REQUESTED	950 975	975		
H040.55	REQUEST_INITIAL_GMT_UPDATE		BIT 15-INITIAL GMT UPDATE REQUESTED =1 ISSUE MTU SVC FOR INITIAL GMT UPDATE =0 NO GMT UPDATE REQUESTED	950 975	975		
H050	TIME_MGMT_BLANKING_FLAGS	CAAB_TM_BLNK_FLG	FLAGS FOR CONTROLLING DISPLAY OF GMT AND MET DELTA VALUES. THE EIGHT COPIES OF THE ARRAY CORRESPOND TO ITEM'S 1-8 ON TIME MGMT DISPLAY. BITS 2-16 OF EACH ARRAY ARE NOT USED BIT 1=1:ITEM NOT ENTERED,DISPLAY BLANKS =0:ITEM ENTERED,DISPLAY VALUE	950		A(8),BS(16)	
H070	MTU_UPDATE_EVENT	CAAE_MTU_UPDATE	EVENT FOR WAITING ON COMPLETION OF MTU UPDATE AND ACTIVATING ASJ_MTU_UPDATE	148 950 975		E	
H080	MTU_UPDATE_PARAMETER_LIST	CAAV_MTU_PARMS	SVC PARAMETER LIST FOR MTU UPDATES	950 975	148	ST(1)	

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
H080.1	MTU_REQUEST_TYPE_INDICATOR	CAAV_MTU_PARMS.TYPE_IND	MTU UPDATE REQUEST TYPE: UPDAT_MET, UPDAT_GMT, INIT_GMT MASTER_RESET OR RESET_MET	950 975	148	BS(8)	
H080.20	MTU_UPDATE_COINCIDENT_TIME	CAAV_MTU_PARMS.COINC_TIME	TIME AT WHICH MTU UPDATE IS TO OCCUR.	975	148	1	
H080.30	MTU_UPDATE_TIME	CAAV_MTU_PARMS.UPDAT_TIME	TIME VALUE FOR NEW GMT OR MET	975	148	DSC	
H080.40	MTU_UPDATE_DELTA_TIME	CAAV_MTU_PARMS.MET_DELTA	DELTA VALUES FOR GMT OR MET UPDATE	975	148	DSC	
H090	GMT_DELTA_VALUE	CAAV_DELT_GMT	GMT TOTAL DELTA VALUE IN SECONDS (SUM OF ITEMS 5-8)	950	975	DSC	
H100	MET_DELTA_VALUE	CAAV_DELT_MET	MET TOTAL DELTA VALUE IN SECONDS (SUM OF ITEMS 1-4)	950	975	DSC	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
I010	GPC_STATUS_TABLE	CZ2V_GST	TABLE CONTAINING DATA UNIQUE TO A GPC AND REQUIRED IN EACH GPC	830	830	ST(4)	
I010.01	GPC_DUTY_CYCLE	CZ2V_DUTY_CYCLE (TGSTDUTY, CZ2VGST1,CZ2VGST2, CZ2VGST3, CZ2VGST4)	FCCS COMPUTED % OF CPU UTILIZATION	142 800	485 620 660 665 CPT DL	SC, INIT(0), UM(%)	V91Q1586C V91Q1587C V91Q1588C V91Q1589C
I010.02	CURRENT_OPS	CZ2V_CURRENT_OPS (TGSTPOPS TGSTGOPS, TGSTSOPS)	CURRENT OPS FOR EACH MF (PL,GN, SM RESPECTIVELY)	560 820 880	440 485 800 815 820 830 950	A(3),I,INIT(0)	
I010.03	PROGRAM_OVERLAY_ NUMBER	CZ2V_PROG_OVLY (TGSTPGOV)	PROGRAM OVERLAY NUMBER CURRENTLY RESIDENT	820 880	820 880	I,INIT(0)	
I010.04	MAJOR_FUNCTION_ OVERLAY_NUMBER	CZ2V_MF_OVLY (TGSTMFOV)	MAJOR FUNCTION OVERLAY NUMBER CURRENTLY RESIDENT	820 880	820 880	I,INIT(0)	
I010.05	MEMORY_ CONFIGURATION_ NUMBER	CZ2V_MC (TGSTMCNC)	MEMORY CONFIGURATION NUMBER CURRENTLY RESIDENT	405 820 880	405 620 800 810 830 DL CPT	I,INIT (0)	V91U1546C V91U1547C V91U1548C V91U1549C
I010.06	IOP_TRANSMITTER_ STATE	CZ2B_ACT_XMITR1 (TGSTTCCT)	MASK OF ACTUAL IOP TRANSMITTER STATE FOR BUSES 1-15 BEGINNING WITH BIT 2	351 355 830	DL 148 400 485 660 665 815 830	BS(16),INIT(0)	V91M8706P V91M8740P V91M8774P V91M8808P
I010.07	IOP_TRANSMITTER_ STATE	CZ2B_ACT_XMITR2	MASK OF ACTUAL IOP TRANSMITTER STATE FOR BUSES 16-24 BEGINNING WITH BIT 1	355 830	DL 440 660 665 830	BS(16),INIT(0)	V91M8723P V91M8757P V91M8791P V91M8822P
I010.08	IOP_RECEIVER_STATE	CZ2B_ACT_RECVR (TGSTTCCR)	MASK OF ACTUAL IOP RECEIVER STATE FOR BUSES 1-24 BEGINNING WITH BIT 2	335 355	400 440 815 830	BS(32),INIT(0)	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
I010.09	BUS_MASKS	CZ2B_BUS_MASK1 (TGSTBMSK)	MASK OF I/O BUSES 1-15 FOR WHICH I/C IS TO BE INHIBITED BEGINNING WITH BIT 2	355 MPW	148 205 620 660 665 830 CRT DL	BS(16),INIT(0)	V91M7922P V91M7957P V91M8011P V91M8011P V91M8055P
I010.10		CZ2B_BUS_MASK2	MASK OF I/O BUSES 16-24 FOR WHICH I/C IS TO BE INHIBITED BEGINNING WITH BIT 1			BS(16),INIT(0)	V91M7940P V91M7974P V91M8038P V91M8038P V91M8072P
I010.11	DATA_PATH_MASKS	CZ2A_DPM (TGSTDPM)	MASK OF DATA PATHS INCLUDING BUSES 10-23 FOR IUA 11-17 BEGINNING WITH BIT 1	MPW 244 290 355	DL 148 290 465	A(7),BS(16), INIT(0)	V91M4000P V91M4017P V91M4034P V91M4051P V91M4068P VP1M4085P V91M4102P V91M4204P V91M4221P V91M4238P V91M4255P V91M4272P V91M4300P V91M4317P V91M4419P V91M4436P V91M4453P V91M4470P V91M4487P V91M4500P V91M4517P V91M4619P V91M4636P V91M4653P V91M4670P V91M4687P V91M4704P V91M4721P
I010.12	COMMON_SET_MASK	CZ2B_CS (TGSTCSM)	COMMON SET MASK BEGINNING WITH BIT 12 FOR GPCI WHERE OTHER MEMBERS OF SET ARE REPRESENTED BY A 1 BIT	360 361 368 370 700	DL 485 490 700 750 830 950	BS(16),INIT(0)	V91M2087P V91M2121P V91M2155P V91M2200P

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MNL
					955		
I010.13	REDUNDANT_SET_MASK	CZ28_RS (TGSTRSM)	REDUNDANT SET MASK BEGINNING WITH BIT 12 FOR GPC1 WHERE OTHER MEMBERS OF SET ARE REPRESENTED BY A 1 BIT	320 351 368 392 820 880	DL 320 495 560 660 665 815 820 830	BS(16),INIT(0)	V91M2070P V91M2104P V91M2138P V91M2172P
I010.14	GST_STATUS_FLAGS	CZ28 STATUS(TGSTIND)	GST STATUS FLAGS.BITS 1,2,6-8 NOT USED	142 149 368 375 560 700 820 870 880	148 149 485 620 660 665 710 750 870 CPT DL	BS(16),INIT(0001)	
I010.15	DUTY_CYCLE_HIGH_FLAG		DUTY CYCLE EXCEEDS DUTY CYCLE HIGH LIMIT. BIT 3	142	620 660 665 CPT DL	BT,INIT(0)	
I010.16	DOWNLIST_FORMATTER_ENABLED_FLAG		DOWNLIST FORMATTER ENABLED BIT 4	560 700 820 880	710 750	BT,INIT(0)	
I010.17	OVERLAY_IN_PROGRESS_FLAG		RECONFIGURATION OF RS OR MAIN MEMORY IN PROGRESS.BIT 5	820 870 880		BT,INIT(0)	
I010.18	CAM_STATUS_VOTES		SET OF THIS GPC'S VOTES AGAINST OTHER GPCS IN RS. BITS 9-13 CORRESPOND TO GPCS 1-5 RESPECTIVELY	375 820 880	CPT 620	BS(5),INIT(0)	V91X1561X V91X1562X V91X1563X V91X1564X V91X1566X V91X1567X V91X1568X V91X1569X V91X1571X V91X1572X V91X1573X V91X1574X V91X1576X V91X1577X

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
							V91X1578X V91X1579X
I010.19	TIME_SOURCE		TIME SOURCE FOR MANAGEMENT BITS 14-16 MTU1-1 MTU2-2 MTU3-3 GPCPT-4 GPCMT-5	149	620 660 665 CPT DL	BS(3),INIT(1)	V91Q1710C V91Q1711C V91Q1712C V91Q1713C
I010.20	MAJOR_FUNCTION_ DISPLAY_CODE	CZ2V_MF_ DPS(TGSTMFD)	MAJOR FUNCTION DISPLAY CHARACTER CODE FOR EACH OF TWO LINES PROVIDED ON THE OPS CM PAGE BITS 1-8: CHARACTER FOR LINE 1 BITS 9-16: CHARACTER FOR LINE 2 PL-P GNC-G SM-S	800	CRT 620 485	C(2),INIT( )	
I010.21	OPS_NUMBER_DISPLAY_ CODE	CZ2B_OPS_DPS	OPS NUMBER DISPLAY CODE FOR EACH OF TWO LINES PROVIDED ON THE DPS CM PAGE BITS 1-12 UNUSED BITS 13-16 OPS NUMBER	800	620 660 665 CPT DL	A(2),BS(15),INI T(0)	V91U1511C V91U1516C V91U1512C V91U1517C V91U1513C V91U1518C V91U1514C V91U1519C
I010.22	DATA_PATH_ CONTROLLER_FC_PL_ LDB	CZ2B_DP_ DISP(TGSTDPD)	DATA PATH CONTROLLER FOR FC,PL,AND LD BUSES FIRST HALFWORD: BITS 1-4 FF1-4 PRIMARY PORTS BITS 5-8 FF1-4 SECONDARY PORTS BITS 9-10 MMU1-2 BITS 11-12 PF1 PORTS 1&2 BITS 13-14 LF1 PORTS 1&2 BITS 15-16 NOT USED SECOND HALFWORD: BITS 1-4 FA1-4 SECONDARY PORTS BITS 5-8 FA1-4 PRIMARY PORTS BITS 9-10 NOT USED BITS 11-12 PF2 PORTS 1&2 BITS 13-14 LF2 PORTS 1&2 BITS 15-16 NOT USED	244 355	CRT 485 620	A(2),BS(16),INI T(0)	
I010.23	DATA_PATH_ CONTROLLER_DK_GSE	CZ2B_DK_GSE_DP	DATA PATH CONTROLLER FOR DK AND GSE BUSES BITS 1-3 DK BUSES 1-3 BITS 4-5 GSE BUSES 1-2	800	CRT 620 800	BS(16),INIT(0)	

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
			BITS 6-16 NOT USED				
I020	T_ZERO	CZ2V_TZERO(CZ2VTZER)	BASE TIME AT WHICH DOWNLINK BEGINS A NEW CYCLE (FRAME 0)	700 700	485 700 820 980	SCD,UM(SECONDS)	
I030	T_SIP	CZ2V_TSIP	PROCESS SCHEDULING TIME BASE	700	CPT DL 700	SCD,UM(SECONDS)	V92W4200C V92W4204C V92W4206C V92W4208C
I040	DPS_STATUS_FLAGS	CZ2B_DPS_STATUS(CZ2BSTAT)	DST STATUS FLAGS BITS 1,6,9-13 NOT USED	700 860 910 950	149 485 560 660 665 910 820 830 880	BS(16),INIT(A000 0)	
I040.01	GSE_POLL_FLAG		GSE POLL FLAG ENABLING LDB I/O PROCESSING.BIT 2	860 910	CPT DL 830	BT,INIT(0)	
I040.02	MEMORY_SOURCE		MEMORY SOURCE TO USE FOR MAIN MEMORY OVERLAY BIT 0-0 FORCE ACCESS FROM MMU 1-ACCESS MMU ONLY IF NOT ALREADY RESIDENT	860 TLD	CPT DL 560 820 880	BT,INIT(1)	V93X4371X
I040.03	RM_WATCH_DOG_TIMER_FLAG		TIMER FLAG ENABLING GPC RM PROCESSING.BIT 4	860 TLD	CPT DL	BT,INIT(0)	V91X1533X
I040.04	RM_VOTER_FLAG		VOTER FLAG ENABLING GPC RM PROCESSING BIT5	860 TLD	CPT DL	BT,INIT(0)	
I040.05	MF_LINE1_DISPLAY		SET TO 1: USED BY RTC CMD IN DFG BIT 6	810			
I040.06	MF_LINE2_DISPLAY		SET TO 0: USED BY RTC CMD IN DFG BIT 7:				
I040.07	FORCE_SELECTED_TIME_SOURCE		CODE SPECIFYING TIME SOURCE TO BE USED AT NEXT MTU RM COMPUTATION BITS 14-16	149 950	149	BS(3),INIT(0)	
I050	GPC_PRIME_ID	CZ2V_GPCP((CZ2VGPCP)	PRIME GPC ID	351 700 880 820	142 148 148 350		

