Licensed Material – Property of IBM
LY20-2226-0

Program Product

IBM System/370 Energy Management System Logic Manual

Program Number 5740-U11

This publication is intended to provide a description of the internal logic and operation of the IBM System/370 Energy Management System. The intended audience includes user programmers, IBM systems engineers and support center personnel.

IBM

First Edition (March 1976)

This edition applies to Version 1, Modification Level 1, of the program product IBM System/370 Energy Management System (5740-U11) and to all subsequent versions and modifications until otherwise indicated in new editions or Technical Newsletters.

Changes are continually made to the information herein. Therefore, before using this publication, consult your System/360 and System/370 Bibliography (GA22-6822) for the editions that are applicable and current.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments has been provided at the back of this publication. If this form has been removed, address comments to: IBM Corporation, Technical Publications/Industry, Dept. 825, 1133 Westchester Avenue, White Plains, New York 10604. Comments become the property of IBM.

CONTENTS

PREFACE

This manual describes the internal logic and method of operation of the System/370 Energy Management System. This manual is to present information to the Program Support Representative, Systems Engineer, and Systems Programmer who maintain the system. The reader should be familiar with the concepts presented in the following prerequisite publications:

IBM System/370 Energy Management System General Information Manual (GH20-1496)

IBM System/370 Energy Management System Program Reference Manual (SH20-1742)

IBM System/370 Energy Management System Operations Guide (SH20-1743)

IBM System/7 Energy Management System General Information Manual (GH20-1495)

IBM System/7 Energy Management System Program Reference Manual (SH30-1040)

IBM System/7 Energy Management System Logic Manual (LY30-0105)

IBM System/370 Special Real Time Operating System PRPQ Description and Operations Manual (SH20-1773)

IBM System/370 Special Real Time Operating System PRPQ Systems and Logic Manual (LY20-2228)

IBM System/370 Display Management System PRPQ Systems and Logic Manual (LY20-2227)

IBM System/370 Display Management System PRPQ Description and Operations Manual (SH20-1745)

IBM Energy Management System Generation Manual (SH30-0108)

CHAPTER 1. INTRODUCTION

The System/370 Energy Management System is a computer program system designed for use by the electric utility industry. It automates the monitoring and control of electric power generation and distribution. While it does not remove the need for human operators, it does ease the load by presenting the information needed to control the energy system. The System/370 Energy Management System can, through programmed data monitoring coupled with programmed power generation and distribution control, improve the security of the power network and reduce the cost of power.

The System/370 Energy Management System is dependent upon one other program product and two program RPQ's for its operation. The first of these is the System/7 Energy Management System Program Product which operates within a System/7 and interfaces with the IBM 3707 Remote Data Acquisition Control Station and any data or control points that may be attached locally to the System/7. The Energy Management System utilizes a data collection network consisting of one or more IBM System/7 computers and a variable number of IBM 3707s with associated common facilities. The System/370 and System/7 Energy Management System Program Products contain programs that provide an Energy Management System interface between the two types of computers.

The primary operator interface is through one or more IBM 5985 display stations supported by a Display Management System Program RPQ. This device displays information in color to the operator. Information is input to the System/370 Energy Management System through a typewriter-like keyboard and an additional group of key switches, all part of the 5985 display station. The System/370 Energy Management System contains display definitions which are necessary for normal operation. Other definitions are dependent upon the power network to be monitored and therefore must be defined by the user.

The other program RPQ which the System/370 Energy Management System is dependent upon is the Special Real Time Operating System which satisfies System/370 Energy Mangement System's needs for the following realtime functions:

- Data base definition, initialization, and management
- Asynchronous tasking (independent)
- Time dependent task creation/queuing
- Realtime message handler
- Duplicate data set support
- Data base logging

The Special Real Time Operating System enhances the OS/VS1 services to support realtime applications. The services provided by OS/VS1 are still available as a program or system of programs which utilize the Special Real Time Operating System. In some cases, the functions of the Special Real Time Operating System are similar to those of OS/VS1 when it performs through the Special Real Time Operating System service which is tailored to the needs of a realtime application.

Facilities are provided to execute user-written programs in the computer with the System/370 Energy Management System and to use the data that is collected by the System/370 Energy Management System.

CHAPTER 2. METHOD OF OPERATION DIAGRAMS

The following diagrams present the functional flow of the System/370 Energy Management System. The first overview diagram (2.0) shows the major subsystem in the system. Each subsystem box has a diagram number to provide a reference to the subsystem overviews. The subsystem diagrams use the same format to show the overview of the function and to reference detail diagrams.

The detail diagrams describe the specific functions, show specific input and output items, and refer to other detail diagrams. Diagram 1.0 explains the symbols used in the detail diagrams.

