PERIPHERAL UNIT CONTROLLER
SOFTWARE SUBSYSTEM DESCRIPTION
2-WIRE NO. 1/1A ELECTRONIC SWITCHING SYSTEM

CONTENTS PAGE

CONTENTS PAGE

CONTENTS PAGE

Figures

Tables

1. GENERAL

INTRODUCTION

1.01 This section describes the functional operation of the peripheral unit controller (PUC) software common programs for the No. 1/1A Electronic Switching System (ESS).

1.02 This section is reissued to include pidents PUDA and PUEA. Change arrows are used to indicate these additions and other significant changes.

1.03 This description of the PUC software programs is based on the ▶1E7 and 1AE7◀ generic programs.

1.04 Part 5 provides a listing of the abbreviations and acronyms used in this section and Part 6 contains additional reference material.

SCOPE OF SECTION

1.05 This document delineates the PUC software subsystem as follows:

- Part 2—PUC Operational Overview
- Part 3—PUC Software Overview
- Part 4—PUC Common Software Programs.

2. PUC OVERVIEW

CONFIGURATION

2.01 The PUC is a general purpose microprocessor based system which controls the digital carrier trunk (DCT) and data link (DL) facilities.

2.02 The PUC, Fig. 1, consists of two duplicated controllers, each consisting of a hardcore,

memory, ESS-PUC interface, and a PUC-peripheral interface. The controllers are normally operated in the duplex mode, synchronized by means of a clock. They are initially brought into sychronization by a firmware routine coordinated by the two controllers.

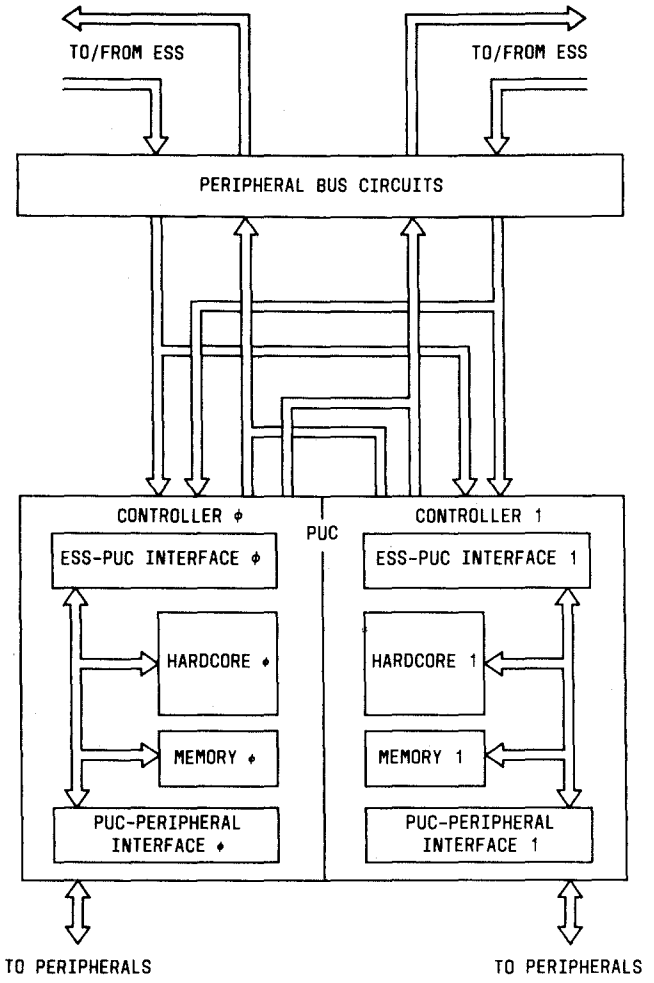
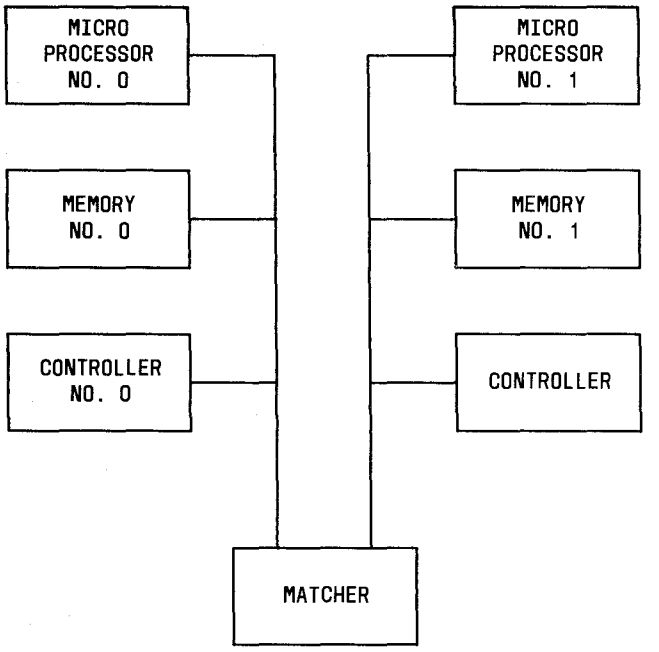


Fig. 1—PUC Block Diagram

2.03 Each hardcore contains two microprocessors, a read only memory (ROM), a random access memory (RAM) which is a write-read memory, and a ♦maintenance circuit which matches the operations of the two microprocessor complexes of the hardcore (Fig. 2).♦



♦Fig. 2—Block Diagram of Hardcore♦

2.04 The status and control of the PUC is maintained via bipolar central pulse distributor (CPD) points, unipolar CPD points, and ferrods.

A. ESS-PUC Interface

2.05 A block diagram of the ESS-PUC interface is shown in Fig. 3.

2.06 The ESS-PUC interface contains a mode and enable circuit, a first-in/first-out buffer (FIFO), and a scan memory. The mode and enable circuit allows the ESS to control and monitor the maintenance states of the controllers, enable the FIFO buffer, and monitor the controller for failures. The FIFO buffer is a first-in/first-out queue in which the ESS places orders that are unloaded periodically by the hardcore. The scan memory is a RAM in which the hardcore stores call processing and maintenance information.

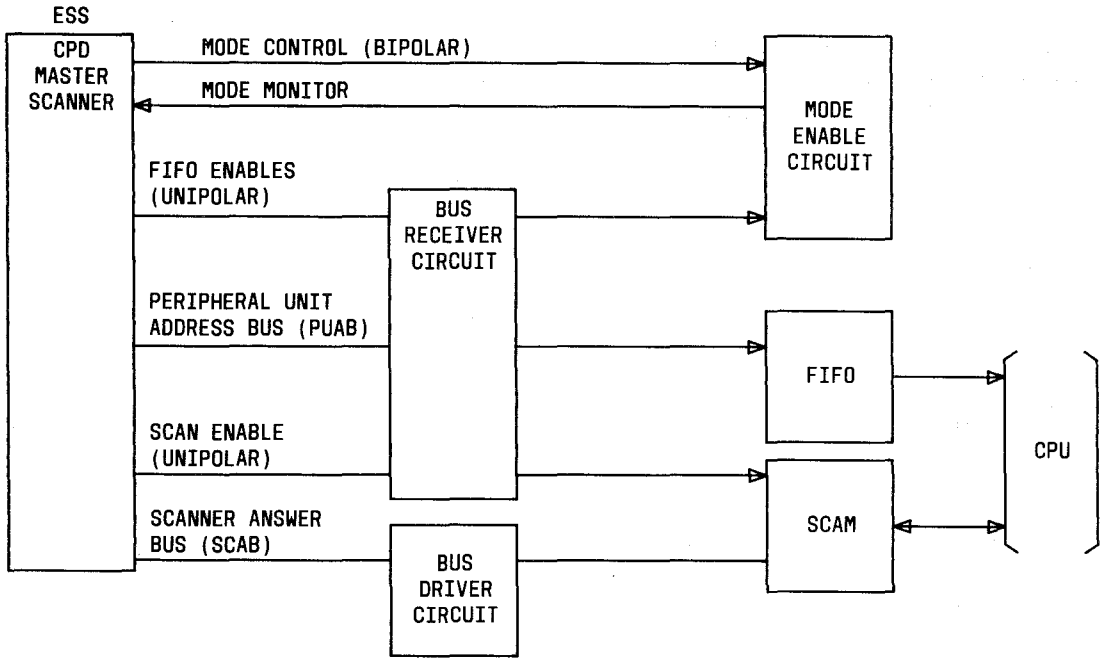


Fig. 3—ESS-PUC Interface

Mode and Enable Circuit (MEN)

2.07 Figure 4 shows a block diagram of the mode and enable circuit.

2.08 The mode and enable (MEN) circuit contains the mode flip-flops which establish the maintenance states of the controller, register controller failures fault flip-flops, and the flip-flops for enabling the FIFO. The mode control block permits the ESS to control and monitor the controller maintenance states in the mode flip-flops. It also contains bus receivers which interface the CPD leads, and scanner drivers which operate the control windings of the scan points used for monitoring the mode flip-flops. The FIFO enable block of the mode and enable circuit control the loading of the FIFO. This block consists of the enable flip-flops and the gating circuitry. The failure indicator block, which allows the ESS to monitor the controller for failures, contains failure flip-flops controlled by the hardcore and scanner drivers, which allow the ESS to monitor the state of the failure flip-flops.

Data Input FIFO Buffer

2.09 Figure 5 depicts a simplified diagram of the FIFO.

Data Input FIFO Controller (DIFC)

2.10 The data input FIFO controller (DIFC) consists of four parts:

(a) Control signal generator—Generates the signals required to allow the PUC to read the output registers

(b) Sequencer—Controls the timing of FIFO read and write operations and prevents simultaneous access by the ESS and hardcore

(c) Address controller—Determines the address at which FIFO is to be read or written

(d) Maintenance matcher—Compares the outputs of the duplicated hardware, and if unequal, signals a failure.