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
					368		
					440		
					495		
					600		
					660		
					665		
					815		
					830		
					950		
I060	MASS_MEMORY/MAJOR_FUNCTION_ASSIGNMENT	CZ2B_MM_MF(CZ2BMMMF)	TABLE SPECIFYING WHICH MMU TO ACCESS FOR EACH MF.ONLY THE SM ENTRY IS USED, BITS 3 AND 4 REPRESENT MMU1 AND 2 RESPECTIVELY FOUR HALFWORDS ARE USED,EACH REPRESENTING A MF AS FOLLOWS: PL GN,SM & OFFLINE (STANDBY)	860 ILD	810	A(4),BS(16) INIT(0000,2000, 2000,1000)	
I060.01	MASS_MEMORY/SM_ASSIGNMENT	CZ2BMMSM	SM ENTRY ONLY IS USED	860 ILD		BS(16),INIT(200 0)	
I070	STRING_DEFINITION_TABLE	CZ2B_STRNG_DEF (CZ2BSDT)	DEFINITION OF BUSES PER STRING BITS 1 AND 26-32 NOT USED BITS 2-25 REPRESENT BUSES 1-24 RESPECTIVELY BITS 11&15 ON(1) FOR BUSES 10 & 14 BITS 12 & 16 ON(1) FOR BUSES 11 & 15 BITS 13 & 17 ON(1) FOR BUSES 12 & 16 13 & 17 18-22	351	350 355 485	A(5)BS(32) INIT(00220000 00110000 00088000 00044000 00003E00)	
I080	CURRENT_STRING_ASSIGNMENT_TABLE	CZ2B_STRNG_ASSIGN(CZ2VSAT)	DEFINITION OF GPC COMMANDING WHICH STRINGS BITS 1-12 NOT USED BITS 13-16 GPC ID (INTEGER) IN COMMAND OF STRINGS 1-5	351 368	CPT 350 392 830	A(5),BS(16) INIT(0)	V91X1721X V91X1722X V91X1723X V91X1724X V91X1725X
I090	STRING_MODING_ASSIGNMENT_TABLE	CZ2B_MODE(CZ2BMODE)	DEFINITION OF MODE FOR EACH STRING BITS 1,7-16 NOT USED BITS 2-6 REPRESENT STRINGS 1-5 RESPECTIVELY 1=MODE 6(PRIMARY PORT) 0=MODE 7(SECONDARY PORT)	351 ILD	OL	BS(16),INIT(7C0 0)	V91U4110C
I100	BTU_BITE_ACQUISITION_TABLE	CZ2V_BTU_BITE	CONTIGUOUS SET OF HALFWORDS FOR STORING BTU BITE	280 361 405 820 880 920	485 660 665 920 CPT OL	ST(1),INIT(0)	

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
I100.01	FLIGHT_CRITICAL_BTU_BITE	CZ2B_BTU_BITE_FC	CONTIGUOUS SET OF 8 HALFWORDS FOR STORING THE BTU BITE FROM THE 8 FLIGHT CRITICAL MDM'S	820 880 920	CRT DL	A(8),BS(16) INIT(0)	
					485 660 665		
I100.02	FLIGHT_AFT_1_BTU_BITE	(CZ2BTFA1)	FLIGHT AFT 1 BTU BITE	820 880 920	CRT DL	BS(16),INIT(0)	V72M5480P
I100.03	FLIGHT_AFT_2_BTU_BITE	(CZ2BTFA2)	FLIGHT AFT 2 BTU BITE	820 880 920	CRT DL	BS(16),INIT(0)	V72M5500P
I100.04	FLIGHT_AFT_3_BTU_BITE	(CZ2BTFA3)	FLIGHT AFT 3 BTU BITE	820 880 920	CRT DL	BS(16),INIT(0)	V72M5520P
I100.05	FLIGHT_AFT_4_BTU_BITE	(CZ23TFA4)	FLIGHT AFT 4 BTU BITE	820 880 920	CRT DL	BS(16),INIT(0)	V72M5540P
I100.06	FLIGHT_FORWARD_1_BTU_BITE	(CZ2BTFF1)	FLIGHT FORWARD 1 BTU BITE	820 880 920	CRT DL	BS(16),INIT(0)	V72M5400P
I100.07	FLIGHT_FORWARD_2_BTU_BITE	(CZ2BTFF2)	FLIGHT FORWARD 2 BTU BITE	820 880 920	CRT DL	BS(16),INIT(0)	V72M5420P
I100.08	FLIGHT_FORWARD_3_BTU_BITE	(CZ2BTFF3)	FLIGHT FORWARD 3 BTU BITE	820 880 920	CRT DL	BS(16),INIT(0)	V72M5440P
I100.09	FLIGHT_FORWARD_4_BTU_BITE	(CZ2BTFF4)	FLIGHT FORWARD 4 BTU BITE	820 880 920	CRT DL	BS(16),INIT(0)	V72M5460P
I100.10	PAYLOAD_BTU_BITE	CZ2B_BTU_BITE_PL	CONTIGUOUS SET OF 2 HALFWORDS FOR STORING THE BTU BITE FROM THE 2 PAYLOAD MDM'S	264 800 920 880 920	CRT DL	A(2),BS(16),INIT(0)	
I100.11	PAYLOAD_FORWARD_1_BTU_BITE	(CZ2BTPF1)	PAYLOAD FORWARD 1 BTU BITE	920 880 920	CRT DL	BS(16),INIT(0)	V72M5600P
I100.12	PAYLOAD_FORWARD_2_BTU_BITE	(CZ2BTPF2)	PAYLOAD FORWARD 2 BTU BITE	820 880 920	CRT DL	BS(16),INIT(0)	V72M5620P
I100.13	PCMMU_BTU_BITE	CZ2B_BTU_BITE_PCM	PCMMU BTU BITE	820	CRT	BS(16),INIT(0)	V72M8261P

FCOS/CI COMMON COMPOOL (CZ2_COMMON)

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
				880	DL 920		
I100.14	MMU1_BTU_BITE	CZ2B_BTU_BITE_MM1 (CZ2BTMM1)	CONTIGUOUS SET OF 2 HALFWORDS FOR STORING THE BTU BITE FROM MASS MEMORY UNIT 1	280 282 820 880	CRT DL 282	A(2),BS(16) INIT(0)	
I100.15	MMU2_BTU_BITE	CZ2B_BTU_BITE_MM2 (CZ2BTMM2)	CONTIGUOUS SET OF 2 HALFWORDS FOR STORING THE BTU BITE FROM MASS MEMORY UNIT 2	280 820 880	CRT DL 282	A(2),BS(15) INIT(0)	
I100.16	MTU1_BTU_BITE	CZ2B_BTU_BITE_ MTU1(CZ2BTMT1)	MTU1 BTU BITE	149 361 820 880	660 665	BS(16),INIT(0)	
I100.17	MTU2_BTU_BITE	CZ2B_BTU_BITE_ MTU2(CZ2BTMT2)	MTU2 BTU BITE	149 361 820 880		BS(16),INIT(0)	
I100.18	MTU3_BTU_BITE	CZ2B_BTU_BITE_MTU3 (CZ2BTMT3)	MTU3 BTU BITE	149 361 820 880		BS(16),INIT(0)	
I100.19	DEU_BTU_BITE	CZ2B_BTU_BITE_DEU	CONTIGUOUS SET OF 12 HALFWORDS FOR STORING THE BTU BITE FROM THE DEU'S	405 820 880	CRT DL 660 665	A(3,4),BS(15),I NIT(0)	V72M5650P V72M5660P V72M5680P V72M5690P V72M5720P V72M5740P V72M5750P V72M5760P V72M5810P V72M5830P V72M5840P V72M5850P
I110	NOMINAL_STRING_ ASSIGNMENT_TABLE	CZ2B_NOM_STRNG (CZ2BNOMS)	NOMINAL STRING ASSIGNMENT TABLE FOR EACH OF 4 MEMORY CONFIGURATIONS.EACH TABLE CONTAINS GPC ID PER STRINGS 1-5 BITS 1-12 UNUSED BITS 13-16 GPC ID	850 860 110	CRT DL 351 485 860	A(4,5),BS(16) INIT(1,2,3,4,1 1,2,3,4,1,1,1,1,1 1,1,1,1,1,1,1,1	
I120	REQUESTED_GPC_FOR_ GSE_COMMAND	CZ2V_LDB_GPC_REQ (CZ2VLDBC)	CREW INPUT OF DESIRED GPC TO COMMAND GSE I/O		CRT 485	I,INIT(1)	V93X4064X
I130	REQUESTED_BUS_FOR_ GSE_I/O	CZ2V_LDB_BUS_REQ (CZ2VLDBB)	CREW INPUT OF DESIRED LAUNCH DATA BUS	700	CRT	I,INIT(1)	V93X4065X

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
I140	CS_AND_RS_MASKS_FOR_DPS_CM_PAGE	CZ2B_SET(CZ2BSET)	CS/RS MASKS TO DETERMINE SET TO DISPLAY ON DPS CM PAGE BITS 1-5 REPRESENT GPC'S 1-5 IN REDUNDANT SET BITS 6-10 REPRESENT GPC'S 1-5 IN COMMON SET BITS 11-16 NOT USED	390		BS(16)	
I150	COMPUTER_CRT_ASSIGNMENT_TABLE	CZ2V_CCATS	GPC ID FOR EACH MAJOR FUNCTION FOR EACH DEU	830	830	ST(3)	
I150.01	GPC_TO_COMMAND_DEU	CZ2V_CCAT, (CZ2VCCAT)	GPC TO COMMAND DEU 1,2,OR 3 UPON SELECTION OF PL,GN,OR SM MAJOR FUNCTION.EACH ARRAY OF 3 INTEGERS REPRESENTS GPC ID TO COMMAND A DEU WHENEVER THE RESPECTIVE MAJOR FUNCTION IS SELECTED	830	830	A(3),INIT(0,1,1,0,2,2,0,3,3)	
I160	MEMORY_CONFIGURATION_NBR_FOR_GPC/STRING_CHANGE	CZ2V_MC_REQ (CZ2VCIN)	MEMORY CONFIGURATION NUMBER INPUT BY CREW FOR GPC/STRING DISPLAY ON DPS CM	850 860	CRT 485 860	1,INIT(0)	V91X1592X
I170	MAJOR_FUNCTION_PER_MEMORY_CONFIGURATION	CZ2V_MF_MC	MAJOR FUNCTION WHICH IS PERMITTED TO INITIATE GIVEN MEMORY CONFIGURATION	850 860	CRT	C(2),INIT( )	
I180	OPS_PER_MEMORY_CONFIGURATION	CZ2V_OPS_MC	OPS WHICH IS PERMITTED TO INITIATE GIVEN MEMORY CONFIGURATION	850 860	CRT	I,INIT(0)	V91X1594X
I190	GPC_PER_MEMORY_CONFIGURATION	CZ2B_GPC_MC	GPC SET SELECTED FROM GPC RECONFIGURATION TABLE (I2I0) BITS 1-5 REPRESENT PRESENCE (1) OR ABSENCE (0) OF GPC'S 1-5 RESPECTIVELY	850 860	CRT	BS(16),INIT(0)	
I200	STRING_PER_MEMORY_CONFIGURATION	CZ2B_STRNG_MC	GPC ID TO BE IN COMMAND OF STRINGS 1-5 UPON REQUEST OF A MEMORY CONFIGURATION. SUBJECT OF NOMINAL_STRING_ASSIGNMENT_TABLE (I110) BITS 1-13 NOT USED BITS 14-16 GPC ID	850 860	CRT	A(5),BS(16),INIT(0)	
I210	GPC_SET	CZ2B_GRT_GPC_SET	SETS OF GPC'S REQUIRED PER MEMORY RECONFIGURATION BITS 1-5 REPRESENT PRESENCE (1) OR ABSENCE (0) OF GPC'S 1-5 RESPECTIVELY	810 860	560 810 820 860	A(4),BS(16),INIT(8000,8000,8400,0,8400)	
I210.01	SIMPLEX_ONLY		BIT 6 INDICATES SIMPLEX ONLY FOR EACH MEMORY CONFIGURATION	810 850	810 850		
I220	FREEZE_DRY_MEMORY_	CZ2V_MC_FRZ	CREW INPUT SPECIFYING MEMORY		485	I,INIT(0)	