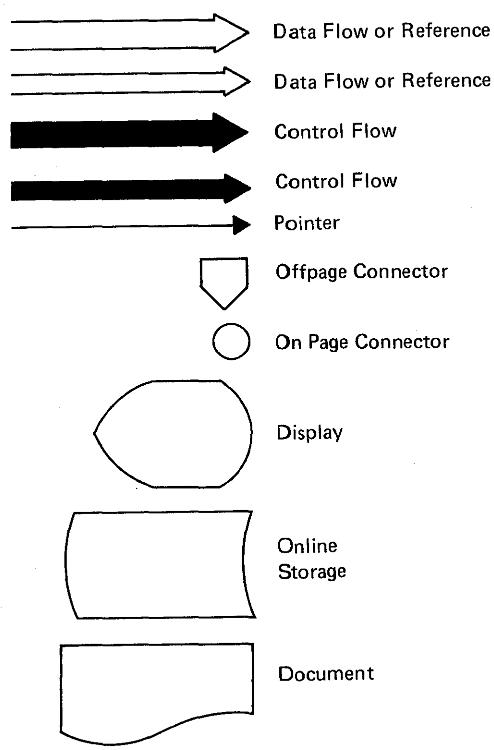


DIAGRAM 1.0: Legend

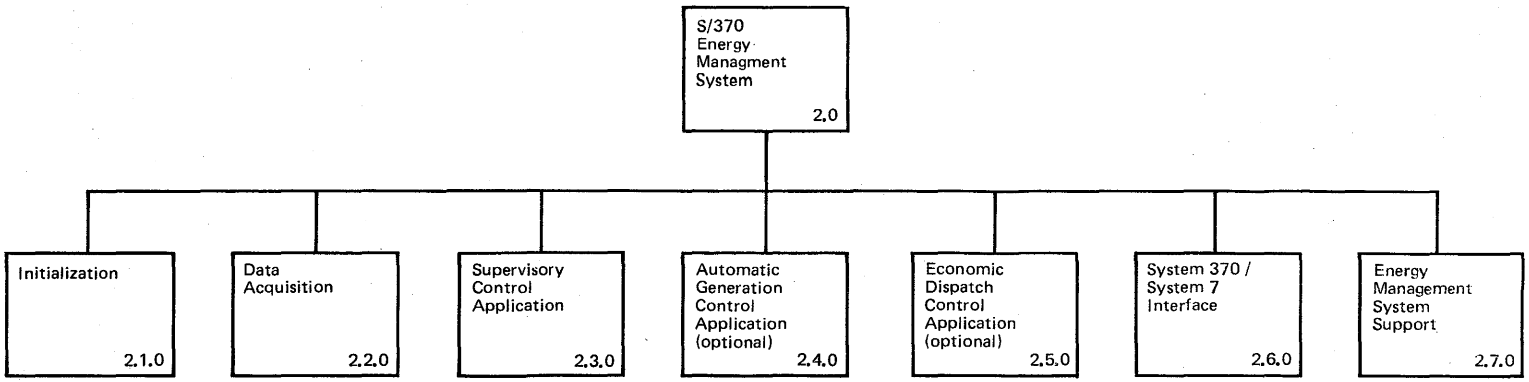


DIAGRAM 2.0: System 370 Energy Management System Functions

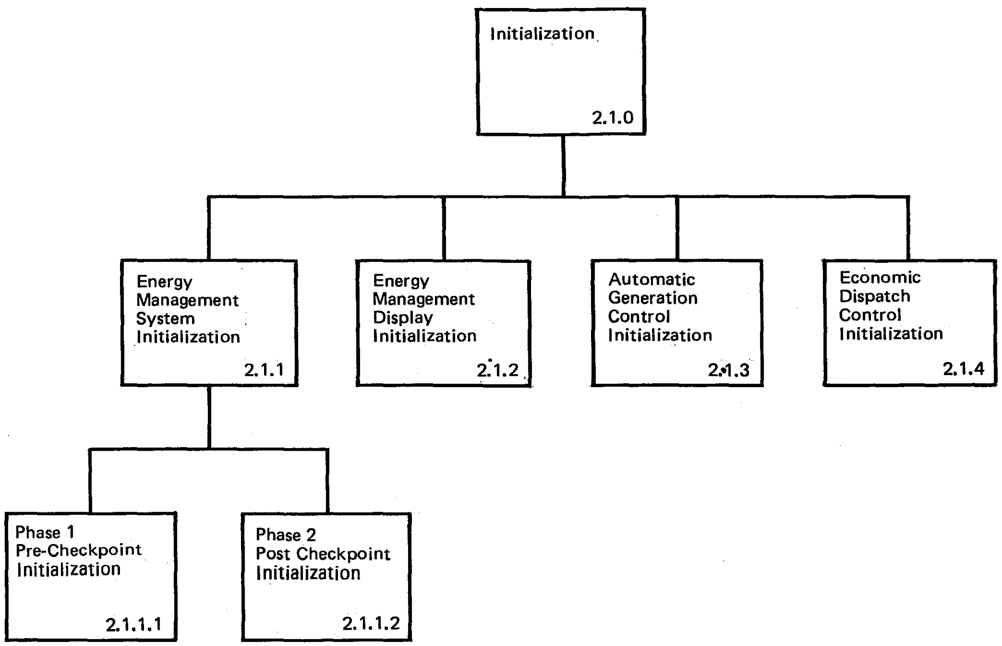


DIAGRAM 2.1.0: Initialization Functions

2.1.0

Input

Data base arrays

TAS7COMM
CASCAN
CASYS
CAPLNAME
CAWBNAME
CAWBCONF
TAS7HIER
CARCB

Process

1. If second PATCH, then (12)
2. Initialize system task structure.
3. Invoke phase one Display Mgmt initialization.
4. Obtain space for the fixed system control and work space.
5. Initialize fixed system control space.
6. Initialize fixed system work space.
7. Issue entities for all status items.
8. Initialize all S/7 support control and work space.
9. Invoke phase one supervisory control initialization.
10. Return to Special Real Time Operating System.
11. Page Fix all items in page fix array. (18)
12. Invoke phase two Display Mgmt initialization.
13. Initialize CAPLNAME array.
14. Invoke the S/7 I/O processor.
15. Vary the S/7s in service.
16. Initialize wallboard arrays.
17. Event the initialization completion.
18. Return to Special Real Time Operating System.

Output

Data base arrays

TAS7COMM
TAS7HIER
CASCAN
CASYS
CAPLNAME
CAWBNAME
CAWBCONF
CARCB

DIAGRAM 2.1.1: Energy Management System Initialization

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. Initialization is performed in two phases. One before and one after the RESTART WRITE		
2. PATCH all energy management tasks with program DOMCTCBI		
3. PATCH DPOIDISM		
4. Called subroutine DOMTP1IN		
5. Called subroutine DOMTP1IN		
6. PATCH DOMTSINT This program is DOMTRESI PATCHed by Special Real Time Operating System as directed through the job step input stream		
7. Called subroutine DOMCSENT		
8. PATCH DOMTINFO		
9. PATCH DOMCINIT		
11. PATCH DPPPXFIX Special Real Time Operating System program	DOMCINIT	2.1.1.2
12. PATCH DPOIDISM		
13. Call subroutine DOMTAPLI		
14. PATCH DOMTS7IO	DOMTS7IO	2.6.3.0
15. VARYS7 macro from subroutine DOMTP2IN		
16. Call subroutine DOMCUBIN		
17. SCEVENT,type=system		

DIAGRAM 2.1.1

Page of LY20-2226-0
Updated August 31, 1976
By TNL: LN20-3620

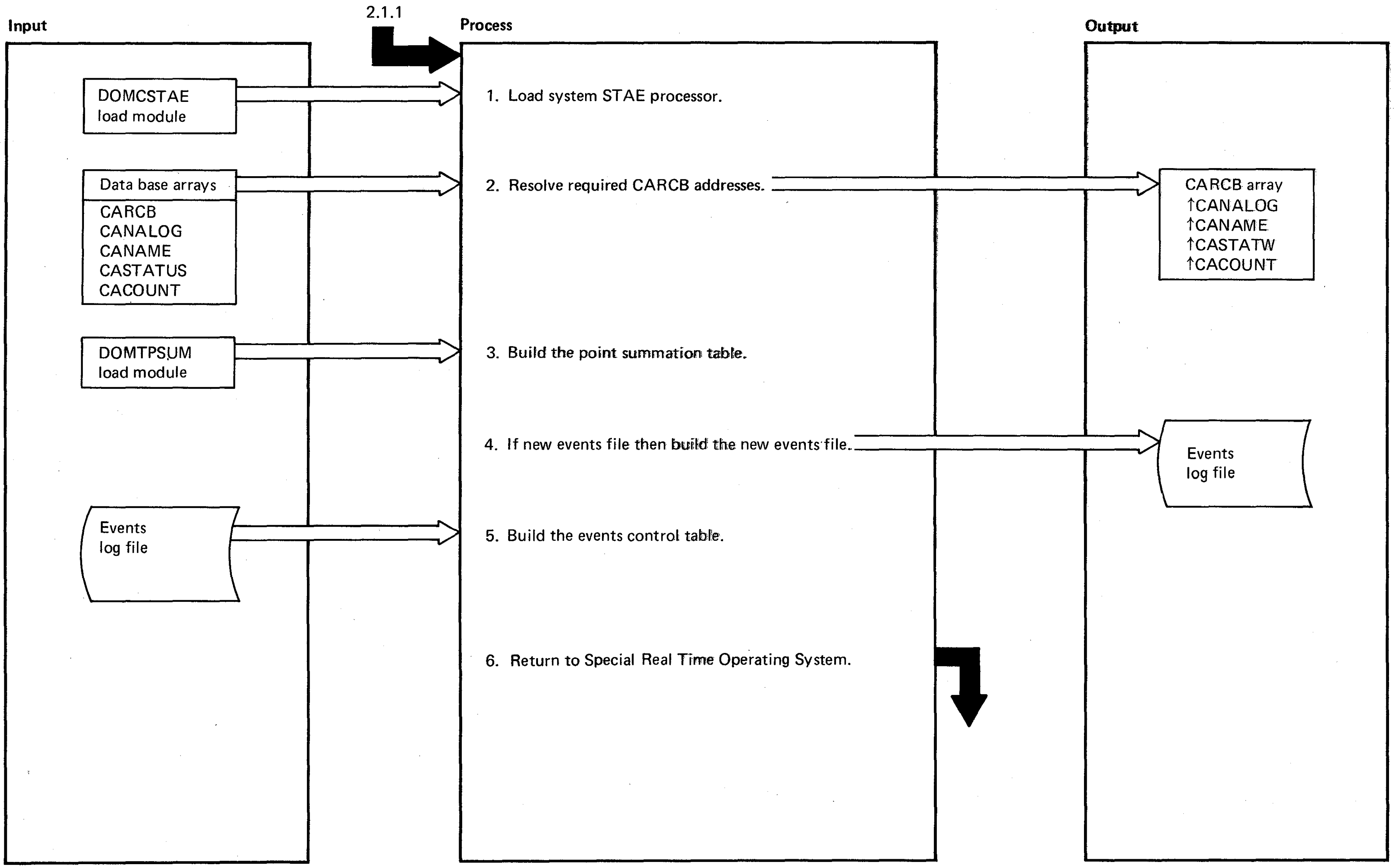


DIAGRAM 2.1.1.1: Phase One (DOMCINIT)

