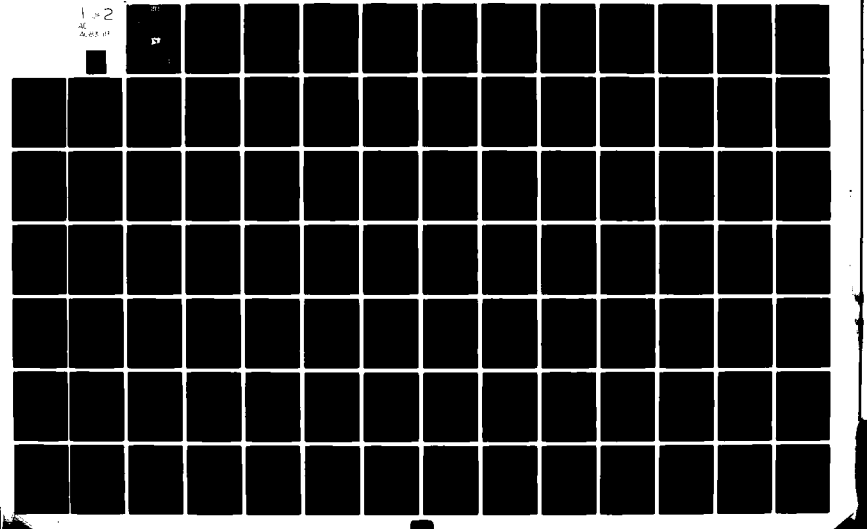
AD-A083 118 AIR FORCE AVIONICS LAB WRIGHT-PATTERSON AFB OH F/G 9/2
COMPUTER PROGRAM DEVELOPMENT SPECIFICATION FOR IDAMST OPERATION--ETC(U)
JUL 76
UNCLASSIFIED AFAL-TR-76-209-ADD-2 NL



AFAL-TR-76-209, Addendum #2

AFAL-TR-76-209 ADDENDUM #2

LEVEL III

SPECIFICATION NUMBER

CODE IDENT

PART 1 OF TWO PARTS

(DATE) 30 JULY 1976

COMPUTER PROGRAM DEVELOPMENT SPECIFICATION FOR IDAMST OPERATIONAL FLIGHT PROGRAM EXECUTIVE, SOFTWARE TYPE B5. Addendum 2.

(11) 30 Jul 76

(14) AFAL-TR-76-209-ADD-2, SPEC-SD-2041-PT-1

(12) 144

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

AIR FORCE AVIONICS LABORATORY
AFAL/AAA-1
WRIGHT-PATTERSON AFB, OHIO 45433

Table of Contents

Table of Contents - Cont.

LIST OF FIGURES

LIST OF FIGURES (CONT'D)

LIST OF TABLES

LIST OF TABLES (CONT'D)

1.0 SCOPE

1.1 Identification

This specification establishes the requirements for performance, design, test, and qualification of a computer program identified as Operational Flight Program (OFP), Executive Software for the Integrated Digital Avionics for the Medium STOL Transport (IDAMST).

1.2 Functional Summary

The IDAMST Executive Software Systems provides the system software services for the application software. It has been divided into three major functions denoted as Master Executive, Local Executive and Monitor Executive Functions. These functions provide services for the execution of real-time applications, data bus management, system control, common data utilization and Remote Terminals communication.

2.0 Applicable Documents

The following documents of the exact issue shown form a part of this specification to the extent specified herein. In the event of conflict between the documents referenced herein and the contents of this specification, the contents of this specification shall be considered a superseding requirement.

Specifications:

1) OFP IDAMST Error Handling and Recovery System Software (EHARS), SD2042

2) System Specification Type A SI 1010, June 1976

3) Control/Display System Segment Spec Type A SR 5020.

Other Publications:

1) Specifications for IDAMST Software Technical Report

2) DAIS Mission Software Executive Specification F33615-75-C-1181, 26 December 1975

3.0 REQUIREMENTS

3.1 Program Definition

3.1.1 Hardware Interfaces

The IDAMST Executive System interfaces with the following elements of hardware: a Bus Control Interface Unit (BCIU), Remote Terminals, Mass Memory, a Processor Control Panel (PCP), and Processors.

3.1.1.1 Bus Control Interface Unit (BCIU)

The Bus Control Interface Unit (BCIU) shall provide the interface control and data transfer function required to connect a Processor with two multiplexed data buses. The BCIU shall be directed to operate in a mode by its interfacing processor. The following are the modes in which the BCIU shall be capable of operating:

a. Remote Mode, providing transfer of date in both directions between the Processor and either of the two Buses, providing status replies on the appropriate bus in response to commands, and special internal operations and interrupts to the associated processor upon receipt of certain special commands on the data buses.

b. Master Mode, providing control of the data bus based upon instructions fetched from the memory of the Processor through the Direct Memory Access (DMA) Channel by the BCIU.

This Master Control mode shall result in:

1. The BCIU issuing Bus Commands to other devices on the Data Buses.

2. Participating in data transfers on the buses (when the instruction dictates it).

3. Checking status responses from devices on the data buses.

4. Checking formats of the data bus operation.

5. Reporting of error conditions to the processor.

At any time, there shall only be one BCIU in Master Mode.

3.1.1.1.1 Instruction Format

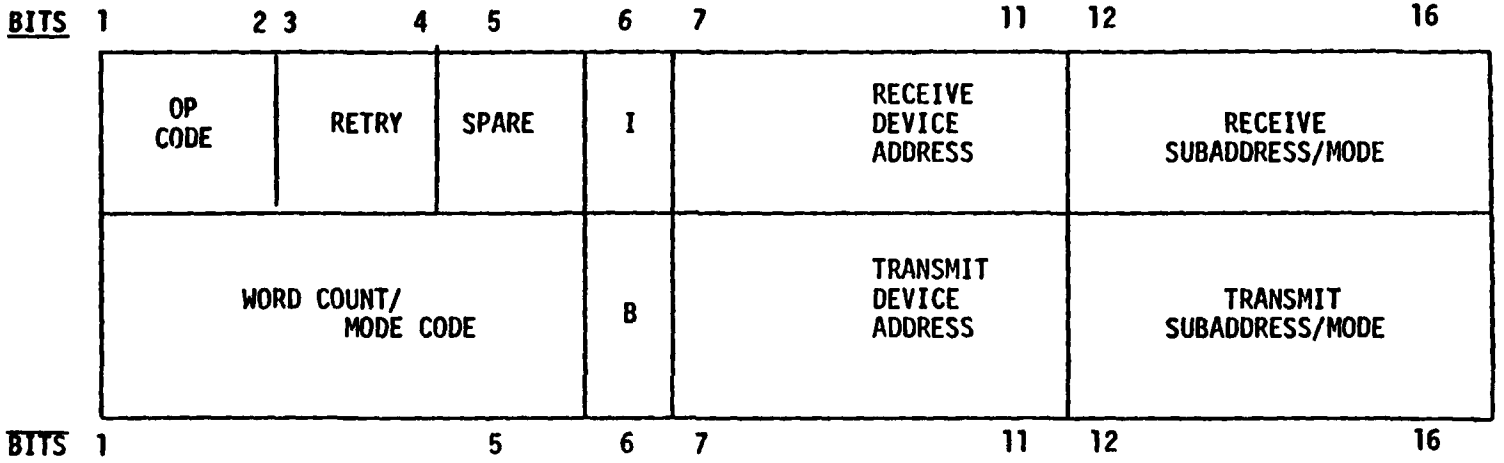
The BCIU instruction list is composed of pairs of instructions accessed by the BCIU using a DMA channel. The BCIU sequentially interprets instruction pairs to determine the action required. The format of the instruction pair is shown in Figure 1.

Each of the fields in the two word instruction have the following uses:

a. OP CODE - These two bits deterine the function of the command.

00 = Halt the BCIU. This is always the last command in a list and denotes that no other command is to be performed. When the BCIU executes this instruction the Halt bit is set inthe Internal Status Register and a BCIU level 1 interrupt will be generated.

01 = Link. This OP code is used to link sections of the command list. Thus, the individual instructions of the command list need not occupy contiguous memory locations. The second word of the instruction is used as the address of the next two word instruction. The other fields of the instruction are ignored except for the interrupt (I) field.



B = BIT TO DEFINE WHICH BUS THIS BUS OPERATION SHALL BE PERFORMED ON. (LOGIC 1 SELECTS BIM2)

I = BIT TO INDICATE WHETHER AN INTERRUPT TO THE PROCESSOR UPON SUCCESSFUL COMPLETION OF THE BUS OPERATION. (LOGIC 1 PRESENTS PROGRAMMED CONTROLLED INTERRUPT [PCI] TO PROCESSOR)

RETRY = INDICATOR TO SOFTWARE OF NUMBER OF RETRIES ALLOWED BY THIS 2 WORD INSTRUCTION (MAXIMUM OF THREE RETRIES PER INSTRUCTION)

BCIU INSTRUCTION FORMAT

FIGURE 1.

10 = No Operation. This OP code has two uses. The first is to cancel commands which the Master Processor no longer wishes the Master BCIU to perform.

The second is to create a processor interrupt before the next BCIU instruction is generated. A typical use of the latter case is sending Mode Commands. The Mode Data Register must be set before the command is sent. Thus, the interrupt causes a BCIU pause which permits the Master Processor to set the MDR and then set the Continue Bit in the PCR to reset BCIU processing.

For this OP code only the interrupt field is examined. All other options are ignored.

11 = Bus Operation. For this operation the rest of the fields are interpreted as reception or transmission across the Bus.

b. RETRY - If the transmission attempted by this instruction was not successfully completed, and this field is not zero, then the transmission will be retired up to three times.

c. SPARE - This bit is not used.

d. I - If this bit is set, successful completion of this instruction will cause an interrupt. The PCI bit in the ISR will be set. The interrupt will be of level 1. The discussion accompanying the No Operation Code explains the use of this bit, although the bit may be used in any of the four instructions.

e. RECEIVE DEVICE ADDRESS - This field contains the address of the terminal to receive the message. This field is only used for BCIU instruction OP code "Bus Operation" (11). If the Receive Device Address is not the address of the Master BCIU (as contained in the BCIU address field of the PCR), then a Receive Command will be formed by concatenating the Receive Device Address Field, a bit denoting Receive, the Receive Subaddress/

Mode field, and the Word Count/Mode Code field. This receive command will then be transmitted across the Bus.

If the Receive Device Address field is the address of this BCIU and the Receive Subaddress/Mode field is not zero (i.e., this is not a Mode Command), then the Receive Subaddress field will be used to load the Data Address Register (see Section 3.1.1.1.2.15) which will then determine where the received message will be stored.

f. RECEIVE SUBADDRESS/MODE - This field describes the message to be received. The use of this field is described in the Receive Device Address field. If this address were zero it would indicate that this is a Mode Command.

g. WORD COUNT/MODE CODE - For mode commands this field contains the number of the command. For Receive/Transmit commands this field contains the number of data words to be transmitted.

h. B - This field indicates which Bus will be used for data transmission. If this bit is zero, Bus number one will be used. If this bit is one, Bus number two will be used.

i. TRANSMIT DEVICE ADDRESS - This field is similar to the Receive Device Address except that it is the address of the terminal which will send the message.

If the address is not that of this Master BCIU, then Transmit Command will be formed by concentrating the Transmit Device Address field, the Transmit bit, the Transmit Subaddress/Mode field and the Word Count/Mode Code field.

If the Transmit Device Address field is the address of this terminal then the Data Address Register will be formed (see Section 3.1.1.1.2.15) and the data will be written into the Bus from that address.

For Mode Commands the Transmit Device Address field is the address of the terminal to receive the Mode Command and the Receive Device Address field is the address of the Master BCIU.

It is an error for the Receive Device Address field and the Transmit Device Address field to be the same device. This error will cause an interrupt of level 1 and the IVI bit will be set in the Internal Status Register.

j. TRANSMIT SUBADDRESS/MODE - The use of this field has been discussed in the description of the Transmit Device Address field.

For mode commands, both the Transmit Subaddress and Receive Subaddress will be zero.

3.1.1.1.2 BCIU Registers

The registers on the BCIU control its mode of operation, provide information for the master processor and provide information to its local processor. There are sixteen, 16-bit registers accessible to the processor through the PIO.

These registers and their respective PIO addresses are listed in Table I. Their description follows:

3.1.1.1.2.1 Processor Control Register (PCR)

This register's format is illustrated in Figure 2. The description of this format follows:

a. Master - This bit is set to logic 1 by the processor, to direct the BCIU to operate in Master Mode.

b. GO - Set to logic 1 by the processor to indicate the BCIU is to enter an operational mode. A logic 0 indicates the termination of an operational mode. A HALT instruction in master mode will set it to logic 0.

TABLE I. BCIU REGISTERS

PIO ADDRESS	
0	PROCESSOR CONTROL REGISTER (PCR)
1	INTERNAL STATUS REGISTER (ISR)
2	BASE ADDRESS REGISTER (BAR)
3	INSTRUCTION ADDRESS REGISTER (IAR)
4	BUILT-IN-TEST REGISTER (BITR)
5	MODE DATA REGISTER (MDR)
6	LAST COMMAND REGISTER (LCR)
7	STATUS CODE REGISTER (SCR)
8	MASTER FUNCTION REGISTER (MFR)
9	POINTER REGISTER (PR)
10	DATA ADDRESS REGISTER (DAR)
11	WORD COUNT REGISTER (WCR)
12	XMIT STATUS WORD REGISTER (XSWR)
13	RECV STATUS WORD REGISTER (RSWR)
14	INSTRUCTION WORD REGISTER 1 (IWR1)
15	INSTRUCTION WORD REGISTER 2 (IWR2)

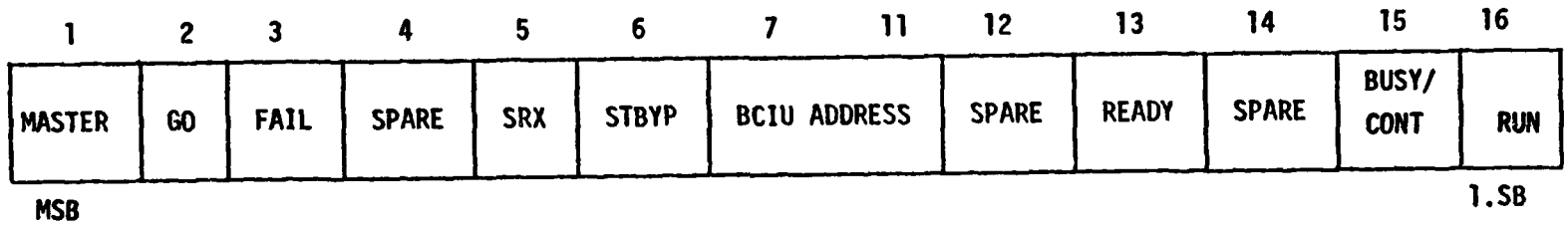


FIGURE 2. PROCESSOR CONTROL REGISTER (PCR)

c. FAIL - Set to logic 1 after detecting an error in self-test.

d. SPARE - Set to logic 0.

e. System Reset Acknowledge - Set to logic 1 by the processor to indicate acknowledgement of the power-on-reset interrupt.

f. Self-Test By-Pass - Set to logic 1 by the processor indicate that the BCIU is not to perform self-test.

g. BCIU Address - These 5 bits shall be set by the processor to indicate the address on the BCIU.

h. SPARE - Set to logic 0.

i. READY - Set to logic 1 by the BCIU after completing its power-on initialization.

j. BUSY/CONT - Set to logic 1 by the remote processor to indicate the BCIU is to enter BUSY state. It is set to logic by the BCIU after having been directed to exit BUSY state.

In master mode, the bit is set to logic by master processor to indicate to the BCIU that an interrupt has been processed.

k. RUN - Set to logic 1 by BCIU after being directed to enter an operational mode or upon exciting a BUSY state. It is set to by the BCIU after terminating an operational mode.

3.1.1.1.2.2 Internal Status Register (ISR)

This register shall be set only by the BCIU. It contains indications of the cause of a BCIU generated interrupt. This register is cleared by the BCIU prior to processing a new instruction or command.

This register's format and the meaning of each bit is indicated in Figure 3. The interrupt levels generated by these bits are also indicated in this figure.

A description of each bit follows:

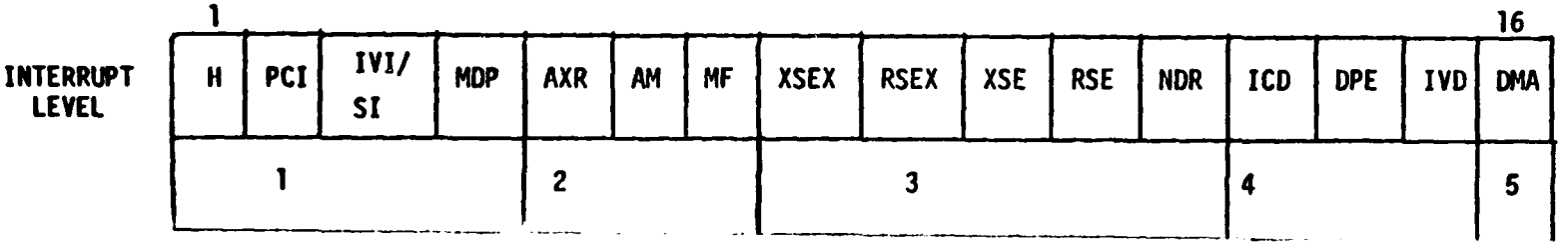
a. HALT (H) This bit shall be set to logic 1, in Master Mode only, to indicate that the BCM has processed a HALT instruction. The operational mode (Master) shall be terminated.

b. Program Controlled Interrupt (PCI) This bit shall be set to logic 1, in Master Mode only, after completion of 2 word instruction operation in which PCI was requested (PCI=1).

c. Invalid Instruction (IVI) In Master Mode only, this bit shall be set to logic 1 if the Device Address within the Receive field of the 2-word instruction is equal to the Device Address within the Transmit field.

d. System Interrupt (SI) In REmote Mode only, this bit shall be set to logic 1 upon receiving the System Interrupt Mode Command.

e. Mode Data Present (MDP) This bit shall be set to logic 1, in Master Mode only, after successfully receiving the Data Word associated with Mode Operations (Interrupt results from mode operations 3, 10, 11, and 13 - Refer to paragraph 3.2.1.1.1.).



	BIT	MEANING
1	1	H - HALT
	2	PCI - PROGRAM CONTROLLED INTERRUPT
	3	IVI/ SI - INVALID INSTRUCTION (MASTER MODE ONLY) / SYSTEM INTERRUPT (REMOTE MODE ONLY)
2	4	MDP - MODE DATA PRESENT
	5	AXR - A SYNC MSG INT. Ø = RECV. 1 = XMIT
	6	AM - A SYNC MSG INT.
	7	MF - MASTER FUNCTION (REMOTE MODE ONLY)
3	8	XSEX - XMIT STATUS EXCEPTION
	9	RSEX - RECV STATUS EXCEPTION
	10	XSE - XMIT STATUS ERROR
	11	RSE - RECV STATUS ERROR
	12	NDR - NO DATA RECEIVED
4	13	ICD - INCOMPLETE DATA
	14	DPE - DATA PARITY ERROR
	15	IVD - INVALID DATA
5	16	DMA - DIRECT MEMORY ACCESS (ERROR)

FIGURE 3. INTERNAL STATUS REGISTER (ISR)

f. Asynchronous Message Xmit/Recv (AXR) In Master or Remote Modes, this bit shall be set in conjunction with Bit 6 (AM) to indicate whether the BCIU was the Receiver (AXR=0) or the Transmitter (AXR-1) of an asynchronous message (Sub-Address=31).

g. Asynchronous Message (AM) In Master or Remote Modes, this bit shall be set to logic 1 after successful completion of an asynchronous bus message operation (Sub-Address=31).

h. Master Function (MF) This bit shall be set to logic 1, in Remote Mode only, after receiving the Master Function Mode Command.

i. Transmit Status Exception (XSEX) This bit shall be set to logic 1, in Master Mode only, after receiving an excepted, valid status word associated with a Remote device in Transmit Mode in which the Message Error, Terminal Failure, or Status Code is non-zero. The status word shall be placed intact within the Xmit Status Word Register.

j. Receive Status Exception (RSEX) This bit shall be set to logic 1, in Master Mode only, after receiving an expected, valid status word associated with a Remote device in Receive Mode in which the Message Error, Terminal Failure, or Status Code is non-zero. The status word shall be placed intact within the Received Status Word Register.

k. Transmit Status Error (XSE) This bit shall be set to logic 1, in Master Mode only, if an expected status word associated with a Remote device in Transmit Mode is not received, is received invalidly, is received validly with bad parity, or is received validly with good parity with a Device Address that does not match the Transmit Device Address within the 2-word instruction.

l. Receive Status Error (RSE) This bit shall be set to logic 1, in Master Mode only, if an expected status word associated with a Remote Device in Receive mode, is not received, is received invalidly, is received validly with bad parity, or is received validly with good parity with a Device Address that does not match the Receive Device Address within the 2-word instruction.

m. No Data Receive (NDR) This bit shall be set to logic 1, in Master Mode only, after commanding a remote device to transmit one or more data words and the first such data word has not arrived within 60 microseconds after status word reception.

n. Incomplete Data (ICD) This bit shall be set to logic 1, in Master Mode only, after receiving at least one expected data word and with further data words expected, the next data word is not received within 60 microseconds after reception of the last data word.

o. Invalid Data (IVD) This bit shall be set to logic 1, in Master Mode only, after an expected data word was received with Parity Error indicated. Data word reception continues.

p. Direct Memory Access Error (DMA) This bit shall be set to logic 1, in Master or Remote Mode, after an unrecoverable DMA Error is detected while attempting to fetch an instruction word, a pointer word, or a data word from main memory or while attempting to store a tag word or a data word into main memory.

3.1.1.1.2.3 Base Address Register (BAR)

This register shall be set only by a Processor for the associated BCIU (Master/Remote) and shall contain the most significant 10 bits of a pointer word address within main memory for a given data transfer operation. The addressed pointer word shall contain the true data block address.

3.1.1.1.2.4 Instruction Address Register (IAR)

This register shall be met only by a Processor whose associated BCIU is to operate in Master Mode. The register shall contain the main memory address of the initial 2-word instruction executed, the BCIU shall modify the register in order to reflect the address of the next instruction to be executed. The register shall be unused in Remote Mode.

3.1.1.1.2.5 Last Command Register (LCR)

This register shall be used only in support of the Transmit Last Command Mode Command. In Remote mode, the BCIU shall place commands which are received validly and directed to the particular BCIU into this register. Exceptions shall be Transmit Status Word, Transmit Bit Word, and the Transmit Last Command itself.

3.1.1.1.2.6 Built-In Test Word Register (BITR)

This register shall be used to either maintain the Built-In Test Word (Remote Mode), or to temporarily hold Terminal Failure or bus monitoring of own transmission information (Master Mode). The format of a BCIU BIT word is shown in Figure 4, and described in the following paragraphs.

a. Power-On-Reset This bit shall be set to logic 1 if the BCIU performs Power-On Initialization.

b. Power Supply Failure This bit shall be set to Logic 1 in the event of failure.

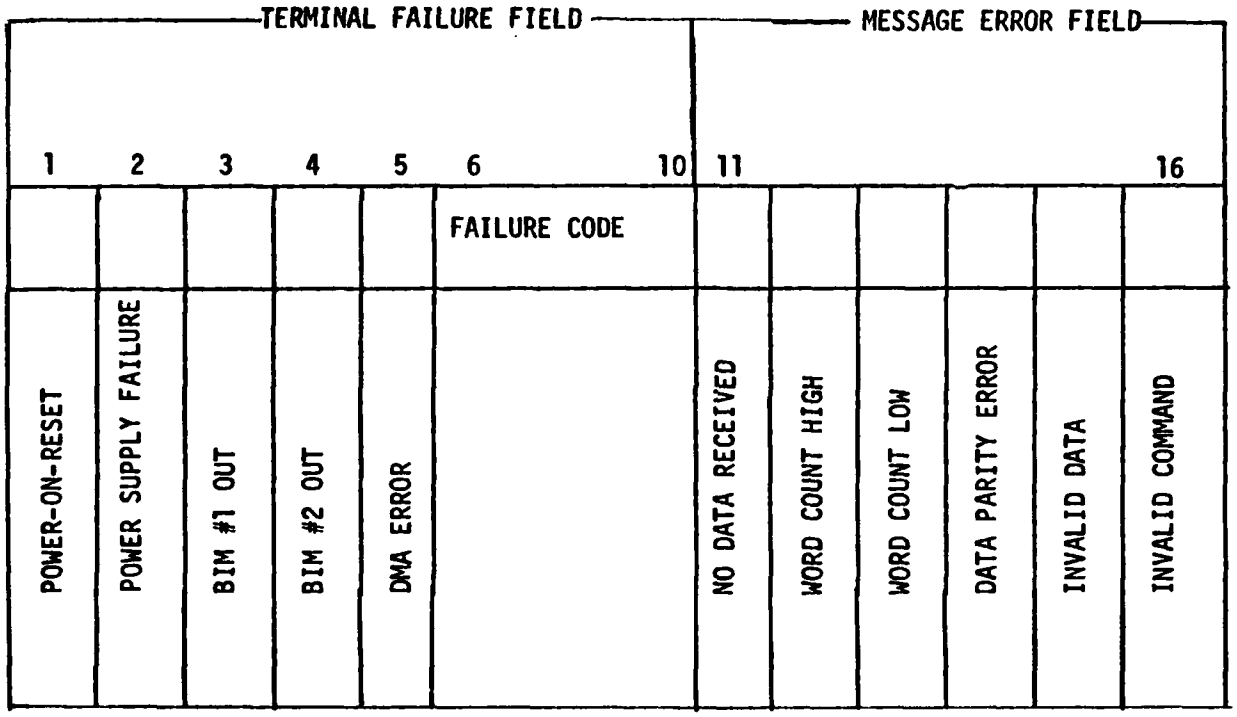


FIGURE 4. BCIU BUILT-IN-TEST (BIT) WORD FORMAT

c. BIM 1 Out This bit shall be set to logic 1 by the Remote Mode BCIU after powering down BIM 1 as a result of receiving a Remove Power BIM 1 Mode Command. The BIT shall indicate that power has been removed from BIM 1.

d. BIM 2 Out This bit shall be set to logic 1 by the Remote mode BCIU after powering down BIM 2 as a result of receiving a Remove Power BIM 2 Mode Command. The bit shall indicate that power has been removed from BIM 2.

e. DMA Error This bit shall be set to logic 1 by the Remote Mode BCIU after an unrecoverable direct memory access error is detected while fetching data words from or storing data words (excluding tag words) into main memory.

f. Failure Code Errors The failure code shall be set to indicate detected self-test failures as follows:

o No failure	00000
o BIM #1 failure	10001
o BIM #2 failure	10010
o MROM Parity Error	10011
o BCM Data Flow Error	10100
o BCM DROM Error	10101
o BCM SEQ Error	10110
o PIM DMA Data Flow Error	10111

g. No Data Received This bit shall be set to logic 1 by the Remote Mode BCIU after having been directed to receive one or more data words and the first such data word has not arrived within 75 microseconds after command word reception.

h. Word Count Low This bit shall be set to logic 1 by the Remote Mode BCIU after having been directed to receive two or more data words, at least one such data word has arrived, but the next expected data word does

not arrive within 60 microseconds of last data word reception.

i. Word Count High This bit shall be set to a logic 1 by the Remote Mode BCIU after detecting another Data Word after the word count is zero.

j. Data Parity Error This bit shall be set to logic 1 by the Remote BCIU after an expected data word was received with Parity Error indicated. Data word reception continues.

k. Invalid Data This bit shall be set to logic 1 by the Remote mode BCIU after an expected data word was received with RECV WORD INVALID indicated. Data word reception continues.

l. Invalid Command This bit shall be set to logic 1 by the Remote BCIU after receiving a mode command in which the mode code designates an invalid operation for the BCIU.