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
	CONFIGURATION		CONFIGURATION TO BE LOADED BUT NOT EXECUTED				
I230	FREEZE_DRY_GPC	CZ2V_GPC_FRZ	CREW INPUT SPECIFYING GPC TO BE LOADED			I,INIT(0)	
I240	CAM_COUNTERS	CZ2V_CAM_CNTRS (CZ2CNTGI)	EACH GPC'S COUNT OF VOTES AGAINST ALL OTHER GPC'S. FIVE GROUPS OF FIVE HALFWORDS EACH,WHERE A GROUP IS FOR A GPC (1-5) AND A HALFWORD IS THE GPC'S COUNT OF VOTES	375 820 880	485 660 665	A(5,5),I INIT(0)	
I250	PRIME_GPC_TIME_ MICROSECONDS	CZ2V_MET_MSEC (CZ2VMETM)	MET FROM PRIME GPC FOR SECONDARY INITIALIZATION	142 149 205		I,UM(MICROSECONDS)	
I260	PRIME_GPC_TIME_ HALF_HOURS	CZ2V_MET_HALFHRS (CZ2VMETH)	MET FROM PRIME GPC FOR SECONDARY INITIALIZATION	142 149 205		I,UM(30 MINUTES)	
I270	CURRENT_MF	CZ2V_DCU_MF	CURRENT MAJOR FUNCTION FOR EACH DFU	405	405 485 830	A(3),I	
I280	I/O_ERROR_LOG	CZ2V_IO_ERR_LOG (CZ2VIOER)	THE I/O ERROR LOG IS USED TO RECORD INFORMATION ABOUT I/O ERRORS	246	246 485 660 665	ST	
I280.1	GPC_SELF_I/O_ERROR_ LOG	CZ2B_IO_ERR TIOESTAK	THE CIRCULAR I/O ERROR LOG FOR GPC SELF, DESCRIBED BY THE FOLLOWING FIELDS	246	246 660 665 DL	A(5,5),BS(15)	
I280.2	I/O_ERROR_ PARAMETERS	TIOESTAK,TIOEPRMS, CZ2VIOER)	FIELD DESCRIBING PARAMETERS OF THE I/O OPERATION RECEIVING THE ERROR	246		BS(32)	V92M4322P V92M4339P V92M4400P V92M4417P V92M4477P V92M4494P V92M4550P V92M4567P V92M4620P V92M4637P
I280.3	I/O_ERROR_BCE_ NUMBER		TIOEPRMS BITS 1-5 CONTAIN THE BCE NUMBER OF THE I/O TRANSACTION IN ERROR	246		BS(5)	
I280.4	I/O_ERROR_BCE_ ELEMENT_NUMBER		TIOEPRMS BITS 6-10 CONTAIN THE BCE ELEMENT NUMBER OF THE I/O TRANSACTION IN ERROR	246		BS(5)	

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
I280.5	I/O_ERROR_DEVICE_ID		TIOEPRMS BITS 11-17 CONTAIN THE DEVICE ID OF I/O TRANSACTION	246		BS(7)	
I280.6	I/O_ERROR_OP_CODE		TIOEPRMS BITS 18-21 CONTAIN THE OP CODE OF THE TRANSACTION	246		BS(4)	
I280.7	I/O_ERROR_RESIDUAL_WORD_COUNT		TIOEPRMS BITS 22-32 CONTAIN THE RECUAL WORD COUNT	246		BS(11)	
I280.8	I/O_ERROR_IOP_STATUS_REGISTER	TIOESRGS	THE IOP STATUS REGISTER	246	DL	BS(32)	V92M4356P V92M4373P V92M4434P V92M4451P V92M4511P V92M4528P V92M4584P V92M4601P V92M4654P V92M4671P
I280.9	I/O_ERROR_TIME_TAG	TIOEGMTS	TIME OF ERROR	246	DL	BS(16),UM=(512 MICROSECOND UNITS)	V92W4390C V92W4468C V92W4545C V92W4618C V92W4698C
I280.10	I/O_ERROR_LOG_INDEX	CZ2B_IO_ERR_NDX(TIOEINDX, CZ2BERRX)	THE INDEX INDICATES THE LATEST ENTRY I/O ERROR LOG)	246	DL 246	BS(16),INIT(-1)	V92U4320C
I280.11	LATEST_I/O_ERROR_ENTRIES	CZ2B_IO_ERR_GPC (TIOELIST)	ERROR ENTRIES FOR THE LATEST I/O ERRORS FOR EACH GPC	246	485 660 665	A(5,6),BS(16)	
I280.12	GPC_X_I/O_ERROR_PARAMETERS	TIOEPRML	SAME AS TIOEPRMS LATEST ERROR OF GPC X	246	DL	BS(32)	V92M4700P V92M4717P V92M4777P V92M4794P V92M4850P V92M4867P V92M4927P V92M4948P NULL NULL
I280.13	GPC_X_I/O_ERROR_IOP_STATUS_REGISTER	TIOESRGL	IOP STATUS REGISTER	246	DL	BS(32)	V92M4734P VP2M4751P V92M4811P VP2M4828P V92M4884P VP2M4901P V92M4961P VP2M4988P

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
I280.14	GPC_X_I/O_ERROR_TIME_TAG	TIOEGMTL	TIME OF LATEST I/O ERROR OF GPCX	246	DL	BS(16),UM=(512 MICRO_SECOND UNITS)	V92W4768C V92W4845C V92W4918C V92W5005C
I280.15	GPC_X_I/O_ERROR_COUNT	TIOERCNT	CUMULATIVE COUNT OF I/O ERRORS FOR GPC X	246	DL 246	BS(16)	V92Q4770C V92Q4847C V92Q4920C V92Q5007C
I290	GPC_ERROR_LOG (CZ2VERLG)	CZ2V_GPC_ERR_LOG	THE LOG IS USED BY PROCESS ERROR MANAGEMENT TO RECORD INFORMATION ABOUT GPC ERRORS WHICH OCCUR DURING SYSTEM EXECUTION		183 485 660 665	ST	
I290.1	CIRCULAR_GPC_ERROR_LOG_INDEX	CZ2B_ERR_LOG (TLOGINDX)	INDEX INTO THE CIRCULAR ERROR LOG TABLE (TLOGNTRY) TO THE NEXT AVAILABLE ENTRY	183	183 DL 660 665	BS(16)	V92Q4250C
I290.2	GPC_SELF_CIRCULAR_GPC_ERROR_LOG	CZ2B_ERR_LOG(2) (TLOGNTRY)	CIRCULAR ERROR LOG TABLE WHICH CONTAINS INFORMATION ABOUT THE LATEST 5 ERRORS WHICH HAVE OCCURRED IN GPC SELF	183	DL 660 665	A(15),BS(16)	
I290.3	GPC_ERROR_TIME_TAG (TLOGSLFT)	TLOG_SLFT	TIME OF ERROR FOR THIS ENTRY	183	DL	BS(16),UM=(512 MICROSECOND UNITS)	V92W4256C V92W4264C V92W4270C V92W4276C V92W4282C
I290.4	GPC_ERROR_IDENTIFICATION	TLOGSLFC	FIELD IDENTIFYING ERROR GROUP CODE AND BSP FIELD TO QUALIFY TLOGSLFA	183	DL	BS(16)	V92U4252C V92U4258C V92U4266C V92U4272C V92U4278C
I290.5	GPC_ERROR_BSP		TLOGSLFC BITS 0-3 CONTAINS THE BSP FIELD WHICH QUALIFIES THE ERROR ADDRESS (TLOGSLFC)	183		BS(4)	
I290.6	GPC_ERROR_CODE		TLOGSLFC BITS 4-9 CONTAIN THE ERROR CODE OF THIS ENTRY	183		BS(6)	
I290.7	GPC_ERROR_GROUP		TLOSSLFC BITS 10-15 CONTAIN THE ERROR GROUP OF THIS ENTRY			BS(6)	
I290.8	GPC_ERROR_ADDRESS	(TLOGSLFA)	PSW ADDRESS AT THE TIME OF THIS ERROR OR OCT PSW ADDRESS IF CYCLE OVERRUN	183	DL	BS(16)	V92U4254C V92U4260C V92U4268C V92U4274C V92U4280C

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
I290.9	LATEST_GPC_ERROR_ENTRIES	CZ23_ERR_GPC (TLOGLSTE)	THE LATEST GPC ERROR ENTRIES FOR EACH GPC INDEXED BY GPC ID	183	485 660 665	A(5,4),BS(16)	
I290.10	GPC_X_ERROR_COUNT	(TLOGLSTN)	THE CUMULATIVE GPC ERROR COUNT FOR GPC X	183	DL	BS(16)	V92Q4288C V92Q4296C V92Q4306C V92Q4314C NULL
I290.11	GPC_X_ERROR_TIME_TAG	(TLOGLSTT)	THE TIME OF THE LAST ERROR FOR GPCX	183	DL	BS(16)	V92W4286C V92W4294C V92W4304C V92W4312C
I290.12	GPC_X_ERROR_IDENTIFICATION	(TLOGLSTC)	FIELD IDENTIFYING THE ERROR GROUP CODE AND BSR FIELD TO QUALIFY TLOGLSTA	183	DL	BS(16)	V92U4284C V92U4290C V92U4300C V92U4308C
I290.13	GPC_X_ERROR_ADDRESS	(TLOGSTA)	PSW ADDRESS AT THE TIME OF THE LAST ERROR FOR GPC X	183	DL	BS(16)	V92U4285C V92U4292C V92U4302C V92U4310C
I300	STATUS_FLAGS	CZ2B_STATUS_FLAG (CZ2BFLAG)	STATUS FLAGS WHICH ARE GPC UNIQUE & NOT REQUIRED AS COMMON INFORMATION BITS 4-32 NOT USED	495 620 700 820 830 910	325 620 830 910	BS(32)	
I300.01	PROTECT_INTERRUPT		INITIALIZED BY READ/WRITE SPEC AND RESET BY FCOS PROGRAM MODIFICATION	325 910	910		
I300.02	GSE_POLL_STATE		GSE POLLING ACTIVE IN THIS GPC BIT 2	495 820 830 870	830		
I300.03	CYCLIC_DISPLAY_INITIALIZE_FLAG		SIGNAL TO CYCLIC TO SELF INITIALIZE ON FIRST EXECUTION BIT 3	620 710	620		
I310	REDUNDANCY_MANAGED_DISCRETES	CZ2B_DIA_RM1(CZ2BDIA1 CZ2B_DIA_RM2	REDUNDANCY MANAGED DISCRETE REGISTER A. BITS 1-16. SAME. BITS 17-32.		800 800	BS(16),BS(16)	
I320	ICC_STATUS_FLAGS	CZ2B_ICC_FLAG	UPDATE STATUS OF SELECTED PARAMETERS WHICH MUST BE TRANSFERRED ACROSS ICC AT SIP TIME	351 355 405 560	142 149 170 375	A(3),BS(16)	

FCOS/CI COMMON COMPOOL (CZ2_COMMON)

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
				700	405		
				750	750		
				800			
				810			
				815			
				820			
				860			
				870			
				880			
				910			
				920			
				950			
I320.01	ICC_FIRST_PART_DST	(CZ2VIF1)	HALFWORD 1 BIT 1 INITIAL TRANSFER OF FIRST PART OF DST TO NEW GPC				
I320.02	ICC_LAST_PART_DST		BIT 2 INITIAL TRANSFER OF LAST PART OF DST TO NEW GPC				
I320.03	ICC_GST		BIT 3 TRANSFER OF ENTIRE GST TO OTHER GPC'S	920	920		
I320.04	ICC_DATA_PATH_MASKS		BIT 4 CHANGE IN: CZ2B_DP_DISP CZ2B_OK_GSE_DP	355			
I320.05	ICC_DUTY_CYCLE		BIT 5 CHANGE IN: CZ2V_DUTY_CYCLE		142		
I320.06	ICC_CURRENT_LOAD		BIT 6 CHANGE IN: CZ2V_CURRENT_OPS CZ2V_PROG_OVLY CZ2V_ME_OVLY CZ2V_MC	820 880	560 810		
I320.07	ICC_BUS_MASKS		BIT 7 CHANGE IN: CZ2B_ACT_XMITR1 CZ2B_ACT_XMITR2 CZ2B_ACT_RECVR CZ2B_BUS_MASK1 CZ2B_BUS_MASK2 CZ2B_DPM	355			
I320.08	ICC_CS/RS		BIT 8 CHANGE IN: CZ2B_CS CZ2B_RS	820 880			
I320.09	ICC_GST_STATUS		BIT 9 CHANGE IN: CZ2B_STATUS	375 560 820 870 880	142		