Intentionally Blank

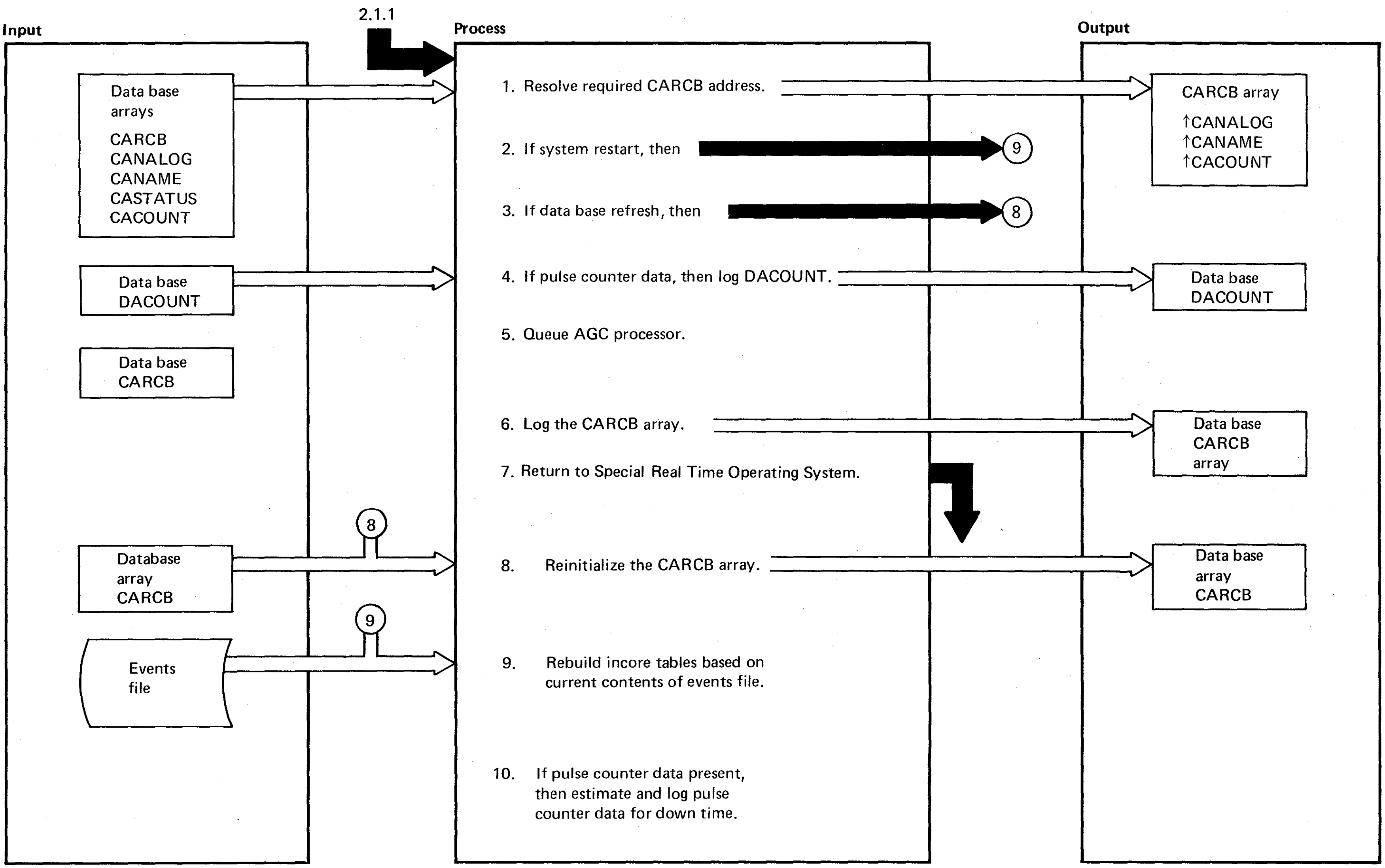


DIAGRAM 2.1.1.2: Phase Two (DOMCINIT)

Input

DASTATUS
CASTATUS

Process

11. Rebuild CASTATUS array.

DOMCWSTA

12. Reissue all necessary alarms.

DOMCIALM

13. Return to Special Real Time Operating System.

Output

CASTATUS
array

DIAGRAM 2.1.1.2

EXTENDED DESCRIPTION

Notes	Modules	Diagram
	DOMAEDCI	2.1.4
	DOMALFCI	2.1.3

DIAGRAM 2.1.1.2

Intentionally Blank

2.1.1

Input

Input patch list
Patch parm list
AID
DCE
XCVT
EMSCVT
CASYS

Process

1. If DCEUSER field is zero retrieve area to be used for Control Element Address Table and zero area just retrieved.

2. If DCEUSER field is zero then place address of Control Element Address Table in DCEUSER field.

3. Turn on background lites for system defined program function keys.

4. Return to system.

Output

Control element address table (CEATAB)

Display Management System

DIAGRAM 2.1.2: Energy Management Display Initialization

EXTENDED DESCRIPTION

Notes	Modules	Diagram
This function is executed whenever page one of the sign on display DOMDSGON is requested on the screen 1. The Control Element Address Table is used to hold pointers to Remote Control Element areas (RCE) for functionally distinct display areas (i.e., alarm, events, sensor based data, etc.) 2. The CEATAB is pointed to by the DCEUSER field in the DCE		