3.1.1.1.2.7 Status Code Register (SCR)

This register shall be used in Remote Mode only and shall be set and reset by the Remote Mode Processor. The actual status code shall be the nine (9) least significant bits of the register and shall be merged into any status word prior to status word bus transmittal by the Remote BCIU.

3.1.1.1.2.8 Master Function Register (MFR)

This register shall be used only in support of the Master Function Mode Command. In Master Mode and in accordance with Master Function processing, the contents of the register shall be transmitted to the Remote device as a data word immediately following the command word. It shall be the Master Processor's responsibility to set the register. In Remote Mode, the Remote Mode BCIU shall place the received data word, in response to the Master Function mode command, into the Master Function Register. It shall be the Remote Processor's responsibility to then interpret the contents of the register.

3.1.1.1.2.9 Instruction Word Register 1 (IWRI)

This register shall be used in Master Mode only to hold the first half of the current 32-bit instruction.

3.1.1.1.2.10 Instruction Word Register 2 (IWR2)

This register shall be used in Master Mode only to hold the second half of the current 32-ibt instruction.

3.1.1.1.2.11 Xmit Status Word Register (XSWR)

This register shall be used in Master Mode only to hold any status word received from a Remote Device in Transmit Mode, in which the Message Error, Terminal Failure, or Status Code fields were non-zero.

3.1.1.1.2.12 Received Status Word Register (RSWR)

This register shall be used in Master Mode only to hold any status word received from a Remote device in Receive Mode, in which the Message Error, Terminal Failure, or Status Code fields were non-zero.

3.1.1.1.2.13 Mode Data Register (MDR)

In Master Mode, and only in accordance with performing a certain class of mode commands, the contents of this register shall be transmitted to the Remote device as a data word immediately following the command word. The Master Processor shall be responsible for setting the register.

In Remote Mode, the MDR shall be undefined for the Mode Operation defined.

3.1.1.1.2.14 Pointer Register (PR)

This register shall be set by a BCIU operating in either Master or Remote mode and shall contain the initial data area address for a given data bus operation involving main memory data transfers. The register shall be used in Tag Word Operations.

3.1.1.1.2.15 Data Address Register (DAR)

This register shall be set by a BCIU operating in either Master or Remote mode and shall be used to indicate the main memory address of the next data word to be fetched/stored in support of a given bus operation. The register shall be derived from the Pointer Register and in all cases (Receive or Transmit) that value shall be initially incremented by 1 to get over the Tag Word. This value then becomes the address to fetch/store the first data word. As each word is fetched/stored, the BCIU shall increment the register value by 1 to affect sequential data word fetch/stores.

3.1.1.1.2.16 Word Count Register (WCR)

This register shall be derived from the Bus Command and set by the BCIU in either Master or Remote Mode. In Bus Operations involving data word transfers, it shall indicate the remaining number of data words to be transferred. The register shall be decremented by 1, by the BCIU, as each data word transfer is performed.

3.1.1.1.3 Interrupt Generation

The BCIU shall examine the Program Controlled Interrupt Indicator within the Instruction Word One Register (IWR1). If set to logic 1, the BCIU shall set the PCI indicator within the ISR to logic 1. (see Figure 3). The BCIU shall begin to examine the contents of the ISR from right to left, one field at a time. If any field is found to be non-zero, the BCIU shall discontinue the examination and present the corresponding level interrupt as indicated in Figure 3.

3.1.1.2 Remote Terminals

3.1.1.2.1 Basic Characteristics

The Remote Terminal (RT) provides the interface between the IDAMST Multiplex System and an Aircraft Subsystem.

The RTs provide for Bus communication with the IDAMST processors (as described in Section 3.1.1.1.2).

The subaddress field of each Transmit or Receive Command acts as a message identifier. The message is formatted by the RT for correct interface with the Interface Modules (IM) which relay (or accept from) the signals to the aircraft subsystems.

The RT also as a buffer, holding the message until correct transmission has occurred.

The RT performs all the error checking and setting of error and status bits of a remote BCIU.

3.1.1.2.2 RT Functions

The RT shall contain the registers, logic, decoders, buffers, comparators and control sequences required to perform the following functions:

a. Receive Command Words from the Bus.

b. Detect Command Words directed to this RT.

c. Receive Data Words from the Bus (one at a time) if directed to do so by the received Command Word.

d. Transmit Data Words through the Bus to the data bus (one at a time) if directed to do so by the received Command Word.

e. Transmit Status Words through the Bus to the data bus as directed by the received Command Word.

f. Perform Mode Operations when and as directed by received Command Words.

g. Distribute received Data Words to the proper channels of the proper IMs.

h. Input Data Words from the proper channels of the proper IMs for transmission to the data bus.

i. Maintain the Status Word and the Built-In-Test (BIT) Word of the RT by performing continuous and periodic self test functions within the RT.

j. Maintain an Activity Word and Error Word for monitoring status of serial digital IM's.

k. Maintain a Last Command Register for verification of command receipt in the event of an invalid response.

l. Perform Bit and Word Masking.

3.1.1.3 Processor Control Panel (PCP)

The IDAMST Processor Control Panel is illustrated in Figure 5 and its description follows.

3.1.1.3.1 IDAMST Bus Power Switches

The function of these switches is to provide the required signal to the power control unit to turn on and off the power supplied to the multiplex elements (Remote Terminal side A and B, and the Bus Control Interface Units). These switches shall also control power to all other processor control panel functions. These switches shall be push-on, push-off, and backlighted to indicate the "on" condition.

3.1.1.3.2 Processor Power Switches

The function of these switches is to provide the control signal to the power control unit to turn on or off each processor. One switch shall be supplied for each processor. The processor "power on" signal shall also be supplied to the advisory caution panel circuitry to control the processor failure indication. The switches shall be push-on, push-off and backlighted as described below.

3.1.1.3.3 Processor Interrupt - Startup/Restart

This switch, when depressed, shall enable the startup/restart interrupt to each Processor. The processor shall enter the Startup/Loader program and perform complete system restart as defined in the System Control Procedures. This switch shall be a momentary switch and backlighted while depressed.

3.1.1.3.4 Processor Interrupt - Reconfiguration

This switch, when depressed, shall enable the reconfigure interrupt to each Processor and cause the Master Executive performing system control (either Master Executive in Master Processor or Monitor Processor) to

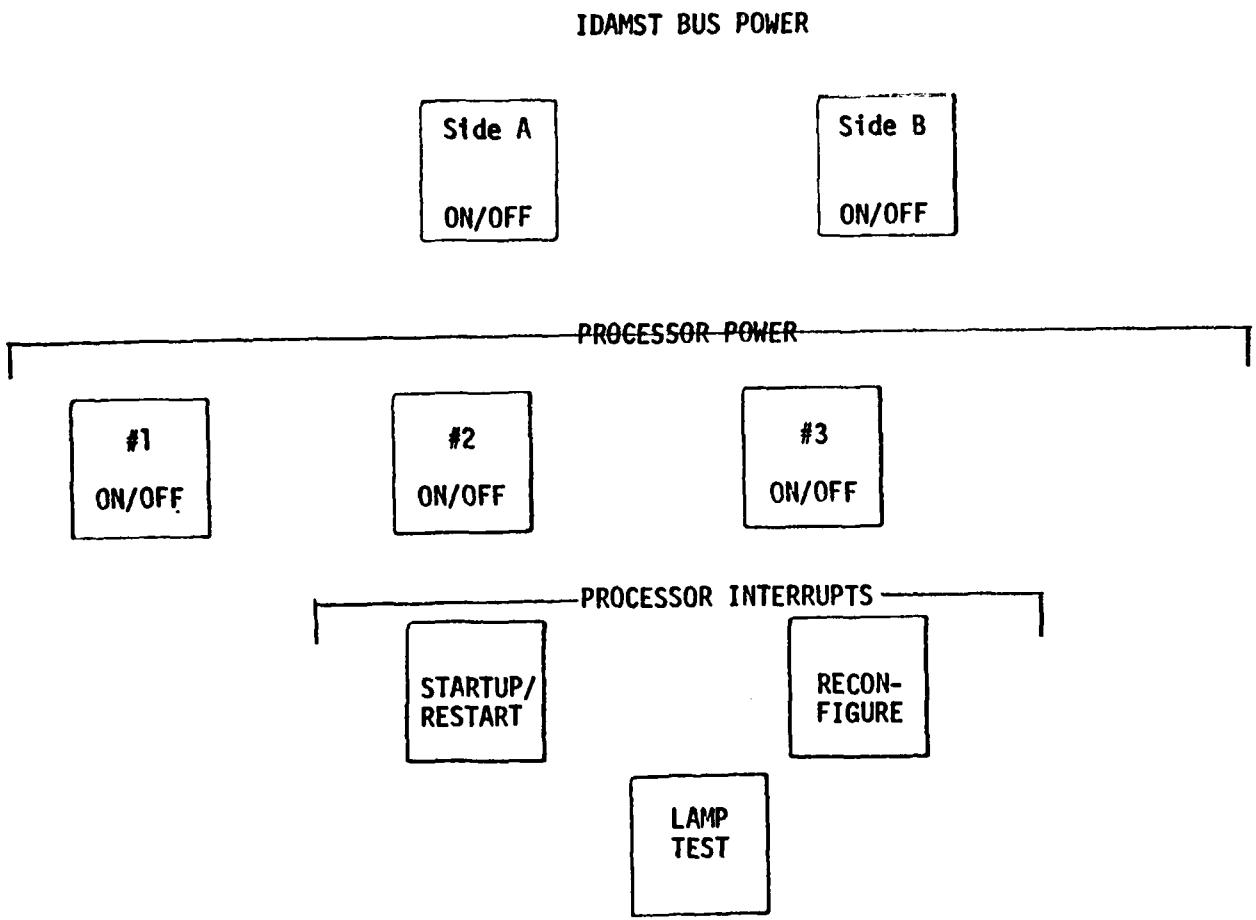


Figure 5 - IDAMST Processor Control Panel (PCP)

initiate reconfiguration. Reconfiguration is performed after one or more processors have failed; the system is in either the recovery or backup mode; and the pilot manually initiates reconfiguration.

3.1.1.3.5 Press to Test

The function of this switch shall be to test all lights on the PCP.

3.1.1.3.6 Switch Indicators

3.1.1.3.6.1 IDAMST BUS Power and Processor Interrupts

These switches shall be backlighted to indicate the "on" condition.

3.1.1.3.6.2 Processor Power

These switches shall be backlighted as follows:

a. White - Indicates the switches have been depressed

b. Green - Indicates ("GO") that power has been supplied to the processor and the "Processor GO/NO-GO" signal has been set to the "GO" state within the previous 40 msec.

c. Red - Indicates ("Fail") processor power is "on" and the absence of the "GO" signal for more than 40 msec.

3.1.2 Software Interfaces

The IDAMST Executive software interfaces with the Application software developed for IDAMST and a pre-processor software system (PALEFAC) that allocates and initializes the executive tables.

The IDAMST Application Software has been defined to consist of Events, Tasks, Comsubs, Compool Blocks and Real-Time Pseudo Statements.

Tasks and Comsubs are processing modules containing executable code and local data. Compool Blocks and data modules used for communication between tasks. Events are boolean values used for control interactions between tasks.

Real-Time Pseudo Statements are the means through which the Application Software will communicate with the Executive.

3.1.2.1 Events

Events are used for control communication between tasks. An event has two possible values: on and off.

Any Task may have an associated Task Activation Event. Such an Event is set on when the Task is Activated and set off when the Task returns to Inactive or Uninvoked state. The Activation Event associated with a Task must have the same name as the Task.

Any Compool Block may have an associated Compool Update Event. Such an Event is set on when the Compool Block is updated, either by a Task or an RT. The Update Event associated with a Compool Block must have the same name as the Compool Block.

Minor Cycle Events are set on by the Executive according to specified rates and phases. They may only be referenced in Event Condition Sets.

3.1.2.2 Tasks

Tasks are the principal processing module within the IDAMST Application software. All tasks have been defined to exist in a given "state" at a given time. These "states" are illustrated in Figure 6. The "states" of tasks are controlled by the application software through the Real-Time Pseudo Statements (see Paragraph 3.1.2.5). A task shall become "invoked" only after being scheduled by another task, otherwise it shall remain uninvoked.

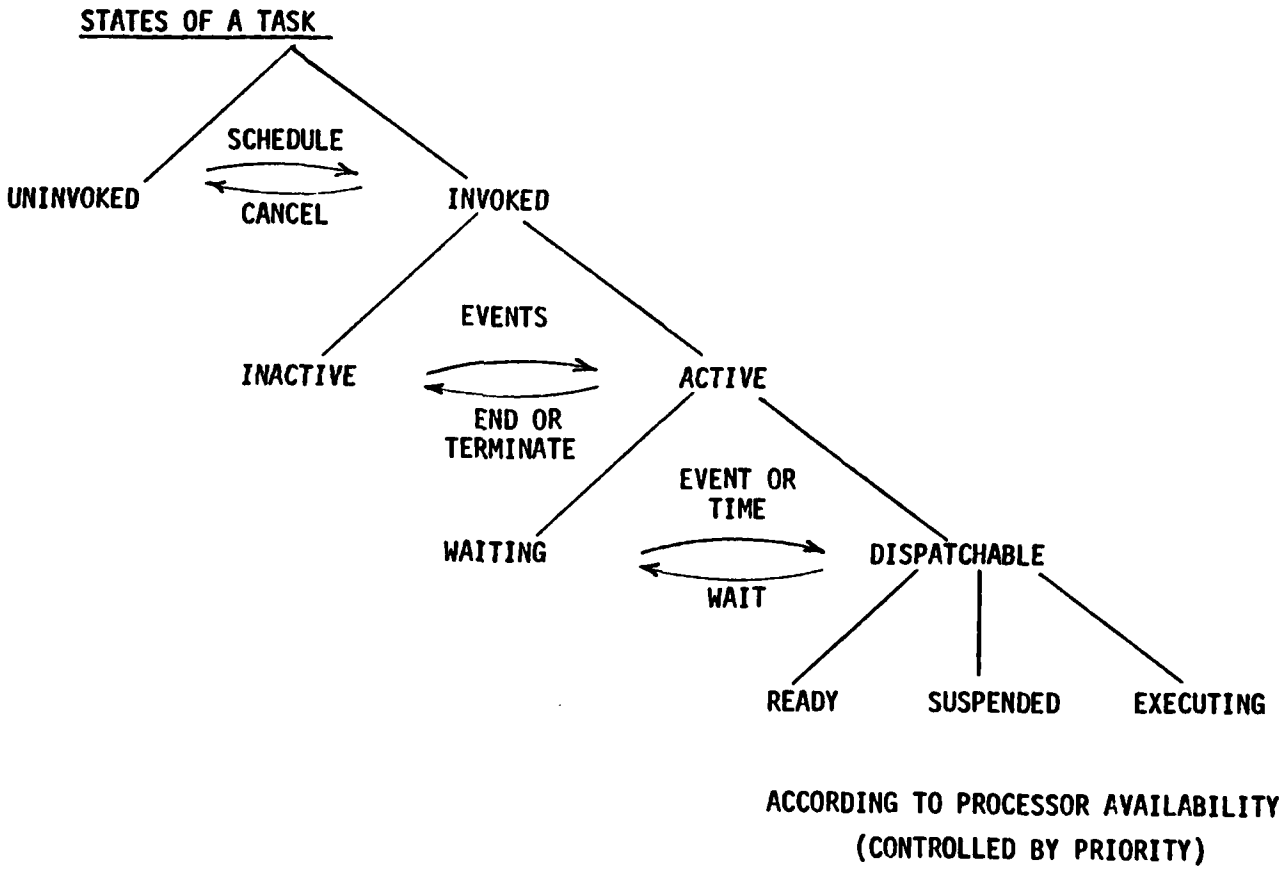


Figure 6 - States of a Task

Immediately after being scheduled, a Task is Inactive; however, it has the potential to become Active, depending upon its Event Condition Set. The Event Condition Set is a collection of Conditions, each of which may be either "on" or "off." Each Condition has a "desired" value. When all the conditions in the Event Condition Set have their desired values, if the Task is Inactive, the Executive will put it into Active state. A Task may have a null Event Condition Set, in which case it can only be Inactive momentarily.

Each Condition in an Event Condition Set is associated with a set of Events. When any of these Events is set on, the Condition is set on; when any of these Events is set off, the Condition is set off. One Event may be associated with more than one Condition in an Event Condition Set. In addition, one Condition may be associated with a "Minor Cycle Event." These are Executive-generated Events which are set "on" at certain specified times and are otherwise inaccessible to the Application Software. If a Condition is associated with a Minor Cycle Event, it may not be associated with any other Event.

A Condition may be either Latched or Unlatched. A Condition associated with a Minor Cycle Event must be Unlatched. The sole difference between a Latched and an Unlatched Condition is that upon the Scheduling or Activation of a Task, the Unlatched Conditions are set to the undesired value. Thus, a Task can only be Activated by an Unlatched Condition when the value of that condition is changed to the desired value subsequent to the last Scheduling or Activation of the Task. By contrast, Latched Conditions are changed only when one of their associated Events is changed. Therefore, a Task with only Latched Conditions in its Condition Set will be immediately Activated after it is Scheduled if all the Conditions were satisfied before the Schedule Statement.

A Task may return from Active to Inactive state from two causes: either because it completes execution, or because it is forcibly Terminated by another Task. In either case, immediately after it returns to Inactive state, the Event Condition Set is evaluated, and if all the Condi-

tions have their desired values, the Task is immediately re-activated.

When a Task is Activated, it is immediately put into Dispatchable state. If, at any point during its execution, a Task executes a Wait Statement, the Executive will place it into Wait state until the specified condition is satisfied, upon which the Task will again become Dispatchable.

All Dispatchable Tasks should theoretically be executed immediately. However, since there may be more than one Dispatchable Task at any time within any one of the DAIS Processors, Tasks are ordered by Priority to resolve possible conflicts. Whenever the Executive in any Processor is not called upon for immediate action, it selects the highest Priority Dispatchable Task, and causes the Processor to execute it.

Some tasks are declared "Privileged Tasks" and are considered to have the highest priority. Thus, as soon as they are scheduled and their Event Condition Set is satisfied, they immediately become executable with the highest priority.

If a Task is Active but has not yet been executed, it is said to be Ready. If it has been in the process of execution, but has been interrupted by a higher priority Task, it is said to be Suspended. If it is executing, it is said to be Executing.

Any given Task may only be Scheduled by one Task, which is called its Controller. Two Tasks with a common Controller are said to be "siblings." The Tasks Scheduled by any Task are said to be its "sons." If a Task has no sons, it is said to have no "descendents;" otherwise, its descendents are its sons and all the descendents of its sons.

Only a Task's Controller may Cancel or Terminate it; however, when a Task is Cancelled or Terminated, all of its descendents are Cancelled or Terminated. If a Task attempts to Cancel or Terminate itself, it will Cancel or Terminate all of its descendents, but will leave its own state unchanged.

3.1.2.3 Comsubs

In addition to Tasks, the IDAMST Application Software may include another kind of processing module, known as the "Comsub." A Comsub is a Jovial J73/I based procedure declared external to any Tasks. A Comsub may be called from many Tasks; there is a copy of each Comsub in any processor containing a Task from which the Comsub may be called.

A Comsub communicates with a Task which calls it only through its parameters and/or function result. No Comsub may execute any Real-Time Pseudo-Statements; however, one Comsub may call another.

When a Task calls a Comsub, the Task is considered to be executing within the code of the Comsub. Thus, it is possible for one Task to be suspended within the code of a Comsub at the same time that another Task is executing within the same Comsub. In other words, a Comsub must be reentrant. To implement this, every Task has a Comsub Local Storage Area assigned by PALEFAC for storage of local data by the Comsubs which it calls. At any time, there is a Comsub Stack Pointer which points to the area available for storage to the next called Comsub. This Comsub Stack Pointer is considered to be part of the process state of the Task, and is saved upon the occurrence of an Interrupt.

3.1.2.4 Compool Blocks

All communication of data between Tasks or between Tasks and the external environment (RT's) is done by means of "Compool Blocks."

No Task may directly access a Compool Block; instead, a Task references a "Local Copy" which has size and attributes identical to the Compool Block. A Task may copy the Compool Block into its Local Copy by a READ Statement, or copy the Local Copy into the Compool Block by a WRITE statement. From the point of view of the Application Software, READS, WRITEs, occur instantaneously, so a Compool Block can never be read when it has been partially updated by a WRITE.

Compool Blocks are divided into three classes: Input, Output, and Inter-task. Input Compool Blocks can only be accessed by Tasks in a READ statement. Their values are determined by RT's. Output Compool Blocks can only be accessed by Tasks in a WRITE statement; their values are "received" only by RT's. Intertask Compool Blocks are used solely for communication between Tasks.

Since a Compool Block may be accessed in more than one processor and also, possibly, in an RT, Compool Blocks may exist in multiple copies. Any processor in which a Compool Block is read has a Physical Copy of the Block; any RT which references the Block, or any processor which only WRITEs the Compool Block, is considered to have a Virtual Copy of the Block. To maintain consistency between the various copies of a Compool Block, the Executive must send Compool Update Messages across the Data Bus. Compool Blocks are further classified according to when these Update Messages are sent as: Synchronous, and Asynchronous.

Synchronous Compool Blocks are updated from a single authoritative Copy, whether in a processor or an RT, at a specified rate and phase. All copies of an Asynchronous Compool Block are updated when any of those copies is changed, either by the hardware of an RT or by a WRITE statement within a processor.

3.1.2.5 Real-Time Pseudo Statements

The application software requests services from the Executive system through Real-Time Pseudo Statements. These pseudo-statements are interpreted by the Executive software into calls to different functions as explained in Section 3.2.1.3. The statements implemented in the IDAMST system are:

a. SCHEDULE
b. CANCEL
c. TERMINATE
d. WAIT
e. SIGNAL
f. WRITE
g. READ

3.1.2.6 Palefac

The PALEFAC software system is responsible for the allocation and initialization of the Executive Tables driving the IDAMST Executive software. These tables describe the attributes and inter-relations of the various components of the application software, i.e., tasks, events, compool blocks, comsub.

PALEFAC shall generate two types of executive tables: Local Executive Tables and Master Executive Tables. The Local Executive Tables are used for control of Tasks, Events, Compool Blocks and Comsub. The Master Executive Tables are used by the Master Executive to control the operations of the master processor and its interface with the master BCIU.

These tables have been described extensively in Section 3.2 and have been identified with the particular function involved with the data.

3.2 Detailed Functional Requirements

3.2.1 IDAMST Local Executive Functions

The IDAMST local Executive shall reside in each processor within the IDAMST federated system and it shall be functionally identical in each processor.

The functions of the Local Executive shall be to:

1. Perform services as requested by the application software. These services are:

   a. Event signalling
   b. Task scheduling
   c. Task termination
   d. Task cancellation
   e. Wait
   f. Compool Read/Write

2. Perform event handling, and task checking and dispatching.

3. Control the interface with the BCIU with regard to Local Executive Communications.

4. Respond to minor cycle interrupt and participate in the transmission and reception of synchronous messages.

5. Participate in asynchronous message transmission and reception.

Figure 7 illustrates the functions interrelationship.

Figure 8 illustrates a top level functional flow as exercised by the IDAMST Local Executive.

3.2.1.1 Function One - Local Executive Control

The purpose of this function is to provide single point control of the Local Executive by maintaining the proper sequencing of its functions. The Local Executive Control processes requests from the Application Software Interface and the Hardware Interface Functions.

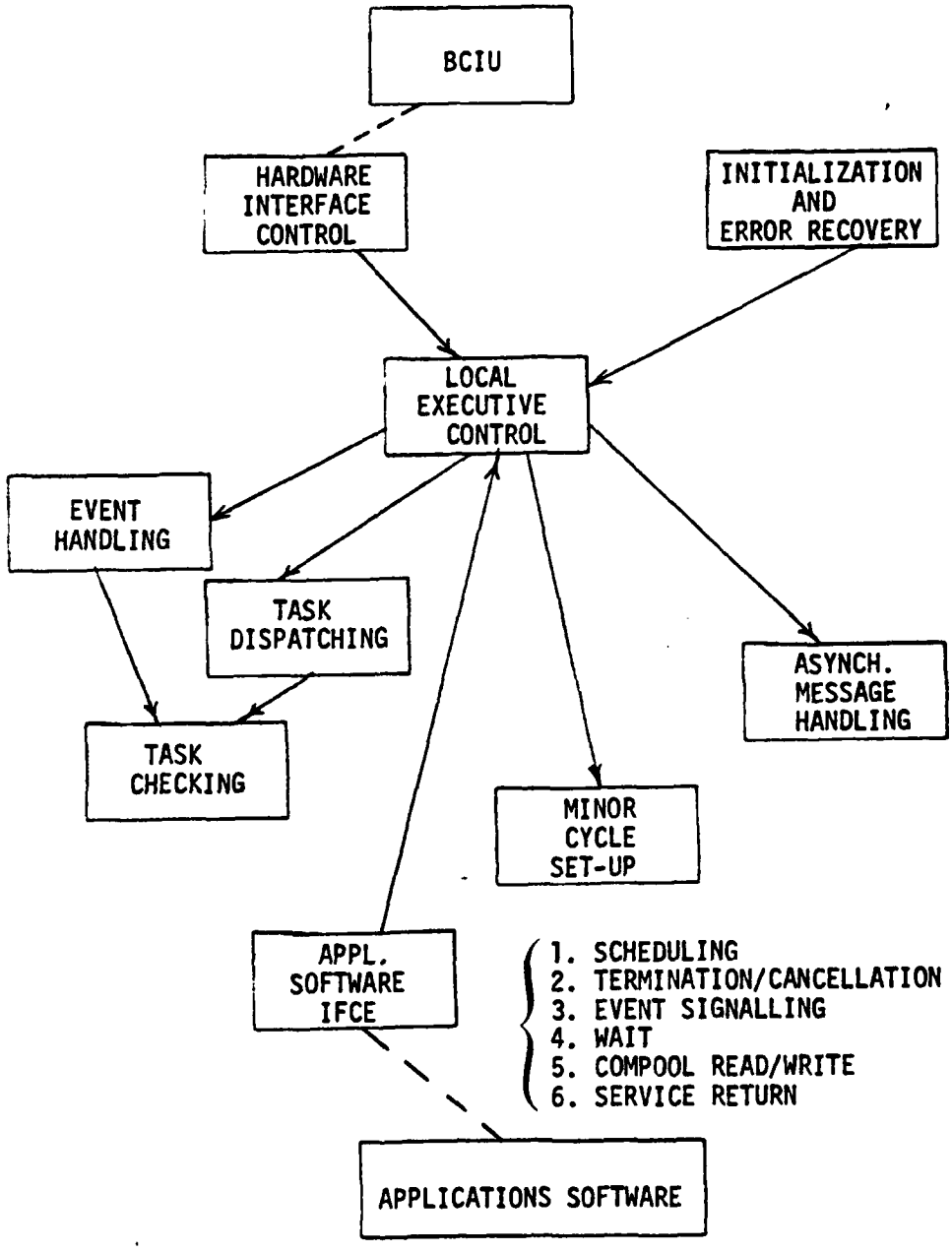
3.2.1.1.1 Inputs to the Local Executive Control Function

Inputs are listed in Table II.