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
I320.10	ICC_MAJOR_FUNCTION_DISPLAY		BIT 10 CHANGE IN: CZ2V_MF_OPS CZ2V_OPS_OPS				
I320.11	ICC_DST_STATUS	FICCGPCP	BIT 11 CHANGE IN: CZ2B_OPS_STATUS CZ2V_CRT_D CZ2B_MM_MF	351 910 950			
I320.12	ICC_STRING_MODE		BIT 12 CHANGE IN: CZ2B_STRNG_DEF CZ2V_STRNG_ASSGN CZ2B_MODE	351			
I320.13	ICC_BTU_BITE		BIT 13 CHANGE IN: CZ2V_BTU_BITE	820 880			
I320.14	ICC_NOMINAL_STRING		BIT 14 CHANGE IN: CZ2B_NOM_STRNG	850	850		
I320.15	ICC_SYNC_FAIL		BIT 15 FLAG SIP TO CAUSE ICC MESSAGE TO BUS CONFIGURATION WHEN FAIL SYNC OCCURS				
I320.16	ICC_FORCE_O		BIT 16 FLAG SIP TO CAUSE ICC MESSAGE TO BUS CONFIGURATION WHEN FORCE OPS 0-00 OCCURS	815	815		
I320.17	ICC_CAM_GPC1	(CZ2VIF2)	HALFWORD 2 BIT 1 CHANGE IN: CZ2V CAM CNTRS FOR GPC1	820 880			
I320.18	ICC_CAM_GPC2		BIT 2 CHANGE IN: CZ2V_CAM_CNTRS FOR GPC2	820 880			
I320.19	ICC_CAM_GPC3		BIT 3 CHANGE IN: CZ2V_CAM_CNTRS FOR GPC3	820 880			
I320.20	ICC_CAM_GPC4		BIT 4 CHANGE IN: CZ2V_CAM_CNTRS FOR GPC4	820 880			
I320.21	ICC_CAM_GPC5		BIT 5 CHANGE IN: CZ2V_CAM_CNTRS FOR GPC5	820 880			
I320.22	ICC_IO_ERR_GPC1		BIT 6 CHANGE IN: CZ2B_IO_ERR_GPC FOR GPC1	246			
I320.23	ICC_IO_ERR_GPC2		BIT 7 CHANGE IN: CZ2B_IO_ERR_GPC FOR GPC2	246			

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
I320.24	ICC_IO_ERR_GPC3		BIT 8 CHANGE IN: CZ2R_IO_ERR_GPC FOR GPC3	246			
I320.25	ICC_IO_ERR_GPC4		BIT 9 CHANGE IN: CZ2R_IO_ERR_GPC FOR GPC4	246			
I320.26	ICC_IO_ERR_GPC5		BIT 10 CHANGE IN: CZ2R_IO_ERR_GPC FOR GPC5	246			
I320.27	ICC_GPC_ERR_GPC1		BIT 11 CHANGE IN: CZ2R_ERR_GPC FOR GPC1	183			
I320.28	ICC_GPC_ERR_GPC2		BIT 12 CHANGE IN: CZ2R_ERR_GPC FOR GPC2	183			
I320.29	ICC_GPC_ERR_GPC3		BIT 13 CHANGE IN: CZ2R_ERR_GPC FOR GPC3	183			
I320.30	ICC_GPC_ERR_GPC4		BIT 14 CHANGE IN: CZ2R_ERR_GPC FOR GPC4	183			
I320.31	ICC_GPC_ERR_GPC5		BIT 15 CHANGE IN: CZ2R_ERR_GPC FOR GPC5				
I320.32	ICC_LAST_MAJOR_FUNCTION		BIT 16 CHANGE IN: CZ2V_DEU_MF				
I320.33	ICC_COMPUTER_CRT_ASSIGNMENT_ETC	(CZ2VIF3)	HALFWORD 3 BIT 1 CHANGE IN: CZ2V_LOB_GPC_REQ CZ2V_LOB_BUS_REQ CZ2V_CCAT CZ2R_SET				
I320.34	ICC_GPC_RECONFIGURATION_ETC		BIT 2 CHANGE IN: CZ2V_MC_REQ CZ2V_ME_MC CZ2V_OPS_MC CZ2R_GPC_MC CZ2V_STRNG_MC CZ2R_GRT_GPC_SET	850	850		
I320.35	ICC_FREEZE		BIT 3 CHANGE IN: CZ2V_MC_FRZ CZ2V_GPT_FRZ BIT 4-6 NOT USED.				
I320.36	ICC_DOWNLIST_DEU1		BIT 7 CHANGE IN: CDMV_UI_DOWNLIST_FORMAT_ID FOR DEU1				
I320.37	ICC_DOWNLIST_DEU2		BIT 8 CHANGE IN: CDMV_UI_DOWNLIST_FORMAT_ID				

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
			FOR DEU2				
I320.38	ICC_DOWNLIST_DEU3		BIT 9 CHANGE IN: COMV_UI_DOWNLIST_FORMAT_ID FOR DEU3				
I320.39	ICC_TUTORIAL_DEJ1		BIT 10 CHANGE IN: COMV_UI_DOWNLIST_TUTORIAL FOR DEU1				
I320.40	ICC_TUTORIAL_DEJ2		BIT 11 CHANGE IN: COMV_UI_DOWNLIST_TUTORIAL FOR DEU2				
I320.41	ICC_TUTORIAL_DEU3		BIT 12 CHANGE IN: COMV_UI_DOWNLIST_TUTORIAL FOR DEU3 BIT 13-16 NOT USED.				
I330	FAIL_TO_SYNC_GPC	CZ2B_FAIL_SYNC_SET (CZ2BFSS)	MASK OF GPCS WHICH FAILED SYNC. BITS 1-11 NOT USED. BITS 12-15 REPRESENT GPCS 1-5. 1=GPC FAILED SYNC.	368	485 830 950	BS(16)	
I340	PRIME_AT_FAIL_TO_SYNC	CZ2V_PRIME_GPC (CZ2VPRIM)	PRIME GPC AT TIME OF SYNC FAILURE.		368	I	
I350	FORCE_OPS_0_ICC	CZ2B_ICC_FRC_0	ICC MESSAGE DATA FOR OPS 0 FORCE. HALFWORD 1 BITS 1-8 GPC ID OF SELF BITS 9-16 PRIME GPC ID HALFWORD 2 BITS 1-8 PS MASK BITS 9-16 OPS NUMBER FOR MAJOR FUNCTION 1 HALFWORD 3 BITS 1-9 OPS NUMBER FOR MAJOR FUNCTION 2 BITS 9-16 OPS NUMBER FOR MAJOR FUNCTION 3	815	485	A(3),BS(16)	
I360	DEU_SELF_TEST_EVENT	CZ2E_SELF_TEST	EVENT FOR CONTROLLING DEU LOADER EXECUTION.	700		E INIT (LATCHED)	
I370	LDB_ON	CZ2K_LDB_ON	DPS CM DISPLAY CHARACTERS			C(2) INIT(*)	
I380	LDB_OFF	CZ2K_LDB_OFF	DPS CM DISPLAY CHARACTERS			C(2) INIT(*)	
I390	DOWN_ARROW	CZ2K_DWN_ARROW	DPS CM DISPLAY CHARACTERS			C(2) INIT( )	

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
I400	VOTE_FAIL	CZ2K_VOTE_FAIL	DPS CM DISPLAY CHARACTERS			C(2) INIT (F)	
I410	MODE	CZ2K_MODE	DPS CM DISPLAY CHARACTERS			C(4) INIT(SPPN)	
I420	ICC_BUFFER_FREEZE_FLAG	CZ2V_ICC_BUF_FRZ (CZ2BFFZT)	GATE TO SHUT OFF SUBSEQUENT ICC BUFFER ENTRIES.	361 490	480 495	I	
I430	ICC_BUFFER_POINTER	CZ2V_ICC_BUF_PCINT	POINTER TO NEXT ITEM IN ICC BUFFER	480 485 490	480 485 750	I,INIT(1)	
I440	NEW_GPC_FLAG	CZ2B_NEW_GPC (CZ2BNEWG)	MASK FOR SPECIFYING NEW GPC IN CS. BITS 1-11 NOT USED. BITS 12-16 REPPESENT GPCS 1-5. 1=GPC DETECTED.	700 750	750	BS(16)	
I450	LAST_MAJOR_FUNCTION	CZ2V_MF_OLD	LAST MAJOR FUNCTION ACTIVE PRIOR TO CURRENT MAJOR FUNCTION.	405 560 700 790 830	405	A(3),I INIT(-1)	
I460	BITE_UPDATE_FLAG	CZ2V_BITE_UPD_VAL	FLAG TO CONTROL CYCLIC OR ONE SHOT BIU BITE READ. 1=CYCLIC 0=ONE SHOT	820 880	660 665 910	I INIT(0)	
I470	PROCEED_WITH_RECONFIGURATION	CZ2E_REC_PROCEED	EVENT TO CONTROL OPERATION OF GPC RECONFIGURATION	820 870 880	820 880	E INIT(FALSE)	
I480	RECONFIGURATION_MESSAGE_TYPE	CZ2V_REC_MSG_TYPE	ICC MESSAGE TYPE FOR GPC RECONFIGURATION. 1-RECONFIGURE WITH OLD GPCS. 2-RECONFIGURE WITH NONE OF OLD GPCS. 3-PROCEED NOW WITH RECONFIGURATION 4-OVERLAY COMPLETED. 6-SPECS CANCELED AND DISPLAY CHANGED.	870	880	I INIT(0)	
I490	RECONFIGURATION_TABLE_INDEX	CZ2V_REC_GRT_INDEX	INDEX INTO GPC_RECONFIGURATION_TABLE FOR REQUESTED MC.	560 870	560 810 820 880	I INIT(0)	
I500	RECONFIGURATION_DEU_SOURCE	CZ2V_REC_DEU	DEU NUMBER OF REQUESTING DEU FOR AN OPS.	560	560 820	I INIT(0)	
I510	RECONFIGURATION_MAJOR_FUNCTION	CZ2V_REC_MF (CZ2VRMFT)	MAJOR FUNCTION OF OPS REQUEST.	560	560 820	I INIT(0)	

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
					880		
I520	RECONFIGURATION_OPS_REQUEST	CZ2V_REC_OPS	OPS NUMBER OF REQUESTED OPS.	560 820	560 820 880	I INIT(0)	
I530	RECONFIGURATION_MODE_REQUEST	CZ2V_REC_MODE	MODE NUMBER OF REQUESTED OPS.	560 820	560 820	I,INIT(0)	
I540	RECONFIGURATION_STATUS	CZ2V_REC_XERR	SIGNAL STATUS IN RECONFIGURATION ATTEMPT. 0=W/O OVERLAY W/O ERROR 1=W/O OVERLAY WITH ERROR 100=WITH OVERLAY W/O ERROR 101=WITH OVERLAY WITH ERROR 3=IN STANDBY 103=COMPLETE WITH OVERLAY IN STANDBY.	560 810 820 880	560 810 820 880	I,INIT(0)	
I550	RECONFIGURATION_OVERLAY_STATUS	CZ2B_REC_OVL_STATUS	STATUS OF GPCS ATTEMPTING OVERLAY.	820	810	BS(16)	
I550.01	RECONFIGURATION_ERROR_MASK		BITS 1-5 GPCS WITH ERROR IN OVERLAY.	880			
I550.02	RECONFIGURATION_ATTEMPT_MASK		BITS 6-10 GPCS ATTEMPTING OVERLAY.	810	810	INIT(0)	
I550.03			BITS 11-16 NOT USED.				
I560	RECONFIGURATION_REDUNDANT_SET	CZ2B_REC_RS	MASK OF GPCS FOR RECONFIGURATION.	870	980	BS(16),INIT(0)	
I560.01	RS_SECONDARY_MASK		BITS 1-5 SECONDARY GPC SET.			BS(5)	
I560.02	RS_REQUIRED_MASK		BITS 6-10 REQUIRED GPC SET.			BS(5)	
I560.03	RS_CURRENT_MASK		BITS 11-15 CURRENT GPC SET.			BS(5)	
I570	LAST_SIP_TIME	CZ2V_REC_TSIP	TIME OF LAST SIP EXECUTION	870 880	870	SCD,UM(SECONDS)	
I580	OPS-TRANSITION-PARAMETERS	CZ2_RM-OTP	PARAMETERS TO TAKE EFFECT AT NEXT OPS TRANSITION	560		STI	
I580.01	SUM_WORD_DELTA	CZ2V_SUM_WD_DIFF CZ2VSUMW	NOMINAL DIFFERENCE ALLOWED BETWEEN SUM WORDS OF GPCS IN RS		375	BS(32) INIT(0)	
I580.02	SYNC_TIMEOUT_COUNTER	CZ2V_SYNC_TO_CNTR CZ2VSYNC	COUNT OF MAXIMUM NUMBER OF LOOPS THROUGH SYNC ALGORITHM WHICH DETERMINES THE TIME TO WAIT ON OTHER GPCS	205	363 364	ID INIT 4000	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
I580.03	CAM_NOGO_COUNTER	CZ2V_CAM_NGN_CNT CZ2VCAMN	COUNT OF MAXIMUM ALLOWED ERRORS BEFORE SETTING GPC FAILED DISCRETE	375	375	I INIT 3	
I590	UPDATE_OPS_TRANSITION_PARAMETERS	CZ2V_UPDATE_RM_OTP	UPDATED PARAMETERS TO TAKE EFFECT AT OPS TRANSITION	560		ST1	
I590.01	UPDATE_SUM_WORD_DELTA		SAME AS I580.1				
I590.02	UPDATE_SYNC_TO_COUNTER		SAME AS I580.02				
I590.03	UPDATE_CAM_NOGO_COUNTER		SAME AS I580.03				
I600	DEU_SELF_TEST_INTERFACE_DATA	CZ2V_READ_BUF	DATA INTERFACE BETWEEN DISPLAY CONTROL AND DEU SELF TEST	520 995	520 995	ST2	
I600.01	DISPLAY_BUFFER_AVAILABILITY_INDICATOR	CZ2V_RBUF_DISP	INDICATOR USED TO REFLECT AVAILABILITY OF DISPLAY BUFFERS. SET ZERO INDICATES BUFFER AVAILABLE AND SET NON-ZERO INDICATES BUFFER IN USE	520 560 995	520 560 995	I	
I600.02	DISPLAY_BUFFER_ALLOCATION_INDICATOR	CZ2V_RBUF_P	INDICATOR USED TO REFLECT ANY PENDING READ REQUESTS FOR THE DISPLAY BUFFER. SET ZERO INDICATES NO PENDING READ REQUESTS	520 560 995	520 560 995	I	
I510	ICC_BUFFER	CZ2V_ICC_BUF CZ2VTCC1,CZ2VICC2, CZ2VICC3,CZ2VICC4	BUFFER FOR MESSAGES SENT AND RECEIVED OVER ICC	490 750 800 830	480 485 490 495 650 665 680 700 710 750 820 870 955	ST4	
I610.01	SUM_WORD	CZ2V_SUM_WD TICCRMSW	RM SUM WORD FOR GPC	750	375	ID, INITO	
I610.02	DISCRETE_IN_REGISTER_A	CZ2B_DIA1, TICCDIA1, CZ2B_ DIA2,TICCDIA2	FRESH INPUT OF DISCRETE INPUT REGISTER A	800	560 660 665 700 710 820		