2.11 ***Control Signal Generator:*** The control signal generator is the principal means by which the hardcore accesses the FIFO. The hardcore performs the following actions on the FIFO:

(1) Clears the output register which initiates the movement of the next word in the FIFO to be moved to the output register

(2) Reads the output registers and the words-in-use counter

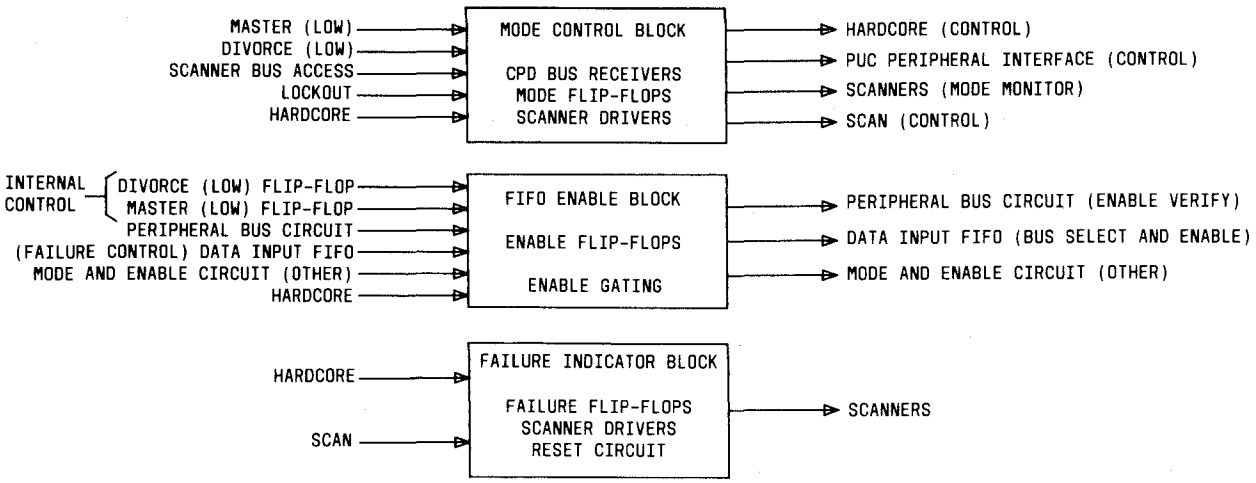


Fig. 4—Mode and Enable Circuit Block Diagram

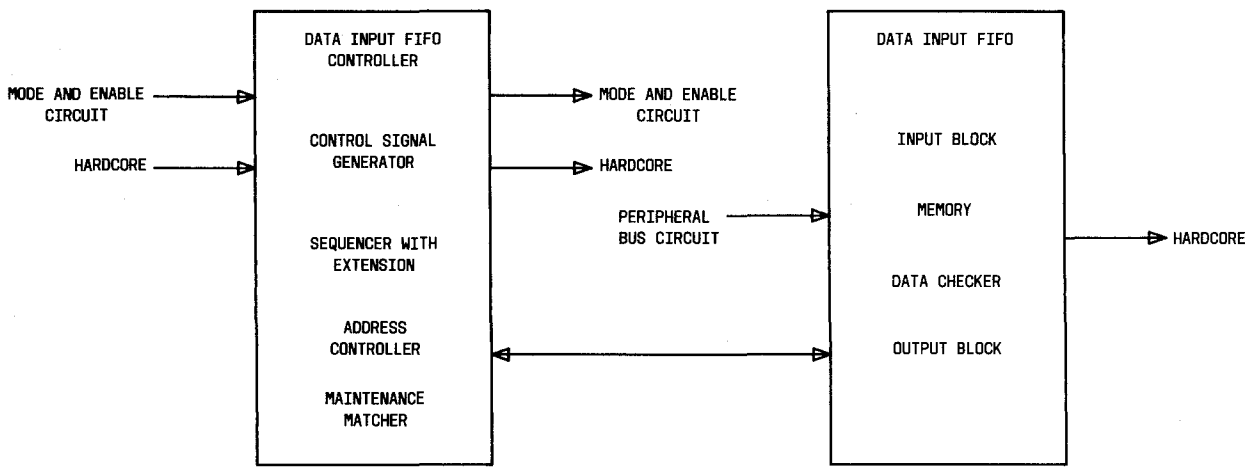


Fig. 5—Data Input FIFO (DIF) and DIF Controller (DIFC) Block Diagram

(3) Forces a mismatch in the maintenance matcher for maintenance checking

(4) Resets counters and pointers

(5) Resets various flip-flops.

2.12 ***Sequencer:*** The sequencer contains two counters: sequencer A and sequencer B. Sequencer A controls the ESS write operation. Sequencer B controls the hardcore read-to-output register and ensures that reads and writes do not conflict.

2.13 ***Address Controller:*** The address controller contains an input counter, an output count-

er, and a words-in-use counter. In addition, it contains a multiplexer which selects the pointer used to address the memory.

2.14 ***Maintenance Matcher:*** The maintenance matcher contains a comparator which monitors selected signals in the FIFO controller for mismatches and signals a failure by setting a flip-flop. Maintenance access is provided at every input of the comparator, allowing the hardcore to force mismatches, thus verifying the correct operation of the matcher.

Data Input FIFO (DIF)

2.15 Figure 6 shows a block diagram of the data input FIFO (DIF).

2.16 The DIF consists of four essential parts:

(a) Input block—Receives the ESS data from the peripheral unit address bus (PUAB)

(b) Output block—Controls the flow of data to the PUC

(c) Memory—Contains two hundred fifty-six 24-bit words

(d) Data checker—Generates check bits which are stored in a check bit memory during write operations and used to check the data for validity.

B. Scan Memory

2.17 The scan memory (SCAM) is the principal means by which information is passed from the controller to the ESS. The SCAM appears to the ESS as a group of six scanners. It appears to the hardcore as a block of memory. The hardcore can both read and write the SCAM. The SCAM receives enables and addresses from the ESS and returns 16 bits of data via the scanner answer bus.

2.18 A block diagram of the SCAM is shown in Fig. 7. It contains the following:

- Addressing block
- ESS read sequencer
- Control signal generator
- Memory block
- Data input-output block.

Addressing Block

2.19 The addressing block transforms the enables, starts the sequencer, and generates the enable verify. When an enable is received at the input register, it is encoded into a binary address. This address is then decoded and combined with the original signal to produce the enable verify. The enable circuitry also

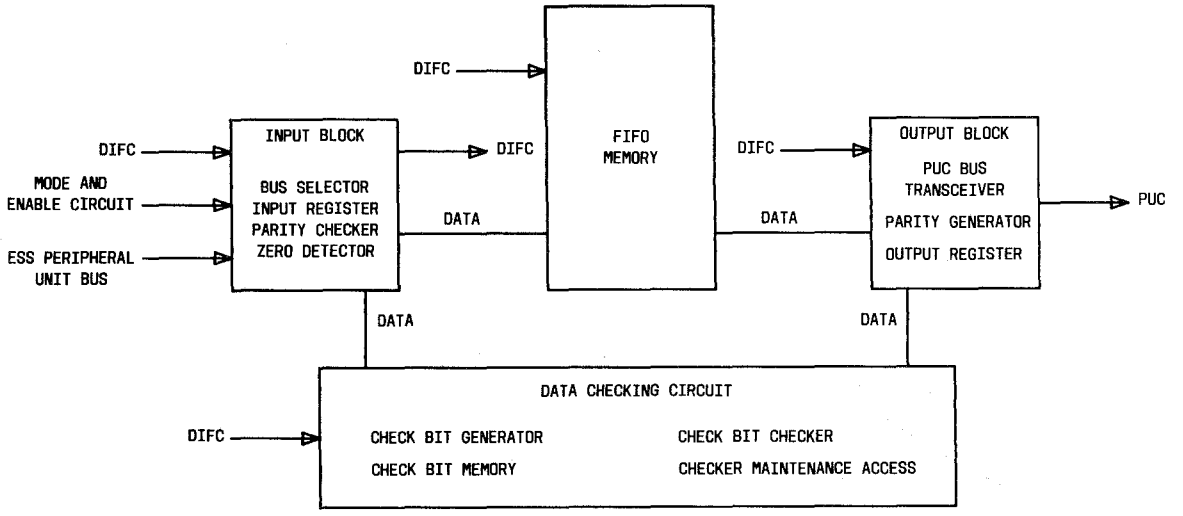


Fig. 6—DIF File Block Diagram

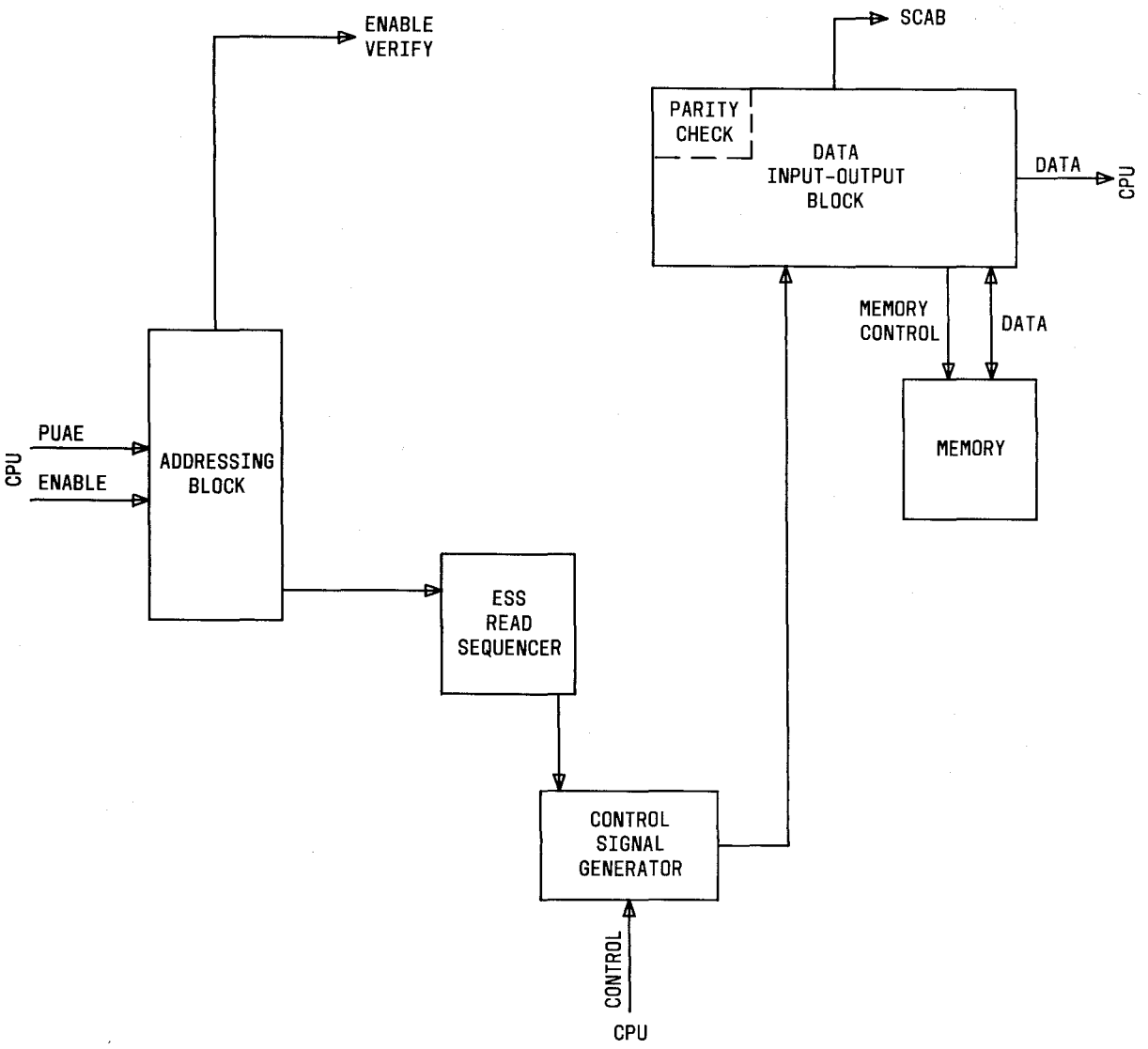


Fig. 7—Block Diagram of SCAM

controls the multiplexer which selects the peripheral unit bus.