In order to identify messages received from Remote Terminals, two tables of predetermined information are used. These are called the Terminal Originator Address Table (TOAD) and the Subaddress Name Keys Table (SNAKE).

3.2.1.1.1.1 Terminal Originator Address Table (TOAD)

This table, illustrated in Figure 9 shall appear once in every processor. An entry in this table is generated for each of the 32 possible Remote Terminals. Thus, the RT address is to index into this table.



MAJOR FUNCTIONS OF LOCAL EXECUTIVE

Figure 7

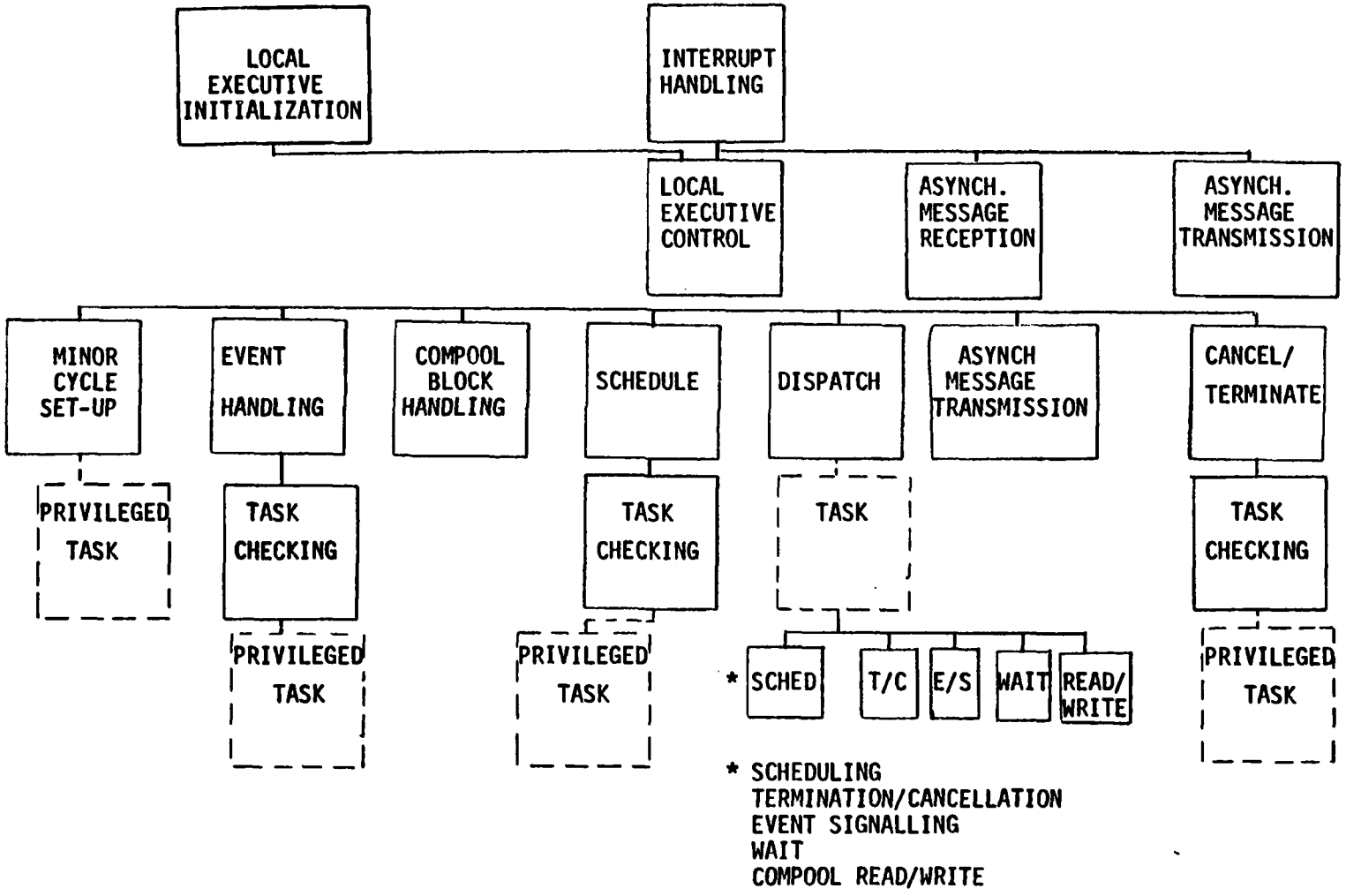


Figure 8 - LOCAL EXECUTIVE FUNCTIONAL FLOW

TABLE II - INPUTS TO THE LOCAL EXECUTIVE CONTROL FUNCTION

DATA NAME	SOURCE	REFERENCE
Minor Cycle Pending Flag	Synchronous Message Processing	
Asynchronous Reception Pending Flag	Asynchronous Message Processing	
Event Queue	Task Checking Function	
	Task Dispatching Function	
	Compool Block Handling Function	
Reception Queue		
RT Reception	Asynchronous Message Processing	
Executive Tables (TOAD, SNAKE)		

ITEM	DESCRIPTION
1	Number of Asynchronous Messages from this RT
2	Pointer to First Message described in SNAKE Table (Fig. 3.2.1.1-2)
o This table to contain one entry for each of the 32 possible Remote Terminals	

Figure 9 Terminal Origination Address Table (TOAD) Description

This table shall specify the number of synchronous messages associated with the RT and shall provide a pointer to the message descriptions located in the Subaddress Name Keys Table. If there are no messages originating from this RT, its entry in this table shall be null.

3.2.1.1.1.2 Subaddress Name Keys Table (SNAKE)

This table, illustrated in Figure 10 shall appear once in every processor. It shall contain an entry for each possible asynchronous message from the remote terminals to the processor. All messages from an RT shall be contiguous within this table and they shall be indexed by the pointer stored in TOAD (ref. para. 3.2.1.1.1.1) and identified by the subaddress accompanying the RT message.

3.2.1.1.2 Local Executive Control Processing

This function controls the sequence of operations performed by the Local Executive. These operations are a consequence of:

a. The reception and processing of a minor cycle (synchronous) interrupt.

b. A service request originating in an application or Executive Task.

c. The processing of an Asynchronous Reception.

d. The performance of System Initialization.

Figure 11 demonstrates the Local Executive Control Processing sequence.

Upon being entered, the Function will determine if a minor-cycle set-up is due. If a minor-cycle set-up is not pending the Event Queue is immediately searched. After the Local Executive Control Function dequeues an Event from the Event Queue, control is passed to the Event Handling Function.

ITEM	DESCRIPTION
1	RT sub-address originating the message
2	Pointer to Asynchronous DDB for this message (ref. para. 3.2.1.3.1)
o This table to contain an entry for each possible asynchronous message from RT's.	

Figure 10 Sub-address Name Keys Table (SNAKE)

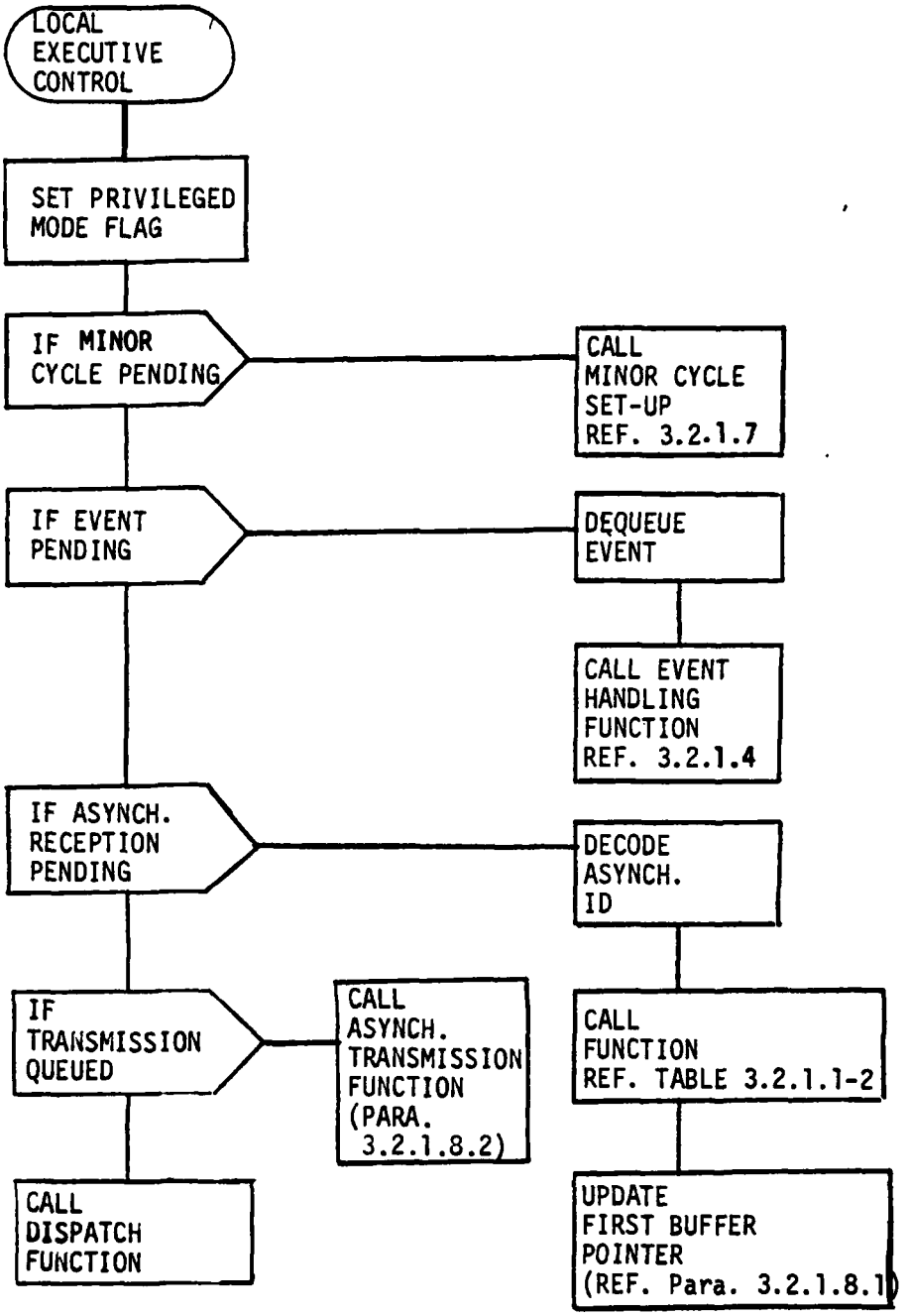


Figure 11 LOCAL EXECUTIVE CONTROL PROCESSING

If an asynchronous reception is pending in the Reception Queue, the Asynchronous ID is decoded, and the proper function is invoked accordingly. If the asynchronous message was sent by a Remote Terminal (RT), the Executive Tables associated with the Remote Terminal are examined. These tables, as described in Section 3.2.1.1.1, specify the Data Descriptor Block associated with the RT. If the source of the asynchronous message is not an RT, the type of message and the parameters to pass to the specified function are determined from the ID. The types of Asynchronous ID's, the Functions invoked to service them, and the parameters passed to those Functions are listed in Table

After the asynchronous message is processed, it is dequeued from the Reception Queue. Otherwise, if the processing had not been accomplished successfully, the ID would still be available in the Reception Queue.

Finally, the Task Dispatching Function is called.

The local executive sets the Privileged Mode flag when it is entered and resets the flag before passing control to a non-privileged task. The privileged mode flag is used to prevent local executive routines or "privileged" tasks from being re-entered before completion. If an interrupt occurs when the privileged mode flag is set, the interrupt processing routine shall return to the point of interruption when it completes execution. Similarly, when a Privileged Mode Task makes an Executive Service Request, the Local Executive shall return control directly to the task.

3.2.1.1.3 Outputs from the Local Executive Control Functions

The actual output of this Function is the initiation of the various Functions in the exercise of the sequence control. Each one of these Functions do require several inputs for their processing as listed in the description of each specific function and Table III.

TABLE III FUNCTIONS INVOKED TO SERVICE ASYNCHRONOUS RECEPTIONS

Type of Message	Function Invoked	Input Parameters
RT Transmission Compool Block Update	Compool Block Handling (Para. 3.2.1.3.2.5)	1) Internal/External Flag = External 2) DDB
Event Signal	Event Handling (Para. 3.2.1.4)	1) Internal/External Flag = External 2) Event Table Entry 3) Desired Value
Schedule Request	Schedule (Para. 3.2.1.3.2.1)	1) Task Table A Entry
Cancel Request	Cancel/Terminate (Para. 3.2.1.3.2.2)	1) Task Table A Entry 2) Cancel/Terminate Flag = Cancel
Terminate Request	Cancel/Terminate (Para. 3.2.1.3.2.2)	1) Task Table A Entry 2) Cancel/Terminate Flag = Terminate

3.2.1.2 Function Two - Hardware Interface Control Function

The Hardware Interface Control Function shall have as its prime objective the proper conduction of communication between the Local Executive and the Bus Control Interface Unit (BCIU). This communication shall be accomplished through the reading and loading of the BCIU registers.

This Function shall be responsible to process all interrupts received from the BCIU and, as a consequence, invoke the proper functions to service the interrupts. It shall accept asynchronous messages for transmission, supply the Local Executive with asynchronous messages received and accept and enqueue minor cycle numbers as a result of a synchronous interrupt reception.

Interrupts received as a result of terminal failure or Data communication error shall cause the invocation of the Error/Failure Control Function.

3.2.1.2.1 Inputs to Hardware Interface Control Function

The inputs to this Function are listed in Table IV

3.2.1.2.1.1 BCIU Registers

The BCIU registers referred to in Table IV are listed in Table I and described in Paragraph 3.1.1.1.2.

3.2.1.2.1.2 BCIU Interrupts

The BCIU activates six levels of interrupts directly related to the contents of Internal Status Register (ISR). As described in Paragraph 3.1.1.1.3, asynchronous message transmission and reception are related to Level 2 interrupts. The Master Function and the reception of synchronous transmission is related to Level 3 interrupt.

Power-On Initialization generates a Level 1 interrupt.

TABLE IV INPUTS TO HARDWARE INTERFACE CONTROL FUNCTION

DATA NAME	SOURCE	REFERENCE
BCIU Registers: Internal Status Register Master Function Register	BCIU	
INTERRUPT Signal	BCIU	

3.2.1.2.2 Hardware Interface Control Function Processing

The Hardware Interface Control Function shall be invoked upon the reception of an interrupt from the BCIU. This function shall invoke the proper Local Executive function to service the interrupt. This processing is illustrated in Figure 12.

If the "Privileged Task Mode" flag is set, the status of the processor (program counter, registers, condition status, etc.) prior to the interrupt is locally saved to allow for immediate return after the interrupt is serviced. If the "Privileged Task Mode" flag is not set, the processor status is saved in the Local Processor Tasks Table B entry for the interrupted task.

The origin of the interrupt is identified, if necessary, reading the Internal Status Register (ISR) of the BCIU. Then, the appropriate Executive Function is invoked to service the interrupt, namely, asynchronous reception, asynchronous transmission or the Error/Failure Control Function.

Upon return, the "Privileged Task Mode" flag is checked. If it is on, the processor will return to the state prior to the interrupt. Otherwise, return will be through Function 1, Local Executive Control and eventually the Dispatcher.

3.2.1.2.3 Hardware Interface Control Function Outputs

On exiting this function, the only parameter actually output by this function is the Minor-Cycle Pending Flag.

All other outputs indirectly concerned with this Function shall be output by the Functions being invoked.

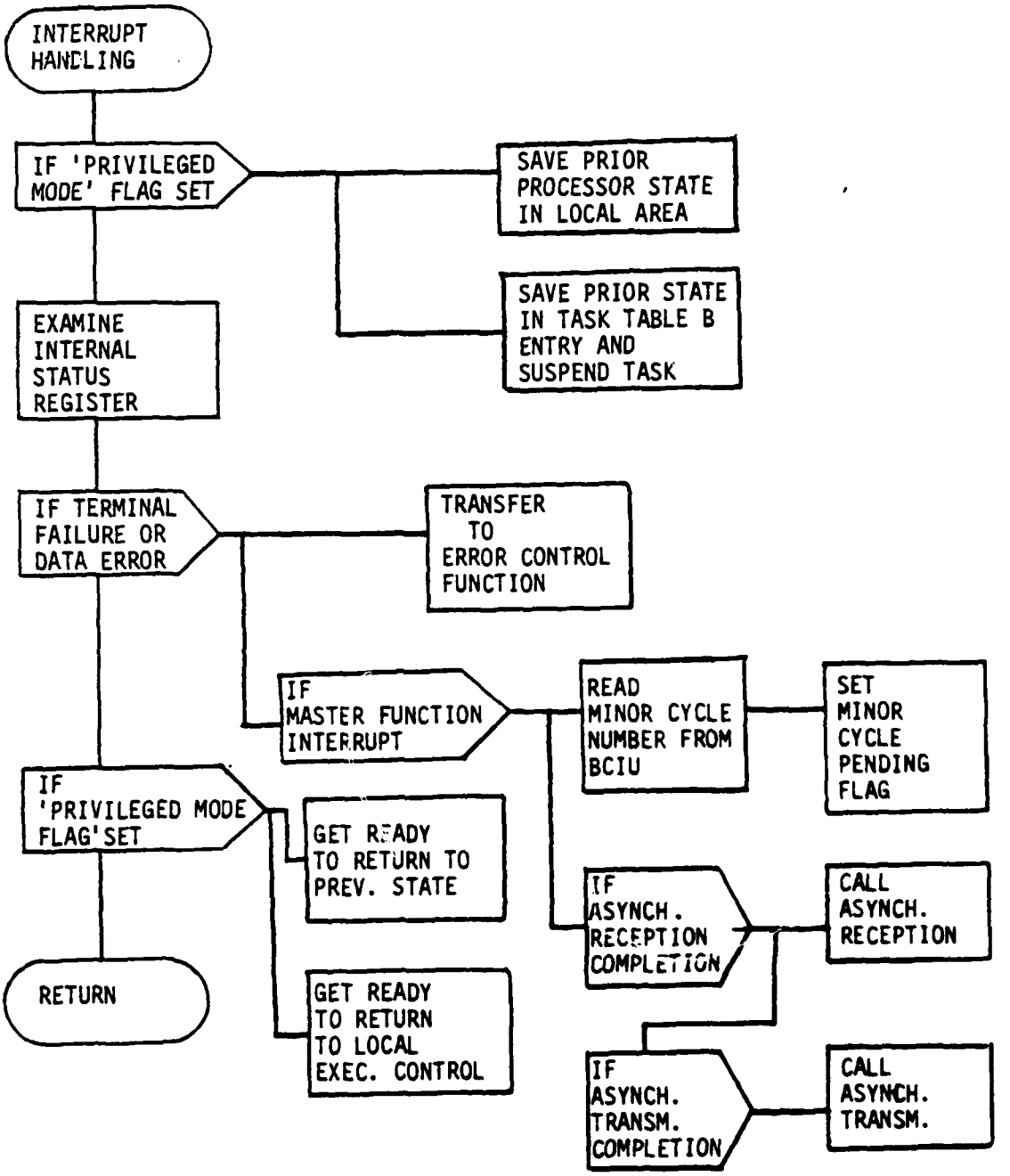


Figure 12 Interrupt Handling Processing

3.2.1.3 Function Three - Application Software Interface Control Functions

This function serves as an interface between the application software and the executive services. Each service is initiated by explicit task action through the exercise of the Real Time Pseudo-Statements. Included in the category of local executive servicessare the following:

1. Task scheduling
2. Task termination and task cancellation
3. Event signalling
4. Compool read/write
5. Wait - Absolute Time
   - Relative Time
   - Latched Event
   - Unlatched Event

Upon termination of the executive service, an Executive Service Return Function shall be activated. This function determines whether control must be relinquished to the calling task or more executive functions need to be exercised.

3.2.1.3.1 Inputs to Application Software Interface Control Function

The inputs to the local executive services function are listed in Table V. The source of these inputs is the requesting application or executive task.

Task Table A and Data Descriptor Blocks are preloaded tables. Their description follows.

Task Table A

This table is illustrated in Figure 13. This table shall appear once in every processor. It shall contain an entry for any task residing in the processor and for the controller and sons of this task, whether resident in this processor or not.

TABLE V INPUTS TO APPLICATION SOFTWARE INTERFACE CONTROL FUNCTION

ROUTINE	INPUT PARAMETERS	SOURCE
SCHEDULE	1. TASK TABLE A ENTRY OF TASK TO BE SCHEDULED	REQUESTING TASK
CANCEL	1. TASK TABLE A ENTRY OF TASK TO BE CANCELLED	
TERMINATE	1. TASK TABLE A ENTRY OF TASK BEING TERMINATED	
EVENT SIGNALLING	1. DESIRED EVENT VALUE 2. EVENT TABLE ENTRY (REF. TABLE VII)	
COMPOOL READ	1. COMPOOL BLOCK DDB ADDRESS 2. LOCAL STORAGE INTO WHICH EACH COMPOOL BLOCK IS TO BE READ	
COMPOOL WRITE	1. COMPOOL BLOCK DDB ADDRESS 2. LOCAL AREA TO BE WRITTEN FROM	
WAIT:		
ABSOLUTE TIME	1. ABSOLUTE TIME	
RELATIVE TIME	2. RELATIVE TIME	
LATCHED EVENT	1. DESIRED EVENT VALUE 2. EVENT TABLE ENTRY	
UNLATCHED EVENT	1. DESIRED EVENT VALUE 2. EVENT TABLE ENTRY (REF. TABLE VII)	

Task Table A shall be ordered according to the invocation tree, according to the following rules:

a. The controller of a task always precedes the task.

b. If Tasks A and B are siblings and A precedes B, and A is not the controller of B, then all offsprings of A precedes B.

Asynchronous Data Descriptor Block (DDB)

Every physical or virtual copy of a compool block within a processor has an associated DDB. An asynchronous DDB area shall appear once in every processor. An asynchronous DDB is generated for each compool block that is read, written or updated by a task in this processor.

Figure 14 illustrates the components of an asynchronous DDB.

Item #1: Is off because this is an Asynchronous DDB.

Item #2: Is on if there is a physical copy of this compool block within this processor. This item indicates whether Item #7 is present.

Item #3: Is on if there is an Update Event for this compool block within the processor. This item indicates whether Item #8 is present. Note that this item can be on only if Item #2 is on.

Item #4: Is on if there is a virtual copy of this compool block in an RT and this compool block is written within this processor. This item indicates whether Item #9 is present.

Item #5: Is the number of non-local physical copies of this compool block to be updated from this processor. It is zero if the compool block is not written within this processor. This item indicates the number of items of types #10 and #11.

Item #6: Is the number of words in the compool block, including the MC Tag Word.

ITEM	DESCRIPTION
1	NON-RESIDENT BIT
2	PROCESSOR NUMBER
3	INDEX TO TASK TABLE ENTRY
4	NON-RESIDENT BIT FOR CONTROLLER
5	PROCESSOR NUMBER FOR CONTROLLER
6	INDEX TO TASK TABLE ENTRY FOR CONTROLLER
7	INVOKED/UNINVOKED BIT
8	NUMBER OF DESCENDANTS

FIGURE 13 TASK TABLE A

Item #1: ON if the task is non-resident.

Item #2: Processor number where a non-resident task resides. If the task is resident to this processor, this item will be zero.

Item #3: For non-resident task, pointer to task Table A in the appropriate processor. For resident task, points to entry in Local Processor Task Table B (ref. Table X).

Item #4 - #6: Point to the controller in the same way that Items #1 -#3 point to the task. If this task is the highest task in the hierarchy tree, these items will be set to zero.

Item #7: ON if the task is uninvoked.

Item #8: The total number of descendants of this task with entries in this processor's task Table A.

ITEM	DESCRIPTION
1	SYNCH BIT
2	LOCAL COPY BIT
3	UPDATE EVENT BIT
4	REMOTE TERMINAL TRANSMIT BIT
5	NUMBER OF NON-LOCAL PHYSICAL COPIES (INDICATES NBR. OF PAIRS 10, 11)
6	NUMBER OF WORDS IN COMPOOL
7	ADDRESS OF LOCAL COPY
8	OFFSET TO UPDATE EVENT IN EVENT TABLE (TABLE VII)
9	REQUEST VECTOR FOR REMOTE TERMINAL TRANSMISSION
10 }	OFFSET TO DDB OF PHYSICAL COPY WITHIN EACH PROCESSOR
11 }	REQUEST VECTORS FOR UPDATING NON-LOCAL PHYSICAL COPIES

FIGURE 14 ASYNCHRONOUS DATA DESCRIPTOR BLOCK

Item #7: If present, is the starting address of the local physical copy of the compool block.

Item #8: If present, is an offset from the beginning of the Event Table to the entry for the Update Event associated with this compool block.

Item #9: If present, is the Request Vector for updating the virtual copy of this compool block within an RT.

Item #10: If present, are offsets from the beginning of the DDB area within each processor with a physical copy of this compool block to the DDB for that physical copy.

Item #11: If present, are the Request Vectors for sending updates for non-local physical copies of this compool block.

Synchronous DDB

Synchronous DDB's are used for Synchronous Compool Blocks. All Synchronous DDB's within a processor are contained in a single contiguous synchronous DDB area. A synchronous DDB area shall appear once in every processor.

A synchronous DDB is illustrated in Figure 15

3.2.1.3.2 *Application Software Interface Control Function Processing*

This function consists primarily of a collection of Executive Service Routines. Whenever a task requests an executive service by means of a Real Time Pseudo Statement, a call to an executive service routine is effected. The return from the service routing is done through this control function in order to determine whether return should be directly to the calling task or to satisfy any other request from the Local Executive.