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	OST	ATTRIBUTES	MML
					830		
I610.06	ICC_PRIME_TIME			205	142 140		
I610.10	INTERNAL_TIME_ MICROSECONDS	CZ2B_GPC_MT TICCGMTH	MICROSECONDS OF GMT UP TO 1 HALF HOUR FOR GPC	149	142 660 665 955	ID INIT 0 UM MICROSEC	
I610.14	INTERNAL_TIME_HALF_ HOURS	CZ2B_GPC_MT2 TICCGMTM	HALFHOURS OF GMT FOR GPC	149	142 149 660 665	I, INIT 0 UM 30MINUTES	
I610.18	ICC_MESSAGES	CZ2B_ICC_MSG_BUF TICCBUFF	UP TO 120 HALFWORDS OF ICC MESSAGES	480 485 490	490 495 680 830 870	A 121,BS 16	
I620	GPC RECONFIGURATION	CZ2V_GRT_TAB	GPC RECONFIGURATION TABLE		800 820 860 880	ST(4)	
I620.01	MEMORY CONFIGURATION	CZ2V_GRT_MC	MEMORY CONFIGURATION NUMBER		820 860 880	I INIT 1,2,3,4	
I620.02	PROGRAM_OVERLAY	CZ2V_GRT_POVL	PROGRAM OVERLAY NUMBER		820 880	I INIT 4,5,7,7	
I620.03	MAJOR FUNCTION_ OVERLAY	CZ2V_GRT_MFOVL	MAJOR FUNCTION OVERLAY NUMBER		820 880	I INIT 3,3,6,6	
I620.04	MAJOR_FUNCTION	CZ2V_GRT_MF_VALUE	MAJOR FUNCTION CODE 1=PAYLOAD 2=GNC 3=SM		560 800 850	A(2),I INIT(2,2,3, 3,3,0,0)	
I620.05	OPS_NUMBER	CZ2V_GRT_OPS_NBR	OPS NUMBER TO REQUEST OVERLAY		560 850	I, INIT 2,3,8,8	
I620.06	MAXIMUM_NUMBER_ MODES	CZ2V_GRT_MAX_MODE	MAXIMUM NUMBER OF MODES IN OPS		560	I,INIT 1,5,1,1	
I630	OPS_ITEM	CZ2B_OPS_ACK_ITEMS	RECORD OF ITEM INPUT ENTERED ON OPS CM DISPLAY	800		BS(16) INIT(0)	
I640	ICC_MESSAGE_TABLE	CZ2V_ICC_MSG_TAB	ICC MESSAGE TABLE USED TO IDENTIFY MESSAGES TRANSFERRED ACROSS ICC.BITS		485 490	A(28)BS(16),INI T(0,0,3100,3100 ,3100,3100,3100	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
			1-4 ROUTING CODE: 3,14=CALL,15=MOVE BITS 5-8 PROCESSOR ID 1=APG,2=SIP,3= ADD,4=FIOP M BITS 9-15 NAME INDEX BIT 16 CONTENTION FLAG			,0,0,3200 3200,3200,0,0,0 ,0,0,5400,3100, 3300,3300,3300, 3300 3300, F300, F300, 3300, F000	
I650	ICC_ADDRESS_TABLE	CZ2V_LNAMES	ICC ADDRESS TABLE TO PERMIT HAL VARIABLE NAMES TO BE MANIPULATED AS DATA	485	485 490	ST70	
I650.01	ICC_ADDRESS_NAME	CZ2V_ICC_NM	HAL VARIABLE NAME DEFINED AT EXECUTION TIME	485	485 490	A2,BS(16)	
I660	OLD DISCRETES			700 800	800	BS(5) INIT'00011'	
I670	LIGHT_AND_ALARMS_COUNTER	CZ2V_LAP_CNT	LIGHT AND ALARMS PROCESS FREQUENCY COUNTER	700 750	750	I,INIT 1	
I680	MCDS_INPUT_BUSY	CZ2V_MCDS_IN	FLAG INDICATING BUSY STATE OF DMI MCDS_IN 400.	400	400 500 505 560 820 880	I, INIT 0	
I690	DEU_SHARED_EVENT	CZ2E_SHRD_EVT	EVENT SET BY SEVERAL DIFFERENT PROCESSES TO SET OR RESET DEU_POLL FLAG	400 560 790 800 820 830 880 980 995	400	E	
I700	LDB_EQUIVALENT_DEU_MESSAGE_EVENT	CZ2E_LDB_EQ_EVT	EVENT SET BY LDB_MESSAGE_ROUTER TO REQUEST AN EQUIVALENT DEU MESSAGE BE PROCESSED	400 440	400	E	
I710	KEYBOARD_MESSAGE_PROCESSED_EVENT	CZ2E_KYBD_PROC_EVT	EVENT SET BY MCDS FUNCTIONS PROCESSING TO SET DEU_POLL FLAG WHEN A KEYBOARD MESSAGE HAS BEEN PROCESSED	400 500	400	E	
I720	LDB_EQUIVALENT_DEU_INPUT_TABLE	CZ2V_DIT	STORAGE AREA FOR LDB EQUIVALENT DEU MESSAGE IN SAME FORMAT AS DEU INPUT TABLE	440	400	ST(3)	
I720.01	EQUIVALENT_STATUS	CZ2V_DIT_STAT	EQUIVALENT DEU INPUT STATUS		400	A(4),I	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
I720.02	EQUIVALENT_HEADER	CZ2B_DIT_HDR	EQUIVALENT DEU MESSAGE HEADER SEE DESCRIPTION FOR I060.02	440	400	INIT(0) BS(16)	
I720.03	EQUIVALENT DEU_ NUMBER OF KEYS	CZ2V_DIT_NKEYS	NUMBER OF KEYSTROKES IN DEU EQUIVALENT MESSAGE	440	400	I	
I720.04	EQUIVALENT_DEU_ KEYBOARD_MESSAGE	CZ2V_DIT_KYBD_MSG	KEYSTROKE CODES MAKING UP THE EQUIVALENT DEU MESSAGE SEE DESCRIPTION FOR J060.04	440	400	A(40),T	
I730	EQUIVALENT_DEU_ NUMBER	CZ2V_EQ_DEUNO	DEU FOR WHICH THE LOB EQUIVALENT MESSAGE IS INTENDED	440	400	I	
I730	CASE_NUMBER	CZ2V_CASE	CASE NUMBER FOR THE SHARED EVENT WHERE: 1=RN TO SB OR SB TO RN 2=TURN DEU_POLL_FLAGS ON 3=TURN DEU_POLL_FLAGS OFF 4=FORCE MAJOR FUNCTION CHANGE 5=TURN DEU_POLL_FLAG OFF 6=TURN DEU_POLL_FLAGS ON		400		
I740	DEU_PROCESSED	CZ2B_DEU_PROC	BITS CORRESPONDING TO DEU'S THAT HAVE BEEN PROCESSED: BIT 1=DEU 1 PROCESSED BIT 2=DEU 2 PROCESSED BIT 3=DEU 3 PROCESSED BITS4-16= NOT USED	505	400	BS(16)	
I750	DEU_NUMBER	CZ2V_DEUNUM	DEU INVOLVED IN BEING LOADED OR IN SELF TEST	790 990 995	400	I	
I760	MAJOR_FUNCTIONS_ CHANGE_DEU'S	CZ2B_MF_CHNG	BITS CORRESPONDING TO DEU'S FOR MAJOR FUNCTION CHANGES ARE TO BE FORCED: BIT 1= DEU 1,FORCE MF CHANGE BIT 2= DEU 2,FORCE MF CHANGE BIT 3= DEU 3,FORCE MF CHANGE BITS 4-16 = NOT USED	400 830	400	BS(16)	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
W010	GPC_ADDRESS_ID STATUS	COJB_R_W_L	THIS ARRAY REFLECTS THE STATUS FOR EACH ADDRESS ID ENTERED ON THE MEMORY READ/WRITE SPEC.	910 915	910 915	A(6),BS(16)	
W010.1	ADD_ID_BLANKING_ BIT		BIT 1 STATUS BLANKING INDICATOR. WHEN ON CORRESPONDING ITEM IS BLANKED OUT BIT 2-6 NOT USED	910 915	910 915		
W010.2	ADD_ID_VALIDITY_ BIT		BIT 7 DATA VALIDITY BIT-WHEN ON DATA FOR ITEM IS ACCEPTABLE BIT 8-16 NOT USED	910	910 915		
W020	DESIRED_DATA_ STATUS	COJB_R_W_D	THIS ARRAY REFLECTS THE STATUS FOR EACH DESIRED VALUE ENTERED ON THE MEMORY READ/WRITE SPEC.	910	910	A(6),BS(16)	
W020.1	DESIRED_DATA_ BLANKING_BIT		BIT 1 STATUS BLANKING INDICATOR. WHEN ON CORRESPONDING ITEM IS BLACKED OUT BIT 2-6 NOT USED				
W020.2	DESIRED_DATA_ VALIDITY_BIT		BIT 7 DATA VALIDITY BIT-WHEN ON DATA FOR ITEM IS ACCEPTABLE BIT 8-16 NOT USED	910	910		
W030	CURRENT_DATA_ STATUS	COJB_R_W_C	THIS ARRAY REFLECTS THE STATUS FOR EACH CURRENT DATA VALUE DISPLAYED ON THE MEMORY READ/WRITE SPEC	910 915	910 915	A(6),BS(16)	
W030.1	CURRENT_DATA_ BLANKING_BIT		BIT 1 STATUS BLANKING INDICATOR- WHEN ON CORRESPONDING ITEM IS BLANKED OUT BIT 2-6 NOT USED	910 915	910 915		
W030.2	CURRENT_DATA_ VALIDITY_BIT		BIT 7 DATA VALIDITY BIT-WHEN ON DATA FOR ITEM IS ACCEPTABLE BIT 8-16 NOT USED				
W040	GPC_ADDRESS_ID_ SAVE_AREA	COJB_R_W_LOCID	THIS ARRAY IS THE SAVE AREA INTO WHICH THE REQUESTED GPC ADDRESS IDS ARE STORED FOR DISPLAY PURPOSES	910 915	910 915	A(6),BS(32)	
W050	DESIRED_DATA_SAVE_ AREA	COJB_R_W_DSRDO	THIS ARRAY IS THE SAVE AREA INTO WHICH THE REQUESTED DESIRED DATA IS STORED FOR DISPLAY PURPOSES	910	910	A(6),BS(32)	
W060	CURRENT_DATA_SAVE_ AREA	COJB_R_W_CURRO	THIS ARRAY IS THE SAVE AREA INTO WHICH THE CURRENT DATA IS STORED FOR DISPLAY PURPOSES	910 915	910 915	A(6),BS(32)	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
W070	ITEM_EXECUTE_STATUS	CDJB_R_W_EXC_STAT	THIS WORD INDICATES THE STATUS OF THE VARIOUS PARTS OF THE DISPLAY CONTROLLED BY ITEM EXEC ENTRIES	910	910	BS(16)	
W070.1	DATA_MODE_SELECT		BIT 1 WHEN ON INDICATES THE DATA MODE IS ACTIVE	910	910		
W070.2	CODE_MODE_SELECT		BIT 2 WHEN ON INDICATES THE CODE MODE IS ACTIVE BIT 3 NOT USED	910	910		
W070.3	LOC_ID_PRESENT_IND		BIT 4 DATA ENTERED FOR LOC ID BIT 5 DATA ENTERED FOR DESIRED VALUE	910	910		
W070.4	DESIRED_VALUE_PRESENT_IND		BIT 6&7 NOT USED BITS 10-16 NOT USED	910	910		
W070.5	DWNLST_ADD_ID_FLG		BIT 8 WHEN ON INDICATES DOWNLIST ADDRESS INPUT				
W070.6	DWNLST_WRD_CNT_FLG		BIT 9 WHEN ON INDICATES WORD COUNT HAS BEEN INPUT				
W080	DATA_READ_OUT_AREA	CDJV_DATA_AT_LOC ID	THIS VARIABLE IS THE READ OUT AREA USED BY THE PROGRAM MODIFICATION MACRO WHICH DOES THE MEMORY READ	910 915	910 915	DSC	
W090	READ_WRITE_END_OF_LOOP_INDICATOR	CDJV_FLAG1	THIS INDICATOR IS SET WHENEVER A "DO FOR X UNTIL Y IS ON" TYPE LOOP IS EXECUTED TO REDUCE THE NUMBER OF TIMES THE LOOP IS EXECUTED	910	910	I	
W100	DATA_WRITE_OUT_AREA	CDJV_INIT_SGL	THIS VARIABLE IS THE WRITE OUT AREA USED BY THE PROGRAM MODIFICATION MACRO WHICH DOES THE MEMORY WRITE	910	910	I	