DIAGRAM 2.1.2

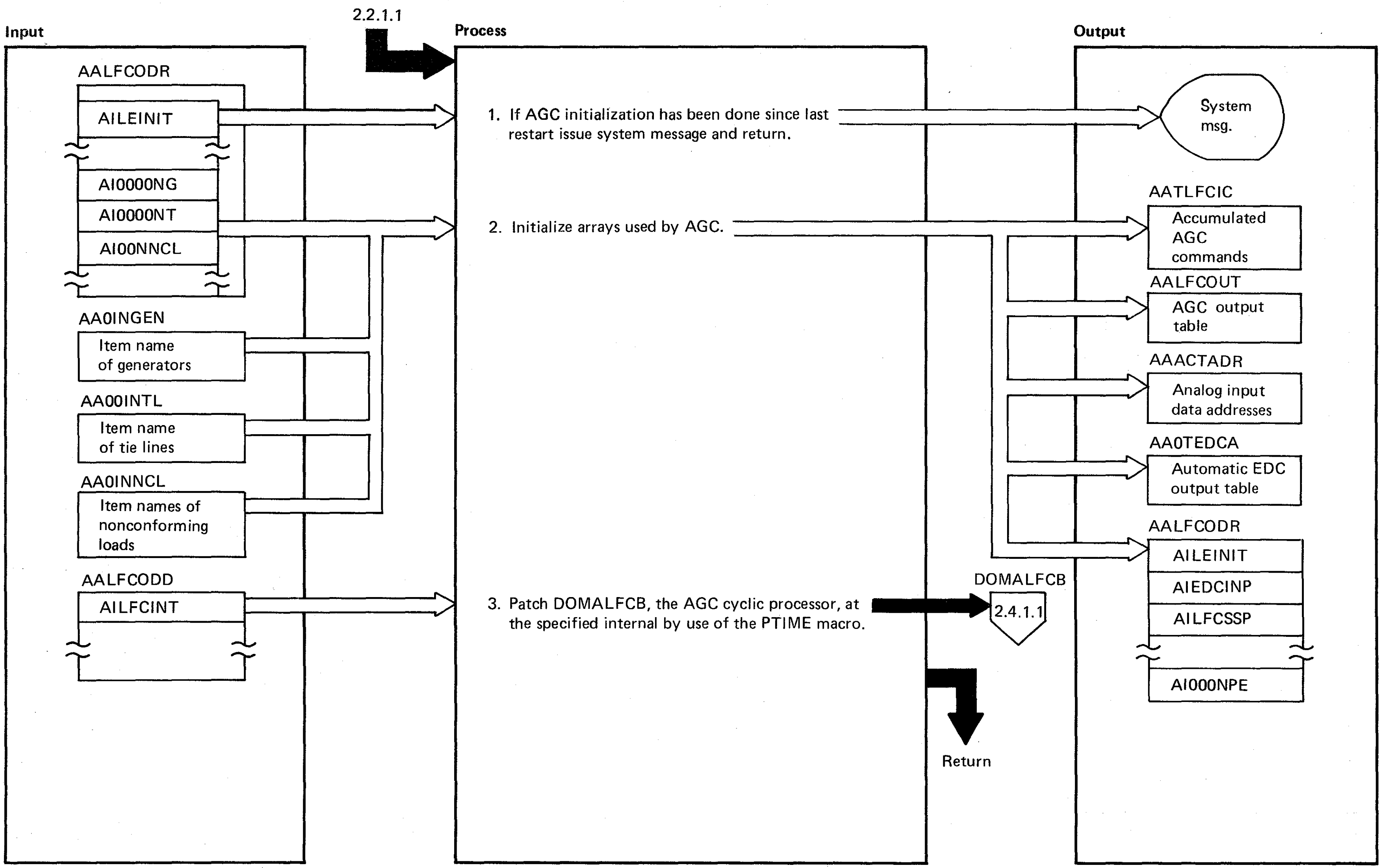


DIAGRAM 2.1.3: Automatic Generation Control Initialization

EXTENDED DESCRIPTION

Notes	Modules	Diagram
0. DOMALFCI is PATCHed by DOMTAGCI during post checkpoint/restart initialization after the first scan has been processed		
1. AGC initialization has been done if AILEINIT is non zero	DOMALFCI	
2. AILEINIT = 1; AGC initialized AIEDCINP = 0; EDC output not available AILFCSSP = 1; AGC suspended NPE = NG + NT + NNCL; the number of elements in the power vector is equal to the number of generators plus the number of tie lines plus the number of non conforming loads	DOMALFCI	

DIAGRAM 2.1.3

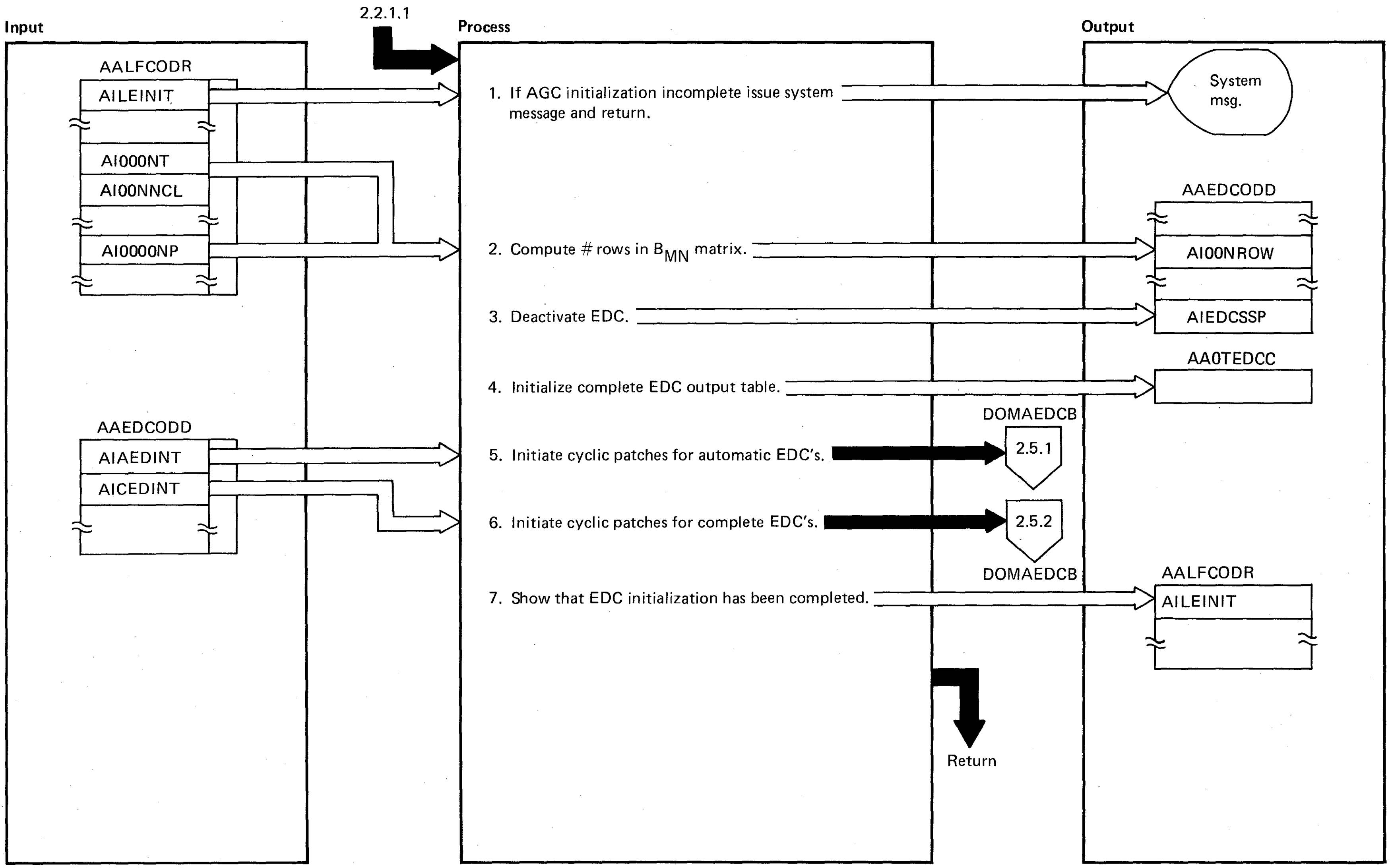


DIAGRAM 2.1.4: Economic Dispatch Control Initialization Program

EXTENDED DESCRIPTION

Notes	Modules	Diagram
0. DOMAEDCI is patched by DOMTAGCI during post checkpoint/restart initialization after DOMALFCI has been executed		
1. If AILEINIT = 0 AGC initialization is incomplete; if AILEINIT ≠ 1 multiple initializations of EDC are being attempted	DOMAEDCI	
2. The number of rows in the B_{MN} matrix is equal to the number of generation plants + the number of tie lines + the number of non-conforming loads	DOMAEDCI	
3. To do this set AIEDCSSP to 1	DOMAEDCI	
5. PATCH ID = 1 for automatic EDC	DOMAEDCI	
6. PATCH ID = 2 for complete EDC	DOMAEDCI	
7. To do this set AILEINIT = 2	DOMAEDCI	