ITEM	DESCRIPTION
1	Synchronous Bit
2	Number of Words in Compool Block
3	Absolute Address of Compool Block
4	Period -1
5	Phase

Item #1: On for a Synchronous DDB

Item #2: Number of Words in Data Block

Item #3: Starting Address of Compool Block

Item #4: Number of Minor Cycle between successive transmission of this block

Item #5: Phase on which the Compool Block is transmitted

FIGURE 15 SYNCHRONOUS DDB

3.2.1.3.2.1 Task Scheduling

The Task Scheduling Service shall be accomplished as illustrated in Figure 16.

3.2.1.3.2.2 Task Termination/Cancellation

This service shall forcibly terminate or cancel a task, as requested, and act upon the tasks descendants.

When a task is terminated, it is put into an Inactive State; a cancellation request will establish the task as uninvoked. A termination or cancellation request may be for self-termination or cancellation or may specify a descendant. A request on a descendant shall also affect the descendant's descendant. On the other hand, a request for self-cancellation or self-termination shall be interpreted as a request to cancel or terminate only its descendants.

The Cancellation/Termination processing is illustrated in Figure 17.

3.2.1.3.2.3 Event Signalling

The event signalling services requests to set an event state to either "1" or "0". Upon receipt of this request, the proper event information will be stored in the Event Queue. The Event Handling Function will proceed to exercise the request as described in paragraph 3.2.1.4.

3.2.1.3.2.4 Wait

Wait service is requested by a dispatchable task to place itself into a wait state until a specified time occurs or a specified Event attains a specified value.

An Absolute Time Wait places the task into Wait state until a specified absolute time. If the specified time has already occurred, this statement is a No-Op.

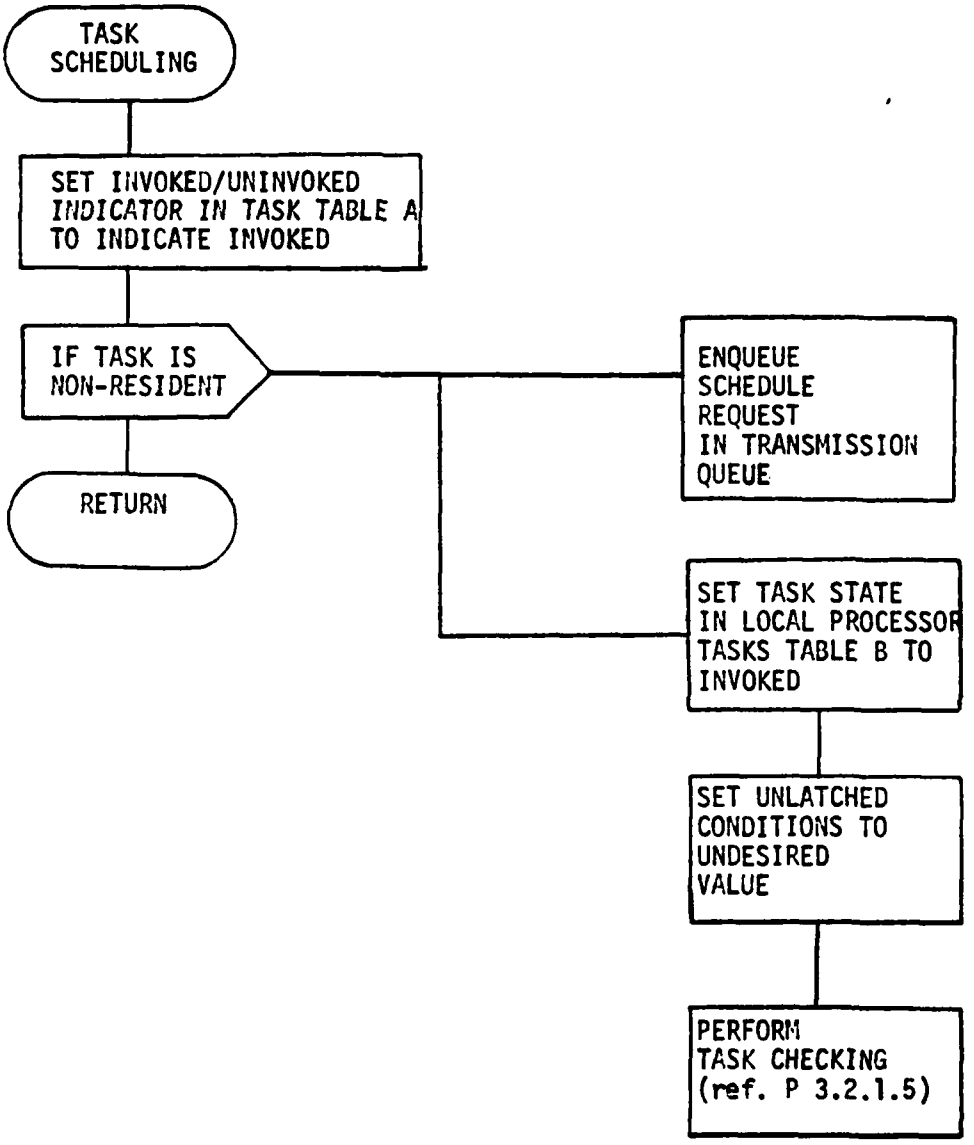


Figure 16 Task Scheduling Processing

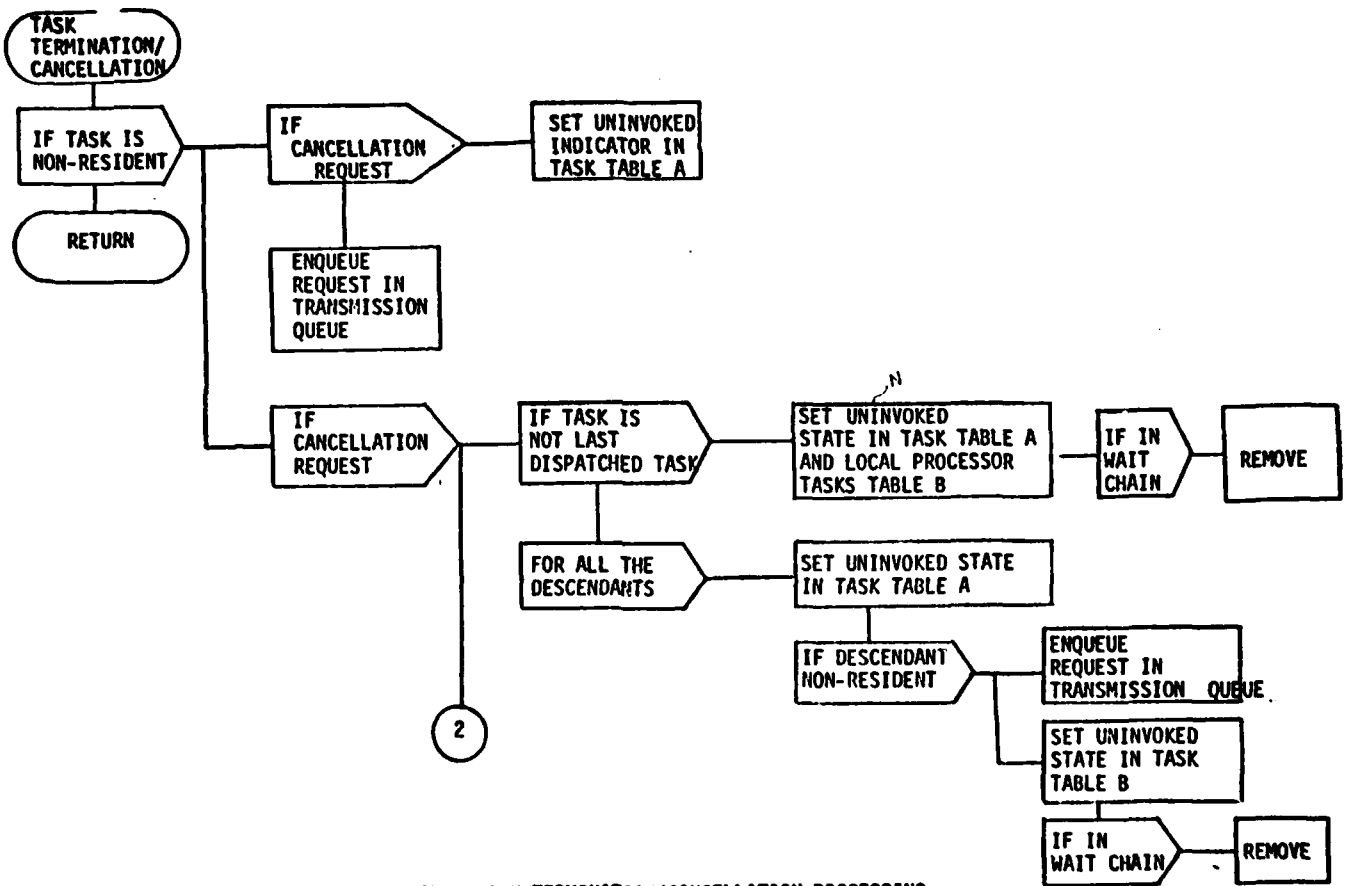


FIGURE 17. TASK TERMINATION/CANCELLATION PROCESSING

(Sheet 1 of 2)

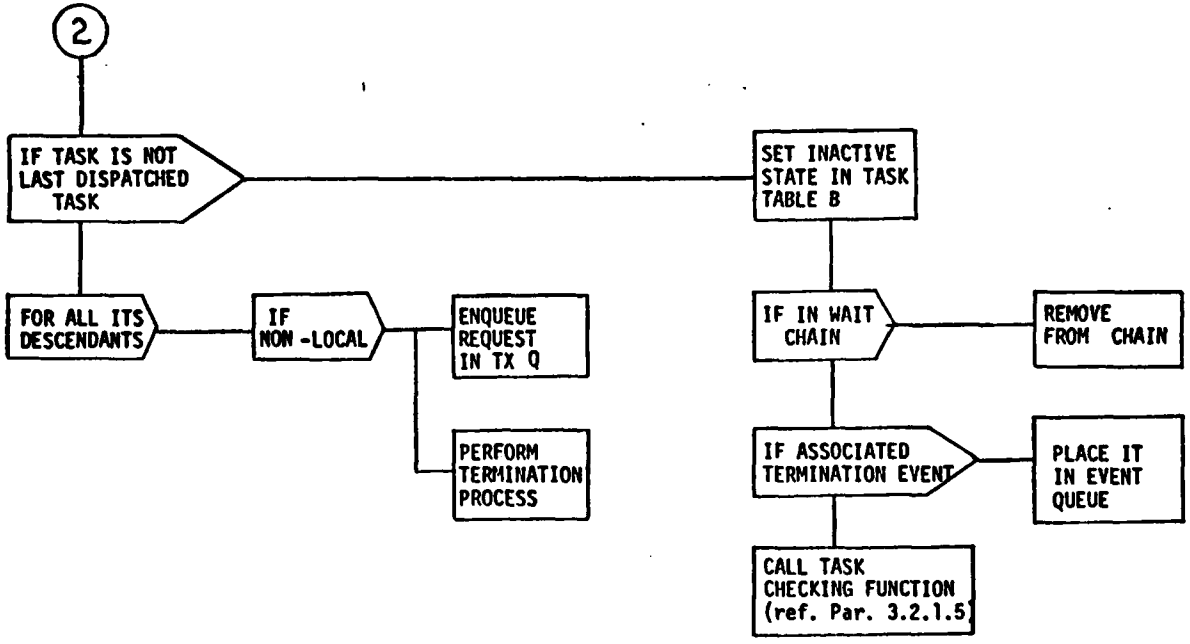


FIGURE 17. TASK TERMINATION/CANCELLATION PROCESSING

(Sheet 2 of 2)

A Relative Time Wait places the task into Wait state for a specified period of time. If the specified period is non-positive, this statement is a No-Op.

A Latched Wait places the task into Wait state until a specified Event reaches a specified "desired value." If the Event already has the desired value, this statement is a No-Op.

An Unlatched Wait places the task into Wait state until the specified Event is changed to the specified value. This statement is never a No-Op.

This service processing is illustrated in Figure 18.

3.2.1.3.2.5 Compool Read/Write

Compool blocks, classified according to when they are updated, are named Synchronous and Asynchronous.

Synchronous Data Blocks are updated from a single authoritative copy at a specified rate and phase. Asynchronous Compool Blocks are updated whenever any particular copy is changed, either by the hardware of an RT or by a WRITE statement within a processor.

The first word of each physical and virtual copy of a compool block in a processor shall consist of a "Minor Cycle Time Tag" indicating the last time the physical copy was updated. The generation of this "Time Tag" has been explained in Paragraph 3.2.1.2.

Compool Block Processing is illustrated in Figure 19.

3.2.1.3.2.6 Executive Service Return Function

This function is called by all executive service routines on their way to relinquish control. As illustrated in Figure 20, this function shall determine whether there is a need to perform additional local executive functions. These additional functions could have been as a

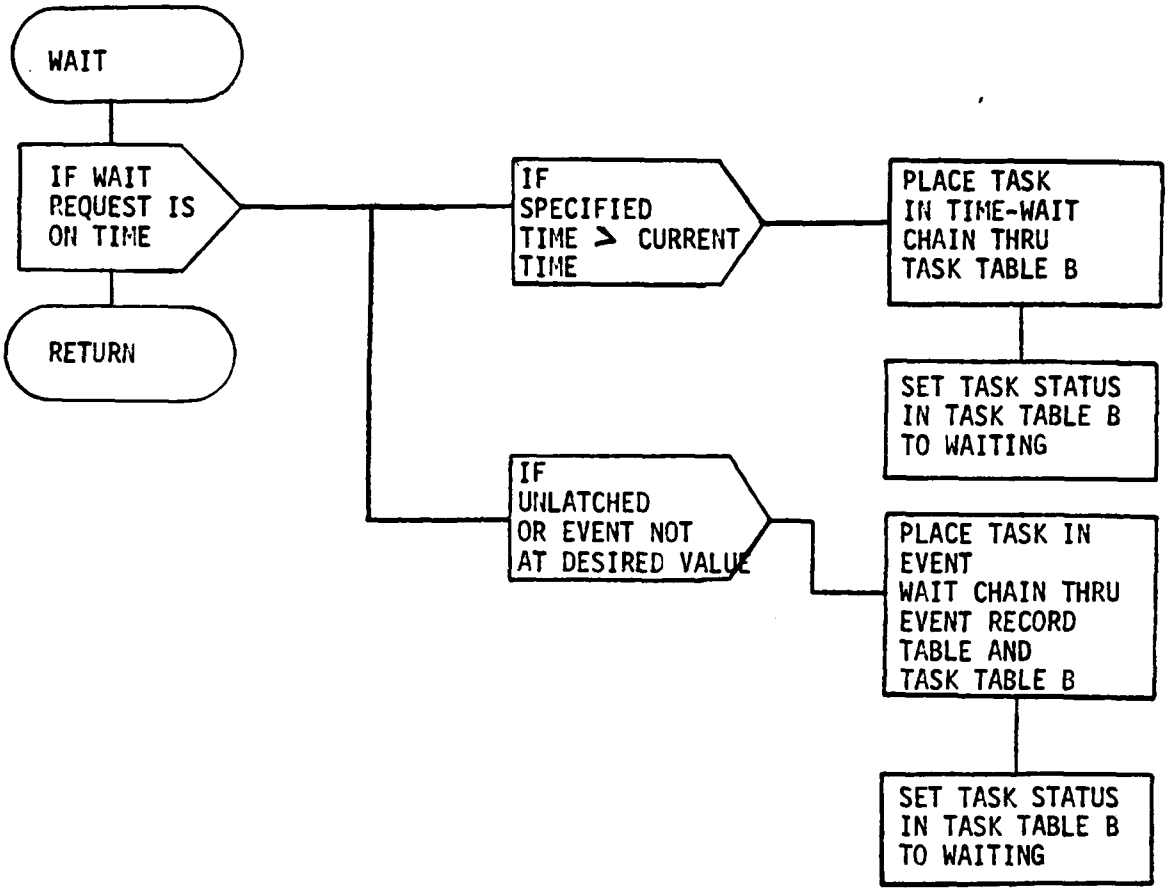


Figure 18 - WAIT PROCESSING

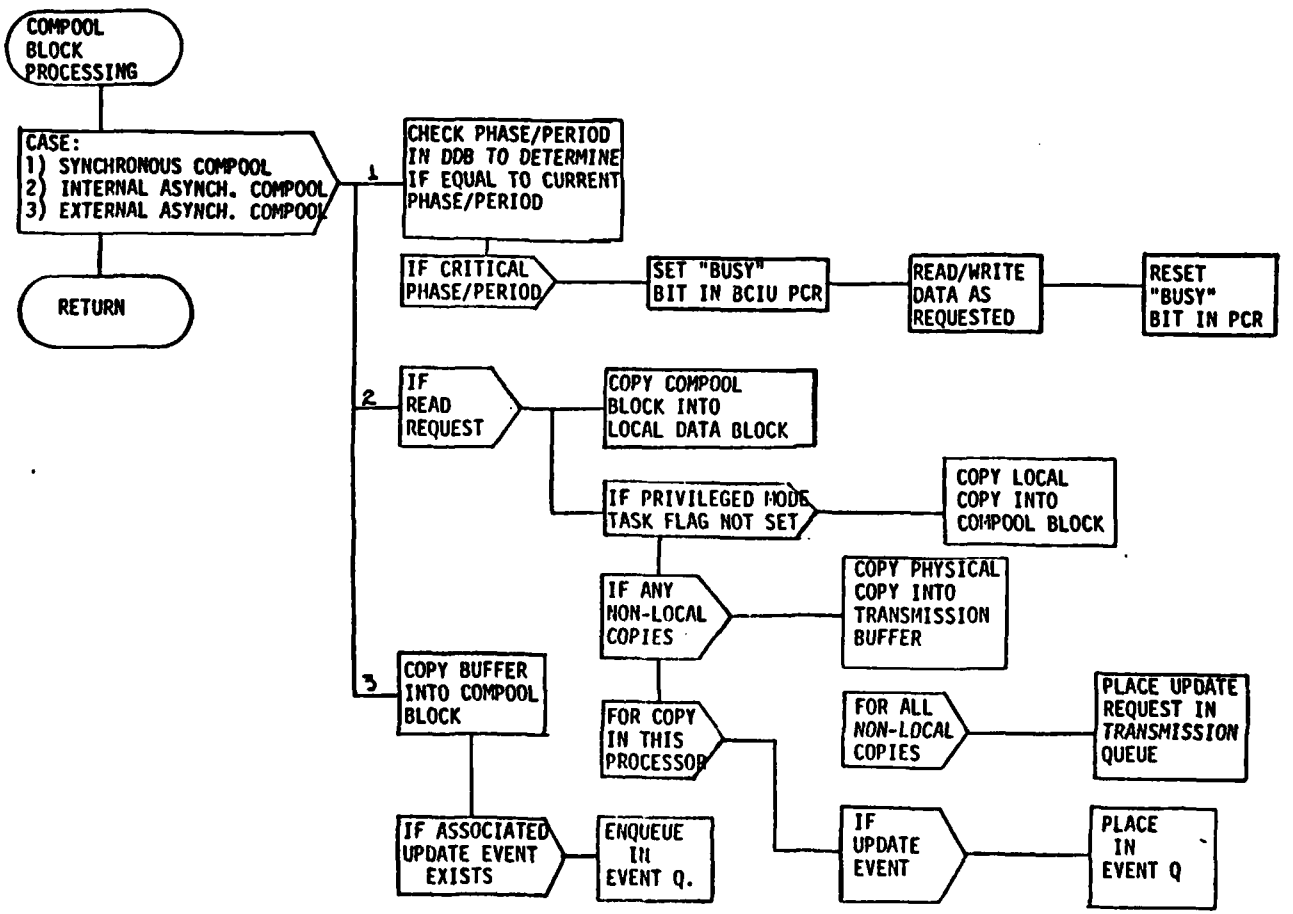


FIGURE 19. COMPOOL BLOCK PROCESSING

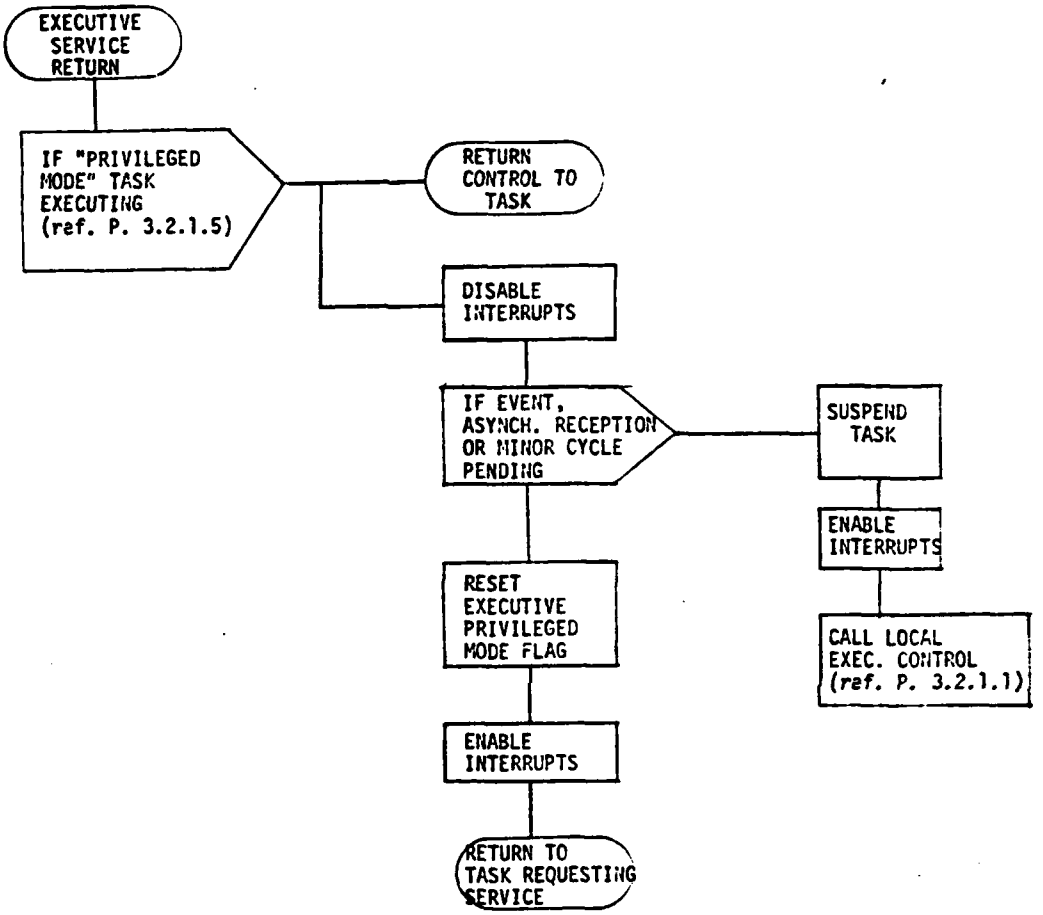


FIGURE 20: EXECUTIVE SERVICE RETURN PROCESSING

result of asynchronous or synchronous messages received while executing the executive services in a privileged mode or the executive services themselves requested further services.

The first thing that this function shall do is inspect the status of the task that requested the services. If the task was operating as a "Privileged Mode" Task, the local executive shall return immediate control to the task, rather than perform the checking mentioned above and servicing the pending requests.

3.2.1.3.3 Outputs from Application Software Interface Control Functions

The outputs received from this control function are listed in Table VI.

The Wait chain listed in this table is a result of the Wait function. It is a chain of tasks waiting for the same type of condition. There shall be one Wait chain for tasks waiting on time, one for each event waited on, and one for each event whose complement is waited on.

All tasks in same Wait chains are tied together by the forward and back pointers in their Local Processor Tasks Table B entries (ref. Par. 3.2.1.5.1). The Wait chain on time is ordered by time. The first task in a chain waiting on an event or the complement of an event is located by a pointer in the Event Record Table entry for the event (ref. Table VII).

TABLE VI OUTPUTS FROM APPLICATION SOFTWARE INTERFACE CONTROL FUNCTIONS

DATA NAME		DESTINATION	REFERENCE
Invoke Indication		Task Table A, Task Table B	3.2.1.3.2.1
Uninvoked Indication		Task Table A, Task Table B	3.2.1.3.2.2
Inactive Indication		Task Table B	3.2.1.3.2.2
Event Status		Event Record Table	3.2.1.3.2.3 3.2.1.3.2.6
Compool Data		Compool Block Compool Local Copy	3.2.1.3.2.5 3.2.1.3.2.5
Wait Chain		Task Table B	3.2.1.3.2.4 3.2.1.3.3

3.2.1.4 Function Four

The Event Handling Function is exercised from requests by tasks in the local processor or, through transmitted messages, from tasks related to a remote processor or remote terminals.

All the information concerning events is kept in an Event Record Table. Each processor contains an event record for each Event contained within the processor. The Event Handling Function shall act on Event Condition Sets as specified in the Event Record Table. This Event Record Table is illustrated in Table VII.

3.2.1.4.1 Inputs to Event Handling Function

The inputs to this function are shown in Table VIII.

3.2.1.4.2 Event Handling Processing

The IDAMST Executive System receives requests for event processing from application software tasks or executive tasks. These requests will consist of commands to set an Event to a value of "1" or "0". The processing of this function is accomplished as demonstrated in Figure 21. The processing of this function consists of setting the desired value of an Event into the Event Record Table. If the table indicates that there are copies of this event in other processors, an asynchronous message is formulated and enqueued.

The Event Record Table indicates any local tasks with the Event as part of their Event Condition Set. Thus, these conditions are set in the respective Task Table B entries.

Finally, the Task Checking Function is invoked to check the condition status of these tasks.

3.2.1.4.3 Outputs from the Event Handling Function

Outputs from this function are shown in Table IX.

TABLE VII EVENT RECORD TABLE

ITEM	DESCRIPTION	COMMENT
1	Event Value	"0" to "1"
2	Number of Non-Local Copies	Number of processors, other than this one, which references this event (1 or 2)
3	Number of Local Tasks Pointed To	Number of Tasks within this Processor that have this Event in the Event Condition Set
4	Pointer to Task Waiting on Event	Address for Task (if any) waiting on this Event
5	Pointer to Task Waiting on Complement	
6	Processor of 1st Non-Local Copy	Number of Processor with this Event in an Event Record Table
7	Offset Value into Events Table for this Event in other Processor	
6	Processor of 2nd Non-Local Copy	There will be one Set for each Copy specified in Item #2
7	Offset Value for this Event Table	
8	Bit Position in Task 1	Position within the Event Condition Set for Task Listing this Event
9	Entry # of Task Table B Entry for Task 1	Entry # in Local Processor Task Tables for the Task with this Event in its Condition Set Items 8, 9 will be repeated for each Task specified in Item #3.