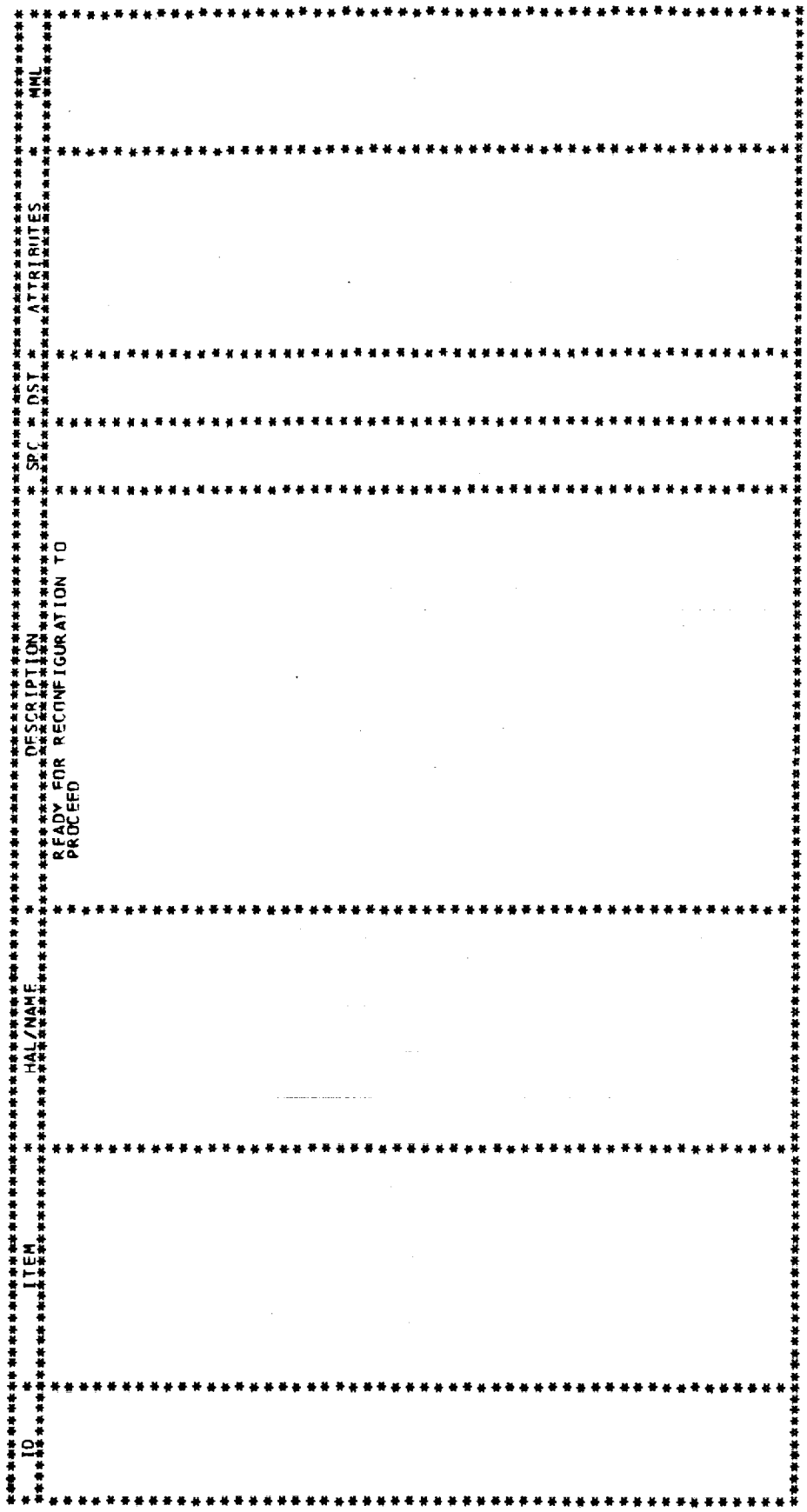
ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
X001	DEU_LOAD_ERROR_FLAG	AIGB_IO_ERR_FLG	FLAG USED TO INDICATE ANY ERROR CONDITIONS DURING LOAD OF DEU.	790	790	B0	
X002	ARB_A_ARRAY_MEMBER	ARB_A	SAVE AREA FOR CURRENT GPC ID	810	810	I	
X003	ARB_I	ARB_I	INTEGER USED FOR LOOP COUNTER.	810	810	I	
X004	ARB_OVERLAY_COUNTER	ARB_OVL_CNT	COUNTER FOR GPCS IN OVERLAY	810	810	I	
X005	ARB_ERROR_COUNTER	ARB_ERR_CNT	COUNTER FOR GPCS THAT HAD ERRORS WHILE DOING THE OVERLAY.	810	810	I	
X006	AREV_GRT_OPS_COPY	AREV_GRT_OPS_STRUCT	INDEX VALUE BASED UPON MEMORY CONFIGURATION NUMBER AND IS USED TO RETRIEVE DATA PERTINENT FOR THAT MEMORY CONFIGURATION FROM GPC RECONFIGURATION TABLE.	850	850	I	
X007	AREV_MEM_CONFIG_NO	AREV_MC_NUM	MEMORY CONFIGURATION NUMBER	850	850	I	
X008	AREB_MEM_CONFIG_FOUND	AREB_MC_FOUND	FLAG USED TO INDICATE REQUESTED MEMORY CONFIGURATION VALID.	850	850	BS(1)	
X009	AREV_ITEM_CNTR	AREV_ITEM_COUNTER	INTEGER USED FOR ITEM COUNTER.	850	850	I	
X010	AREV_LOOP	AREV_LOOP	INTEGER USED FOR LOOP COUNTER.	850	850	I	
X011	AREB_ITEM_NUM	AREB_ITEM_NUM	INTEGER USED TO INDICATE WHICH ITEM IS BEING PROCESSED.	850	850	BS(1)	
X012	AREB_ITEM_ERROR	AREB_ITEM_ERROR	FLAG USED TO INDICATE ERROR CONDITION.	850	850	BS(1)	
X013	AREV_ODD_NUM_TAB	AREV_ITEM_NUM_ODD	ARRAY OF ITEM NUMBERS USED FOR GPC DESELECT.	850	850	A(4),I	
X014	AREV_STRNG_NUM	AREV_ITEM_STRG_NUM	ARRAY OF ITEM NUMBERS USED FOR STRING ASSIGNMENTS.	850	850	A(5),I	
X015	ARF_LOOP1_FLAG	ARF_FLAG1	INDICATION OF WHEN END OF DO LOOP REACHED.	860	860	BS(16)	
X016	ARF_CASE_NO	ARF_K	VARIABLE INDICATING CASE NUMBER REQUIRED.	860	860	I	
X017	ARF_DO_CASE_INDEX	ARF_DO_CASE_INDEX	ARRAY USED TO DETERMINE THE CASE NUMBER REQUIRED.	860	860	A(22),I	
X018	ARF_I	ARF_I	DO LOOP COUNTER VARIABLE	860	860	I	
X020	ARF_ICC_MSG_TYPE	ARF_VT_ICC_MSG	VARIABLE USED TO PASS DATA TO THE	860	495	A(2),BS(16)	

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
			ICC COLLECTOR.				
X021	ARF_C	ARF_C	DO LOOP COUNTER VARIABLE	860	860	I	
X022	ASB_LINE_NO	ASB_I	INDICATION OF LINE NUMBER BASED ON ITEM INPUT.	910	910	I	
X023	ASB_CASE_NO	ASB_K	INDICATION OF CASE NO. REQUIRED FOR PROCESSING.	910	910	I	
X024	ASB_ITEM_O	ASB_ITEM_O	SAVE AREA FOR ITEM INPUTS.	910	910	BS(32)	
X025	ASB_MEM_INDEX	ASB_MEM_INDEX	INTEGER REPRESENTATION OF ITEM INPUTS.	910	910	A(6),I	
X026	ASB_GSE_ICC_MSG	ASB_GSE_ICC_MSG	VARIABLE USED TO PASS DATA TO THE ICC COLLECTOR.	910	495 910	A(2),BS(16)	
X027	ASB_LOOP_CNT	ASB_J	DO LOOP COUNTER	910	910	I	
X028	ASB_LINE_CNT	ASB_K	DO LOOP COUNTER	910	910	I	
X029	ASB_MEM_LOC	ASB_A	RESULT OF CURRENT MEMORY LOC. REQUEST MINUS THE MAX GPC ADDRESS	910	910	I	
X030	ASB_MEM_MAX	ASB_MEM_MAX	CONSTANT CONTAINING MAX. GPC ADDRESS.	910	910	I	
X031	ASB_LOC_THIS_LINE	ASB_ITEM_O	GPC LOCATION ID FOR CURRENT LINE	910	910	BS(32)	
X032	ASH_D	ASH_D	DO LOOP COUNTER	915	915	I	
X033	ASH_READ_FLAG	ASH_READ FC.DFLGSVC	PARAMETER WITHIN PARAMETER LIST USED TO INVOKE THE PGMMOD SVC INDICATING READ REQUEST.	915	915	I	
X034	ASH_READ_STRT	ASH_READ FC.DPMSTRT	PARAMETER WITHIN PARAMETER LIST USED TO INVOKE THE PGMMOD SVC INDICATING THE GPC ADDRESS.	915	915	I	
X035	ASH_WORD1_DATA	ASH_FW1_DATA	AREA INTO WHICH DATA READ IS STORED	915	915	BS(32)	
X036	COINCIDENT_TOTAL	ASJVL_CTIME	COINCIDENTAL TOTAL TIME IN SECONDS	975	975	DSC INIT(0)	
X037	CURRENT_MET	ASJVL_MET_TIME	CURRENT MET OBTAINED FROM MET_SVC SVC	975	975	DSC INIT(0)	
X038	COINCIDENT_TIME_IN_MINUTES	ASJVL_CO_TIME	COINCIDENT TIME IN MINUTES FOR MTU UPDATE	975	975	DI INIT(0)	
X039	TIME_SOURCE_DATA	ASCBL_SOURCE	BIT PATTERNS FOR DISPLAYING TIME SOURCE SELECTED:		950	A(4),BS(3)	

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
			001=A1 ACCUMULATOR 1 SELECTED 010=A2 ACCUMULATOR 2 SELECTED 011=A3 ACCUMULATOR 3 SELECTED 100=PT GPC PRIME TIME SELECTED				
X040	ERROR_MESSAGE_FLAG	ASCBL_FLAG	FLAG USED TO INDICATE ERROR CONDITION FOUND AND ERROR MESSAGE ISSUED.	950	950	BO INIT (FALSE)	
X041	INPUT_DEU	ARDVL_CRT	INPUT DEU ON CMPTR/CRT MESSAGE OR DEU RECEIVING MAJOR FUNCTION CHANGE	830	830	I	
X042	INPUT_MF	ARDVL_MF	NEW MF SETTING OF DEU	830	830	I	
X043	COMMANDER_SET	ARDVL_CMDR_SET	REDUNDANT SET OF NEW BUS COMMANDER OR NEW BUS COMMANDER ALONE IF SIMPLEX	830	830	BS(5)	
X044	ASSIGNED_OPS_0_GPC	ARDVL_CRT_FLG	GPC ID OF GPC IN OPS 0 ASSIGNED DEU CONTROL VIA CMPTR/CRT KEY	830	830	A(3),I,INIT(0)	
X045	NEW_BUS_COMMANDER	ARDVL_NEW_CMDR	GPC ID OF GPC TO BE ASSIGNED COMMAND OF THE SPECIFIED BUS	830	830	I	
X046	RUN_GPC	ARDVL_RUN	GPC ID OF GPC IN RUN MODE SUPPORTING THE NEW MF OF A DEU	830	830	I	
X047	STANDBY_GPC	ARDVL_STBY	GPC ID OF GPC IN STANDBY MODE SUPPORTING THE NEW MF OF A DEU	830	830	I	
X048	ZERO_GPC	ARDVL_ZERO	ID OF LOWEST GPC IN OPS 0 FOR MF PROCESSING	830	830	I	
X049	MF_FLAG	ARDVL_MF_FLG	FLAG FOR BUS CONFIGURATION TO INDICATE MF DEU ASSIGNMENT	830	830	BO,INIT(FALSE)	
X050	HANDOVER_STATUS	STAT	STATUS INDICATOR FOR BUS HANDOVER COOPERATION	830	830	BO	
X051	INPUT_CMPTR	ARDVL_CMPTR	INPUT CMPTR CODE FROM CMPTR/BUS KEY	830	830	I	
X052	BUS_NUMBER	ARDVL_BUS	INPUT BUS CODE FROM CMPTR/BUS KEY AND HARDWARE BUS NUMBER FOR BUS ASSIGNMENTS.	830	830	I	
X053	MESSAGE_DEU	ARDVL_DEU_NO	DEU USED TO INPUT CMPTR/CRT OR CMPTR/BUS REQUESTS	830	830	I	
X054	BUS_PARMS_VARIABLE_DATA_1	BUS_PARMS.VAR_DATA 1	PARAMETER LIST VARIABLE FOR RECONFIG SVC	830	350	I	