DIAGRAM 2.1.4

Intentionally Blank

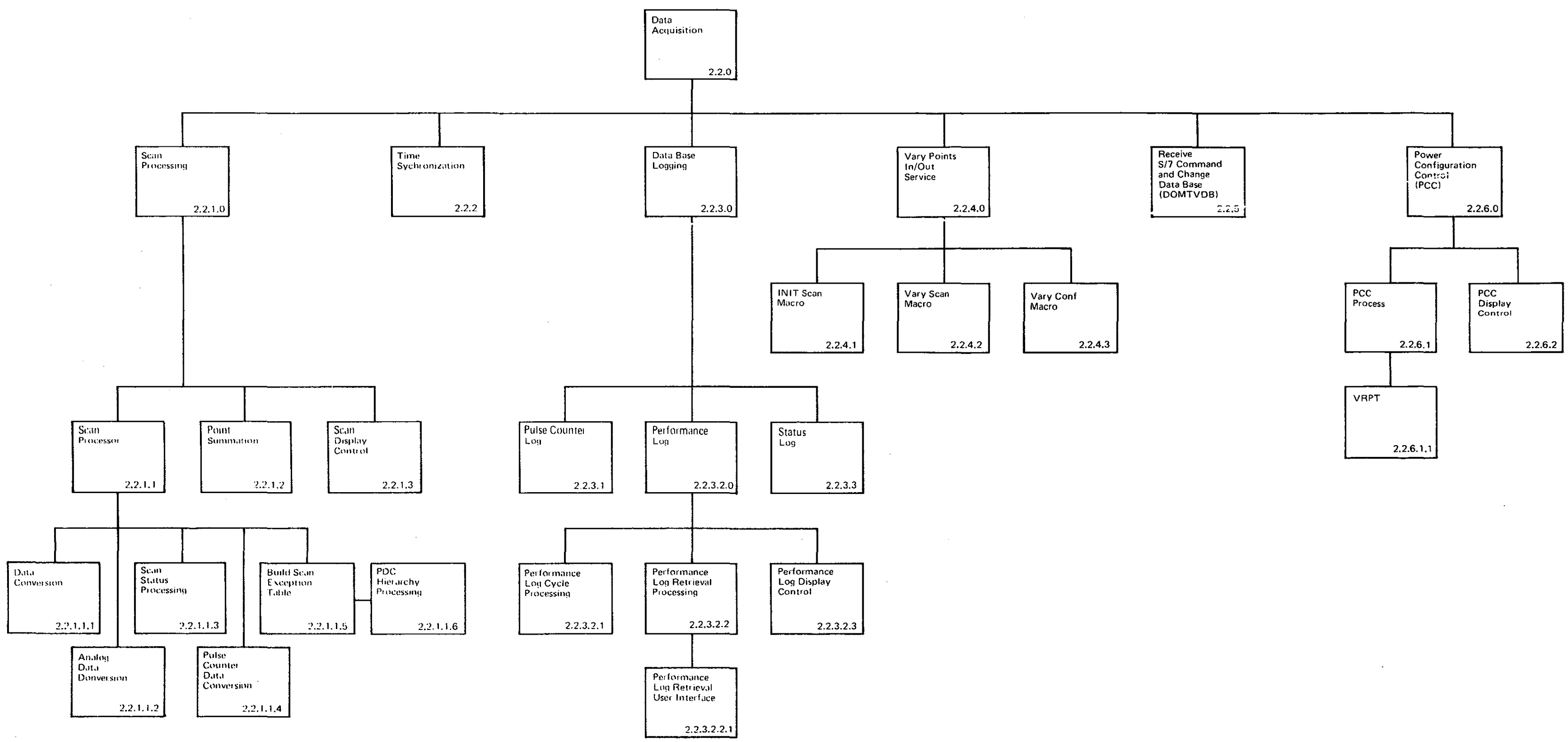


DIAGRAM 2.2.0: Data Acquisition Functions

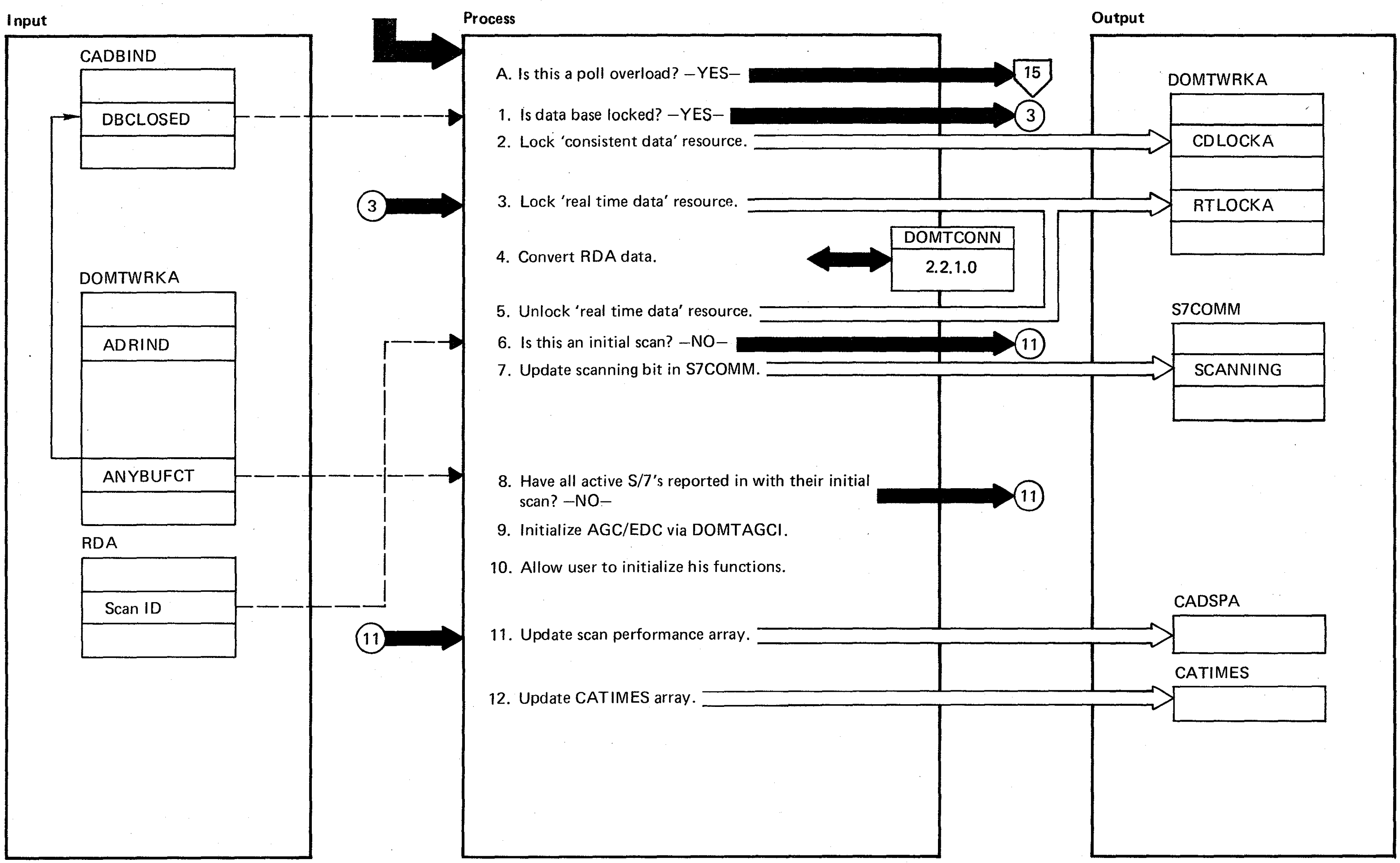


DIAGRAM 2.2.1.1: Scan Processing

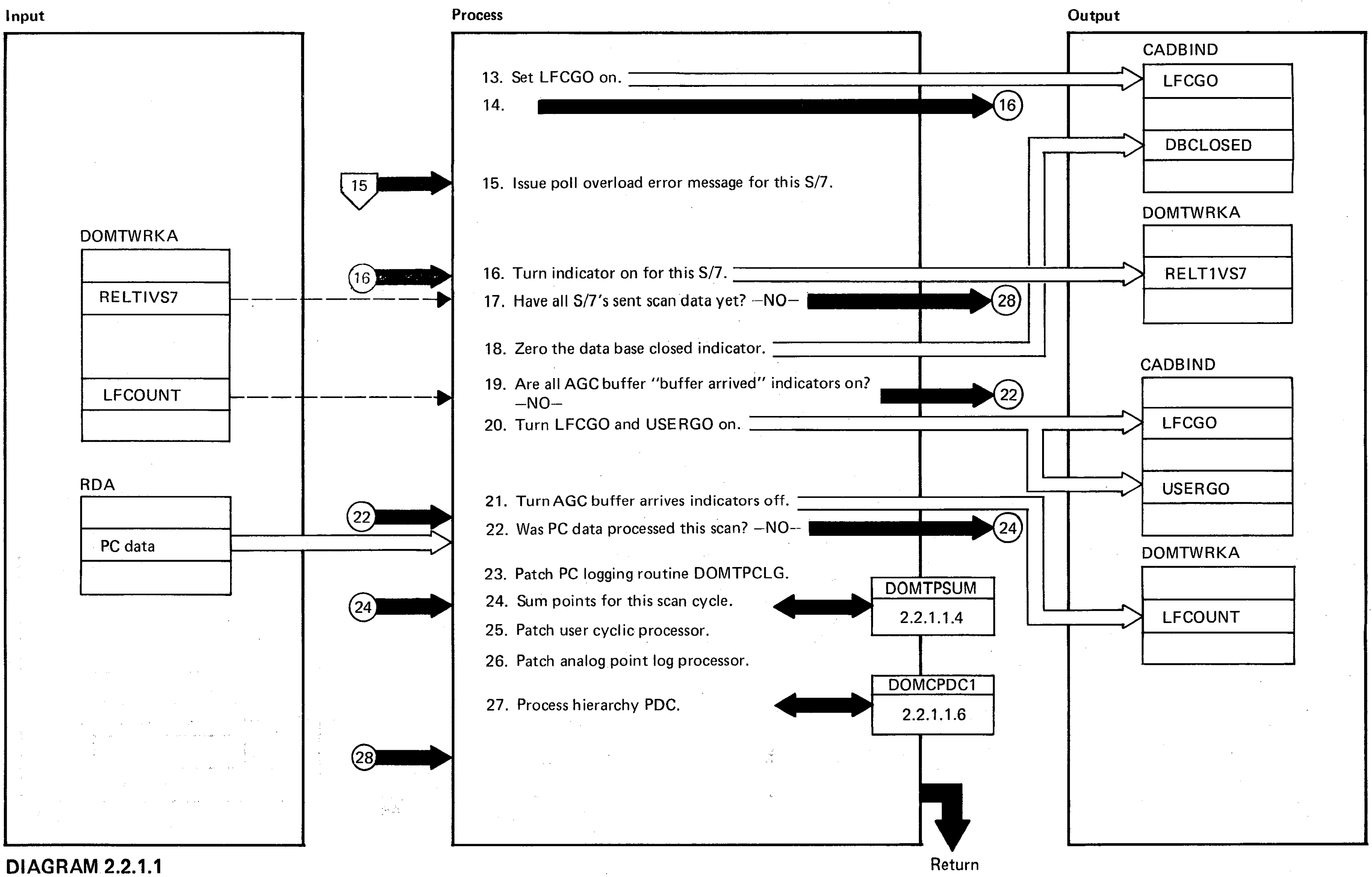


DIAGRAM 2.2.1.1

EXTENDED DESCRIPTION

Notes	Modules	Diagram
2. The consistent data base lock is set when the first buffer arrives in this scan and is not unlocked until all active system/7's have reported In		
3. The real time data lock is used to prevent anyone from using the data base while it is being updated by the data acquistion routines		
6. A scan ID of 128 indicates an initial scan		
8. All active system/7's must report in with their initial scans in order to have a complete set of data in the data base for the initialization routines		
9. PATCH to DOMTAGCI	DOMCINIT	2.1.1
10. PATCH to DOMUSERI	DOMUSERI	NA
12. The system/7 time in the RDA is converted to 10 millisecond units and stored in the CATIMES array. The day of the year, power system time, and time deviation are also updated		
16. Even though the system/7 did not report in with scan data, it did report in and therefore must still be active		