TABLE VIII

INPUTS TO EVENT HANDLING FUNCTION

DATA NAME		SOURCE	REFERENCE
Event Record Table Entry		Application Tasks	
Desired Value of Event		Synch. Message Handling Asynch. Message Handling	
Event Record Table		Pre-loaded	

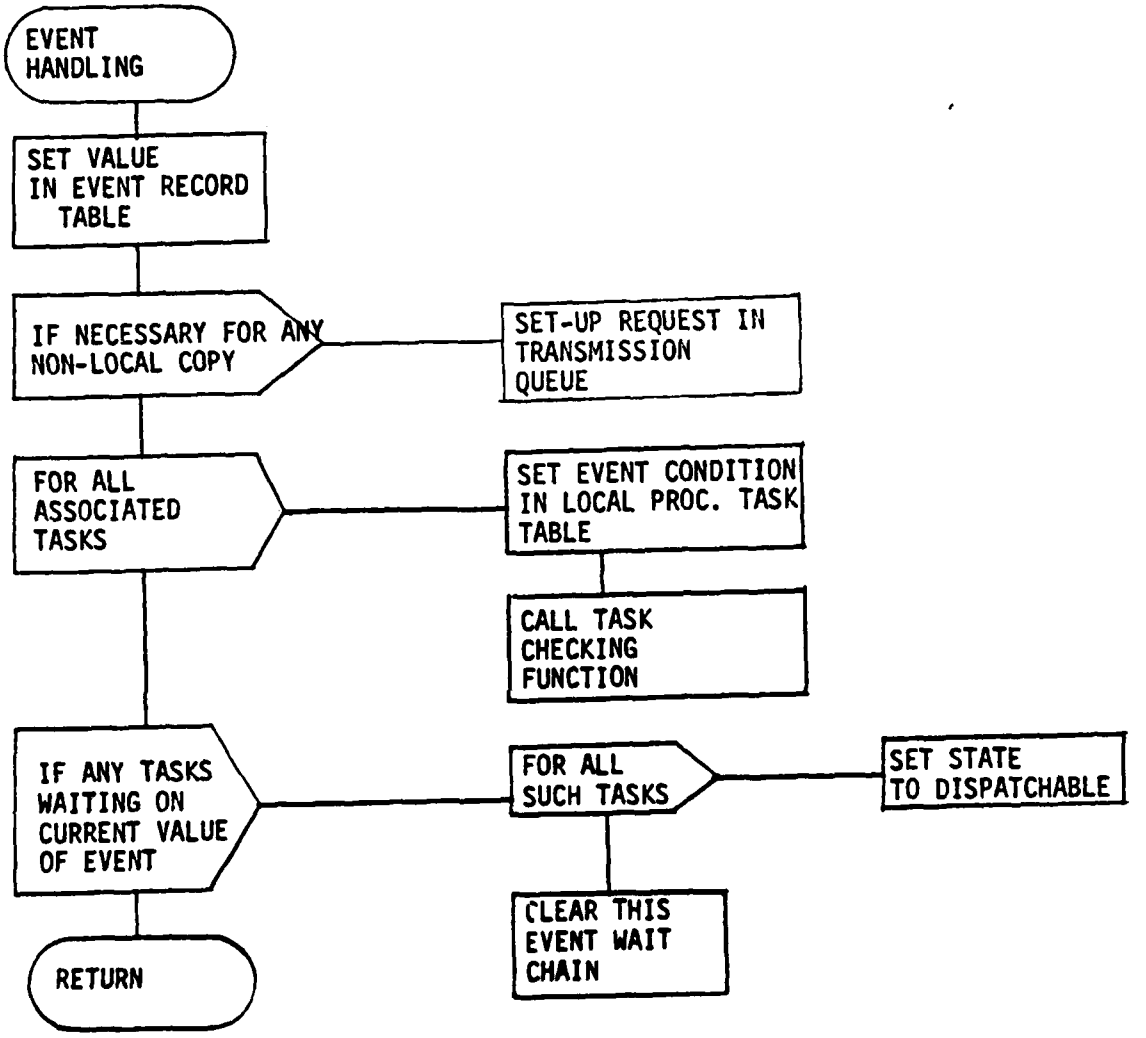


FIG. 21 EVENT HANDLING PROCESSING

TABLE IX OUTPUTS FROM EVENT HANDLING FUNCTION

DATA NAME		DESTINATION	REFERENCE
Updated Event Value		Event Record Table Task Table Record	
Transmission Queue		Asynchronous Message Handling - Local Executive	

3.2.1.5 Function Five - Task Checking - Local Executive

The purpose of this function is to determine whether a given task should change states and become Active.

3.2.1.5.1 Inputs to the Task Checking Function

Related to each task resident in a processor there exists a table internal to the Local Processor. The entries to this table, described in Table X are ordered according to the tasks priority.

The inputs to the Task Checking Functions shall be, as listed in Table XI, the entries to the tasks on the Local Processor Tasks Table B.

3.2.1.5.2 Task Checking Function Processing

This function will be invoked by the Local Executive Control:

a) Each time a task is invoked.
b) Each time an Event, except Minor Cycle Events, is signalled and posted in the Local Processor Tasks Table B (ref. Table X).
c) Whenever a Task ends.
d) Whenever a Task is forcibly terminated.

This function is processed as indicated in Figure 22. As this function is invoked, it will verify the status of the Task from the Local Processor Tasks Table. A task can be in either of two states, invoked or uninvoked, as shown in Figure 6 and explained in Section 3.2.2.2 If in an invoked state, the Task Event Condition Set is checked against the desired Event Condition Set as established in the input table. If all conditions are satisfied, the Task state is changed to Active and Dispatchable and all its Unlatched Events Conditions are complemented. If the task is indicated as a Privileged Mode task, it will be called immediately and control is transferred to the task. Either way, if the task has an Activation/Termination Event, the Event is put into the Event queue.

3.2.1.5.3 Output from Task Checking Function

The outputs from this function are listed in Table XII.

TABLE X LOCAL PROCESSOR TASKS TABLE B

ITEM	DESCRIPTION	COMMENT
1	Task Status	Uninvoked, Inactive, Waiting, Dispatchable, Subsequent (Reference See Figure 3.1) Values Indicate Priviledged Mode Task.
2	Present Event Condition Set	This Task actual Condition Set
3	Desired Event Condition Set	Condition Set to satisfy before Activation
4	Unlatched Event Mask	A "One" in any Bit Position Points to the corresponding Bit in Item #3 as an Unlatched Event
5	Starting Address of Task	First Executable Statement in Task
6	Initial Value for Comsub Stack Pointer	Address of Beginning of Comsub Stack Area for this Task
7	Save Area for Comsub Stack Pointer	Loc. to contain the current Stack Pointer if the Task is not executing
8	Back Pointer for Wait Chain	Pointers to the Time Queue or Queue of Task waiting on the same Event (Para. 3.2.1.3.2.4)
9	Forward Pointer for Wait Chain	Same comment as for Hem H8.
10	Time or Event Waited On	Parameter the Task is waiting on. If waiting on Event, the High Order Bit will be set.
11	Save Area for Regs. and P.C.	Program Counter & Regs. will be saved during an Interrupt or Wait Function

TABLE X LOCAL PROCESSOR TASKS TABLE B - Cont.

ITEM	DESCRIPTION	COMMENT
12	Restart Address	Contains the Task Starting Address on Task invocation or the Address following the most recent Wait Statement executed. After Task execution is completed successfully, its starting Address will be stored here.
13	Pointer to Activation Event	Pointer on Event Table to the associated Activation Event for this Task.

TABLE XI INPUTS TO TASK CHECKING FUNCTION

DATA NAME	SYMBOL	SOURCE	REFERENCE
Local Proc. Task Table Entries:		Local Exec. Control	
Task Status Indicator			
Present Event Condition Set		Event Handing Function	
Desired Event Set		Predetermined	
Unlatched EVent Task		Predetermined	
Starting Address of Task		Predetermined	

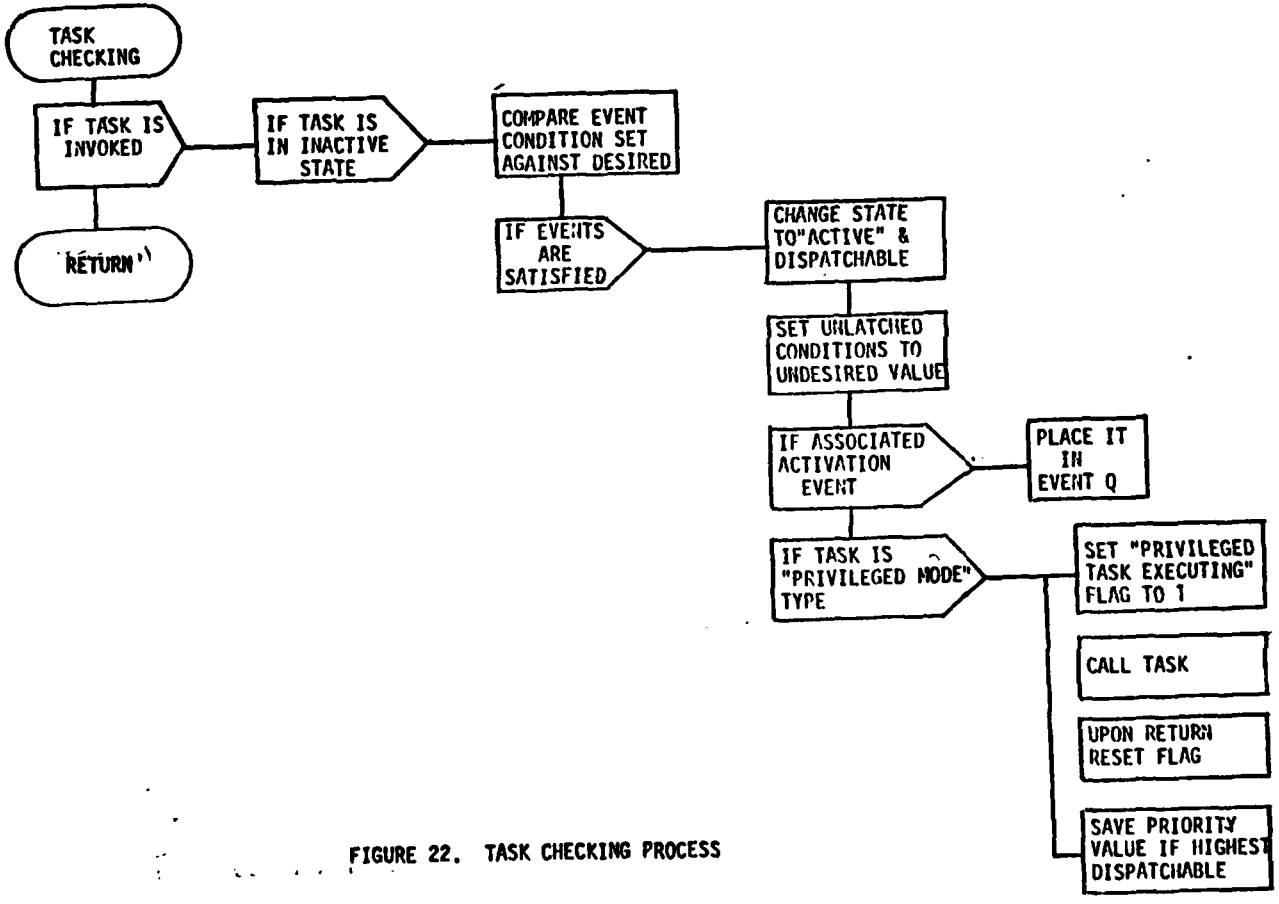


FIGURE 22. TASK CHECKING PROCESS

TABLE XII OUTPUTS FROM TASK CHECKING FUNCTIONS

DATA NAME	SYMBOL	DESTINATION	REFERENCE
Updated Entries in Local Proc. Task Table (Table X) Task Status Unlatched Event Condition Activation Event (if any) in Event Queue			

3.2.1.6 Function Six - Task Dispatching - Local Executive

The main purpose of the Task Dispatching Function is to search for the highest priority dispatchable task in the Processor. Upon finding a task, control is transferred to it.

3.2.1.6.1 Inputs to Task Dispatching Function

The inputs to this function are listed in Table XIII.

3.2.1.6.2 Task Dispatching Processing

The Task Dispatching Function is called by the Local Executive Control Function. It takes place whenever the Event Queue is empty, the Task Checking Function and all other Local Executive Tasks have been completed.

The objective is to determine the highest priority task in the local processor that is in the dispatchable state. If the task is being resumed from an interrupt point, the save area will contain the address at where the task was interrupted and the information contained in all the registers at the time of interruption. If the task is being entered at its beginning, only the task start address will be defined; therefore when initiated, tasks cannot depend upon register initialization.

The Local Executive Control Function, through the Task Checking Function, keeps track of the highest priority task made dispatchable since the last Dispatch. Thus, when the Task Dispatching Function is called, it commences scanning on the task table starting with the highest priority Dispatchable task. If the last Dispatched task is still the Highest Priority Task, then control will be returned to the original task. Otherwise, control will be given to a new task.

Upon normal termination, the dispatched task returns to the Task Dispatching Function. The dispatcher will, in turn, return to the Local Executive Control Function.

TABLE XIII INPUTS TO TASK DISPATCHING FUNCTION

DATA NAME	SYMBOL	SOURCE	REFERENCE
Local Processor Tasks Table			
Information			
Task Status		Local Executive Control	
Starting Address of Task		Predetermined	
Contents of Save Area		Interrupt Control	
Comsub Stack Pointer			
Pointer to Activation Event			
Priority of highest priority task		Task Checking Function	
made dispatchable since last Dispatch			

The processing accomplished by this Function is illustrated in Figure 23.

3.2.1.6.3 Outputs from Task Dispatching Function

The outputs from this Function are listed in Table XIV.

3.2.1.7 Function Seven - Minor Cycle Set-Up Function

The Minor Cycle Set-Up Function shall be initiated by the Local Executive Control Function upon detection of a pending Minor Cycle. This function performs the processing necessary to prepare for synchronous message transmission and reception each minor cycle.

3.2.1.7.1 Inputs to the Minor Cycle Set-Up Function

The inputs to this function are listed in Table XV and described in the following paragraphs.

The Synchronous I/O Tables are used to determine which DMA Pointers to set up in the appropriate DMA Pointer Block on any given Minor Cycle.

a) Synchronous Pointer Table (SYNPTR)

The SYNPTR Table will appear once in every processor that has any dynamically maintained synchronous pointers in the DMA Pointer Block (ref. para. 3.2.1.7.3). It contains two blocks of pointers for each Minor Cycle. One contains the addresses of all Compool blocks received during the Minor Cycle, excluding the Compool blocks whose addresses are fixed within the appropriate DMA Pointer Block. The other contains the addresses of all Compool blocks transmitted during the Minor Cycle but not fixed within the appropriate DMA Pointer Block. If for a given Minor Cycle, all the DMA Pointers within the DMA Pointer Block are of a fixed nature, then the appropriate pointer in SYNPTR for this Minor Cycle shall be null.

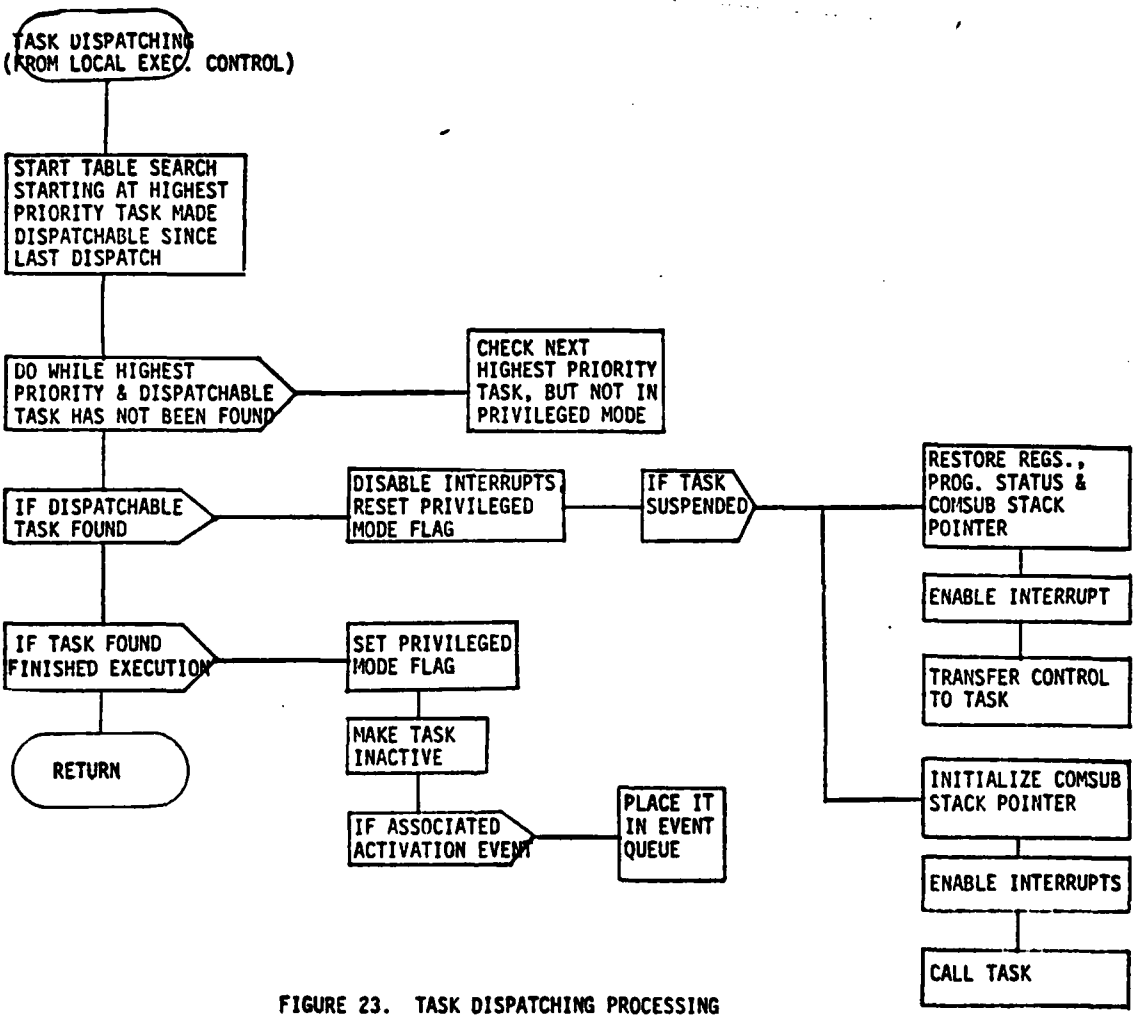


FIGURE 23. TASK DISPATCHING PROCESSING

Table XIV OUTPUTS FROM TASK DISPATCHING FUNCTION

DATA NAME	SYMBOL	DESTINATION	REFERENCE
Local Processor Tasks Table:			
Task Status		Task Checking Function	
Event Queue		Event Handling Function	

TABLE XV INPUTS TO MINOR CYCLE SET-UP FUNCTION

DATA NAME	SYMBOL	SOURCE	REFERENCE
MINOR CYCLE NUMBER			
SYNCHRONOUS I/O TABLES:		PRE-LOADED	
SYNCHRONOUS POINTER TABLE (SYNPTR)			
SYNPTR INDEX TABLE			
POINTER BLOCK DESCRIPTOR			
MINOR CYCLE EVENT GENERATION TABLE			
TIME-WAITING CHAIN OF TASKS			

b) SYNPTR Index Table

The SYNPTR Index Table is used to locate the blocks of Receive and Transmit Pointers within SYNPTR for any Minor Cycle. This table shall appear once in every processor that has dynamically maintained pointers in the DMA Pointer Block and will contain one entry for every minor cycle number. The items in this table are described in Table XVI below.

TABLE XVI SYNCHRONOUS POINTER INDEX TABLE

ITEM NO.	DESCRIPTION
1	Number of variable Synchronous Receive Pointers for this Minor Cycle within SYNPTR.
2	Offset from the beginning SYNPTR to first Receive Pointer in SYNPTR for this Minor Cycle.
3	Number of variable Synchronous Transmit Pointers for this Minor Cycle within SYNPTR.
4	Offset its first Transmit Pointer in SYNPTR for this Minor Cycle.

c) Pointer Block Descriptor

The Pointer Block Descriptor shall appear once in every processor. It is used by the Minor Cycle Set-Up Function to determine which parts of the DMA Pointer Blocks are fixed and which are variable.

It contains four words as shown in Table XVII. They are pointers to the first nonfixed pointer in the respective block of the DMA Pointer Block.

TABLE XVII POINTER BLOCK DESCRIPTOR TABLE

ITEM NO.	DESCRIPTION
1	Address of first variable Receive Pointer in Block 0
2	Address of first variable Transmit Pointer in Block 0
3	Address of first variable Receive Pointer in Block 1
4	Address of first variable Transmit Pointer in Block 1

Minor Cycle Event Generation Table

The Minor Cycle Event Generation Table shall appear once in every processor in the IDAMST system. This table shall be used by the Local Executive to determine which Minor Cycle Event to signal on any given Minor Cycle.

The Minor-Cycle Event Generation Table consists of two interdependent parts. The first part shall contain an entry for every Minor Cycle in a major frame ordered by Minor Cycle number. Minor Cycles shall be numbered in ascending order. Each entry shall contain two items. One entry shall indicate the number of Events for the Minor Cycle. The second entry is an index field containing an offset to the beginning of the second part of the table. This offset points to the beginning of a list of Event Table pointers which point to the appropriate period - phase events in the Event Record Table (Table VII). If there are no Minor Cycle events for a given Minor Cycle, the proper entries in the first part of the table shall be set to zero.

The actual arrangement of Event Record Table pointers may vary within the second part of the Minor-Cycle Event Generation Table. Many distinct Minor Cycles can cause activation of an identical list of Minor Cycle Events, thus the groups pointed to by Part 1 of this table for several different Minor Cycles may be the same.

An illustration of the Minor-Cycle Event Generation Table is shown in Table XVIII.

TABLE XVIII MINOR CYCLE EVENT GENERATION TABLE

PART 1	Number of Events for Minor Cycle 0	Offset to First Event for this Minor Cycle in Part 2
	Number of Events for Minor Cycle 1	Offset for this Minor Cycle in Part 2
	Number of Events Last Minor Cycle	Offset in Part 2
	Blank	Offset into Event Record Table
PART 2		
	Blank	Offset into Event Record Table

3.2.1.7.2 Minor Cycle Set-Up Processing

The main objective of this Function is to generate the address pointers used by the Bus Controller Interface Unit (BCIU) to store or access, via the DMA channel, synchronous message data received or synchronous message to be transmitted. These pointers shall be stored in a table designated as DMA Pointers Table.

This Function also determines which tasks depend on the Minor Cycle to become dispatchable. Those tasks that have their Event Condition Set satisfied by this Minor-Cycle and are "Privileged Mode" tasks, shall become Active, Dispatchable and directly executable by the Minor-Cycle Set-Up Function. This action takes advantage of its knowledge of the nature of the Events it is signalling to bypass the Event Handling Function (3.2.1.4) and the Task Checking Function (3.2.1.5) of the Local Executive.

The processing performed by this Function is illustrated in Figure 24.

In the process of setting up the DMA Pointer Table, this Function makes use of the Minor Cycle number to index into the SYNPTR Index Table. With the information obtained from this table, the SYNPTR for the particular Minor Cycle will be read and the addresses of all Compool blocks to be received and transmitted during the Minor Cycle will be fetched. Making use of the information stored in the Pointer Block Descriptor Table, the Compool blocks addresses shall be stored in the DMA Pointer Tables. The DMA Pointer Tables are described in Paragraph 3.2.1.7.3..

3.2.1.7.3 Outputs from the Minor Cycle Set-Up Function

The items output from this Function are summarized in Table XIX.

The DMA Pointers Blocks Table shall contain the address pointers for the Compool blocks into which receive messages are to be stored by the BCIU and out of which the BCIU will access messages for transmission during a Minor Cycle. This table, as shown in Table XVI is divided into four parts. Two parts are used for even-numbered Minor Cycles and the other two for odd-numbered Minor Cycles.

Part 1 shall contain pointers for message reception while Part 2 shall contain pointers for message transmission. Each part shall contain up to 31 pointers that are accessed by the BCIU as described in Section 3.2.1.2. Word 0 in each block shall not be used as a pointer since a sub-

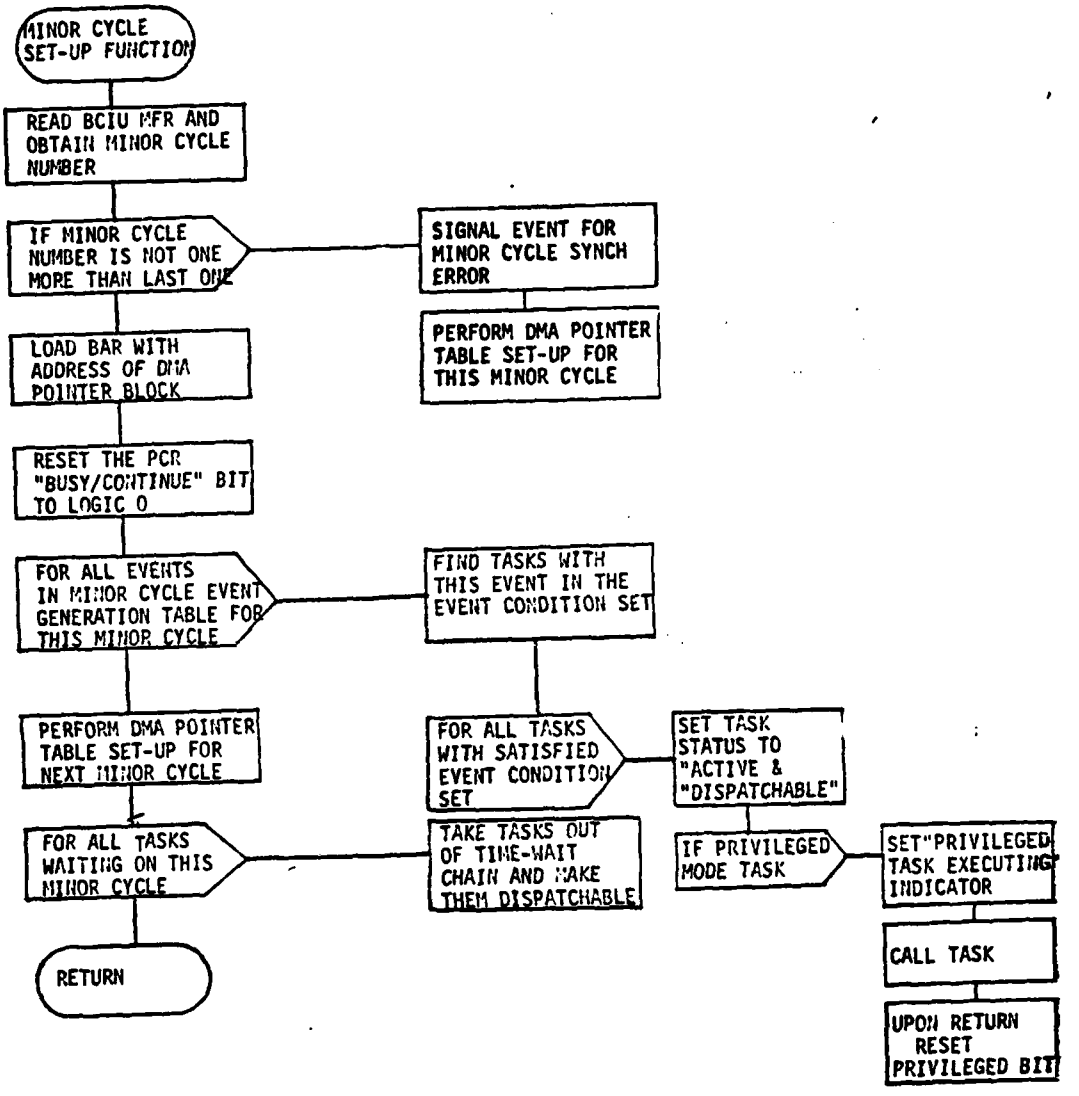


FIGURE 24. PROCESSING OF MINOR CYCLE SET-UP FUNCTION

TABLE XIX OUTPUTS FROM MINOR CYCLE SET-UP FUNCTION

DATA NAME	SYMBOL	DESTINATION	REFERENCE
MINOR CYCLE NUMBER			
DMA POINTERS BLOCKS TABLE		BCIU	
BASE ADDRESS REGISTER (BAR)		BCIU	
RESET OF BCIU PCR BUSY BIT			
EVENT QUEUE		EVENT HANDLING FUNCTION	
UPDATED TASK TABLE B		TASK DISPATCHING FUNCTION	
ENTRIES FOR TASKS WAITING ON THE MINOR CYCLE EVENT			

address of zero indicates a Mode operation. The pointer corresponding to sub address 31, for both reception and transmission, is reserved for asychronous message reception and transmission.