ESS Read Sequencer

2.20 The sequencer governs the operation of the ESS read. It is initiated when the enable (active high) is received from the address block. The sequencer responds with signals that block the hardcore access to the SCAM, enables the memory, gates the data into the scanner address bus, and resets the input registers in the address block.

Control Signal Generator

2.21 The control signal generator decodes instructions received from the hardcore and produces the signals necessary to execute the instructions.

Memory

2.22 The memory consists of two parts, namely, the data memory and check bit memory. Whenever data is written into the data memory, check bits are generated and stored. Whenever data is read

from the data memory, check bits are regenerated and compared with those in the check bit memory.

Data Input-Output Block

2.23 The data input-output block contains the output buffers for the ESS data, the input-output transceivers for the hardcore data, and a parity checker. The parity checker performs the following actions:

(a) Checks parity on data received from the hardcore

(b) Generates parity over either byte of data being sent to the hardcore

(c) Generates parity over the whole word before sending it to the ESS.

PERIPHERAL BUS INTERFACE

2.24 The PUAB and the scanner address bus (SCAB) provide supervision interface between the central control and the PUC. Data is transmitted from the central control to the PUC over the PUAB. The SCAB carries the data that is transmitted from the PUC to the central control.

3. PUC SOFTWARE OVERVIEW

FUNCTION

3.01 The PUC software subsystem programs provide the capability for the use of a PUC to control the switching process between the central control and the DCT and DL transmission facilities. Figure 8 shows a block diagram depicting the PUC subsystem functional interface between the ESS application programs and the PUC.

A. PUC-ESS Communication

3.02 The central control communicates with the PUC via orders. The PUC communicates with the central control via messages loaded in a dedicated area of the PUC memory called the message buffer. The message buffer appears to the central control as a scanner.

3.03 Each PUC contains a duplicated message buffer. Both of the message buffers are normally operated in the duplex mode (both buffers operable, one active and the other on standby).

3.04 Data is transmitted from the central control to the PUC over the PUAB in data blocks

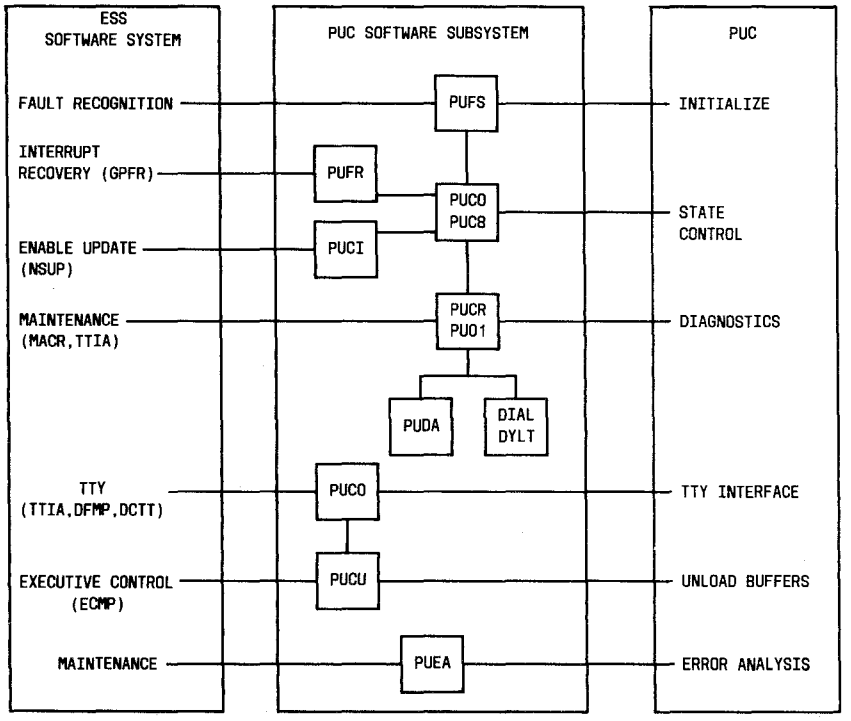


Fig. 8—PUC Software Subsystem Functional Interface

(messages). All data sent to the PUC passes through the FIFO buffer stack.

Buffer Unload

3.05 Message buffers are unloaded during the class C level job schedule. Each time a PUC message buffer is unloaded, a finished unload order is sent to the PUC from the base level program. This order informs the PUC to update the control word that is used to prevent the unloading program from unloading a message the second time. The message buffer, associated with the master controller, is unloaded in the duplex mode. The inactive buffer is unloaded, when requested by maintenance, in the simplex mode.

B. PUC Maintenance

3.06 Software maintenance is based on:

(a) The ESS has ultimate control of all PUC maintenance states.

(b) The PUC provides the first level of fault recovery and all internal diagnostic routines.

(c) The ESS controls diagnostics of the PUC-ESS interface circuitry.

(d) The SCAB contains parity.

SUBSYSTEM PROGRAMS

3.07 Table A lists the PUC software subsystem common programs described in this section.

4. PUC COMMON SOFTWARE PROGRAM DESCRIPTION

PERIPHERAL UNIT CONTROLLER INITIALIZATION (PUCI)

A. Function

4.01 The peripheral unit controller initialization (PUCI) program is entered from the enable table maintenance routines (NSUP) program to:

(a) Initialize PUC fixed memory head cells

TABLE A

PUC SUBSYSTEM COMMON PROGRAMS

ACRONYM	PIDENT LISTINGS		NAME
	NO. 1 ESS	NO. 1A ESS	
DIAL	1A035	6A035	Diagnostic Interpreter
DYLT	1A463	6A463	DIAL Phase Table
PUCI	1A448	6A448	Peripheral Unit Controller Initialization
PUCO	1A448	6A448	Peripheral Unit Controller Input/Output Control
PUCR	1A473	6A473	Peripheral Unit Controller Diagnostic Routines
PUCU	1A450	6A450	Peripheral Controller Unloader
PUDA	1A100	6A100	Peripheral Unit Controller Data Analysis
PUEA	1A831	6A831	Peripheral Unit Controller Error Analysis
PUFR	1A617	6A617	Peripheral Unit Controller F-Level Recognition
PUFS	1A448	6A448	Peripheral Unit Controller F-Scan
PUCO	1A953	6A953	Peripheral Controller State Control Module
PUC1	1A954	6A954	Peripheral Unit Controller State Control Module
PUC2	1A955	6A955	Peripheral Unit Controller State Control Module
PUC3	1A956	6A956	Peripheral Unit Controller State Control Module
PUC4	1A957	6A957	Peripheral Unit Controller State Control Module
PUC5	1A958	6A958	Peripheral Unit Controller State Control Module
PUC6	1A959	6A959	Peripheral Unit Controller State Control Module
PUC7	1A960	6A960	Peripheral Unit Controller State Control Module
PUC8	1A961	6A961	Peripheral Unit Controller State Control Module
PU01	1A473	6A473	Peripheral Unit Controller Diagnostic

(b) Initialize maintenance buffer and application status head cells

(c) Initialize all enable and master scanner number (MSN) tables

(d) Initialize PUC application

(e) Initialize F-scan call store enable tables.

B. Operation

4.02 The program is structured in format consisting of three global program units and thirteen nonglobal program units. Table B lists the subroutines in structured sequence.

4.03 A diagram of the functional interface between the program units is depicted in Fig. 9.

PUC Initialization

Head Cell Initialization—PUCOMP

4.04 Upon entry to global PUCALL from NSUP, subroutine PUCOMP is called to initialize the PUC Compool memory. Subroutine PUCOMP transfers to subroutine PUEAHC which initializes the fixed call store enable table head cell with the param-

TABLE B

PUCI PROGRAM UNITS

NAME	FUNCTION
PUCALL	Initializes all PUCs
PUCOMP	Initializes PUC Compool memory
PUEAHC	Initializes fixed CS enable head cell
PUMAHC	Initializes fixed CS maintenance buffer head cell
PUINIT	Administers initialization of a single PUC
PUIOIN	Initializes enable tables
PUEAIN	Initializes enable table for specific PUC
PUMSIN	Initializes expanded F-scan MSN pointer
PUAPSP	Stores application status block pointer
PUSTATE	Establishes bus and CPD for enable
PUPAPP	Initializes a specific PUC application
PUDLIN	Establishes PUC-RSS interface exchange data for final phase
PUINDT	Establishes PUC-DCT interface exchange data for final phase
PUCIFS	Initializes F-scan MSNs
PUCAPP	Administers non-phase PUC application enables
DCT_INIT	Correct current PUC-DCT SDs and TSs initializes PUC-DCT