ID	ITEM	HAL/NAME	DESCRIPTION	SPC	DST	ATTRIBUTES	MML
X055	CURRENT_BUS_COMMANDER	ARDVL_CURR_CMDR	GPC CURRENTLY COMMANDING THE SPECIFIED BUS	830	830	I	
X056	BUS_NUMBER_FOR_HANDOVER	ARDVL_B_NO	BUS INDICATOR FOR BUS HANDOVER COOPERATION	830	830	I	
X057	BUS_REQUEST_TYPE	BUS_PARMS.TYPE_IND	PARAMETER LIST REQUEST TYPE FOR RECONFIG SVC	830	350	BS(8)	
X058	BUS_PARMS_VARIABLE_DATA_2	BUS_PARMS.VAR_DATA 2	PARAMETER LIST VARIABLE FOR RECONFIG SVC	830	350	I	
X059	INPUT_OVERLAY_NUMBER	ARDVL_MEM	INPUT MEMORY OVERLAY NUMBER FOR OPS INITIATION REQUESTS	830	830	I	
X060	FAIL_SYNC_SET	ARDVL_FAIL_SET	MASK INDICATING GPC'S THAT HAVE FAILED TO SYNC	830	830	BS(5),INIT(0)	
X061	NEW_0_GPC	ARDVL_GPC	GPC THAT FAILED TO SYNC OR WAS FORCED TO OPS 0	830	830	I	
X062	LOWEST_0_GPC	N	LOOP VARIABLE TO DETERMINE LOWEST ID GPC IN OPS 0	830	830	I	
X063	FC_STRING	STRING	LOOP VARIABLE USED TO ASSIGN FLIGHT CRITICAL STRINGS	830	830	I	
X064	BUS_MASK_VARIABLE_DATA	BUS_MASK_PARMS.VAR_DATA	PARAMETER LIST VARIABLE DATA FOR DG_MSK_MGMT SVC	830	355	I	
X065	ICC_BUFFER_NUMBER	B	ICC BUFFER INDEX PASSED IN CALLING SEQUENCE TO BUS_CONFIGURATION_CHANGE	490	830	I	
X066	ICC_ARRAY_INDEX	I	ARRAY INDEX INTO ICC BUFFER PASSED IN CALLING SEQUENCE TO BUS_CONFIGURATION_CHANGE	490	830	I	
X067	ICC_MSG_BUFFER	AIEB_ICC_MSG	LOCAL BUFFER TO GENERATE ICC MESSAGES	750	750	A(2)BS(16)	
X068	LOAD_STATUS	AIEB_STAT	ICC BUFFER LOAD STATUS	750	750	BS(16)	
X069	ICC_MSG_CNT	AIEV_MSG_CNT	COUNT OF MSG.SENT THIS SIP CYCLE	750	750	I INIT(0)	
X070	TEMP_TEST_FLAGS	AIEB_TST	TEMPORARY WORD FOR TESTING PACKED MESSAGE REQUEST FLAGS	750	750	BS(8)	
X071	HALF_INDEX	I	LOOP VARIABLE TO SELECT HALF OF PACKED REQUEST FLAGS	750	750	I	
X072	WORD_INDEX	J	LOOP INDEX TO SELECT PACKED WORD ENTRY	750	750	I	

ID	ITEM	HAL/NAME	DESCRIPTION	SRC	DST	ATTRIBUTES	MML
X073	BIT_INDEX	K	LOOP VARIABLE TO SELECT BIT POSITION OF PACKED REQUEST FLAGS	750	750	I	
X074	ICC_BUF_INDEX	INDX	ICC BUFFER INDEX BASED ON GPC ID	750	750	I	
X075	MTU_RM_SVC	AIEV_MTU_SN	SVC PARAMETER LIST FOR MTU REDUNDANCY MANAGEMENT	750	100	ST(1)	
X076	ICC_REQUEST	AIEV_ICC_SN	SVC PARAMETER LIST FOR ICC I/O	750	100	ST(1)	
X076.15	ICC_REQUEST_WD_COUNT	AIEV_ICC_SN.DWDCT	ICC REQUEST WORD COUNT	750	100	I	
X076.16	ICC_REQUEST_BUFFER	AIEV_ICC_SN.DLOC	ICC REQUEST BUFFER LOCATION	750	100	NA BS(16)	
X077	ICC_MSG_HEADER	AIEV_ICC_HDR	ICC MESSAGE HEADER TABLE	750	100 750	A(44)BS(16)	
X078	BUFFER_AVAILABLE_STATUS	AOPV_STAT	STATUS INDICATING IF A DEU_INPUT_BUFFER HAS BEEN FOUND AVAILABLE =0,NOT AVAILABLE =1,AVAILABLE	815	815	I	
X079	DEU_NUMBER	AOPV_DEU	LOCAL INDEX USED IN DO LOOP FOR TESTING DEU_INPUT_BUFFER	815	815	I	
X080	OPS_0_MESSAGE	AOPK_MSG_HDR	CONSTANT REPRESENTING A HEADER WORD FOR A KEYBOARD MESSAGE	815	815	BS(16)	
X081	OPS_0_KEYBOARD_MSG	AOPK_KYBD_MSG	CONSTANT ARRAY CONTAING THE KEYSTROKES FOR OPS 000 PRO	815	815	A(5),I	
X082	SECONDARY/REQUIRED/CURRENT_RS_MASK	ARC_RS_MASK	TEMPORARY SET OF MASKS OF 3 STATES INVOLVED IN RECONFIGURATION (SECONDARY,REQUIRED,CURRENT)	820	820	BS(16)	
X083	OVERLAY_COMPLETE_EVENT	ARC_OVL_EVT	EVENT SET BY FCOS WHEN OVERLAY HAS BEEN COMPLETED	820	820	E	
X084	ICC_MESSAGE	ARC_ICC_MSG123	PARAMETERS SET TO IDENTIFY ICC MESSAGE TYPE AND DATA	820		A(7),BS(16)	
X085	MESSAGE_TYPE	ARC_MSG_TYPE	RECONFIGURATION MESSAGE TYPES: 1.SOME(ALL) MEMBERS OF ORIGINAL SET ARE MEMBERS OF NEW SET. 2.NONE OF ORIGINAL SET ARE MEMBERS OF NEW SET 3.PROCEED WITH PROCESSING 4.OVERLAY IN SECONDARY GPC'S IS COMPLETE 5.NOT APPLICABLE 6.SEQUENCE_REQUEST_PROCESSOR IS	870	870	I	


ITEM
ID
HAL/NAME
DESCRIPTION
SPC
DST
ATTRIBUTES
MNL
READY FOR RECONFIGURATION TO
PROCEED
SYSTEM CONTROL LOCAL DATA

APPENDIX F

OPS DEPENDENT MODULES

Module Id.	English_Equivalent_Name	HAL Name	OPS 0	OPS 1	OPS 2
790	DEU_Loader	AIG_DEU_LOADER		X	X
950	Time_Management_Specialist_ Function	ASC_TIME_MGMT		X	X
955	Time_Management_Display_ Update_Cyclic_Processor	ASG_CYCLIC_UPDATE		X	X
975	Time_Management_MTU_ Update_Processor	ASJ_MTU_UPDATE		X	X

APPENDIX G

The following subsections of this appendix illustrate process sequencing and relationship for several operational system scenarios. Specific application programming relating to these scenarios is summarized or overlooked so that application changes or additions should not affect scenarios accuracy.

The first scenarios (timeline) illustrates the sequence of processing for MCDS keyboard entries from their introduction to the GPC until a new display has been transferred to the MCDS. The flight software has been loaded, initialized and is executing an Operations Sequence (OPS) within a Major Function.

The keyboard message is [SPEC] xxx [PRO] and [ITEM] x [EXEC].

The processing takes place in the following environment.

- User Interface Control Supervisor which was scheduled at system initialization is now in the WAIT state until an MCDS input is received, or an application control segment requests a service, or a load from Mass Memory is completed.

- Cyclic Display Processing has been scheduled at system initialization and is (as the name implies) cyclically generating Format Control Word Buffers for the MCDS.

The second timeline illustrates activity on the SYNC discrete lines between two redundant set GPCS during normal application processing. The SYNC code values are given in mneumonic symbology rather than binary, and all times are given in microseconds. Three minor cycles are illustrated.

The following abbreviations are used in these timelines.

DEU - Display Electronics Unit

FCOS - Flight Computer Operating System

UI - User Interface

APP - Applications Software (Control Segment)

SC - System Control

SIP - System Interface Processor. Also, a SYNC Discrete value.

RUN - Literally "run". (A SYNC discrete value of all-ones.)

IOC - I/O Complete (A SYNC discrete value)

SVC - Supervisor Call (A SYNC discrete value)

TIMER- Literally "Timer". (A SYNC discrete value.)

CS - Control Segment (application code)

G.1 [SPEC] xxx [PRO] Timeline

#	Software Processor	Subsystem	Action
1.			Crew inputs "SPEC xxx PRO"
2.	DEU Control Program	DEU	buffers the message and sets cue to respond with MSG READY at next poll from GPC
3.	DMI_MCDS_IN	UI	Requests poll of DEU via SVC
4.	FPMSVC	FCOS	passes control to ...
5.	FIOSVC	FCOS	who calls ...
6.	FCMSSYNC	FCOS	to Synchronize RS GPC's
7.	FIOSVC	FCOS	then formats the I/O request and calls ...
8.	FIOPDISP	FCOS	to initiate the I/O request
9.	DEU Control program	DEU	responds with MSG RDY flag in mode status message
10.	DMI_MCDS_IN	UI	receives message and calls
11.	DMM_MCDS_PROCESS	UI	who initiates a read (steps 4-8)
12.	DEU Control program	DEU	responds with queued SPEC xxx PRO message
13.	DMM_MCDS_PROCESS	UI	converts message to internal codes and enables ...
14.	DMC_SUPER	UI	who prepares the message for processing and calls ...
15.	DMC_FUNCTIONS	UI	who recognizes SPEC input and calls
16.	DMC_SEQ_REQ_PROC	UI	who issues a schedule SVC for the control segment

G.1 [SPEC] xxx [PRO] Timeline (cont'd)

#	Software Processor	Subsystem	Action
17.	FPMSVC, FPMSCHED	FCOS	add the control segment PCT to the run queue and enters
18.	FPMDISP	FCOS	who initiates execution of the Control Segment, meanwhile,
19	DNX_BMS	UI	has established grammer interfaces for use by the SPEC Control Segment
20.	DIS_PLAY	APPL	requests DISPLAY XXX via grammer
21.	DIS_PLAY	UI	checks for the proper display in main memory and if necessary requests a fetch of the display from Mass Memory
22.	DMC_NEW_DISPLAY	UI	requests an SVC to move the display background to the DEU
23.	same as 4-8	FCOS	
24.	SPEC CS	APPL	is now awaiting a crew input
25.	The Crew enters [ITEM] x [EXEC]		
26.	DEU Control Program	DEU	(same as 2.)
27.	same as 3-11		
28.	DEU Control Program	DEU	responds with gueried [ITEM] x [EXEC] message
29.	same as 13-14		
30.	DMC_FUNCTIONS	UI	recognizes ITEM input and calls...
31.	DMC_ITEM_PROC	UI	who validates the item input and makes the data available to the SPEC CS via the MACT.