The table shall contain fixed pointers and some that are dynamically set up every time the DMA Pointer Block is used. Thus, if a compool is received or transmitted every even or every odd minor cycle, then its pointer shall remain fixed in the DMA Pointer Block.

3.2.1.8 Function Eight - Asynchronous Message Handling Function

The Asynchronous Message Handling Function shall manage the reception and transmission of asynchronous messages via the BCIU. Asynchronous messages are utilized to perform the following:

a. Invoke a task when a task is located in a different processor than its controller.

b. Task termination/cancellation when a task to be terminated or cancelled is in a different processor than the controller task.

c. Write compool block data when the compool block to be written has copies in other processors.

d. Signal Events when the Event Record has copies in other processors.

e. Update Compool Blocks in other processors or Remote Terminals.

Although the Local Executive determines the necessity for an asynchronous message transmission, the actual transmission of the message is controlled by the Master Executive located in the Master Processor.

3.2.1.8.1 Asynchronous Message Reception Function

The Asynchronous Message Reception Function shall accept an incoming Asynchronous message received through the BCIU and enqueue it is processing.

3.2.1.8.1.1 Inputs to Asynchronous Message Reception Function

The input to this function shall be the Reception Queue as noted in Table XX.

TABLE XX INPUT TO ASYNCHRONOUS MESSAGE RECEPTION

DATA NAME	SYMBOL	SOURCE	REFERENCE
Reception Queue		BCIU	

The Reception Queue, as illustrated in Figure 25 shall consist of the following items:

a. The Request Pending Flag indicating the presence of a message that has not been processed by the Local Executive.

b. Three 33-words buffers for storing the received Asynchronous messages.

c. Pointer to first buffer in queue indicating the next buffer to be processed, i.e., the buffer succeeding the last one fully processed.

d. Pointer to last buffer in queue pointing to the last buffer filled by a reception from the BCIU.

The Reception Queue is filled in by the BCIU. (Section 3.2.1.2) and removed from it by the Local Executive Control Function (Section 3.2.1.1) in a First-In-First-Out basis.

The reception buffers are considered to be arranged cyclically, thus, the last physical buffer succeeds the first physical buffer.

3.2.1.8.1.2 Asynchronous Message Reception Processing

The Asynchronous Message Reception Function is invoked by the Hardware Interface Control Function after receiving a level of interrupt indicating the reception of an asynchronous message.

The Asynchronous Reception Function shall examine the Asynchronous Message ID. If the ID indicates a realignment request, the Asynchronous Transmission Function shall be called to re-transmit the last message. Otherwise, as indicated in Figure 26 , the "Next Buffer" and "Last Buffer" pointers in the Reception Queue are adjusted (ref. para. 3.2.1.8.1.1) and the "Reception Pending" Flag in the same queue is set.

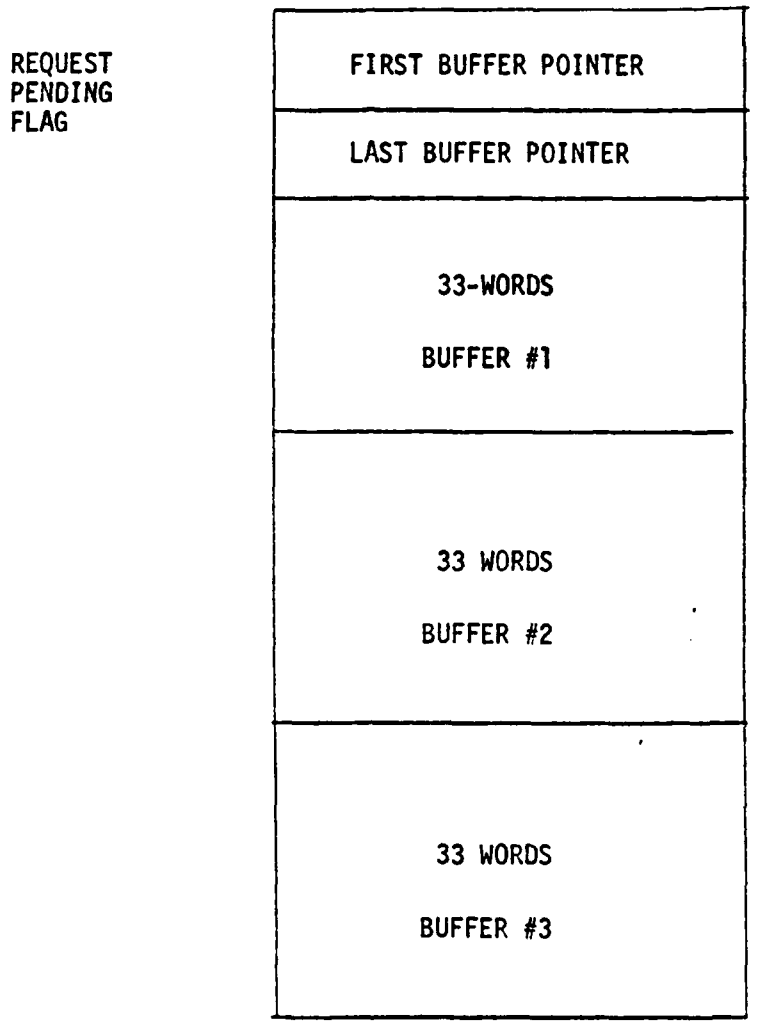
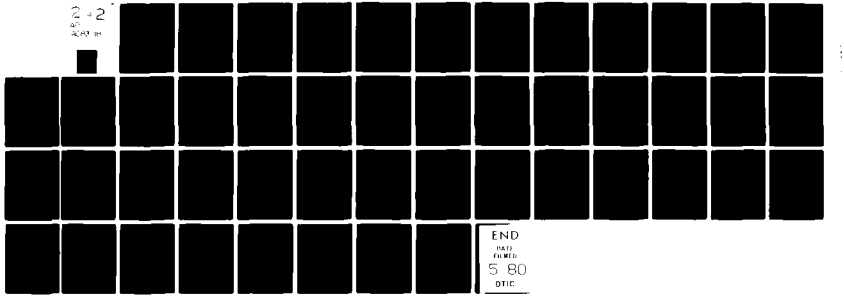


FIGURE 25 RECEPTION QUEUE

AD-A083 118 AIR FORCE AVIONICS LAB WRIGHT-PATTERSON AFB OH F/G 9/2
COMPUTER PROGRAM DEVELOPMENT SPECIFICATION FOR IDAMST OPERATION--ETC(U)
JUL 76
UNCLASSIFIED AFAL-TR-76-209-ADD-2 NL



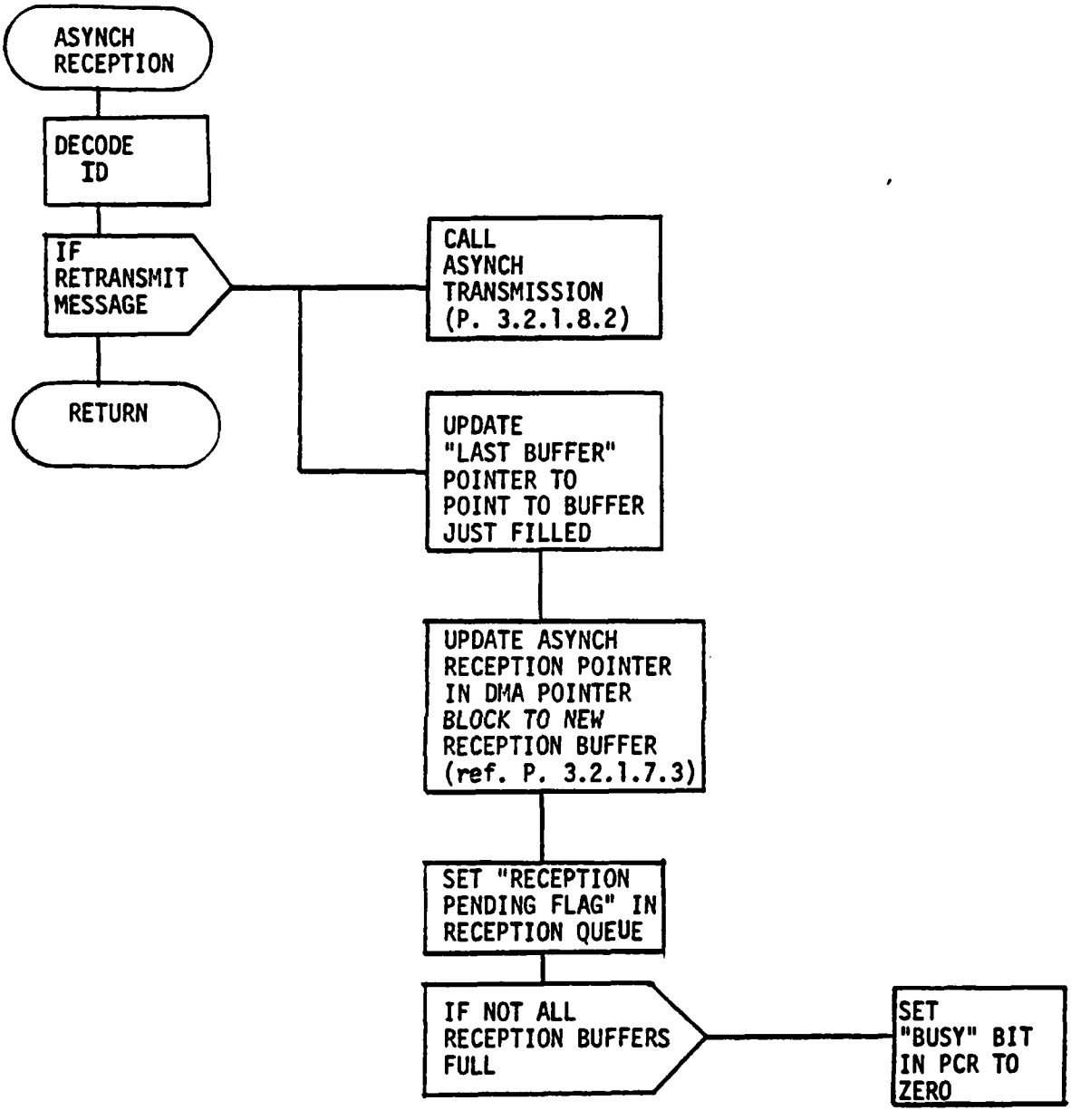


FIGURE 26 - ASYNCHRONOUS MESSAGE RECEPTION PROCESSING

The "BUSY" bit in the BCIU PCR register shall be set to zero only if the Reception Queue is not full with unprocessed messages.

3.2.1.8.1.3 Outputs from Asynchronous Message Reception Function

The outputs from the Asynchronous Message Reception Function are listed in Table XXI.

3.2.1.8.2 Asynchronous Message Transmission Function

The Asynchronous Transmission Function accepts messages enqueued by the Local Executive into a transmission queue and prepares them for transmission by the BCIU. The necessity for the transmission of an asynchronous message is determined by the Local Executive in order to accomplish tasks services as enumerated in paragraph 3.2.1.8.

This function is invoked from:

a) The Interrupt Handling Function (Section 3.2.1.2) after termination of a previous Asynchronous transmission.

b) The Asynchronous Message Reception Function (Section 3.2.1.8.1) upon reception of a message requesting a retransmission of the last message.

c) The Local Executive Control Function on a request to initiate a BCIU transmission.

3.2.1.8.2.1 Inputs to Asynchronous Message Transmission Function

The input to this function is a Transmission Queue as noted in Table XXII.

The Transmission Queue is illustrated in Figure 27. It consists of a number of Message Descriptor Blocks that contain pertinent message information and a series of pointers used in the fetching of the messages to be transmitted. A description of these parameters follows:

TABLE XXI OUTPUTS FROM ASYNCHRONOUS MESSAGE RECEPTION FUNCTION

DATA NAME	SYMBOL	DESTINATION	REFERENCE
"BUSY" Bit in BCIU		BCIU	
PCR Set to Zero			
Updated Reception Queue		Local Executive	
		Control Function	
Asynchronous Reception		BCIU	
Pointer in DMA Pointers Block			

TABLE XXII INPUTS TO ASYNCHRONOUS MESSAGE TRANSMISSION FUNCTION

DATA NAME	SYMBOL	SOURCE	REFERENCE
Transmission Queue:			
Message Descriptor Blocks		Pre-Loaded	
Transmission Pointers			
Transmission Buffer Area			
Buffer Pointers			

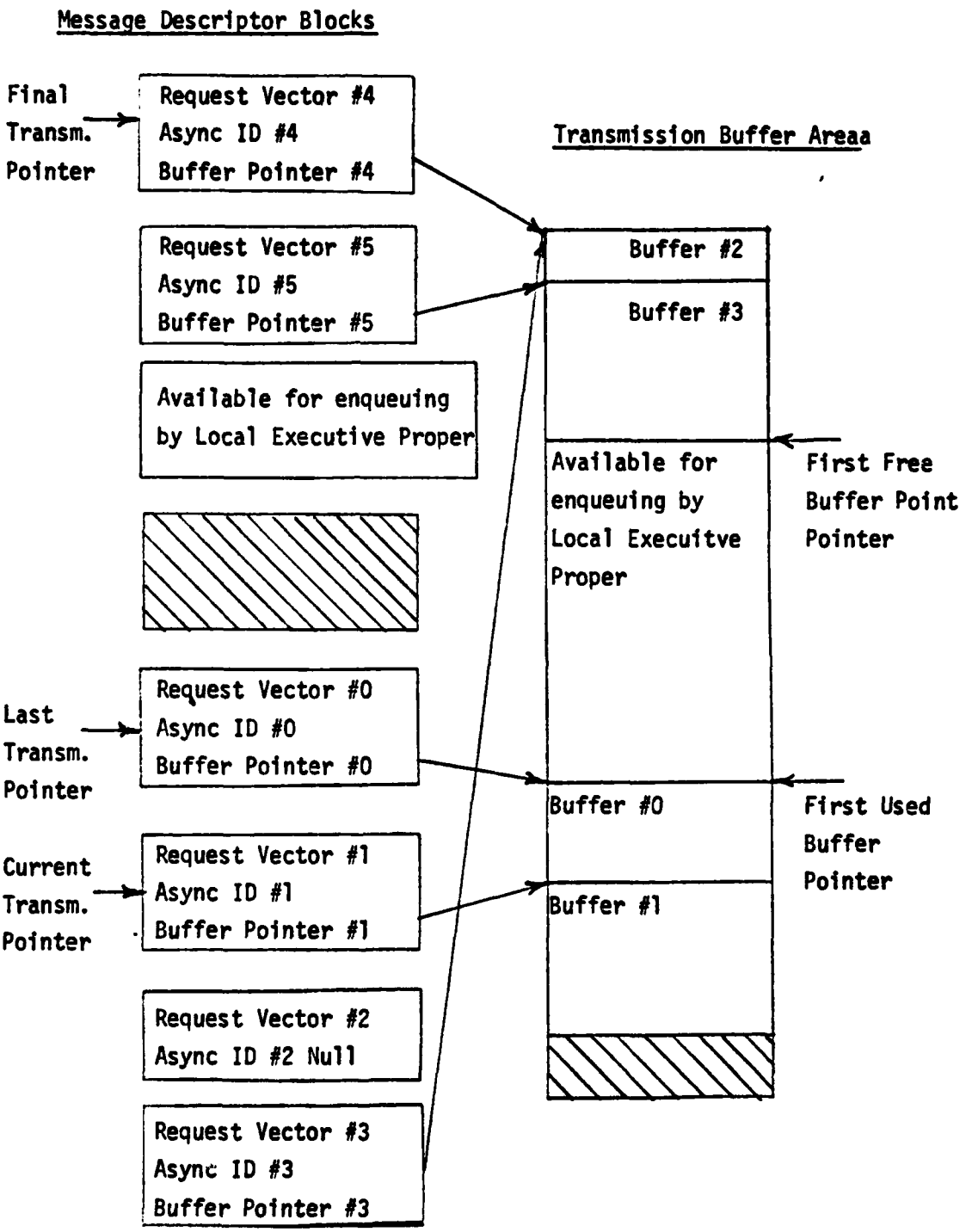


FIGURE 27 TRANSMISSION QUEUE

a) Message Descriptor Blocks

The Message Descriptor Block consists of:

1. Asynchronous Request Vector

The Asynchronous Request Vector is a 9-bit code that identifies each asynchronous message that exists in the system. If the local executive is not in the Master Processor, an Asynchronous Request Vector associated with an Asynchronous message to be transmitted will be loaded into the BCIU Status Code Register (SCR). This Status Code Register will be decoded by the Master Processor and as a consequence the Master Processor will generate the commands required by the asynchronous messages as described in paragraph 3.3.2.3.

If the local executive is located in the Master Processor, this Request Vector will be input to the Master Asynchronous Control Function through its vector stack (ref. para. 3.2.2.3).

2. Asynchronous ID Word

The ID word will be appended to the beginning of the message and the Local Executive receiving the message will make use of this word in identifying the message, (ref. para. 3.2.1.1.2). The Asynchronous ID word will contain the address offset to the message Data Descriptor Block (DDB) (ref. 3.2.1.3) if a Compool Block Handling is invoked. If the message is to be sent to an RT, the ID shall be set equal to 177777, and shall not be appended to the message.

3. Pointer into Transmission Buffer Area

The buffer pointer points to a buffer allocated within the transmission buffer area holding the data to be transmitted. It should be noted that the first two words of each buffer shall contain a Tag Word as described in 3.2.1.2, and the Asynchronous ID word, respectively. The data begins on the third word. If a message has no associated data, the buffer pointer is zero and no buffer is allocated for the message. On the other hand, it is possible for a series of messages to point to the same buffer.

b) Last Transmission Pointer (LTP) pointing to the Message Descriptor Block of the last message transmitted by the BCIU.

c) Current Transmission Pointer (CTP) pointing to the Message Descriptor Block as the message currently set-up for transmission by the BCIU.

d) Final Transmission Pointer (FTP) pointing to the Message Descriptor Block of the message most recently enqueued by the Local Executive.

e) Transmission Buffer Area used to hold the data to be transmitted.

f) First Used Buffer Pointer (FUBP) pointing to the first word in the area occupied by a message, thus unavailable for enqueueing messages.

g) First Free Buffer Pointer (FFBP) pointing to the first word in the buffer area available for storing new transmission messages.

3.2.1.8.2.2 Asynchronous Message Transmission Processing

The Asynchronous Transmission Function shall dispose of messages stored in its transmission queue in a First-In-First-Out basis. To this effect it shall make use of the transmission pointers identified in the Transmission Queue as described in Para. 3.2.1.8.2.1.

Upon being entered, the Asynchronous Message Transmission Function shall have its Current Transmission Pointer (CTP) pointing at the Description Block of the message last transmitted. Thus, upon a request to retransmit the previous message, CTP will be equal to the Last Transmission Pointer (LTP).

Figure 28 illustrates this function processing.

3.2.1.8.2.3 Outputs from Asynchronous Message Transmission Function

The outputs from this function are listed in Table XXIII.

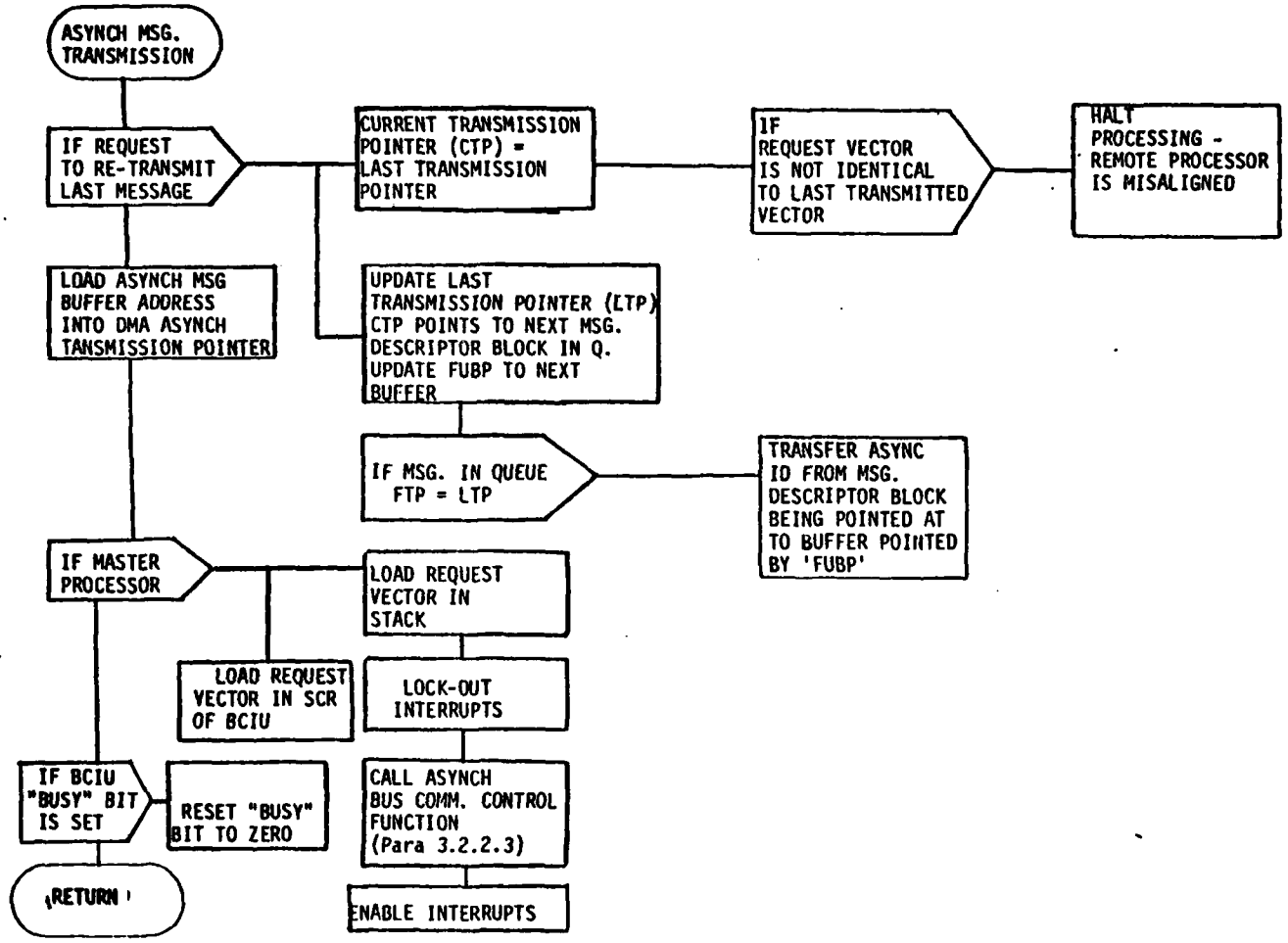


FIGURE 28. ASYNCH. MSG TRANSM. PROCESSING

TABLE XXIII OUTPUTS FROM ASYNCHRONOUS MESSAGE TRANSMISSION FUNCTION

DATA NAME	SYMBOL	DESTINATION	REFERENCE
Updated Transmission Queue		Asynchronous Message Transmission Function	
DMA Asynchronous Transmission Pointer Blocks		BCIU	
Asynchronous Request Vector		In Remote Mode: SCR in BCIU In Master Mode Asychronous Bus Comm. Control	3.2.2.3
"Continue/Busy" Bit in PCR		BCIU	

3.2.1.9 Function Nine - Local Executive Initialization Function

This function is invoked by the system hardware, namely, the ROM, upon system initialization on request from the pilot or on re-initialization as a recovery from a power-down condition.

Its purpose is to initialize or re-initialize the state of the Local Executive and the BCIU associated with the Remote Processor.

3.2.1.9.1 Inputs to the Local Executive Initialization Function

The inputs to this function shall be as listed in Table XXIV.

3.2.1.9.2 Processing of Local Executive Initialization Function

The pilot shall manually power-up the system and turn the "processors on." The pilot shall have the capability to restart the system and thus initialize a start-up sequence.

Each processor initiates operation under control of the start-up program residing in its ROM. This start-up program shall determine whether a normal start-up or a power-transient recovery is to be initiated.

For a normal start-up, the ROM shall perform a processor self-test and initiate the BCIU self-test.

The Processor Control Panel (PCP) discretes shall be read and the processor assignment determined. The BCIU shall be initialized by loading the BCIU address into the PCR bits 7 - 11, setting the Master/Remote (bit 1) bit in the PCR to agree with the processor's assignment and setting the Go bit (bit 2) of the PCR to 1.

After the software has been loaded from mass memory the local processor executive system tables are initialized and set up for a minor cycle 0.

The Local Exec Initialization Processing Sequence is shown in Figure 29.