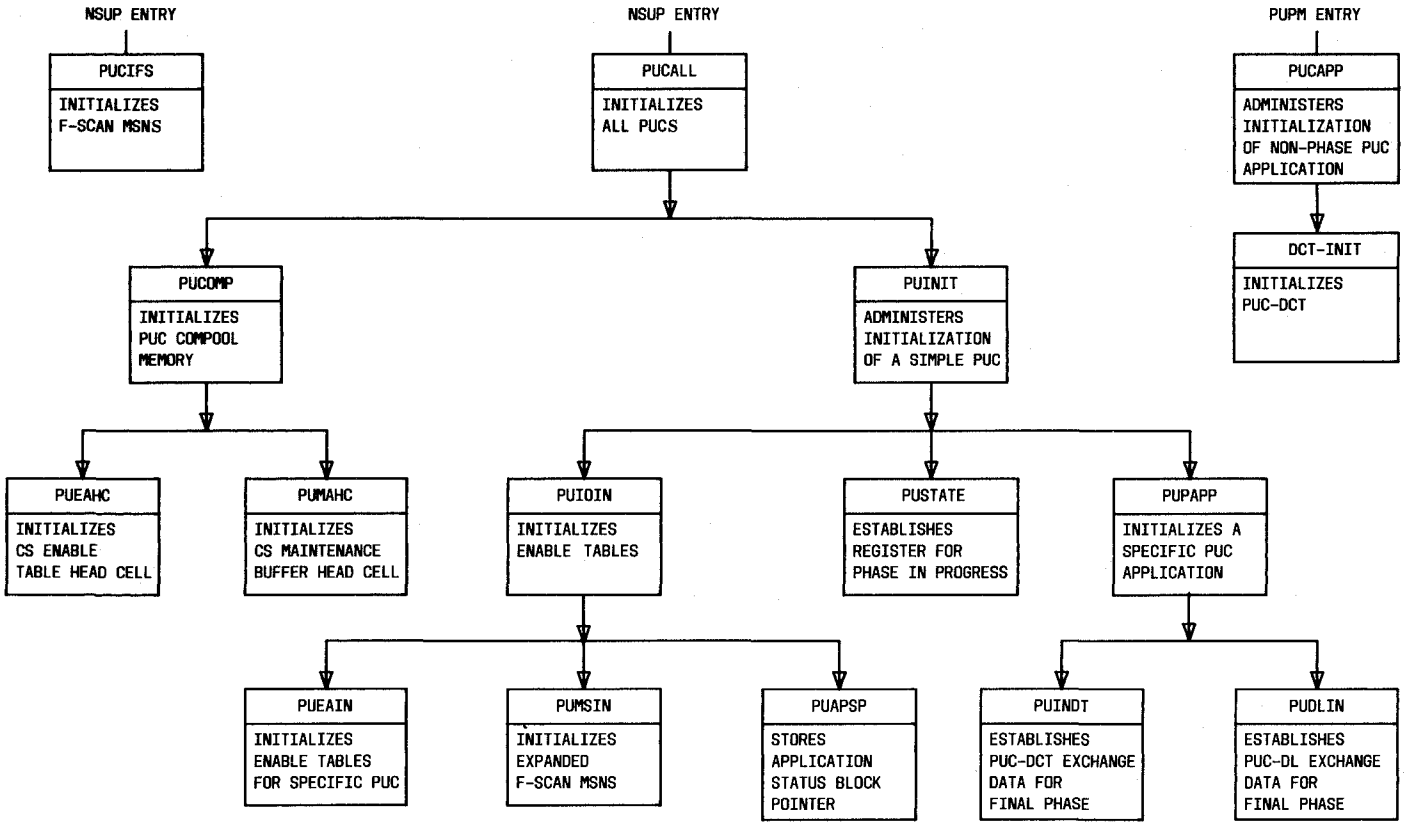


Fig. 9—PUCI Program Structured Diagram

eter stored data. Upon return to PUCOMP, a transfer is made to subroutine PUMAHC. This subroutine initializes the fixed call store maintenance buffer head cell. The buffer end address is placed in the head cell and program control is returned to PUCOMP which gives it to its client, PUCALL.

Initialization Control—PUINIT

4.05 Subroutine PUCALL then calls subroutine PUINIT to administer the initialization of each PUC. This subroutine is the control point for the initialization. It calls two subroutines, PUIOIN and PUSTATE, which determine configuration, request route update via state control, and initiate specific application for each PUC.

Input-Output Memory Initialization—PUIOIN

4.06 Subroutine PUIOIN utilizes subroutines PUEAIN and PUMSIN. Subroutine PUEAIN initializes the current PUC enable block. The PUIOIN subroutine calls subroutine PUMSIN which initializes the pointer to the expanded F-scan MSNs associated with the PUC. After storing the pointer in the enable table, program control is returned to PUIOIN. A transfer is then made to subroutine PUAPSP to store the application status block pointer. It determines the DCT and the Remote Switching System (RSS) applications and associated block memory sizes. Program control is then returned to subroutine PUIOIN.

Phase Initialization—PUSTATE

4.07 Subroutine PUSTATE is called by subroutine PUINIT to establish the peripheral bus and CPD to be used in communicating with the PUC frame. Upon control return, subroutine PUPAPP is called.

PUC Application Initialization—PUPAPP

4.08 Subroutine PUPAPP is a control routine for the emergency action initialization of PUC application data. For DCT applications, subroutine PUINDT is called to establish the PUC-DCT interface exchange data. For RSS applications, subroutine PUDLIN is called to establish the PUC-RSS interface exchange data.

Nonphase Initialization—PUCAPP

4.09 Global PUCAPP is entered from PUC3 and PUC4 to determine the initialization of a specific nonphase application of the PUC. Subroutine DCT_INIT is called by PUCAPP to initialize the PUC-DCT function. This subroutine corrects the current PUC-DCT universal signal distributor (SD) and universal trunk scanner (TS) data. Subroutine PUCDL_INIT is utilized to update the parameter and sample block data for the RSS application.

F-Scan MSN Initialization—PUCIFS

4.10 Global PUCIFS is entered from NSUP to initialize all the F-scan MSNs. The PUCs are initialized eight at a time. When the initialization is completed, program control is returned to NSUP.

PERIPHERAL UNIT CONTROLLER INPUT/OUTPUT CONTROL (PUCO)

A. Function

4.11 The peripheral unit controller input/output control (PUCO) program performs two basic functions, namely, (1) unload maintenance buffer, and (2) service PUC TTY input maintenance support messages.

B. Operation

Main Program Class E Update

4.12 A class E periodic entry is made from the main program (Job #389) to subroutine PUFUPD to ensure that the enable table FIFO pointers are updated. If an update is required, a transfer is made to subroutine PUCFCT which updates the FIFO counts in the enable table.

Main Program Class E Buffer Unload

4.13 Subroutine PUMHUL is entered from the main program as a class E Job #381. This subroutine unloads the maintenance hopper buffer. The removed message is verified per the valid message table and is passed to the appropriate processing routine. All maintenance messages from the PUC pass through this routine. Therefore, PUMHUL acts as an executive for the message handling routines. It enforces a 10-ms time limit on all other subroutines.

4.14 Subroutine PUMMUL is utilized by clients of the PUMHUL subroutine to unload multiple word messages from the maintenance buffer into specified memory areas. The PUMMUL subroutine is

called by the TTY output message subroutines PUCRED, PUCHSH, PUCRIT, and PEUROR and all clients of PUMHUL who receive the multiple word messages.

TTY Input Message Routines

Read and Write Memory

4.15 When the TTY input message PUC-READ is entered, a transfer is made from the TTY input messages—directory and catalog (TTIA) program to subroutine PUREAD. This subroutine formats the data for subroutine PUCSND which sends the order DUMP_32 to the PUC. The DUMP_32 order message is accepted by subroutine PUCRED which also formats the data for the output message PUCREAD. The output message contains data starting with the byte address specified and includes 32 consecutive bytes.

4.16 Subroutine PURITE is entered when TTY input message PUC-WRITE is typed. This subroutine formats the data for subroutine PUCSND which sends the order requesting the PUC to write the data at the specified location in its memory. A PUC-WRITE output message is returned verifying that one byte of data was written as requested. This message is passed to routine PUCRIT which produces a PUC-WRITE complete TTY output message.

4.17 When TTY input message PUC-HASH is entered, a transfer is made to subroutine PUHASH. This subroutine formats the data and requests the PUC to compute a 16-bit checksum over its program store.

4.18 Subroutine PUCHSH accepts the 16-bit checksum message and causes a printout of the TTY output message, a 16-bit octal response.

4.19 Whenever an invalid TTY input message is encountered, subroutine PUEROR formats the data and provides the TTY output error message PUCERROR.

4.20 In the case of PUC maintenance message buffer overflow or an invalid head cell condition, subroutine PUBFAL is called to print the message loaded in the maintenance buffer.

Transmittal of Orders

4.21 Orders generated by TTY input message subroutines, PUREAD, PURITE, and PUHASH are transmitted by the PUCSND subroutine to a given PUC. A regulation action on orders sent per PUC program cycle is maintained so that the PUC is not saturated with ESS orders which limit call processing capabilities. A limit of eight orders can be bypassed by using the force bit.

Data Conversion

4.22 Data (ASCII hexadecimal characters) processed by subroutines PUREAD and PURITE is converted to binary values by subroutine ASCBIN. This subroutine converts up to four numbers and packs the result in the location of the first number.

PERIPHERAL UNIT CONTROLLER UNLOADER (PUCU)

A. Function

4.23 The executive control main program (ECMP) transfers to the peripheral unit controller unloader (PUCU) as one of its class C jobs to unload the call processing and maintenance messages from the PUC message buffer located in the PUC RAM memory.

B. Operation

Message Buffers

4.24 Each PUC has two message buffers (one for each controller) which are normally operated in a duplex mode. The message buffer, associated with the master controller, is unloaded in this mode. The inactive message buffer is unloaded, when requested by maintenance, in the simplex mode (only one buffer is operable). In this case, the inactive message buffer is unloaded immediately after the active one has been unloaded.

4.25 The maintenance messages are separated from call processing messages and stored in a dedicated maintenance call store buffer. Call processing messages are passed.

Program Structure

4.26 The PUCU is divided into seven program units (Fig. 10):

(a) PUPREP—Makes preparation to unload a PUC buffer

(b) UNLOAD—Unloads a single PUC message buffer

(c) TOG_MISMATCH—Processes toggle mismatches

(d) UPDATE_LL—Prints PUC message when control word is invalid

(e) SEND_FIN_ORDER—Sends finished unload order

(f) LOAD_MAINT_MESS—Stores message into maintenance buffer

(g) OFF_LINE—Makes preparation to unload an off-line buffer.

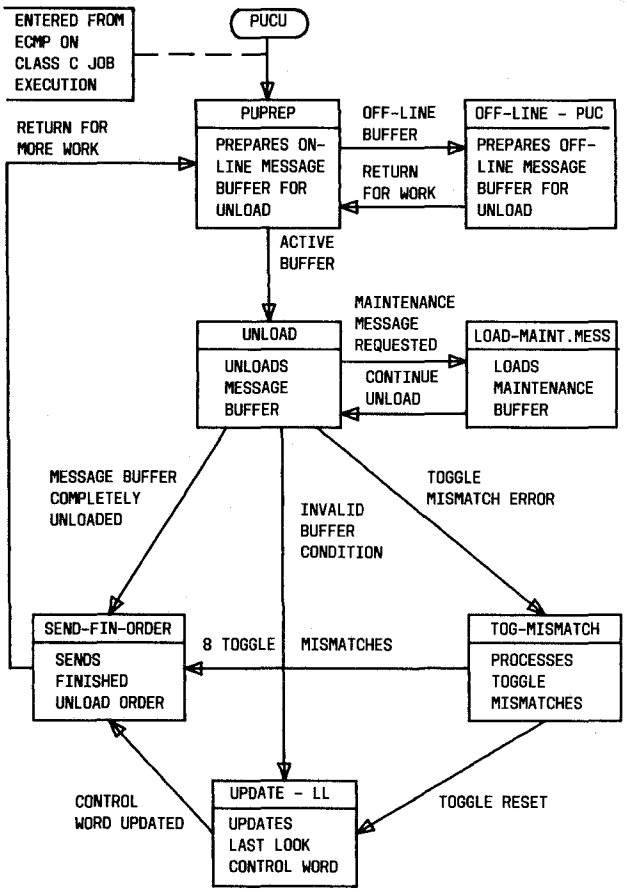


Fig. 10—PUCU Program Structured Diagram

Preparation to Unload

4.27 Subroutine PUPREP is entered from ECMP during its class C job execution to unload the PUC message buffers. This subroutine loads the pointers for the active and off-line buffers which are to be unloaded by the UNLOAD routine.