G.1 [SPEC] xxx [PRC] Timeline (cont'd)

#	Software Processor	Subsystem	Action
32.	SPEC CS	APPL	receives control from UI and performs appropriate ITEM processing
33.	return to 24		

G.2 SYNC Timelines (cont'd)

Cycle #	Time HH/MM.SSSSSS	Time Since Last Change Based on GPC1	GPC1 SYNC Code	GPC1 DELTA Time	GPC2 SYNC Code	GPC2 Delta Time
3	00/13.643921	6192	--	---	SIP	6192
	00/13.643922	1	SIP	6210	--	---
	00/13.643998	76	RUN	76	--	---
	00/13.643999	1	--	---	RUN	78
	00/13.644082	83	--	---	--	---
	00/13.645616	1534	SVC	1518	--	---
	00/13.645743	127	--	---	SVC	1744
	00/13.645842	99	--	---	--	---
	00/13.645870	28	--	---	RUN	127
	00/13.646074	4	RUN	256	--	---
	00/13.647806	1732	SVC	1732	--	---
	00/13.647831	25	--	---	SVC	1761
	00/13.647952	121	--	---	RUN	121
	00/13.647988	16	RUN	162	--	---
	00/13.648629	861	SVC	861	--	---
	00/13.648629	0	--	---	SVC	877
	00/13.648744	115	RUN	115	--	---
	00/13.648744	0	--	---	RUN	115
	00/13.649935	1191	--	---	IOC	1191
	00/13.649943	8	IOC	1199	--	---
	00/13.650067	124	--	---	RUN	132
	00/13.650075	8	RUN	132	--	---
	00/13.651031	956	IOC	956	--	---
	00/13.651033	2	--	---	IOC	966
	00/13.651163	130	RUN	132	--	---
	00/13.651164	1	--	---	RUN	131
	00/13.653462	2298	SVC	2299	--	---
	00/13.653469	7	--	---	SVC	2305
	00/13.653576	107	RUN	114	--	---
	00/13.653582	6	--	---	RUN	113
	00/13.654375	793	SVC	799	--	---
	00/13.654385	10	--	---	SVC	803
	00/13.654502	117	RUN	127	--	---
	00/13.654511	9	--	---	RUN	126
	00/13.659586	5075	IOC	5084	--	---
	00/13.659599	13	--	---	IOC	5088
	00/13.659731	132	--	---	RUN	132
	00/13.659765	34	RUN	179	--	---
	00/13.664010	4245	--	---	TIMER	4279
	00/13.664014	4	TIMER	4249	--	---
	00/13.664069	55	--	---	RUN	59
	00/13.664071	2	RUN	57	--	---
	00/13.666329	2258	--	---	IOC	2269
	00/13.666336	7	IOC	2265	--	---
	00/13.666460	124	--	---	RUN	131
	00/13.666466	6	RUN	130	--	---
	00/13.667013	547	--	---	SVC	553
	00/13.667018	5	SVC	552	--	---
	00/13.667127	109	--	---	RUN	114
	00/13.667131	4	RUN	113	--	---
	00/13.670705	3574	--	---	SVC	3578
	00/13.670712	7	SVC	3581	--	---
	00/13.670818	106	--	---	RUN	113
	00/13.670826	8	RUN	114	--	---
	00/13.671339	513	SVC	513	--	---
	00/13.671454	115	--	---	SVC	636
	00/13.671559	105	RUN	220	--	---
	00/13.671589	30	--	---	RUN	135
	00/13.673283	1694	--	---	IOC	1694
	00/13.673367	84	IOC	1508	--	---
	00/13.673496	131	RUN	131	--	---
	00/13.673507	9	--	---	RUN	224
	00/13.677767	4260	SVC	4269	--	---
	00/13.677963	196	--	---	SVC	4456
	00/13.678076	113	--	---	RUN	113
	00/13.678103	27	RUN	330	--	---
	00/13.683054	4951	SVC	4951	--	---
	00/13.683094	40	--	---	SVC	5016
	00/13.683208	114	--	---	RUN	114
	00/13.683215	7	RUN	151	--	---
4	00/13.683917	702	--	---	SIP	709
	00/13.683923	6	SIP	708	--	---
	00/13.683994	71	--	---	RUN	77
	00/13.684003	9	RUN	80	--	---

G.2 SYNC Timelines (cont'd)

Cycle #	Time HH/MM.SSSSSS	Time Since Last Change Based on GPC1	GPC1 SYNC Code	GPC1 DELTA Time	GPC2 SYNC Code	GPC2 DELTA Time
2	00/13.603922	4350	--	---	SIP	4362
	00/13.603927	5	SIP	4355	--	---
	00/13.603998	71	--	---	RUN	76
	00/13.604003	5	RUN	76	--	---
	00/13.604070	67	--	---	--	---
	00/13.604074	4	--	---	--	---
	00/13.604599	525	--	---	SVC	601
	00/13.604608	9	SVC	605	--	---
	00/13.604697	89	--	---	--	---
	00/13.604713	16	--	---	RUN	114
	00/13.604718	5	RUN	110	--	---
	00/13.606352	1634	--	---	SVC	1639
	00/13.606358	6	SVC	1640	--	---
	00/13.606466	108	--	---	RUN	114
	00/13.606472	6	RUN	114	--	---
	00/13.608577	2105	--	---	IOC	2111
	00/13.608585	8	IOC	2113	--	---
	00/13.608706	123	--	---	RUN	131
	00/13.608716	8	RUN	131	--	---
	00/13.609400	684	--	---	IOC	692
	00/13.609407	7	IOC	691	--	---
	00/13.609531	124	--	---	RUN	131
	00/13.609539	8	RUN	132	--	---
	00/13.611865	2326	--	---	SVC	2334
	00/13.611875	10	SVC	2336	--	---
	00/13.611979	104	--	---	RUN	114
	00/13.611983	9	RUN	115	--	---
	00/13.612746	758	--	---	SVC	767
	00/13.612757	11	SVC	769	--	---
	00/13.612873	116	--	---	RUN	127
	00/13.612886	13	RUN	129	--	---
	00/13.614960	2074	--	---	SVC	2087
	00/13.614967	7	SVC	2081	--	---
	00/13.615075	108	--	---	RUN	115
	00/13.615081	6	RUN	114	--	---
	00/13.616077	996	--	---	SVC	1002
	00/13.616084	7	SVC	1003	--	---
	00/13.616192	108	--	---	RUN	115
	00/13.616199	7	RUN	115	--	---
	00/13.617958	1759	--	---	IOC	1766
	00/13.617968	10	IOC	1769	--	---
	00/13.618088	120	--	---	RUN	130
	00/13.618099	11	RUN	131	--	---
	00/13.624007	5908	--	---	TIMER	5919
	00/13.624011	4	TIMER	5912	--	---
	00/13.624067	56	--	---	RUN	60
	00/13.624069	2	RUN	58	--	---
	00/13.626292	2223	--	---	IOC	2225
	00/13.626298	6	IOC	2229	--	---
	00/13.626423	125	--	---	RUN	131
	00/13.626429	6	RUN	131	--	---
	00/13.629974	3545	--	---	SVC	3551
	00/13.629985	11	SVC	3556	--	---
	00/13.630087	102	--	---	RUN	113
	00/13.630098	11	RUN	113	--	---
	00/13.630546	448	SVC	448	--	---
	00/13.630637	91	--	---	SVC	550
	00/13.630775	138	RUN	229	--	---
	00/13.630780	5	--	---	RUN	143
	00/13.634031	3301	--	---	IOC	3311
	00/13.634172	91	IOC	3397	--	---
	00/13.634304	132	RUN	132	--	---
	00/13.634305	1	--	---	RUN	224
	00/13.637331	3026	SVC	3027	--	---
	00/13.637614	283	--	---	SVC	3309
	00/13.637712	98	RUN	381	--	---
	00/13.637729	17	--	---	RUN	115

G.2 SYNC Timelines

* All times in microseconds

Cycle #	Time* HH/MM.SSSSSS	Time* since Last Change Based on GPC1	GPC1 SYNC Code	GPC1 DELTA Time*	GPC2 SYNC Code	GPC2 DELTA Time*
1	00/13.563923		--	---	SIP	
	00/13.563924	1	SIP		--	---
	00/13.564000	76	--	---	RUN	77
	00/13.564001	1	RUN	77	--	---
	00/13.564079	69	--	---	--	---
	00/13.565341	1271	SVC	1340	--	---
	00/13.565443	102	--	---	SVC	1443
	00/13.565546	103	RUN	235	--	---
	00/13.565546	0	--	---	--	---
	00/13.565562	16	--	---	RUN	119
	00/13.567283	1721	SVC	1737	--	---
	00/13.567304	21	--	---	SVC	1742
	00/13.567421	117	--	---	RUN	117
	00/13.567445	24	RUN	152	--	---
	00/13.569595	2150	--	---	IOC	2174
	00/13.569626	31	IOC	2181	--	---
	00/13.569757	131	RUN	131	--	---
	00/13.569774	17	--	---	RUN	179
	00/13.570696	922	IOC	939	--	---
	00/13.570710	14	--	---	IOC	936
	00/13.570841	131	--	---	RUN	131
	00/13.570874	33	RUN	178	--	---
	00/13.573103	2229	--	---	SVC	2262
	00/13.573150	47	SVC	2276	--	---
	00/13.573262	112	--	---	RUN	159
	00/13.573264	2	RUN	114	--	---
	00/13.574056	792	SVC	792	--	---
	00/13.574060	4	--	---	SVC	798
	00/13.574183	123	RUN	127	--	---
	00/13.574186	3	--	---	RUN	126
	00/13.574438	252	SVC	255	--	---
	00/13.574440	2	--	---	SVC	254
	00/13.574555	115	RUN	117	--	---
	00/13.574556	1	--	---	RUN	116
	00/13.579279	4723	--	---	IOC	4723
	00/13.579284	5	IOC	4729	--	---
	00/13.579411	127	--	---	RUN	132
	00/13.579415	4	RUN	131	--	---
	00/13.584013	4598	--	---	TIMER	4602
	00/13.584014	1	TIMER	4599	--	---
	00/13.584071	57	--	---	RUN	58
	00/13.584073	2	RUN	59	--	---
	00/13.586322	2249	--	---	IOC	2251
	00/13.586344	22	IOC	2271	--	---
	00/13.586474	130	RUN	130	--	---
	00/13.586501	27	--	---	RUN	179
	00/13.587012	511	SVC	538	--	---
	00/13.587038	26	--	---	SVC	537
	00/13.587153	115	--	---	RUN	115
	00/13.587171	18	RUN	159	--	---
	00/13.590570	3399	--	---	SVC	3417
	00/13.590592	22	SVC	3421	--	---
	00/13.590706	114	RUN	114	--	---
	00/13.590729	23	--	---	RUN	159
	00/13.591225	496	SVC	519	--	---
	00/13.591307	142	--	---	SVC	638
	00/13.591494	127	RUN	259	--	---
	00/13.591503	9	--	---	RUN	136
	00/13.593169	1666	--	---	IOC	1666
	00/13.593262	93	IOC	1758	--	---
	00/13.593393	131	RUN	131	--	---
	00/13.593396	3	--	---	RUN	227
	00/13.597386	3990	SVC	3993	--	---
	00/13.597580	194	--	---	SVC	4184
	00/13.597678	98	RUN	292	--	---
	00/13.597694	16	--	---	RUN	114
	00/13.598672	978	SVC	994	--	---
	00/13.598794	122	--	---	SVC	1100
	00/13.598908	114	--	---	RUN	114
	00/13.598920	12	RUN	248	--	---
	00/13.599446	526	--	---	SVC	538
	00/13.599458	12	SVC	538	--	---
	00/13.599560	102	--	---	RUN	114
	00/13.599572	12	RUN	114	--	---

G.3 DISCRETE INPUT

#	Software Processor	Subsystem	Action
1.			Crew sets panel switch to initialize IUS flight Data
2.	up to 40 MS of time passes		
3.	FCMCSYNC	FCOS	reads DI reg A during common set SYNC
4.	up to 1 second of time passes		
5.	FPMIHPC2	FCOS	receives control due to timer interrupt and services TQE for GPC switch monitor (ARA_GPC_SWITCH)
6.	Dispatcher	FCOS	enters ...
7.	ARA_GPC_SWITCH	VI	detects discrete change and "simulates" MCDS input to request SPEC for IUS initialization
8.	same as G.1, #14-#24		
9.	DMC_ITEM_PROC	UI	prepares internally passed ITEM request and makes it available to SPEC CS
10.	SPEC CS	APPL	interprets ITEM Data supplied by UI.
11.	SPEC CS	APPL	performs computations required to prepare IUS data and formats an I/O request and issues an SVC to do the I/O
12.	FPMSVC	FCOS	receives SVC interrupt and calls...
13.	FIOSVC	FCOS	who calls ...
14.	FCMSSYNC	FCOS	to perform redundant set SYNC.
15.	FIOSVC	FCOS	prepares the request and calls
16.	FIOPDISP	FCOS	to activate I/O by starting ...

G.3 DISCRETE INPUT (cont'd)

#	Software Processor	Subsystem	Action
17.	FIOMCNTL	FCOS	to perform the I/O
18.	The data is transferred and ...		
19.	FIOCMPLT	FCOS	receives an I/O complete interrupt and calls
20.	FPMSWTCH	FCOS	to see if the SPEC CS is ready to resume execution. If so
21.	FPMDISP	FCOS	dispatches the ...
22.	SPEC CS	APPL	who sets a parameter for displaying the result of the IUS data transfer
23.	up to 1 second of time passes		
24.	DCI #CYC	UI	updates the display
25.	The crew resets the panel switch		
26.	same as #2-#4		
27.	ARA_GPC_SWITCH		checks discretes and detects panel change and "simulates" an MCDS input to resume display which was on the CRT prior to initial switching.
28.	DMC_SUPER, et al		restores old function to the display.

LIST OF ACRONYMS AND ABBREVIATIONS

AIT	Application Interface Table
AMT	Application Moding Table
BRQE	Bus Reconfiguration Queue Element
BSR	Branch Sector Register
CS	Common Set
CVT	Communication Vector Table
DDT	Dynamic Data Table
DFB	Display Format Buffer
DFG	Display Format Generator
DFT	Display Format Table
DST	DPS Status Table
DSR	Data Sector Register
EQE	Event Queue Element
FMPT	Fault Message Processing Table
GPC_{PT}	GPC Prime Internal Time
GPC_{MT}	GPC Self Internal Time
GPR	General Purpose Register
GST	GPC Status Table
IOQE	I/O Queue Element
IPR	Input Problem Reporting
LGT	Lock Group Table
LOG	GPC Error Log
MF	Major Function
PC	Program Control
PC1	Program Counter 1
PC2	Program Counter 2
PCT	Process Control Table
PDE	Process Directory Entry
PSA	Preferred Storage Area
SSIP	System Services Interface Processor Cycles
TQE	Time Queue Element

☆ U.S. GOVERNMENT PRINTING OFFICE: 1977—771-015/434