TABLE XXIV INPUTS TO LOCAL EXECUTIVE INITIALIZATION FUNCTION

DATA NAME	SYMBOL	SOURCE	REFERENCE
Recognition of Power-On Initialization Interrupt (Level 0)		Interrupt Handling	
Power Shut-Down Flag		Processor	
Initialization Data Base		Pre-Loaded	
PCP Assignment Discretes		Processor Control Panel	

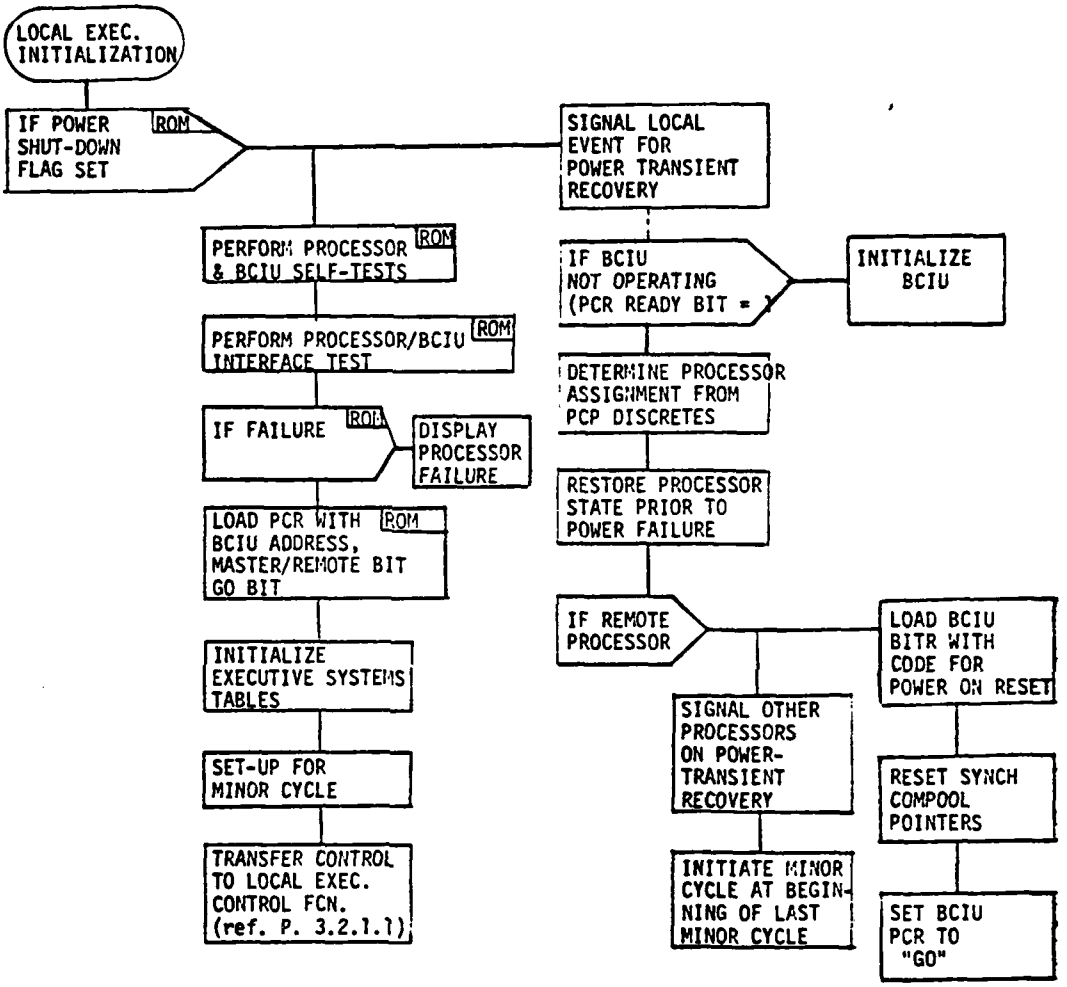


FIGURE 29. LOCAL EXEC. INITIALIZATION PROCESSING

If a remote or monitor processor recovers from a successful power-transient, an Event shall be signalled to start the power-recovery operations.

If the local executive determines that its BCIU is not operating (PCR "Ready" bit 1), it shall initialize the BCIU and through the PCP discretes determine the processor assignment. The local executive shall initialize for power-transient recovery by signalling the Master Processor about the power recovery through the BCIU bit register and setting the BCIU synchronous compool pointers to "null" compools. The master processor, upon receiving indication of a transient recovery, shall restart operation at the end of the last successful minor cycle. Finally, the BCIU PCR shall be set to "Go".

3.2.1.9.3 Outputs from the Local Executive Initialization Function

The outputs from this function are listed in Table XXV.

TABLE XXV OUTPUTS FROM LOCAL EXECUTIVE INITIALIZATION FUNCTION

DATA NAME	SYMBOL	DESTINATION	REFERENCE
Event Values		Event Table	
Task Values		Task Tables	
BCIU Registers		BCIU	

3.2.2 IDAMST Master Executive Functions

The Master Executive shall reside in the master processor and in the monitor processor. Its main function is to manage and control all data bus communication among the processors and remote terminals and direct the initialization and start-up on the software system.

A block diagram of its functions is shown in Figure 30.

3.2.2.1 Function Ten - Master Executive Initialization & Start-Up Functions

The main objective of this function is to establish the proper operational sequence to enter normal system operations.

Each processor in the IDAMST system will contain a basic start-up program residing in a READ-Only-Memory (ROM) module which will control the initiation of operations within the processors.

This start-up program sequence is considered part of the initialization function and as such is being described in the processing description of this function (ref. para. 3.2.2.1.2).

3.2.2.1.1 Inputs to Master Executive Initialization & Start-Up

The inputs to this function are listed in Table XXVI.

3.2.2.1.2 Initialization & Start-Up Processing

This function processing is illustrated in Figure 31 and Figure 32.

The start-up procedure shall be initiated by the pilot going through a sequence of actions using the Processor Control Panel (PCP). These are:

a) Manually turning the power-on for the system. This action shall set a "normal start-up" discrete.

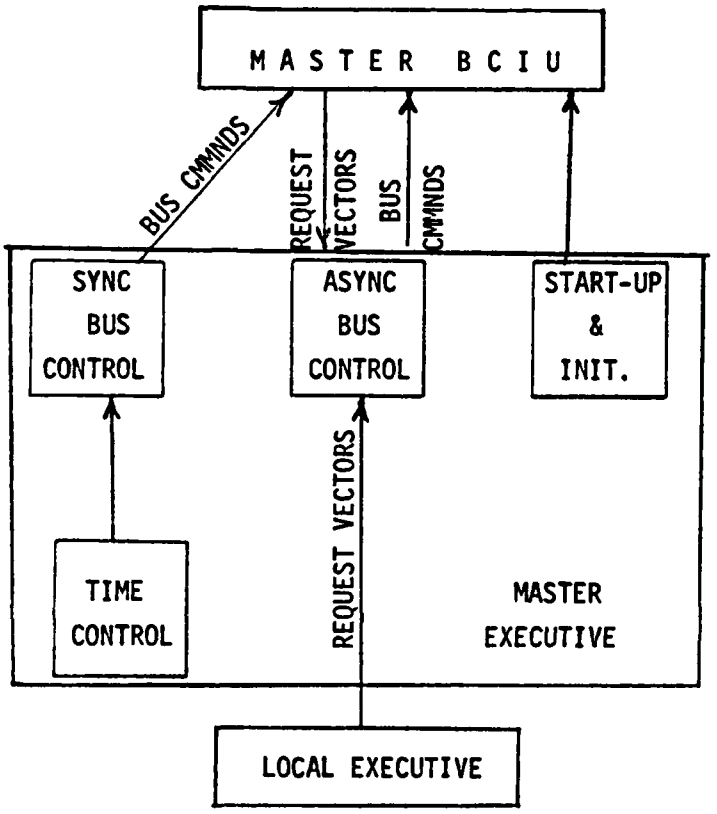


FIGURE 30 MASTER EXECUTIVE FUNCTIONS

TABLE XXVI INPUTS TO M.E. INITIALIZATION & START-UP FUNCTION

DATA NAME	SYMBOL	SOURCE	REFERENCE
Normal Start-Up Discrete		PCP	
Master/Remote		PCP	
PCR "GO" Bit		ROM	
Power Shut Down Discrete		Processors	
Check Sums			
System Software:			
GTP-1, GTP-2, Mission		Mass Memory	
Mission-Dependent Data		Mass Memory	
GTP-1, GTP-2, Mission		PCP	
Indicators			

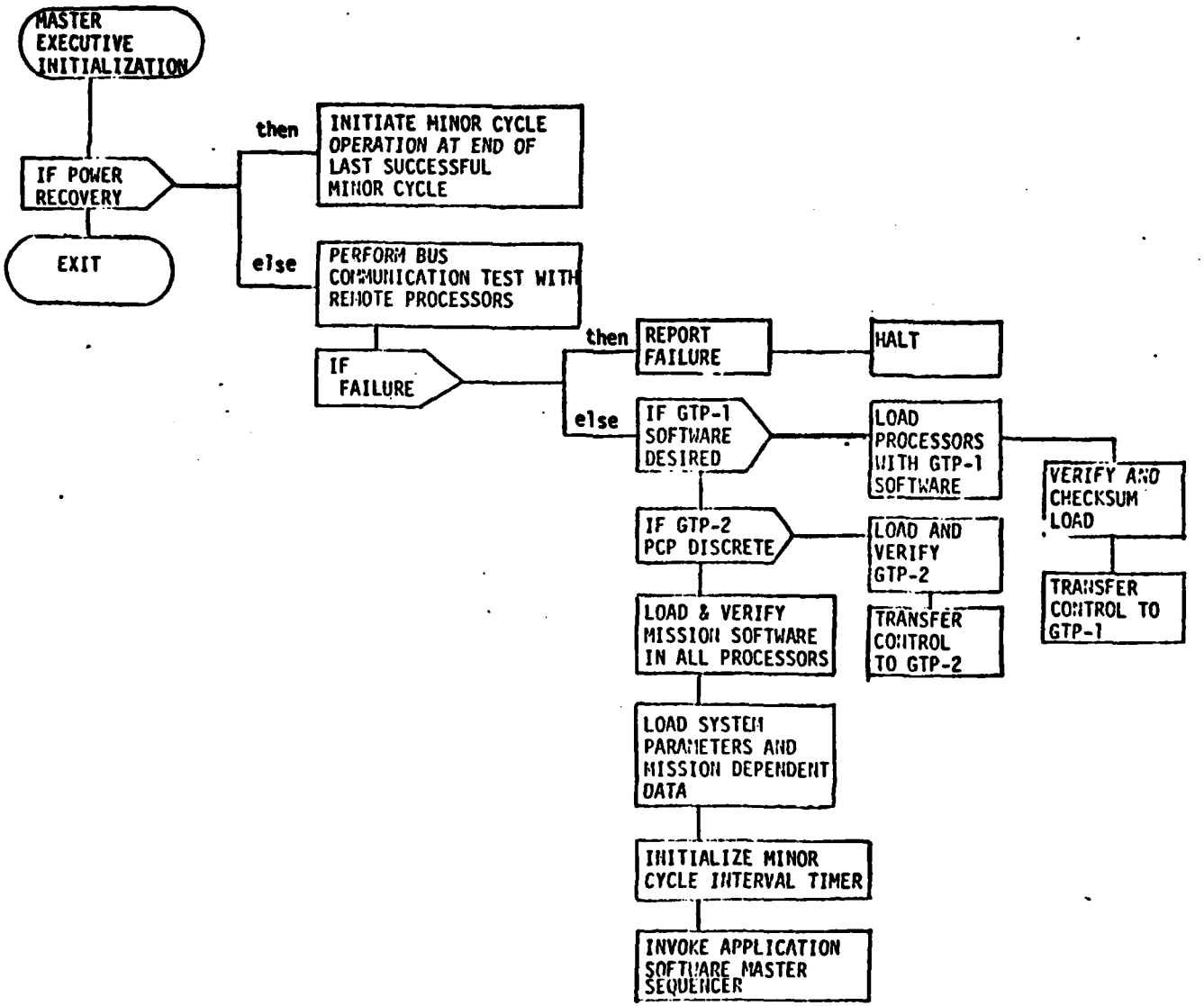


FIGURE 31. MASTER EXECUTIVE INITIALIZATION

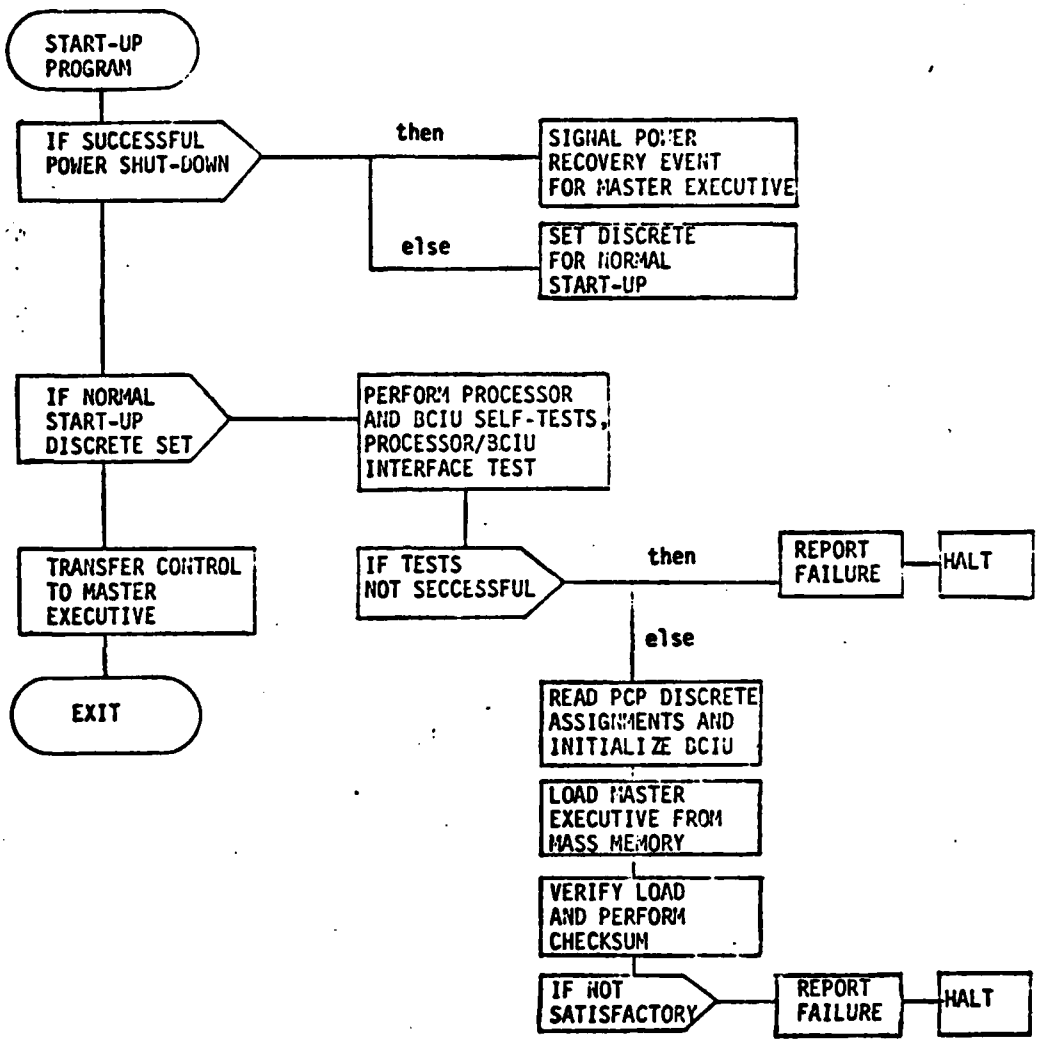


FIGURE 32. MASTER EXECUTIVE START-UP

b) Turning the power-on for every processor.

c) Activating a "START" switch to initiate system start-up.

Upon the start-up program in the ROM taking control, it shall check for a normal start-up sequence ("normal start-up" discrete). If this discrete has not been set, the program will determine if a successful power shut down was accomplished. If successful, the master executive shall signal the other processors of the necessary operation and initiate minor cycle operation at the end of the last successful minor cycle. The Local Executive shall execute its sequence as described in Function Nine, paragraph 3.2.1.9. If the power shut-down was not accomplished successfully, a normal start-up sequence shall be performed.

The ROM start-up program shall perform the processor and BCIU self-tests, the processor/BCIU interface test and read the PCP assignment discretes to initialize the BCIU as per its Master/Remote bit and the device address and setting the PCR "GO" bit. If the tests are satisfactory the Master Executive shall be loaded from mass memory and a check sum performed on the master executive.

The master executive will take control and perform a bus communication test with the other processors.

If a failure occurs in any of the previous tests, i.e. processor self-test, BCIU self-test, processor/BCIU Interface test, Master load verification, Master load check sum, Remote/Master bus communication test, the "Fail" Light" for the respective processor shall be displayed. The pilot shall reassign the processors, turn-off those that failed and restart the system.

After successful testing, the master executive shall determine, from a discrete originating in the Processor Control Panel (PCP) which application software to load and the desired configuration. If the discrete indicates GTP-1, the GTP-1 software shall be loaded and verified from mass memory. A check sum shall also be performed on each processor load.

Upon finishing the execution of GTP-1, GTP-2 may be indicated through the PCP or else regular mission software is to be loaded from mass memory. If there has not been a failure to load the mission software, the system parameters and mission-dependent data shall be loaded from mass memory.

To initiate the normal system operation the minor cycle interval timer shall be initiated and the application software master sequencer invoked.

3.2.2.1.3 Outputs from M.E. Initialization & Start-Up Function

Upon exiting this function the IDAMST system shall have each processor loaded with the proper executive and application software. One processor shall contain master and local executives, another processor shall contain only a local executive and the third processor shall contain a master executive with monitoring functions and a local executive. The application software shall be allocated as per the configuration management.

3.2.2.2 Function Eleven - Master Synchronous Bus Communication Control

The master executive controls the transmission of synchronous message data over the BCIU each minor cycle. These data may be in the form of minor cycle mode commands, actual synchronous messages and/or status word polling messages.

There are sixty-four minor cycles occurring per major cycle where a major cycle is defined as occurring every second. Synchronous messages can be transmitted at different binary rates per major cycle, thus, the possible synchronous data periods are:

1 Every Cycle
2 Every Other Cycle
4 Every Four Cycles
8 Every Eight Cycles
16 Four Times a Major Cycle
32 Twice a Major Cycle
64 Once a Major Cycle

Synchronous messages, while perhaps on the same period as described above, can be scheduled on the BCIU at different phases with respect to the start of the major cycle. Thus every message has associated with it a period and a phase. The phase of a message is its displacement relative to the first minor cycle. A message with a period of two will have its first occurrence in minor cycle 0 or 1 with a phase of 0 or 1 accordingly similarly for a period of four the message can have a phase of 0, 1, 2, or 3. Messages transmitted each minor cycle always have a phase of 0 and a period of 1.

A phase table, as illustrated in Figure 34 , identifies the phases associated with the minor cycles with respect to the transmission rates.

3.2.2.2.1 Inputs to Master Synchronous Control Function

The inputs to this function are listed in Table XXVII. The input tables are described below.

3.2.2.2.1.1 BCIU Instruction Formats

Each BCIU command consists of instruction pairs in the format as illustrated in Figure 33. The BCIU sequentially interpret each instruction pair to determine the action required. Depending on the OP Code contained in the instruction pair, the BCIU will initiate a bus transmission, transmit a masking mode command, perform any bus operations but will use the second word of the instruction as the address of the next instruction pair. The rest of the fields in the instruction pair are described in detail in Section 3.1.1.1.

3.2.2.2.1.2 BCIU Synchronous Instruction List

The BCIU Instruction List contains an instruction pair for each synchronous message that is transmitted on the data bus. If any of the synchronous transmissions require bit or word masking, the instruction pair that affects the transmission will be preceded by one to send the proper Mode Command. A word mask will also require the loading of the BCIU Mode Data Register with the desired word mask.

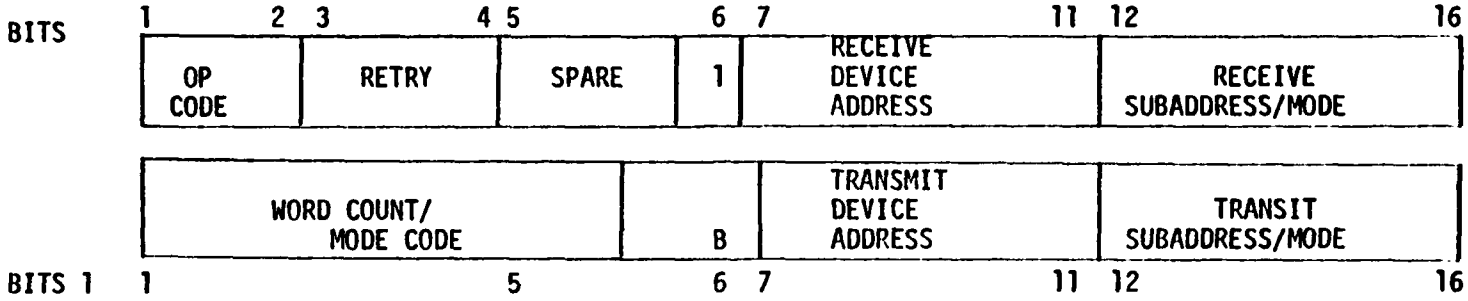
Since each message is not transmitted every minor cycle, the instruction list must be organized so that the proper instruction pairs can be linked together at the start of each minor cycle to form the complete instruction list for that minor cycle. Thus, instruction pairs are organized in instruction blocks. The Synchronous Instruction List will contain one block of instructions for each phase and period for which there are synchronous bus transmissions. The organization of these blocks is illustrated in Figure 34. The last instruction pair in each instruction block is a Link instruction. This link instruction pair is dynamically set by the Master Executive to point to the next block of instructions to be performed during the current minor cycle.

TABLE XXVII INPUTS TO MASTER SYNCHRONOUS CONTROL FUNCTION

DATA NAME	SYMBOL	SOURCE	REFERENCE
Minor-Cycle			
Synchronous Instruction List			
Instruction List Pointer Table			

OP CODES:

0 0 = HALT BCIU
0 1 = LINK (USE SECOND WORD AS ADDRESS OF NEXT 2 WORD INSTRUCTION)
1 0 = NO OPERATION (GO TO NEXT 2 WORD INSTRUCTION)
1 1 = BUS OPERATION

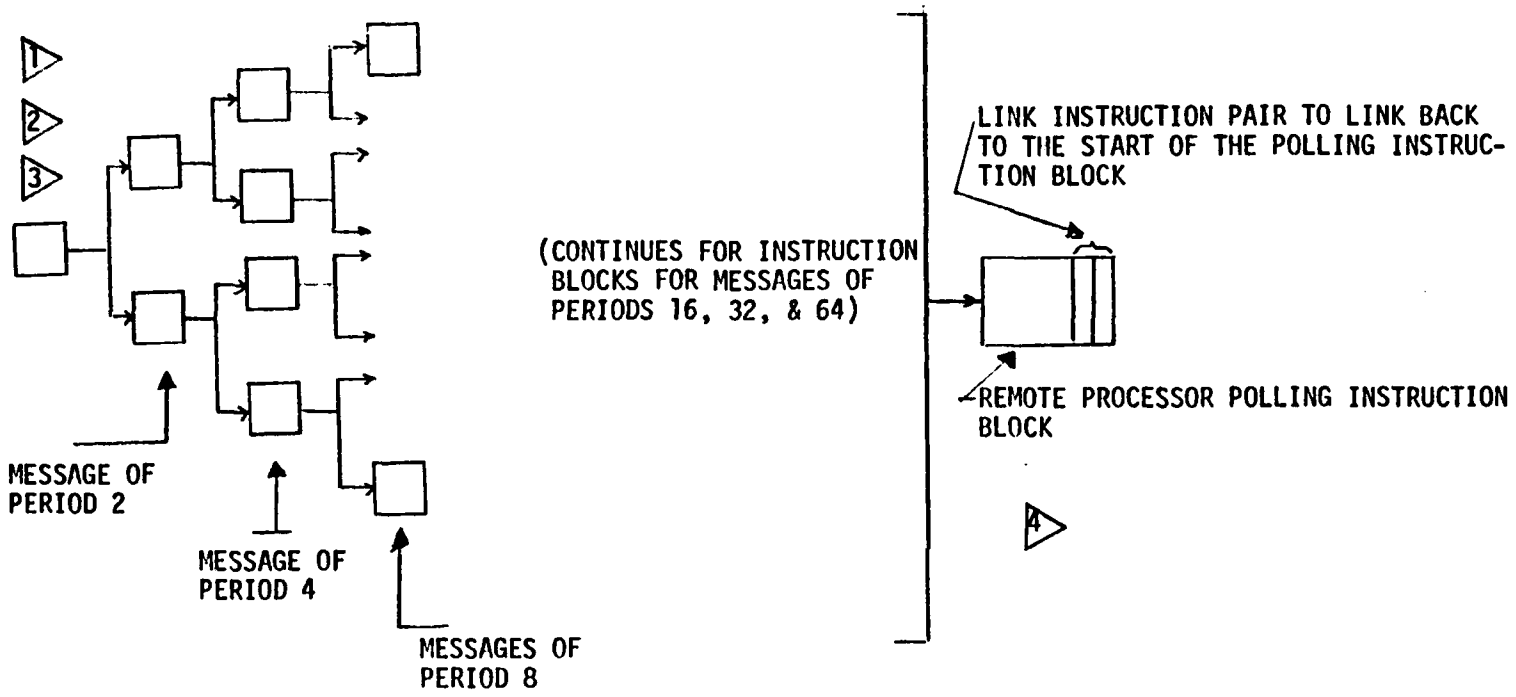


B = BIT TO DEFINE WHICH BUS THIS BUS OPERATION SHALL BE PERFORMED ON.
(LOGIC 1 SELECTS BIM 2)

I = BIT TO INDICATE WHETHER AN INTERRUPT TO THE PROCESSOR UPON SUCCESSFUL COMPLETION OF THE BUS OPERATION. (LOGIC 1 PRESENTS PROGRAMMED CONTROLLED INTERRUPT [PCI] TO PROCESSOR)

RETRY = INDICATOR TO SOFTWARE OF NUMBER OF RETIRES ALLOWED BY THIS 2 WORD INSTRUCTION
(MAXIMUM OF THREE RETRIES PER INSTRUCTION)

FIGURE 33 BCIU Instruction Format



1. SEE FIG. 37 FOR CONTENTS OF 1ST INSTRUCTION BLOCK
2. FOR EACH OF THE PERIODS 2, 4, 8, 16, 32 & 64 THERE MAY BE AN INSTRUCTION BLOCK FOR EACH POSSIBLE PHASE. FOR EXAMPLE, MESSAGES OF PERIOD 16 HAVE 16 PHASE 0 THROUGH 15. IF MESSAGES OF A PARTICULAR PERIOD ARE NOT TRANSMITTED IN A GIVEN PHASE. AN INSTRUCTION BLOCK DOES NOT EXIST.
3. LAST INSTRUCTION PAIR OF EACH BLOCK IS A LINK INSTRUCTION PAIR
4. ALL MESSAGES OF PERIOD 64 (TRANSMITTED ONCE PER SECOND) LINK TO THE REMOTE PROCESSOR POLLING INSTRUCTION BLOCK.

FIGURE 34 INSTRUCTION BLOCK ORGANIZATION

3.2.2.2.1.3 Instruction List Pointer Table

This table is used by the Master Executive to set the appropriate Link instruction pairs every minor cycle. This table is illustrated in Figure 35.