Buffer Unload

4.28 The PUC message buffer is unloaded by subroutine UNLOAD. Upon entry, the buffer control word is examined for a valid condition (start row equals the end row from the last look and toggle bit equals toggle bit in last look). If the control word is invalid, a transfer is made to subroutine UPDATE_LL. The control word is checked to determine if the buffer contains messages. The active message buffer is unloaded and, if requested by maintenance, the inactive message buffer is unloaded.

Toggle Mismatch

4.29 Subroutine TOG_MISMATCH allows time for the PUC to respond to the finished unload order during lightly loaded conditions. A toggle mismatch condition is allowed to fail eight times before another finished order is sent.

Invalid Control Word Print

4.30 Subroutine UPDATE_LL is entered when the control word that was read from the PUC is invalid. This subroutine updates the control word with the contents of the PUC message buffer control word.

SEND_FIN_ORDER

4.31 Subroutine SEND_FIN_ORDER sends the finished unload order to the PUC FIFO from the base level client.

LOAD_MAINT_MESS

4.32 Subroutine LOAD_MAINT_MESS is responsible for loading the maintenance buffer. Messages may be either single word or multiple word messages. If there is insufficient room in the buffer to load a message, the buffer is marked full, message is lost, and the message lost indicator is set.

OFF_LINE_PUC

4.33 Subroutine OFF_LINE_PUC is entered from the PUPREP subroutine when the off-line

indicator is set indicating a request by maintenance. This subroutine unloads the off-line PUC message buffer after the active message buffer is unloaded.

PERIPHERAL UNIT CONTROLLER F-LEVEL RECOGNITION (PUFR)

A. Function

4.34 The purpose of the peripheral unit controller F-level recognition (PUFR) program is to determine and isolate the PUC fault which caused the F-level interrupt.

4.35 Program PUFR contains two global subroutines:

(a) PUFLEV—Locates and isolates the PUC fault

(b) PUFLRC—Entered from CPFR to verify the system was able to switch the CPD and bus.

B. Operation

Interrupt Level Entry

4.36 Subroutine PUFLEV is entered at an F-level interrupt to determine the source of the PUC operational failure. The data defining the PUC is stored via subroutine STFINF located in the CPD fault recognition program (CPFR). A check is then made to determine if the frame is equipped. If so, the F-level counters (CPD activity counts and system activity counts) are updated. An error check is then performed to ensure that a failure is not repeated.

4.37 The updated counters are examined to determine if at least nine failures have occurred within the last eight minutes. If more than nine failures occur, the failing enable activity count is examined. Then if the activity count is greater than three-fourths of the failure counts, maintenance is removed from the controller. The order is retried and, if the retry order fails, the status of the frame is obtained.

Simplex State

4.38 If the frame is in the simplex state, the order is retried using an alternate route.

4.39 If the second order attempt fails on the alternate route and if an alternate SCAB is available, the order is retried again. If this second attempt fails, a fault is indicated. The original SCAB configuration is restored, and the data relating to the failure is formatted for a TTY F-level printout.

4.40 If the second order attempt is a success using the alternate route, an enable update is made, an F-level interrupt is initiated, and maintenance is removed from the frame.

Duplex State

4.41 If the frame is operating in the duplex mode, the enables are updated to the new controller and route. If a failure occurs on the new controller, the order is retried using an alternate route. Then if this second attempt is a success, the enables are updated, an F-level interrupt is initiated, and maintenance is removed from the frame.

4.42 If the order was a success when using the alternate route, the F-level fault is printed by the TTY.

CPD and PU Bus Test

4.43 A transfer is made to subroutines CPPUCK and CP2TST when an enable is required. Subroutine CPPUCK handles the testing of the CPD and peripheral unit (PU) bus for scanner enable orders. Subroutine CP2TST handles the testing of the CPD and PU bus for all other types of orders.

CPD and PU Verification

4.44 Program PUFR is entered from the CPFR program at subroutine PUFLRC to verify that the system was able to switch the CPD and PU bus when the request was made. If the enables are not updated, a fault is reported and maintenance is removed from the frame.

PERIPHERAL UNIT CONTROLLER F-SCAN (PUFS)

A. Function

4.45 The peripheral unit controller F-scan (PUFS) program administers PUC hardware and PUC application faults detected by peripheral fault scanning. It also monitors the state (simplex or duplex) of the PUC for unloading buffer messages.

B. Operation

4.46 Program PUFS contains two global subroutines.

(a) PUFSMM—Determines state (simplex or duplex) of PUC, detects PUC application and hardware faults

(b) PUWORK—Determines whether deactivated peripheral order buffers (POBs) should be reactivated or placed in a failure state.

PUC State

4.47 An entry is made to global PUFSMM to determine the state (simplex or duplex) of the PUC.

4.48 The primary function of PUFSMM is to administer any PUC hardware fault or application fault. The PUFSMM subroutine calls subroutine TRUTYN in basic trunk translation routines (TRBT) program to obtain the PUC auxiliary block address. After determining which controller is active, the hardware and control ferrods are scanned to determine if a PUC controller fault exists.

Duplex State Fault

4.49 If a hardware fault is detected, a delay in POB input-output execution is made by subroutine POB_DELAY. This action allows the detection of possible application faults during the next machine cycle. Subroutine PUPAM of the state control module (PUC0-PUC8) program is called to allow state control to take corrective action on the fault.

Simplex State Fault

4.50 Subroutine PUPAM is then called to allow state control to take corrective action.

Application Fault

4.51 Application faults are detected by performing scans on the auxiliary memory. If an application fault is detected, action is taken to remove the POB associated with the fault. For a PUC-DCT application the most current application failures are separated from the older failures by subroutine POB_AEA_SRCH. The POB associated with each new failure is deactivated. In the case of a failing order, a transfer is made to subroutine MRKOB in the network management (NMRF) program to mark the POB which contains the address enable address (AEA) of the failing order.

Deactivated POB

4.52 Global PUWORK is entered to determine whether a deactivated POB should be either reactivated or failed. The result is based on the success or failure of the PUC retry order and on the failure option.

4.53 The address of a failing POB is located by subroutine NMP4WD of the NMRF programs. Subroutine FOPR_ZERO processes the failure option 0 work. The failing POB is then placed on the maintenance unexpected result list (MURL). Subroutine PUSND, located in the PUCO program, is called to send clear messages to clear the F-scan application fault data.

▶STATE CONTROL MODULE (PUC0-PUC8)

A. Function

4.54 The state control module (PUC0-PUC8) program is made up of nine pidents (PUC0, PUC1, PUC2, PUC3, PUC4, PUC5, PUC6, PUC7, and PUC8). This program administers and maintains control of the state of the PUC maintenance and diagnosis.

PUC Maintenance States

4.55 The PUC maintenance is based on the following 14 possible frame states:

(a) ACTIVE:STANDBY-SUPPORT—This is the state for normal operations. It signifies a frame with full availability and the controllers are up and matching. One controller is designated ACTIVE and the other in a standby support role. This mode provides for a smooth transmission to simplex operation should either controller experience a fault or an access problem is encountered on the ACTIVE controller from the HOST. The application function should be undisturbed by any single fault when in this mode.

(b) ACTIVE:OUT OF SERVICE-OFF-LINE—Routine exercise of the standby controller is accomplished at the midnight routine. When the PUC diagnostic is in MAC and the routine exercise bit has been set, the diagnostic requests the standby controller to be made available for diagnostic. The state of the frame is then classified as ACTIVE:OUT OF SERVICE OFF-LINE. This state thus represents a state with full availability, but operating in a simplex environment for the duration of the diagnostic. Should a fault occur on the controller designated ACTIVE during this period, a hard switch is ordered bringing in the OFF-LINE controller and initializing it.

(c) ACTIVE:OUT OF SERVICE-MANUAL—This state reflects a frame that ostensibly has two all tests pass (ATP) controllers but has removed one to a nonfunctional mode awaiting manual intervention. The transition to this state is effected by: manual request, which can only be accommodated from a ACTIVE:STANDBY SUPPORT state, or by the automatic system reacting to a failure of the standby to achieve STANDBY SUPPORT or the transition to ACTIVE:STANDBY SUPPORT exceeding an hourly allowed threshold. The mechanism for return of this out-of-service-manual controller to the system is via a TTY restore message.

(d) ACTIVE:OUT OF SERVICE-FAULT—This state signifies a viable controller serving the ACTIVE role and an out-of-service controller that is either verified as some tests fail (STF) or assumed to be bad. Entry to this state is effected by an STF of the diagnostic or an abort of the diagnostic if the previous state was ACTIVE:OUT OF SERVICE REMOVED. Manual action via TTY may also place the out-of-service controller in this state from any simplex defined state.

(e) ACTIVE:OUT OF SERVICE-REMOVED—This state is a transitory state in which the out-of-service controller is awaiting the result of an ordered diagnostic. An ATP indication from the diagnostic will cause an attempt to return the frame to the ACTIVE:STANDBY SUPPORT state. An STF of the diagnostic will effect a transition to a fault state, namely, ACTIVE:OUT OF SERVICE FAULT. The ACTIVE:OUT OF SERVICE REMOVED state can be attained by degrading from a ACTIVE:STANDBY SUPPORT state due to a fault; or an upgrading from a maintenance removed state resultant from a diagnostic ATP; or by virtue of a manual request to restore the out-of-service controller from ACTIVE:UNAVAILABLE FORCED state.

(f) ACTIVE: UNAVAILABLE-FORCED—This state provides retreat for a controller to be exempt from recall by the automatic recovery. The state is entered via a TTY request from either the ACTIVE:OUT OF SERVICE FAULT state or the ACTIVE:OUT OF SERVICE MANUAL state. Diagnostic results will not affect the definition of this state.