22. The pulse counter data logging routine is given control only when there has been pulse counter data sent in the RDA		
At the end of every complete scan:		
23. Pulse counter logging routine is PATCHed	DOMTPCLG	2.2.3.1
24. Point summation processor is called	DOMTSSYN	2.2.1.1.4
25. A user cyclic processor is PATCHed	DOMUSERC	NA
26. Analog point log processor is PATCHed	DOMTAPLP	2.2.3.2.1

DIAGRAM 2.2.1.1

Intentionally Blank

Input

Data base
tables
STCT
RCB LIST
CAS7RDAM

Raw data
array card

2.2.1.1.

Process

1. Find the RCB list and RDA map for this System/7.
2. For the terminals defined for this System/7
   DO

   A. Terminal is out of service No processing.
   B. Terminal is in service
      DO

      (1) Call analog data routine if any analog data. — DOMTCACS 2.2.1.1.2

      (2) Call status data routine if any status changes received. — DOMTCSES 2.2.1.1.3

      (3) Call pulse counter data routine if any counter data. — DOMTCPC 2.2.1.1.4

   C. Terminal is PDC information call PDC processor. — DOMCPDC1 2.2.1.1.6

   D. Point to next terminal.

3. If a Scan Exception Table has been built then PATCH the alarm processor.
4. If a wallboard table has been built then PATCH the wallboard processor.

2.2.1.1.