This table will contain an entry for each phase and period in which a synchronous message appears. The entries are arranged in ascending sequence by phase and then by period. Thus, the entry for phase X and period Y is entry (X + Y).

3.2.2.2.2 Processing by Master Synchronous Control Function

This function is initiated upon the occurrence of an interval timer interrupt and its processing is illustrated in Figure 36.

Upon entering, this function shall first determine whether error processing is under way. If this is so the error processing function will be invoked. Otherwise; the function will determine if all minor cycle transmissions have been completed. If they are not completed, the minor cycle shall be extended for one more minor cycle period if the minor cycle has not been extended before. If the minor cycle has been extended before, a message shall be displayed and the master processor halted. This action will cause the monitor processor to take over and manage any reconfiguration request.

If the synchronous bus transmissions for the minor cycle have been completed, the minor cycle number is incremented and loaded into Master Function Register of the BCIU and the Minor Cycle pending bit is set in the Master Processor. A minor cycle phase table shall be generated identifying the phases applicable to all the period for this minor cycle as explained in paragraph 3.2.2.2. Figure 36 illustrates this table for three different minor cycles.

Making use of this table and the Instruction List Pointer Table (para. 3.2.2.1.3), the appropriate instruction blocks shall be linked creating a continuous instruction list to control the BCIU's operation for the

RATE	PERIOD	PHASE	FIRST WORD IN INST. BLOCK	LAST WORD IN INST. BLOCK
64	1	0	Absolute Addresses of First Word in Instruction Block	Absolute Addresses of Last Word in Instruction Block (Second Word of Link Instruction)
32	2	0		
32	2	1		
16	4	0		
	4	1		
	4	2		
16	4	3		
8	8	0		
	8	1		
	-	-		
	-	-		
8	8	7		
4	16	0		
	16	1		
	-			
	-			
	-			
4	16	15		
2	32	0		
	32	1		
	32	2		
	-			
	-			
	-			
2	32	31		
1	64	0		
	-			
	-			
	-			
1	64	63		

Figure 35 Instruction List Pointer Table

MINOR CYCLE 43		60	61
PERIOD	PHASE	PHASE	PHASE
2	1	0	1
4	3	0	1
8	3	4	5
16	11	12	13
32	11	28	29
64	43	60	61

Figure 36 Minor Cycle Phase Table Illustration

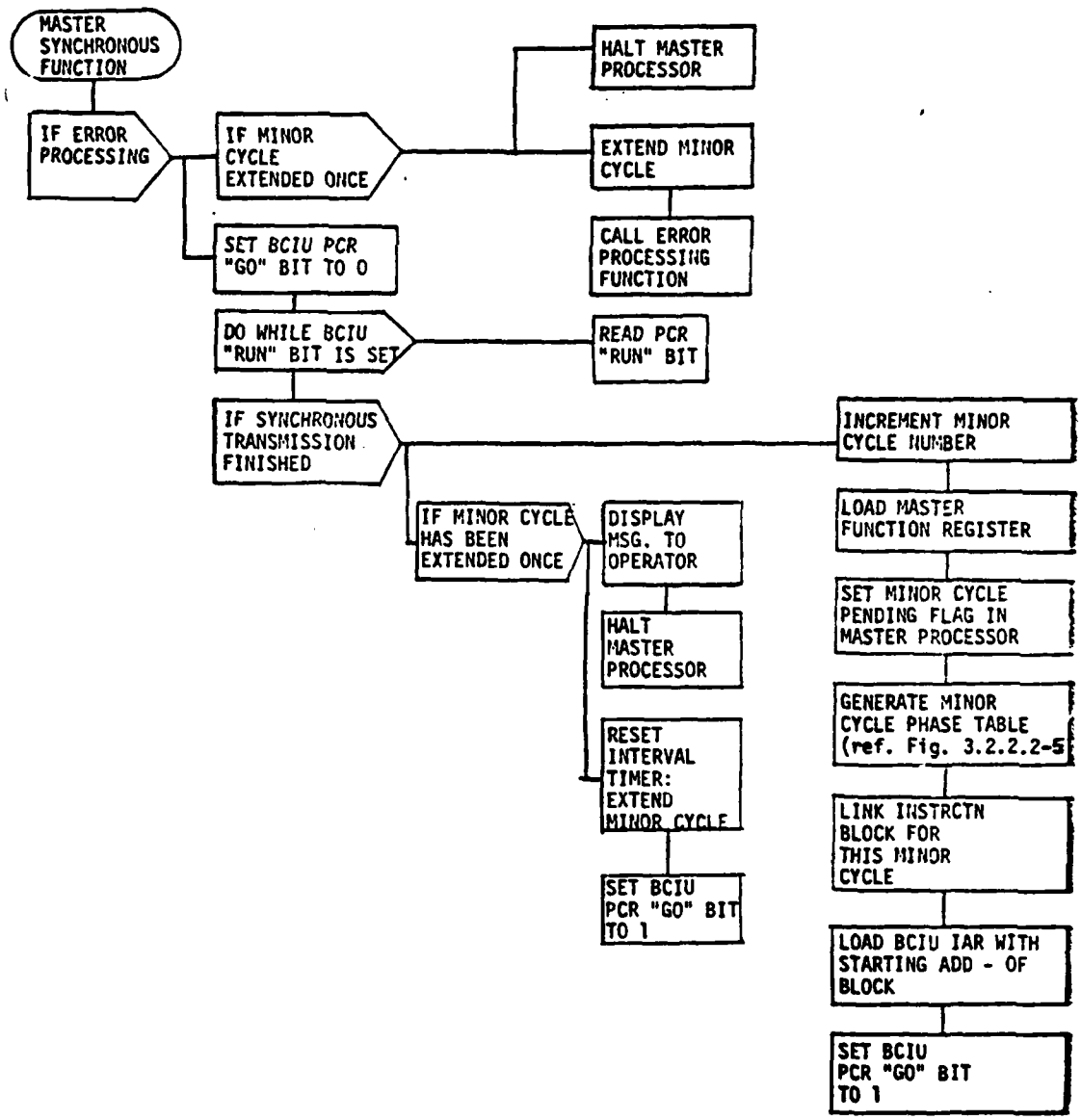


FIGURE 37 MASTER SYNCH. CONTROL PROCESSING

current minor cycle.

The proper register in the BCIU are loaded and the synchronization commands are transmitted to the remote processors.

3.2.2.2.3 Outputs from Master Synchronous Control Function

The outputs from this function are shown in Table XXVIII

3.2.2.3 Function Twelve-Asynchronous Bus Communication Control

The responsibility of this function is to control all asynchronous communication through the BCIU. Requests for asynchronous transmission can be received from the local executive resident in the remote processor or the local executive in the master processor. Requests for transmission are identified to the master executive by the Interrupt Request Vector associated with each asynchronous message.

The asynchronous transmission is formulated as described in the following paragraphs.

3.2.2.3.1 Inputs to Asynchronous Control Function

The inputs to this function are listed in Table XXIX

3.2.2.3.1.1 Request Vectors

The Request Vectors are 9-bits data fields associated with the asynchronous messages existing in the system. Each request vector will point to a particular message, thus a maximum of 512 asynchronous messages can be provided for in the system.

A local executive located in a remote processor requests an asynchronous message by loading the proper interrupt vector into the BCIU's Status Code Register. The local executive located in the Master Processor passes the associated interrupt vector to the master executive by loading it

TABLE XXVII. OUTPUTS FROM MASTER SYNCHRONOUS CONTROL FUNCTION

DATA NAME	SYMBOL	DESTINATION	REFERENCE
MINOR CYCLE NUMBER		BCIU MFR	
MINOR CYCLE PHASE TABLE		SYCHRONOUS MESSAGE RECEPTION & TRANSMISSION	
INSTRUCTION BLOCKS		BCIU	
ADDRESS - FIRST INSTRUCTION BLOCK		BCIU IAR	

TABLE XXIX INPUTS TO ASYNCHRONOUS BUS COMMUNICATION CONTROL FUNCTION

DATA NAME	SYMBOL	SOURCE	REFERENCE
Request from Remote Processor		Level interrupt	
Request from Local Executive in Master Processor		Function	
Request Vectors		Pre-assigned	
Master Request Decode Table		Pre-loaded	
Master Instruction Supplement Table		Pre-loaded	
Master REmote Terminal		Pre-loaded	
Request Table		Pre-loaded	
Remote Table		Pre-loaded	
Remote Asynchronous Table		Pre-loaded	
Master Instruction Keys Table		Pre-loaded	

into an interrupt vector stack. In either case, the master executive uses the interrupt vector as an index into the Master Request Decode Table described in 3.2.2.3.1.2.

3.2.2.3.1.2 Master Request Decode Table

The Master Request Decode Table contains an entry for every processor-to-processor and processor-to-RT asynchronous compool block update message. The interrupt vectors are used to index into this table, thus, the entries will be numbered in ascending, consecutive order. Messages originating in an RT are defined using a different set of tables, Master Remote Terminal Request Tables. There are two different types of entries in this table: one for transmissions involving no masking and one for transmissions involving bit or word masking. For messages with no masking, the entry is a master instruction set to effect the transmission. Messages that entail bit or word masking will have for entry several items relating to the mask description.

Item #1 will be zero to indicate that it is not a master instruction set.

Item #2 will be a pointer to a second table, Master Instruction Supplement Table, describing the bit or word masking.

Item #3 is the word mask that will appear in the bus message. If it is a bit masking, item #3 will be all ones.

3.2.2.3.1.3 Master Instruction Supplement Table

This table will contain an entry generated for any asynchronous message that has word or bit masking and originates at a processor.

Each entry contains two master instruction sets. The first is the Mode command indicating the proper masking; the second is the command which performs the BCIU transmission.

3.2.2.3.1.4 Remote Asynchronous Table

If the asynchronous message received by the Master Processor has originated at a Remote Terminal (RT), the Master Executive will make use of its Remote Terminal Request Tables.

The Remote Asynchronous Table will contain an entry for each of the 32 possible remote terminals. These entries will be indexed by a Remote Terminal identification number.

Item #1 will contain the total number of asynchronous messages transmitted from this RT.

Item #2 will contain an index pointing to this RT's entry in the Remote Terminals Master Instruction Keys Table.

3.2.2.3.1.5 Remote Terminals Master Instruction Keys Table

This table will have an entry for all asynchronous messages originated by a Remote Terminal. All the messages associated with an RT will be contiguous within the table.

The entries to this table are accessed by item #2 of the Remote Asynchronous Table described in 3.2.2.3.1.4.

Item #1 contains a mask associated with the RT Activity Register. This mask's value will identify the particular asynchronous request.

Item #2 contains the Instruction Set necessary to satisfy the RT request.

3.2.2.3.2 Asynchronous Control Function Processing

Asynchronous Processing is performed whenever: (a) Master BCIU receives a status word from a remote processor containing an interrupt vector with an octal value of 001 through 776. This condition shall generate a level 3 interrupt indicating the reception of a valid status word.

(b) The local executive located in the Master Processor passes an interrupt vector to the master executive through the interrupt vector stack as described in Function Seven.

This function shall fetch the top interrupt request vector (the oldest) and use it to index into the Master Request Decode Table described in paragraph 3.2.2.3.1. If this table indicates bit-or-word-maksing message, the Master Instruction Supplement Table will be utilized. As indicated in paragraph 3.2.2.3.1, if the request for transmission comes from a Remote Terminal, the Master Remote Terminal Request Tables shall be used instead.

If the status word received indicates a status word error, the error processing function shall be invoked instead.

The processing performed by this function is illustrated in Figure 38

3.2.2.3.3 Outputs from Asynchronous Control Function

The outputs from this function are listed in Table XXX

3.2.3 Function Thirteen - Monitor Control Functions

The monitor control functions will be performed by the Master Executive residing in the Monitor Processor only. Its functions are:

a. To receive and process asynchronous messages from the master processor updating its bus control data base.

b. To monitor the master processor for minor cycle synchronization operation.

c. To assume control of the system if any processor/BCIU fails.

In the event of the Master Processor or its BCIU's failure, the monitor control function shall assume control of the system automatically.

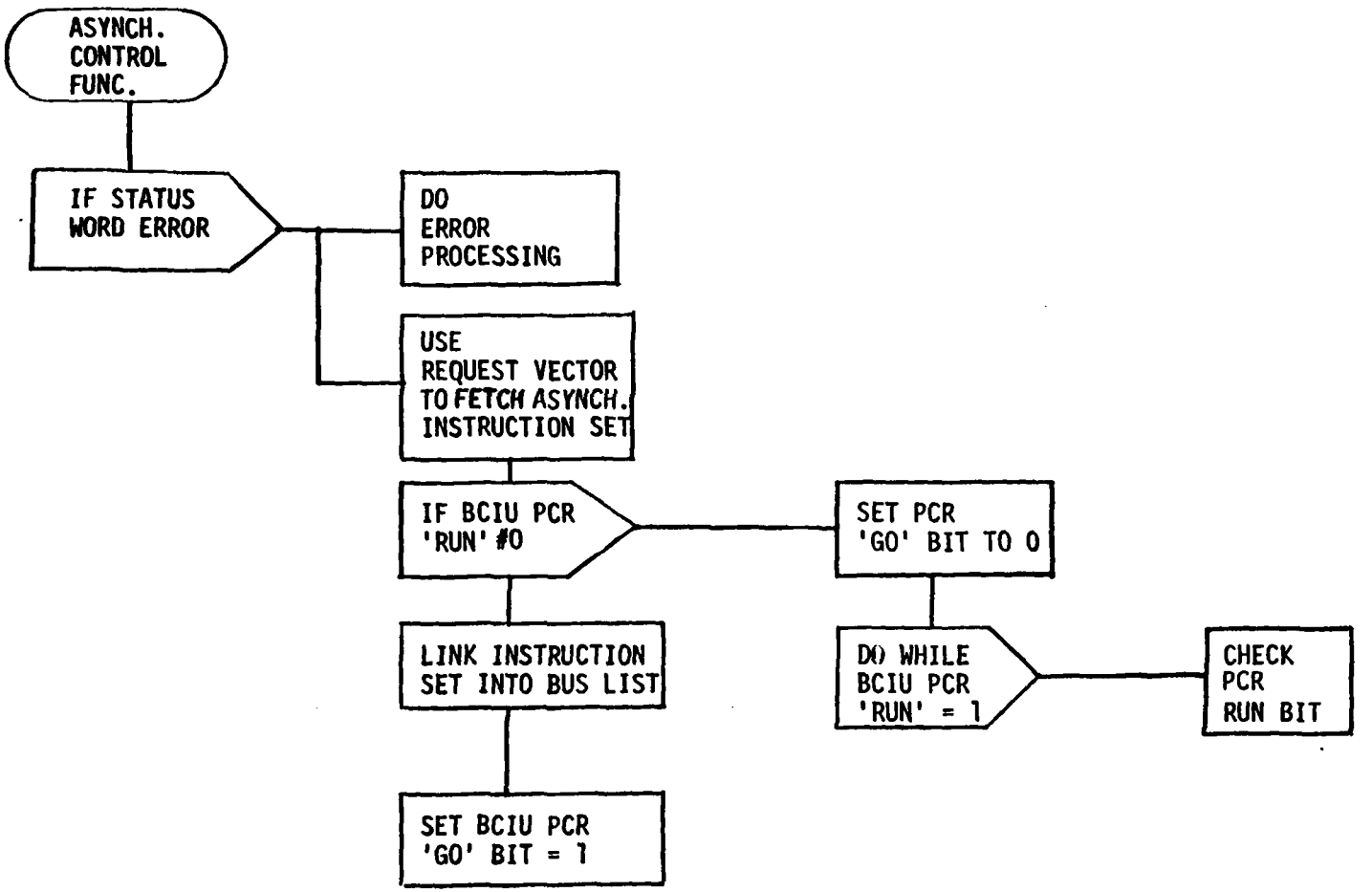


Figure 38 Asynchronous Control Function

TABLE XXX OUTPUTS FROM ASYNCHRONOUS CONTROL FUNCTION

DATA NAME	SYMBOL	DESTINATION	REFERENCE
BCIU PCR BITS		BCIU	
BCIU Instruction pairs for asynchronous message		BCIU	

If a remote processor/BCIU fails, the master executive in the master processor shall confirm the failure and relinquish control to the monitor processor.

The monitor processor shall contain also a local executive system as described in Section 3.2.1. While in monitor mode, the subset of tasks and events residing in the monitor processor shall be updated as per request from the other processors. No data, command or signal transmission shall be generated from the monitor processor while in the monitor mode.

3.2.3.1 Inputs to Monitor Control Function

Inputs to the Monitor Control Function are listed in Table XXXI

3.2.3.2 Monitor Control Processing

The main purpose of the Monitor Executive function is to monitor the performance of the master processor and its BCIU and to assume control of the operating system upon recognizing the master failure.

Upon assuming control, the operator (pilot) shall be informed of the failure and the Monitor Processor shall continue in control of the processing system until reconfiguration is established.

The monitor executive monitors the master processor by monitoring the issuance of minor cycle commands. Upon recognizing the lack of minor cycle reception for N minor cycle lengths, the monitor shall assume system control.

The processing performed by this function is illustrated in Figure 39

3.2.3.3 Outputs from Monitor Control Function

The outputs from this function are listed in Table XXXII

TABLE XXXI INPUTS TO MONITOR CONTROL FUNCTION

DATA NAME	SYMBOL	SOURCE	REFERENCE
Minor Cycle Command		Master Processor	
System Failure Data		Master Processor	

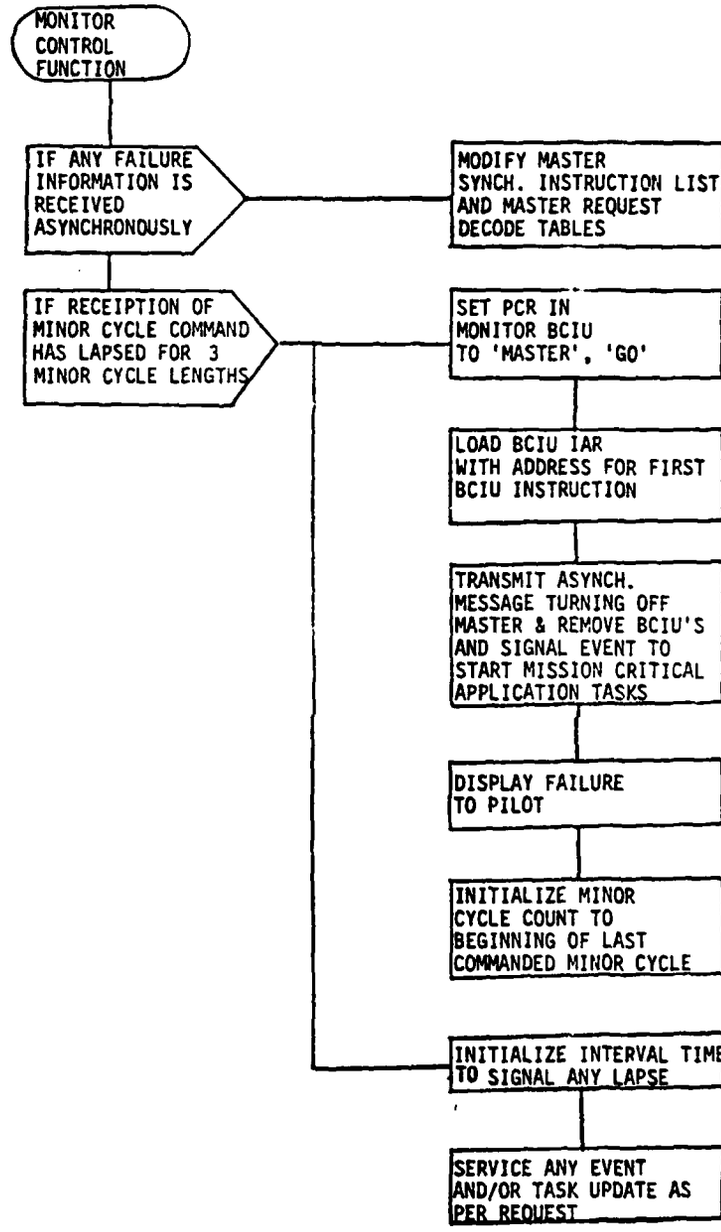


Figure 39 Monitor Control Processing

TABLE XXXII OUTPUTS FROM MONITOR CONTROL FUNCTION

DATA NAME	SYMBOL	DESTINATION	REFERENCE
Message to Pilot Advising of Failure		Displays	
BCIU Registers		BCIU	
Synch. Instruction List Modification		Master *Synchronous* Instruction List	
Asynchronous Messages Modification		Master Request Decode Table	

4.0 QUALITY ASSURANCE PROVISIONS

4.1 Introduction

Tests and evaluations shall be conducted to verify that the performance and design of the OFP-Executive shall meet or exceed the requirements specified in Section 3.0. The test category, verification method, and test requirements for performance/design requirements are specified in the Verification Cross-Reference Index (VCRI), Table XXXIII. The requirements delineated shall be the basis for the test plan and test procedure which shall be written. The four methods given in Table XXXIII of verifying individual requirements are explained as follows:

a. Inspection - Formal verification of a performance of a design requirement by examination of the assembled CPCI at the time and place of qualification testing. Inspection is not often specified as a formal means of verification for a requirement. One set of requirements that might be verified by inspection are the data base requirements, which can be verified by comparing the data base documentation with a system tape listing.

b. Analysis - Formal verification of a performance or design requirement by examination of the constituent elements of a CPCI component. For example, a weapons guidance equation or a coordinate conversion equation might be verified by analysis.

c. Demonstration - Formal verification of a performance or design requirement by observation of a demonstration test. For example, visual demonstration might be used to verify that the displays generated by the CPCI are in the format necessary to satisfy human performance requirements.

d. Review of Test Data - Formal verification of a performance or design requirement by examining the data output after operation of a CPCI component when selected input data are processed. For example, a review of hardcopy printout test data might be used to verify that the content of a specific told-in message is correctly processed. This method is the

TABLE XXXIII VERIFICATION CROSS REFERENCE INDEX

Method Legend: NA Not Applicable

1 - Inspection
2 - Analysis
3 - Demonstration
4 - Review of Test Data

A - Computer Program Test and Evaluation
B - Preliminary Qualification Test
C - Formal Qualification Test
II - Category II Test

SECTION 3 REQUIREMENT REFERENCE	METHOD					TEST CATEGORY				VERIFICATION REQUIREMENT
	NA	1	2	3	4	A	B	C	II	
3.2.1	X									
3.2.1.1		X			X	X	X			4.2.2, 4.2.3, 4.2.4
3.2.1.2		X			X	X	X			" " "
3.2.1.3		X			X	X	X			" " "
3.2.1.4		X			X	X	X			" " "
3.2.1.5		X			X	X	X			" " "
3.2.1.6		X			X	X	X			" " "
3.2.1.7		X			X	X	X			" " "
3.2.1.8		X			X	X	X			" " "
3.2.1.9		X			X	X	X			" " "
3.2.2	X									" " "
3.2.2.1		X			X	X	X			" " "
3.2.2.2		X			X	X	X			" " "
3.2.2.3		X			X	X	X			" " "
3.2.3		X			X	X	X			4.2.2, 4.2.3, 4.2.4

one likely to be used for the majority of qualification testing.

Narrative data pertaining to test categories, amplifying the tabular content of the VCRI are specified in subparagraphs below. Test requirements referenced in the VCRI are specified in 4.2 and subparagraphs thereto.

4.1.1 Category I Test

Category I testing is subdivided into the following broad types:

a. Computer Program Test and Evaluation - Tests conducted prior to and in parallel with preliminary or formal qualification tests. These tests are oriented primarily to support the design and development process.

b. Preliminary Qualification Tests - Formal tests oriented primarily towards verifying portions of the CPCI prior to integrated testing/formal qualification tests of the complete CPCI (see paragraph 4.1.3 below). These tests will typically be conducted by the contractor's design and development facilities.

c. Formal Qualification Tests - Formal tests oriented primarily towards testing of the integrated CPCI, normally using operationally configured equipment at the category II site prior to the beginning of category II testing. This testing will emphasize those aspects of the CPCI performance which were not verified by preliminary tests. The testing requirements which cannot be verified during category I test shall be specified in paragraph 4.1.5.

Qualification of this CPCI shall be accomplished during qualification testing to the maximum extent possible, as a result of preliminary qualification tests (PQT) and formal qualification test (FQP) conducted by the contractor and witnessed/verified by the procuring activity.

4.1.2 Computer Programming Test and Evaluation

Programming test and evaluation which apply satisfy one or both of the following criteria:

(1) They are intended to be the only source of data to qualify specific requirements in Section 3.

(2) They must be accomplished as part of an integrated test program involving other systems/equipment/computer programs.

4.1.3 Preliminary Qualification Tests

These tests will directly support the top-down implementation and verification. Method of verification shall be as specified in Table XXXIII. The following three levels of qualification shall be performed.

a. Unit Design Qualifications shall apply to each module. At this level the characteristics which are of primary interest are the internal workings of the module; logical flow control, numerical results, convergence, scaling, and range.

b. Module Design Qualifications shall apply to each module after it is interfaced with its environment. These tests are basically interface tests; correct internal operations are assumed. The object is to verify that two or more modules work together. To comply with the top-down approach the interfacing tests shall be sequenced from the top to the bottom.

c. System Design Qualifications shall apply to the completely assembled CPCI. This level requires a totally integrated computer program. Such testing discloses errors due to conflicts introduced by data sharing convention violations, improper range of input values, sequencing requirements and communications and control. The internal working of the CPCI is of primary concern with the interfaces of the CPCI with the external environ-

ment deferred to the Formal Qualification Tests.

4.1.4 Formal Qualification Tests (Specified in the Part II Specifications)

4.1.5 Category II Tests (Specified in the Part II Specifications)

4.2 Verification Requirements

This paragraph specifies in greater detail the method used to verify the individual requirements given in Table XXXIII. (This table cross-references the sub-paragraphs of 4.2 which apply).

4.2.1 Performance

The specified function shall be verified with respect to one of the following performance criteria.

a. Accuracy which may be affected by input precision, input frequency, input accuracy, or number of iterations.
b. Execution time
c. Storage used
d. Response time
e. Long term degradation
f. Stability

4.2.2 Priority/Timing

The specified function shall be verified with respect to one of the following priority/timing criteria:

a. Interrupt and return
b. Frequency
c. Consistancy in events
d. Order of processing
e. Scheduling/cancelling consistency
f. Job stacking

4.2.3 Interfaces

The specified function shall be verified with respect to one of the following interface parameters:

a. Data locks
b. Range
c. Consistency
d. Initialization
e. Data organization
f. Human command/response
g. External procedures

4.2.4 Logic Paths

The specified function shall be verified with respect to the correctness of the logic paths by exercising the computer program in operation.

4.2.5 Off-Nominal Conditions

The specified function shall be verified with respect to off-nominal conditions such as:

a. Error detection
b. Error recovery
c. Limitations