(g) ACTIVE MAINTENANCE REMOVED:OUT OF SERVICE-FAULT—This state connotes the same as described for ACTIVE MAINTENANCE REMOVED:OUT OF SERVICE REMOVED except the out-of-service controller is verified as not being an ATP controller. When in this state the TTY PUC-SW-xx. message provides a mechanism to interchange the roles of the controllers.

(h) ACTIVE MAINTENANCE REMOVED:OUT OF SERVICE-REMOVED—This state indicates that neither controller is evaluated as an ATP controller and the out-of-service controller is in a state of evaluation. Transition to this state from any ACTIVE defined state will cause a major alarm.

(i) ACTIVE MAINTENANCE REMOVED: UNAVAILABLE-FORCED—The controller designated UNAVAILABLE-FORCED remains unavailable to the automatic recovery even though a diagnostic may ATP. Manual intervention is required to place this UNAV controller back in the system. The PUC RSTXXX. message will return it to the system.

(j) ACTIVE:OUT OF SERVICE-POWER OFF—This state signifies an up and functioning PUC with reduced availability due to the power down state of the out-of-service controller.

(k) POWER OFF:POWER OFF—This state reflects the consequences of a blown fuse on the controller designated ACTIVE when power has been removed from the alternate. This represents a totally nonfunctional PUC.

(l) ACTIVE MAINTENANCE REMOVED:OUT OF SERVICE-POWER OFF—This state represents a mode where the functional role of the PUC is indeterminate but less than acceptable. Power restored to the out-of-service controller would identify this new resource as ACTIVE.

(m) ACTIVE:PREVIOUS STATE RETURNED—The mode of the PUC in this state is to recognize a power-off condition but treat any restoral of power in a nonnormal sense; ie, restoral of power will not return the controller to the automatic recovery, but rather return it to the state occupied before the removal of power.

(n) ACTIVE MAINTENANCE REMOVED:PREVIOUS STATE

RETURNED—This state indicates the degraded performance of the PUC with the additional constraint that when power is restored to the out-of-service controller no attempt shall be made to declare it ACTIVE in response to the initial declaration that the unit was made unavailable to the system by manual intervention.◀

B. Operation

4.56 The PUC0-PUC8 program contains six global subroutines which provide the following interface:

(a) Supervisory—PUCPWR

(b) TTY—PUSCRE, PUSCSW, PUSCST, PUSCMS

(c) PUC adjudication—PUPAM.

Supervisory

4.57 Subroutine PUCPWR is entered from the system alarm (MALM) program when there is an alarm condition on the PUC. Subroutine PUPAM is called to adjudicate the PUC configuration. The subroutine then transfers to ECMP.

TTY Interface

Change State

4.58 Subroutine PUSCRE provides the craftsperson the capability to change the state of the PUC via the TTY. The message allows the changes of state:

(a) When the PUC is in the duplex mode, the inactive controller may be placed in the maintenance mode.

(b) The manually removed controller may be restored to the system.

(c) If the system has placed a controller in a faulted state or the craftsperson has removed a controller for maintenance, then the input message will make the specified controller unavailable to the automatic system reconfiguration.

(d) The controller may be returned from the maintenance mode to a faulted state.

(e) The controller can be reset and initialized, including a segmented memory pump-up.

Controller Interchange

4.59 The request for interchange of the active and inactive controllers is processed by subroutine PUSCSW, when either controller is in duplex mode and neither is in a manual maintenance environment.

Controller State

4.60 Subroutine PUSCST is entered in response to a TTY input requesting the status of the frame and identification of the active controller.

Controller Maintenance Message

4.61 Subroutine PUSCMS provides means to process the TTY input message used to inhibit or allow PUC state change maintenance type messages to be performed on a priority basis.

PUC Adjudication

4.62 Subroutine PUPAM analyzes the PUC configuration in response to the input diagnostic request.

4.63 The category (administrative or hardware configuration) is determined by subroutine ▶PUACFR◀. The appropriate subroutine is called to perform the necessary actions to allow diagnostics.

4.64 The subroutines perform actions such as:

(a) Determine which controller to diagnose

(b) Perform power and direct scans

(c) Update hardware configuration

(d) Update enable words.

DIAGNOSTIC INTERPRETER (DIAL)

A. Function

4.65 ▶The diagnostic interpreter (DIAL) program contains the subroutines to interpret the test vectors generated by the DIAL macros assembled in program PU01.◀

B. Operation

Global Subroutines

4.66 Global subroutines DIALAU, DIALPD, and DIALST provide interface entries for the maintenance MAC1 interface (MRAM) program. Subroutine DIALAU is entered from the subsystem status table; DIALST is entered from the routine request table for a complete diagnostics; and DIALPD is entered from the routine request table for a partial diagnostic.

4.67 The peripheral unit controller diagnostic (PUCR) program utilizes globals DIALPD, DIALST, DIALTS, and DLVRFY. Subroutine DIALPD is accessed if the input TTY message is a request for a partial diagnostic. Subroutine DIALST provides a complete diagnostic of the PUC when requested via TTY input message PUC-FULDGN.

4.68 If the identified controller is not in a valid state to be diagnosed, subroutine DIALTS will provide a segment break. This subroutine saves the essential registers, transfers to subroutine MACP05 in the maintenance administration control (MACR) program for the segment break and restores the registers.

4.69 During the diagnosis of a PUC controller and after a real-time break has been taken, subroutine DLVRFY is called to verify the peripheral unit bus and CPD paths.

DIAL PHASE TABLE (DYLT)

A. Function

4.70 The dial phase table (DYLT) program is table-oriented and is used in executing the diagnostic phases on the PUC. The program is utilized by the PUCR and DIAL programs.

B. Operation

4.71 Entry is made to DYLT via global subroutine DLOPHT. This subroutine provides the phase table address for the diagnostic phase to be performed on the controller.

PERIPHERAL UNIT CONTROLLER DIAGNOSTIC ROUTINES (PUCR)

A. Function

4.72 The peripheral unit controller diagnostic routines (PUCR) program provides routines necessary to interface the diagnostic of the PUC with the ESS which operates at the base level.

4.73 Program PUCR is composed of eight basic subroutines:

(a) PUCFUL—Provides TTY input interface for PUC diagnostic

(b) PUCPAR—Provides TTY input interface for PUC diagnostic

(c) ⧫PUCMOD—Provides TTY input interface for PUC diagnostic⧫

(d) PUCINT—Provides the initialization of the diagnostic maintenance control scratch area with the PUC unit type auxiliary block data

(e) PUCRDR—Provides the automatic diagnostic request interface with maintenance control

(f) PUCRBS—Initiates segment routine

(g) PUCREP—Terminates phase routine

(h) PUCRAB—Terminates diagnostic routine

(i) PUCRBP—Initiates phase routine.

B. Operation

4.74 In order to diagnose the PUC via the TTY input messages provided by this program, the controller must be in the appropriate maintenance state (off-line controller of a simplex pair). If the requested controller is not in the correct state, the input message is rejected. If, during the diagnostic, the state of the controller changes to an unacceptable state (as a result of system action), the bit will be aborted.

TTY Input or PUC Diagnostic

4.75 Subroutines PUCFUL, PUCPAR, and ⧫ PUCMOD provide the interface for the TTY

input messages PUC-FULDGN, PUC-PARDGN, and PUC-MODDGN◀ respectively. Input parameters are checked for a valid range. If the specified controller can be diagnosed, an entry is made in the maintenance control routine request table (RRT). The PUC-FULDGN message is used to request a complete diagnosis of a PUC. If the request is accepted, either a DR01 or a DR02 TTY output message is returned. The PUC-PARDGN message is used to request a partial diagnosis of a PUC. Diagnostic phases 1 through 6, performed by the PU01 program, will be run automatically even though not specified in the input message. The message parameters provide a selection of phases to be run. ▶The PUC-MODDGN message is the same as PUC-FULDGN except that it allows inputting the number of PROMS, RAMS, and input-output decoders when the number of these devices is being changed.◀

4.76 Subroutine PUCFUL calls subroutine PUCR_TTY to check and set up all the common data required for a full diagnostic printout on the PUC. If the specified PUC can be diagnosed, subroutine PUCR_MAC_RRT is utilized to enter the input request into either the routine request table A or into routine request table B, if there is no room in table A.

4.77 Subroutine PUCPAR utilizes subroutine PUCR_TTY to check and set up all the common data required for a partial diagnostic printout on the PUC. Subroutine PUCRSC is called to check the state of the controller. If the specified PUC is diagnosable, subroutine PUCR_MAC_RRT enters the input request into either the routine request table A or into routine table B, if there is no room in table A.

4.78 ▶Subroutine PUCMOD calls subroutine PUCR_TTY to check and set up all common data required for full diagnostic printout on the PUC. If the specified PUC is in either active-unavailable-force or active-faulted, it will call subroutine PUCR_MAC_RRT to enter diagnostic request into either routine request table A or routine request table B if there is no room on A. The number of PROMS, RAMS and input-output decoders specified in the input will also be stored away for use in the diagnostic. If the number of PROMS, RAMS, or input-output decoders is specified as 0, the value specified by parameters will be used.◀

PUC Auxiliary Block Initialization

4.79 The DIAL program makes a global entry into PUCR at subroutine PUCINT which unloads the data from the unit type auxiliary block into the maintenance control scratch area. The PUCINT address is the first entry in the phase table. This subroutine performs the unit type member number translation for the PUC member. If no auxiliary block exists for the specified member, the diagnostic is aborted. Otherwise, the data is unloaded into the appropriate maintenance control scratch area for the diagnostic.

Automatic Diagnostic Request Check

4.80 Subroutine PUCRDR is entered when maintenance control is searching for diagnostic requests. The PUCRDR subroutine searches the PUC enable blocks looking for diagnostic work. If no work is found, program control is returned to maintenance control for other types of diagnostic work. However if work is found, an indicator is set for maintenance control to perform the work.

Initiate Segment Routine

4.81 Subroutine PUCRBS provides for the DIAL program to begin a segment routine. It is requested upon return from a real-time break during the diagnosis of a PUC. Subroutine DLVRFY is called to verify the current status of the PU bus and CPD paths. The PUC member being diagnosed is verified per subroutine PUCRSC which determines that the PUC member can continue to be diagnosed.

Terminate Phase Routine

4.82 At the end of every phase for which looping is permitted, subroutine PUCREP is entered to check if the current phase is the phase to loop.

> ***Note:*** If a full diagnostic was requested, the entry would be made to pident PUDA before subroutine PUCREP.

If this is the case, this subroutine will overwrite the start and end phase parameters to the expected format (start phase equals current phase and end phase equals 0).

Initiate Phase Routine

4.83 At the beginning of phase 1, subroutine PUCRBP is entered to check if the requested controller is in a valid state to be diagnosed by checking its enable block. If invalid, a segment break is

taken. If after checks the PUC is still not in a valid state for diagnosing, the diagnostic is aborted.