Output

DIAGRAM 2.2.1.1.1: Data Conversion (DOMTCONV)

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. DOMTCONV is called by DOMTSSYN to control the processing of the data in the RDA	DOMTSSYN	2.2.1.1
2.b.1 DOMTCONV calls DOMTCACS to analyze the analog data in the RDA	DOMTSSYN	2.2.1.1.2
2.b.2 DOMTCONV calls DOMTCSES to process the status data in the RDA. Except for initial scan, the RDA contains status data only if there has been a change	DOMTSSYN	2.2.1.1.3
2.b.3 DOMTCONV calls DOMTCPC to process the pulse counter data in the RDA	DOMTSSYN	2.2.1.1.4
2.c. DOMTCONV calls DOMCPDC1 to process the PDC information which is included in the RDA under a dummy terminal number	DOMTSSYN	2.2.1.1.6
3. DOMTCONV patches DOMCALR1 to process any SET entries	DOMCALR1	2.3.1.1
4. DOMTCONV patches DOMCWBPR to process any wallboard changes	DOMCWBPR	2.3.5.1

DIAGRAM 2.2.1.1.1

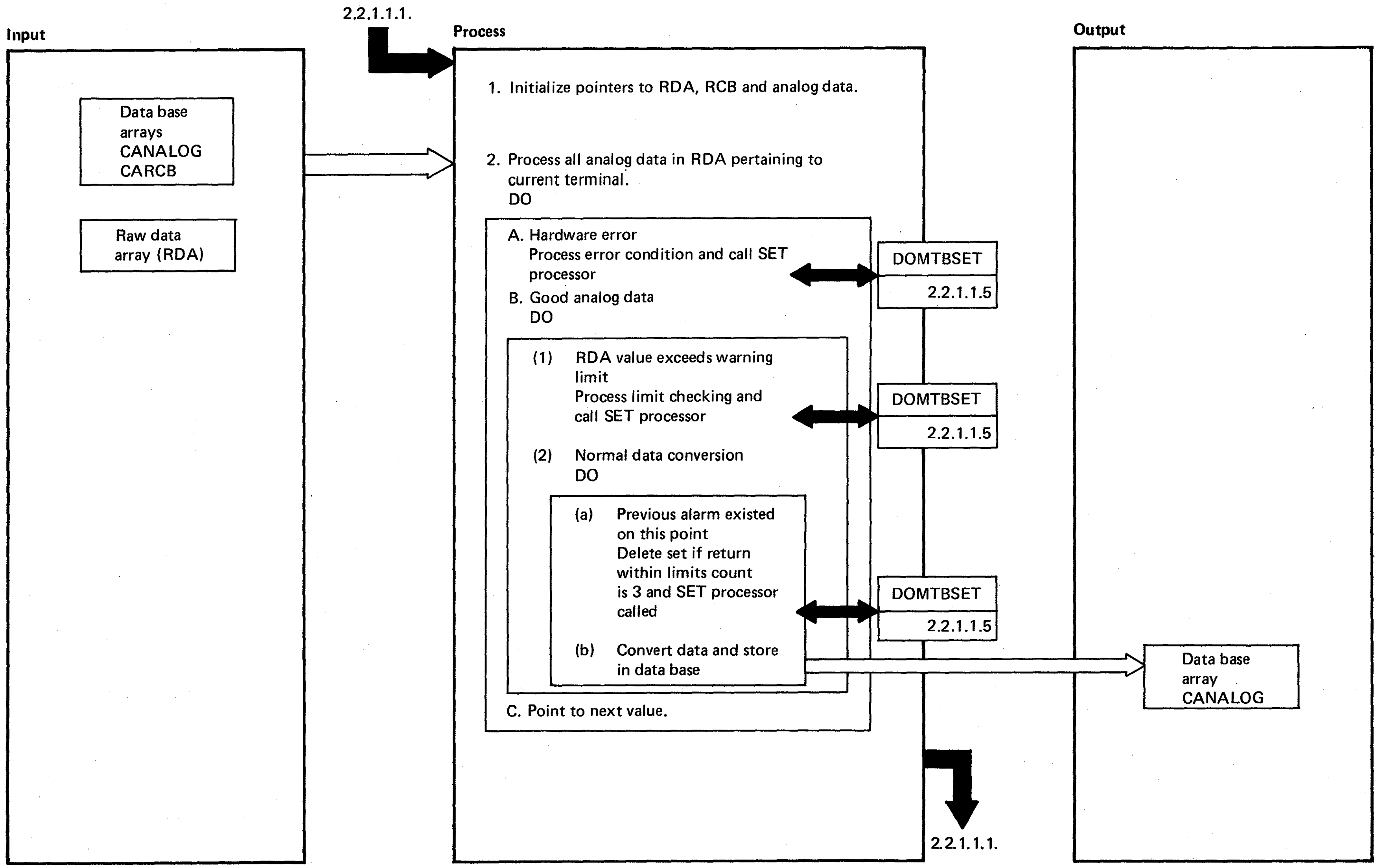


DIAGRAM 2.2.1.1.2: Analog Data Conversion (DOMTCACS)

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. DOMTCACS receives control from DOMTCONV and is passed pointers to the analog data in the RDA and the applicable RCB	DOMTSSYN	2.2.1.1.1
2. DOMTCACS recalls DOMTBSET to include the analog point in the Scan Exception Table (SET) if an error condition exists or if an alarm should be deleted	DOMTSSYN	2.2.1.1.5
2.b.1 The high and low limits for each analog point are defined in the data base. The raw data is compared to these values to determine if the point has an error condition		
2.b.2 When the error condition on a point clears up, it must be good data for 3 consecutive scans before the alarm can be deleted		