Terminate Diagnostic Routine

4.84 Subroutine PUCRAB provides the DIAL program interface abort diagnostic function. This subroutine clears the active indicator in the PUC enable block. For the DCT function, this subroutine calls a special abort routine for initialization of the DCT frame.

PERIPHERAL UNIT CONTROLLER DIAGNOSTIC (PU01)

A. Function

4.85 The purpose of peripheral unit controller diagnostic (PU01) program is to perform diagnostics on the PUC and print the results on the TTY. The diagnostics are applied in phases, each phase is a sequential set of tests to determine the fault.

B. Operation

4.86 The PU01 program contains 20 sets of tests called phases, sequentially applied dependent upon the successful results of prerequisite phases. The TTY output messages provide raw data to be used for interpreting the failing data words. The phases are:

(1) Phase 1—Tests the power ferrod

(2) Phases 2 through 5—Exercise the scan memory and associated circuitry

(3) Phase 6—Tests the FIFO

(4) Phase 7—Verifies a fault will set the fault flip-flops and the controller can reset them

(5) Phase 8—Tests PROM boards of the controller

(6) Phase 9—Tests RAMs

(7) Phase 10—Tests the memory matching and maintenance circuits in the SCAM

(8) Phase 11—Tests the DIF circuitry

(9) Phase 12—Tests the DIFC circuitry

(10) Phase 13—Tests the hardcore matching and maintenance circuits

(11) Phase 14—Tests the input-output matchers

(12) Phase 15—Tests the input-output matchers

(13) Phase 16—Tests the circuit packs associated with input-output decoder

(14) Phase 17—Tests the direct memory access circuit

(15) ◗Phase 18—Tests fault flip-flops

(16) Phase 19—Scans fault flip-flops◖

(17) Phase 20—Performs the administration functions.

Power Ferrod Test

4.87 Phase 1 of the program contains a single test that scans the controller power ferrods. If the test fails, that is, the PUC does not have power, all remaining phases of tests are skipped. One word of raw data is printed indicating power is off. If the power trouble is corrected, a full diagnostic is normally requested.

Scan Memory and Associated Circuitry Test

4.88 A set of tests, which exercise the scan memory and associated circuitry, are performed by program phases 2 through 5.

4.89 These tests are initiated by a reset pulse sent by the ESS on one of the duplicated reset leads. This causes the controller to execute an initialization sequence, then wait for orders to arrive in the FIFO. After waiting long enough for the controller to finish its initialization sequence, the ESS orders the controller to start testing the SCAM. After the controller verifies that each memory location in the SCAM is working properly, it loads test patterns to be read out by the ESS. The ESS reads the test patterns using each of the four possible bus configurations. When the ESS has finished, it orders the controller to clear the SCAM.

4.90 The above procedure is repeated by the ESS using the opposite reset lead to initiate the sequence. The purpose of the second pass is to test the other reset lead.

4.91 The mode flip-flops are set by the ESS before the test is started. The ESS then scans the

flip-flops to verify that they are in the correct state. If the divorce (DIV) flip-flop is reset, orders sent to the on-line controller may reach the controller being diagnosed. Also, if the scanner access (SBA) flip-flop is reset, the controller being diagnosed cannot be connected to the scanner answer bus and all tests will fail.

4.92 The SCAM is tested after the ESS has reset the controller. If when the order is sent, enable-verify fails or the bus is not available, the ESS retries the order on the other bus. When the ESS completes the SCAM tests, it orders the controller to clear the SCAM. A raw data printout is made which summarizes the enable-verify and bus availability information when there is a reasonable chance that the controller did not receive the order.

4.93 Phases 4 and 5 of the program repeat the tests in phases 2 and 3. The only difference is that the opposite reset is used to initiate the sequence.

4.94 Raw data is printed on the TTY to assist the craftsperson in locating the fault in the circuitry associated with the scan memory, PUAB receiver, encoder receiver, and SCAB driver.

FIFO Test

4.95 Upon successful completion of the SCAM tests, the controller performs the FIFO tests (phase 6). A total of 512 tests are performed during phase 6.

4.96 After clearing the SCAM, the controller waits for test patterns to arrive in the FIFO from the ESS. The ESS then sends 512 test patterns which exercise each bit of the FIFO memory. Each test pattern is returned to the ESS via the SCAM. Both the ESS and controller maintain a current test counter. With each test pattern, the controller returns the value of the counter. The ESS then compares the controller's version of the counter with its version to determine if the last test pattern sent was actually received. Any time a test pattern is not received, it is retried on the opposite PUAB.

4.97 The parity is inverted on half of the test patterns. Enable-verify and parity failures are expected on these tests. If these failures do not occur, the test is considered a failure. The TTY printout indicates both of the expected failure patterns.

Mode Flip-Flop Test

4.98 The ability of the ESS to set, reset, and monitor each of the mode flip-flops is tested during phase 7 of the program. The ESS pulses the appropriate CPD leads and then verifies the order by scanning the ferrods.

4.99 If the first test of phase 7 is a success, the ability of the controller is tested to detect a mismatch between the duplicate master flip-flops, the effectiveness of the lock-out flip-flops, and ability of the controller to set and reset the master flip-flops when the lock-out flip-flop is reset.

PROM Board Test

4.100 Phase 8 of the program tests the PROM boards of the controller. Phases 1 through 6 must be performed before running phase 9 tests.

RAM Test

4.101 If tests in phases 1 through 6 pass, phase 9 performs tests on the RAMs. Raw data is printed on the TTY indicating the faulty RAM and circuit pack.

SCAM Memory Circuitry

4.102 Phase 10 is performed, if phases 1 through 6 pass, to check the memory matching and associated maintenance circuits in the SCAM. A 1-word of raw data is printed, if the test fails, which indicates the faulty maintenance circuit.

DIF Circuit Test

4.103 The DIF circuitry is checked by phase 11 of the program. Phases 1 through 6 are prerequisite. If the test fails, a 1-word raw data is printed.

DIFC Circuit Test

4.104 Phase 12 of the program tests the DIFC circuitry. It is performed upon successful completion of tests in phases 1 through 6. If the test fails, a 1-word TTY output message of raw data is printed.

Hardcore Matcher Test

4.105 The hardcore matching circuits and associated maintenance circuitry are checked by

phase 13 tests. If the test fails, a 1-word TTY output message of raw data is printed. This phase is run only if tests in phases 1 through 6 were successful.

Input-Output Matcher

4.106 Phase 14 connects the remaining circuit packs to the internal bus of the controller and does some cursory tests of the input-output matchers. Phases 1 through 6 are prerequisite. If the test fails, a 1-word of raw data is printed indicating the failure.

4.107 The input-output matcher is tested by phase 15. If the associated circuitry is faulty, a 1-word of data is printed.

Input-Output Decoder Test

4.108 Circuit packs associated with the input-output decoder are tested by phase 16. The faulty circuit is indicated by a 1-word TTY printed output message.

Direct Memory Access (DMA) Circuit Test

4.109 The DMA circuit is tested by phase 17. The prerequisite phases are 1 through 6. If the circuit is faulty, a 1-word TTY message is printed.

Flip-Flop Fault Test

4.110 The fault flip-flops are tested by phase 18. ▶ The prerequisite phases are 1 through 6.◀ An order is sent for the controller to clear all fault flip-flops. The ESS then scans to verify that they were cleared. If they are, the ESS orders the controller to set all the flip-flops by inducing faults in the hardcore and scan memory.

▶Fault Ferrod Check Test

4.111 Phase 19 will reset the controller, put the controller in a maintenance access state, and turn on maintenance. After a delay of 500 ms, if the first 18 phases have passed, a scan of the fault ferrods in the standby controller (test controller) will be done. If they fail, additional tests will be done to provide additional test data from both controllers to help resolve phase 19 failures.◀

Administration Test

4.112 Phase 20 performs the necessary administration functions, such as reporting the termination status of the diagnostic to the PUC state control programs.

▶PUC DIAGNOSTIC DATA ANALYSIS (PUDA)

A. Function

4.113 The peripheral unit controller diagnostic data analysis (PUDA) program analyzes the data produced by the PUC diagnostic pident PU01 and identifies the suspected bad circuit packs. Program PUDA then calls Dictionary Trouble Number Production (DOCT) to print out a word description of suspected faults and a list of faulty circuit packs in decreasing order of probability.

B. Operation

4.114 After completion of a diagnostic phase, pident PUCR enters pident PUDA at routine PUCDDA.

> ***Note:*** This pident has no effect until the PUC diagnostic has an STF phase.

4.115 Pident PUDA first checks if the fault determination flag was set on the previous entry to PUDA. If so, PUC state control will be informed that the diagnostic has failed and control will output the suspected fault on the TTY; then the PUC diagnostic will terminate. If the fault determination flag is not set (which means PUDA needs more data to decide) then PUDA branches to 20 different paths corresponding, respectively, to the 20 phases of PUC diagnostic.

4.116 When interpretation of failing phases 1 and 7 through 19 is done by PUDA, a list of circuit packs associated with that particular phase is printed.

4.117 For phases 2 through 6, the operation of PUDA is described as follows:

(a) Phase 2—Word 1 of phase 2 is checked for a zero or nonzero. If nonzero, then the mode and enable flag is set and PUDA exits. If word 1 is zero, then the failing raw data is analyzed according to the combination of PUAB and SCAB that fail. The number of the failing test is pegged against eight different fault counters corresponding to the eight different fault categories. These categories are:

(1) PUAB0 fault

(2) PUAB1 fault

(3) SCAB0 fault

(4) SCAB1 fault

(5) SCAM, SCAM controller (SCAMC) or hardcore (HC) fault

(6) Power to BUS 0

(7) Power to BUS 1 fault

(8) Others (a catch all category that does not belong to the above seven categories).

There are six other flags besides the mode and enable flag to indicate how the data fails. These flags are:

(1) Low byte flag—Set if there is a nonzero value in bits 0 through 7 of the failing data.

(2) Mid byte flags—Set if there is a nonzero value in bits 8 through 15 of the failing data

(3) Parity flag—Set if bit 16 (parity bit) of the failing data is 1

(4) All seems well flag—Set if bit 17 of the failing data is set

(5) Enable verify flag—Set if bit 18 of the failing data is set

(6) Hardcore flag—Set if word 61 of phase 2 has a value of 10.

(b) Phase 3—Perform the same operation as phase 2 except word 56 is checked for MEN circuit failure.

Note: If word 56 is nonzero, the MEN failure flag is set.

(c) Phase 4—PUDA checks if phase 2 has STF and phase 4 is ATP; then reset lead 0 is considered faulty. If phase 2 is ATP and phase 4 has STF, then reset lead 1 is faulty. Once the fault has been identified, PUDA generates the pointers to the table that contains a list of word descriptions of the faults and a set of pointers to the circuit packs table.

(d) Phase 5—PUDA first checks to find if the MEN flag is set or reset. If set, then, the fault is the MEN circuit fault. If the MEN circuit flag is not set, then PUDA finds the maximum of the eight counters that corresponds to the eight different fault categories. Once the fault categories are determined, the six flags are used to determine the exact fault and its associated suspected circuit packs.

(e) Phase 6—Word 1 of phase 6 is checked for zero or nonzero. If word 1 of phase 6 is nonzero, then the suspected problem is DIF, DIFC, or MEN fault. If word 1 of phase 6 is zero, then the remaining data is examined to determine if there is a bus-related problem.

Note: After examining the remaining data and if determining that the suspected problem is not the bus, then the problem is DIF, DIFC or MEN. This determination is obtained by examining the failing data, whether both buses fail or only one bus fails. If both buses fail, then it is a DIF, DIFC, or MEN problem. If one bus fails, then the failing bus—DIF, DIFC or MEN—in that order are suspected. When the exact fault is determined, associated circuit pack numbers are printed.

PERIPHERAL UNIT CONTROLLER ERROR ANALYSIS (PUEA)

A. Function

4.118 There are eight basic areas within the peripheral unit controller error analysis (PUEA) program that support this deterministic performance evaluation. They are:

(1) Heart Beat

(2) Figure of Merit

(3) PUC Error Resolution

(4) Background Firmware Tests

(5) PUC Memory Access

(6) Internal PUC Hardware State Changes

(7) External PUC Hardware Tests

(8) PUC History Data.

B. Operation

Heart Beat

4.119 The Heart Beat test is a repetitive loop-around order originated at the ESS and returned by the active PUC controller once every six seconds. This order goes through the PUC hardware, firmware, and the ESS interface, thereby providing a means of checking all these related areas for functionability. On each transmission a counter is incremented. A similar count of orders received is also maintained. These counts are compared when 25 orders are sent. The resultant success ratio accordingly classifies the functional performance of the given controller. This percentage is used for the determination of PUC relative ability. When heart beat has an efficiency degrading to less than 75 percent, a more definitive evaluation of the controller is implemented. This evaluation is accomplished by using the Figure of Merit test.

Figure of Merit

4.120 This function performs the Figure of Merit (FM) tests. The FM tests check the access, flip-flop control (ferrods), firmware cycling performance, and ability to maintain on board fault detection for each controller.

4.121 The range of FM is 0 to 9, with a 5 being a threshold for usable performance. A completely satisfactory controller will yield an FM=9. Should the access check, the cycling firmware check, or the (ferrod) flip-flop check produce a value of 0 for their respective contributions to the FM, then the value for the FM is set to 0. Otherwise the weighting is as follows:

(a) Access—No route, one route, both routes satisfactory yield a contribution of 0, 1 and 2, respectively.

(b) Cycling firmware—An 88 percent or greater completed cycles provides a contribution of 2; between 1 percent and 88 percent a contribution of 1 and 0 for less than 1 percent.

(c) Flip-Flop Control (ferrods)—For the active PUC controller, the flip-flop control test is performed in two parts. The first part checks the master (MST) and scanner answer bus (SBA) which yields a contribution of 0 or 2. A 2 is given if both flip-flops operate and 0 if either flip-flop is nonoperative.

Note: In this case, a zero will cause the flip-flop test to end and the FM to yield a contributing factor of 0.

The second part checks the divorce (DIV) and lockout (LKO) flip-flop which yield a contribution of 0 or 2. A 2 is given if both flip-flops operate and a 0 if either flip-flop is nonoperative.

Note: In this case, a zero will not cause the flip-flop test to end or the FM to yield a contributing factor of 0.

The maximum contributing factor yield is 4. For the standby controllers the flip-flops are put through 16 combinations of set and reset.

Note: If 4 flip-flop fails, the resultant is a contributing factor of 0. The maximum contributing factor yield is 4.

(d) Maintenance Retention—The maintenance retention test is used to check the ability of the internal firmware maintenance to remain on and produces a contribution of 1 otherwise 0. These tests are performed manually by (via a TTY message) or automatically (triggered by Heart Beat criteria not being satisfied). The results of the FM test, on each controller, determine the action that will be performed (caused by Heart Beat). There are three general actions that are performed. These actions are:

Note: This check occurs only when the PUC is active maintenance remove: out of service-fault state (ACT_MR_OOS_FLT) but the FM performs in any state.

(1) Standby value <5; Active value >5—When this condition occurs, the automatic analysis will not initiate further action. The values returned indicate the active controller is the better of the two controllers and should function properly. Note that the active is not perfect since its inability to handle the heart beat was the reason analysis was activated in the first place. (See output message manual for MPPF Report 4.)

(2) Standby Value >5; Active Value <5—In this case, analyses will perform a positive ac-

tion. The values returned indicate that the standby controller is more functional than the active controller. In this condition, analysis will cause the standby controller to become active. (See output message manual for MPPF Report 4.)

(3) Standby Value <5; Active Value <5—This situation indicates that neither of the two controllers is in a condition to perform in any working configuration. The most appropriate action to take is initialize the frame with the better of the two controllers declared active. (See output message manual for MPPF Report 4.)

PUC Error Resolution

4.122 The PUC Error Resolution performs seven possible areas of error locating tests. These tests consist of PROM, RAM, I/O, HARDCORE, SCAM, MEN, and DIF. A corresponding counter for each test is used to count the number of errors that occur. When the number reaches seven for any of the counters, an entry into the background firmware test occurs.

Background Firmware Test

4.123 The Background Firmware test is used to diagnose the cause of an error in the PROM, RAM, I/O, HARDCORE, SCAM, MEN, or DIF areas. This function occurs manually (via TTY request) or automatically (via entry from PUC Error Resolution function). The automatic tests are performed on the active controller and manually requested tests are run only on the off-line controller.

> ***Note:*** When the background firmware test occurs manually, a single section of PROM, RAM, I/O, HARDCORE, MEN, or DIF is tested. When the background firmware test occurs automatically, a related section of hardware is tested.

PUC Memory Access

4.124 The PUC Memory Access function provides two capabilities:

(a) To scan (which is used to display any part of the SCAM 0 memory), and

(b) To monitor (which is to monitor any word or bit located in SCAM 0 memory).

Internal PUC Hardware State Changes

4.125 The Internal PUC Hardware State changes will provide the capability to single step the PUC through internal hardware states. This will aid in locating a problem with any controller not able to perform state changes successfully.

External PUC Hardware Tests

4.126 The External PUC Hardware tests consist of loop, access, and configuration tests. These tests provide a way of checking the PUC external hardware.

> ***Note:*** These functions can only be performed on the off-line controller.

4.127 In the loop test the ESS continuously sends a series of four orders (all 0s, all 1s, alternating 1010s, alternating 0101s) to the PUC. These bit changes are used to check the ESS interface connections.

4.128 The access test provides a means of checking the controllers off-line. This test checks the combination (See Table C) of PUAB (0 or 1), CPD (0 or 1) and PUC (0 or 1). If the route chosen has trouble, the system will take an F-level interrupt.

TABLE C

PUC ACCESS ROUTE CONFIGURATIONS

PUC CONTROLLER	PUAB	CPD
0	0	0
1	0	1
1	1	0
0	1	1

4.129 The configuration test checks the controllability of the flip-flops (SBA, LKO, DIV, and MST) by setting or resetting any of 16 possible combinations.

PUC History Data

4.130 The PUC History data block records the last four PUC state control entries and recovery

actions. This history is automatically output whenever any PUC frame goes maintenance removed or upon demand. Associated with each block of information is:

- Time of the event
- PUC state (software and hardware)
- Reason for entry
- Requester
- Ferrods for each controller
- Defined active and standby controllers (software and hardware derived).◀

5. ABBREVIATIONS AND ACRONYMS

5.01 The following abbreviations and acromyns are used within this section.

AEA	Address Enable Address
ATP	All Tests Pass
CCU	Combined Channel Unit
CPD	Central Pulse Distributor
CPFR	CPD Fault Recognition Program
DCT	Digital Carrier Trunk
DIAL	Diagnostic Language Program
DIF	Data Input FIFO
DIFC	Data Input Controller
DIV	Divorce
DL	Data Link
DMA	Direct Memory Access
DYLT	Dial Phase Table Program
ECMP	Executive Control Main Program
ESS	Electronic Switching System
FIFO	First-In/First-Out Buffer
FM	Figure of Merit
MALM	System Alarm Program
MEN	Mode and Enable Circuit
MSN	Master Scanner Number
MURL	Maintenance Unexpected Result List
NMRF	Network Management Program
NSUP	Enable Table Maintenance Routines Program
POB	Peripheral Order Buffer
PU	Peripheral Unit
PUAB	Peripheral Unit Address Bus
PUC	Peripheral Unit Controller
PUCI	Peripheral Unit Controller Initialization Program
PUCO	Peripheral Unit Controller Input/Output Control Program
PUCR	Peripheral Unit Controller Diagnostic Program
PUCU	Peripheral Unit Controller Unloader Program
PUC0-8	State Control Module Program
PUFR	Peripheral Unit Controller F-level Recognition Program
PUFS	Peripheral Unit Controller F-Scan Program
PUPM	State Control Module Program
PU01	Peripheral Unit Controller Diagnostic Program
RAM	Random Access Memory
ROB	Remote Order Buffer
ROM	Read Only Memory

RSS	Remote Switching System
SCAB	Scanner Address Bus
SCAM	Scan Memory
SCAMC	Scanner Address Controller
STF	Some Tests Failed
TRBT	Basic Trunk Translation Routines Program
TTIA	TTY Input Message Directory and Catalog Program

6. REFERENCES

6.01 For further information concerning these programs, consult the following references:

- PR-1A035 DIAL Program
- PR-1A448 PUCI Program
- PR-1A448 PUCO Program
- PR-1A448 PUFS Program
- PR-1A450 PUCU Program
- PR-1A463 DIAL Program
- PR-1A473 PUCR Program
- PR-1A473 PU01 Program
- PR-1A617 PUFR Program
- ⧫PR-1A953 PUC0 Program
- PR-1A954 PUC1 Program
- PR-1A955 PUC2 Program
- PR-1A956 PUC3 Program
- PR-1A957 PUC4 Program
- PR-1A958 PUC5 Program
- PR-1A959 PUC6 Program
- PR-1A960 PUC7 Program
- PR-1A961 PUC8 Program
- PR-1A100 PUDA Program
- PR-1A831 PUEA Program.⧫