DIAGRAM 2.2.1.1.2

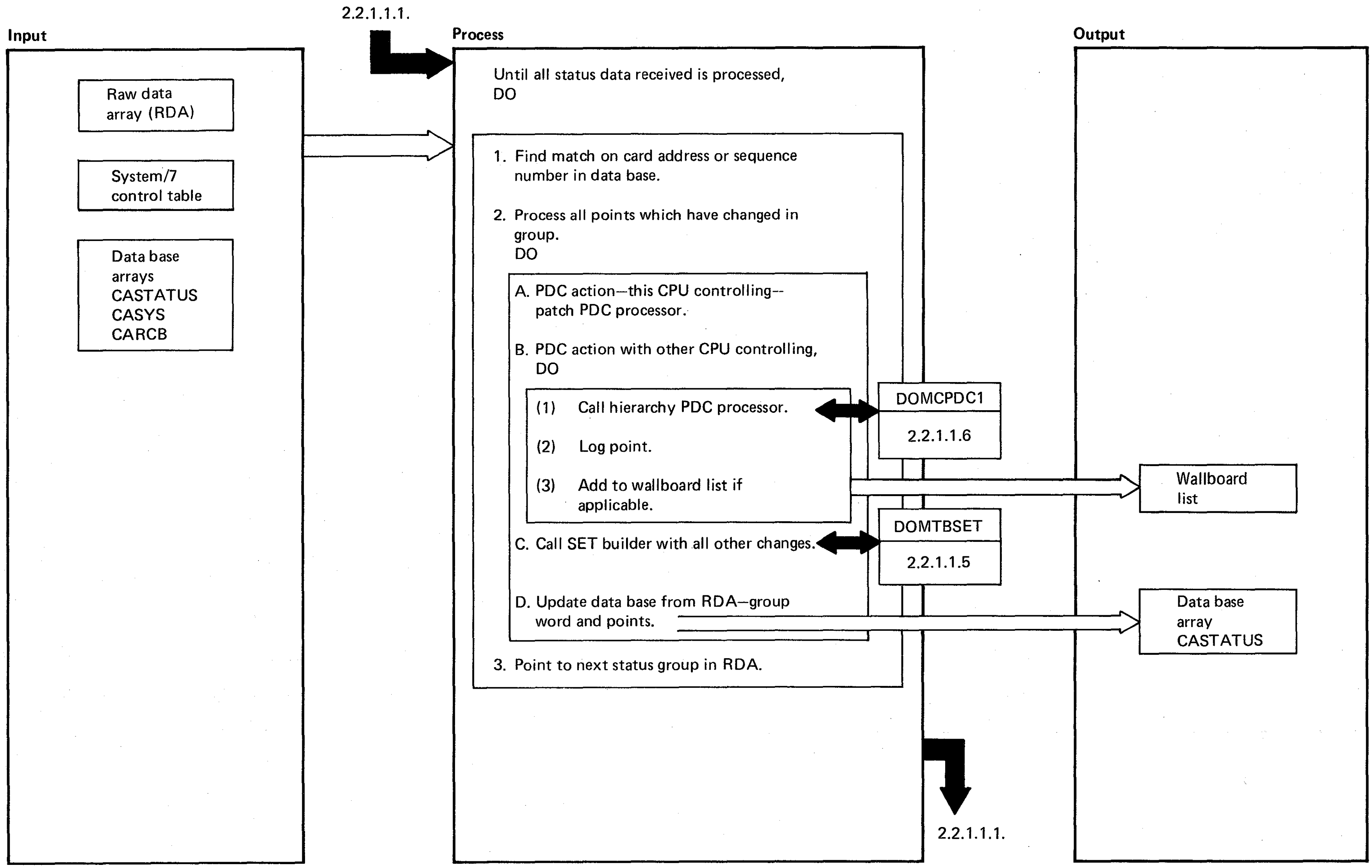


DIAGRAM 2.2.1.1.3: Scan Status Processing

EXTENDED DESCRIPTION

Notes	Modules	Diagram
DOMTCSES receives control from DOMTCONV via a call	DOMTSSYN	2.2.1.1.1
2.a. DOMTCSES patches DOMCDCO1	DOMCDC01	2.3.2.0
2.b.1 DOMTCSES calls DOMCPDC1	DOMTSSYN	2.2.1.1.6
2.b.2 The status point information is passed to the change of status log processor via a PATCH	DOMCSLOG	2.2.3.3
2.b.3 The status point information is added to the wallboard list to be patched to the wallboard processor at the end of processing		
2.c. DOMTBSET is called to add the point to the Scan Exception Table (SET)	DOMTSSYN	2.2.1.1.5

DIAGRAM 2.2.1.1.3

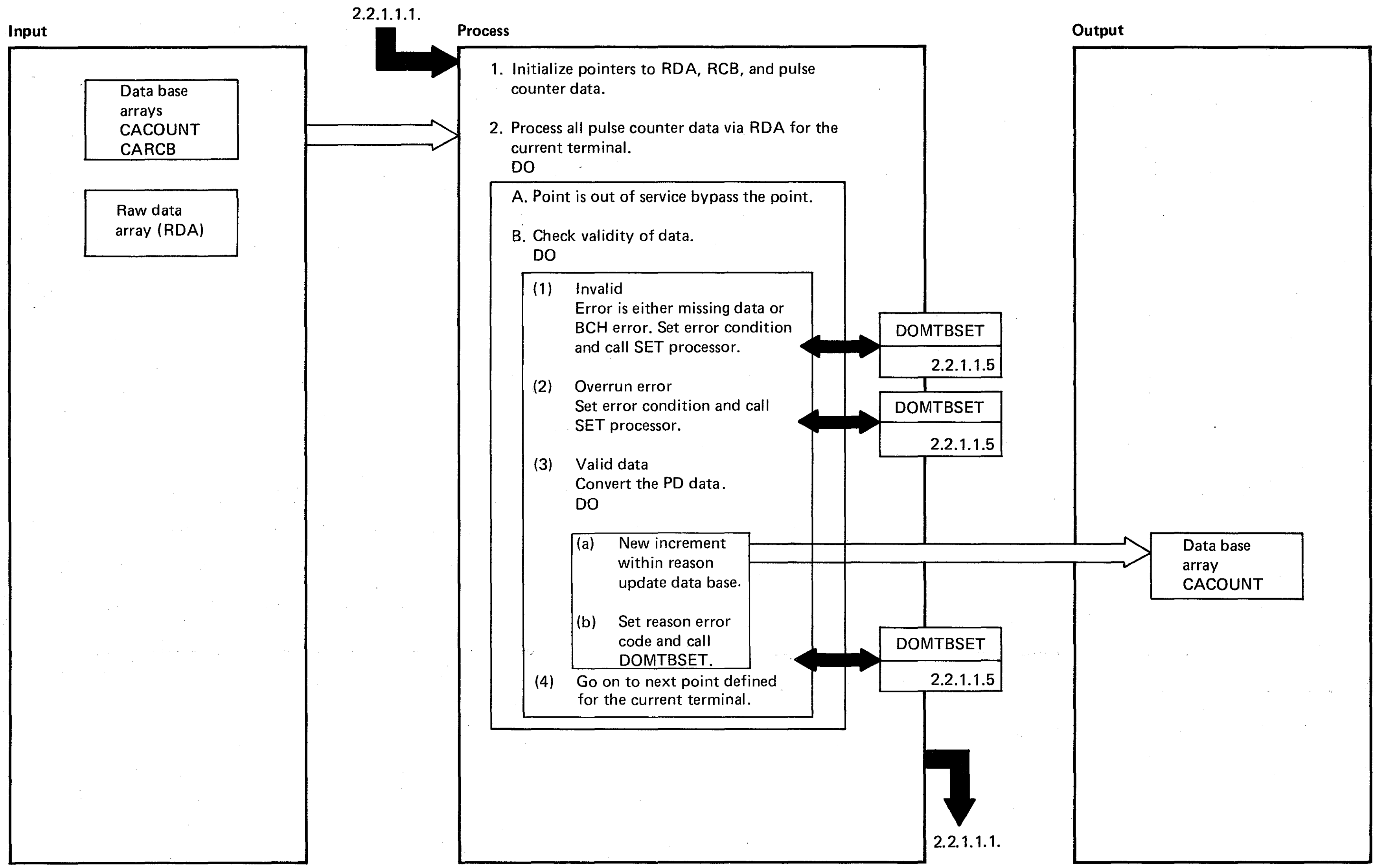


DIAGRAM 2.2.1.1.4: Pulse Counter Data Conversion

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. The pointers to the RDA and RCB are passed by DOMTCONV which calls this module (DOMTCPC)	DOMTSSYN	2.2.1.1.1
2. DOMTCPC is called for each terminal included in the RDA for which pulse counter data has been received		
2.a. No processing is done on points which are out of service		
2.b. The invalid data table (IDT) flags are used to indicate whether the data in the RDA is valid		
2.b. All errors encountered are passed to the Scan Exception Table (SET) builder (DOMTBSET)	DOMTSSYN	2.2.1.1.5
2.b.3 The pulse counter data is converted to floating point and multiplied by an A coefficient. The increment, accumulated count, extrapolation value and time of last good PC data are also updated		
2.b.3.a The new increment is compared with the maximum pulses per line per scan		

DIAGRAM 2.2.1.1.4

Input

DOMTWRKA

- SETHEADP
- CURRENT
- EMTP
- BUFRENDP

Set

Terminal header

2.2.1.1

Process

1. Is this an initial queue or is this the end of the current buffer? NO — (8)
2. GETWA for set buffer.
3. Is this the first buffer? NO — (6)
4. Save buffer address and initialize buffer header.
5. (7)
6. (6) Chain this buffer to the previous one.
7. (7) Reset buffer pointers.
8. (8) Is the terminal header in this buffer the same as the last? YES — (10)
9. Initialize new terminal header portion of buffer and update pointers.
10. (10) Store data, add 1 to entry counter and update current buffer pointer.
11. Return.

Output

DOMTWRKA

- SETHEADP
- CURRENT
- EMTP
- BUFRENDP

Set

Buffer header entry counter terminal counter data

DIAGRAM 2.2.1.1.5: Build Scan Exception Table

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. If this is the first queue for this scan, the buffer pointers will be zero. If not, the pointer indicating the displacement in the buffer for the next entry is the same as the pointer to the end of the buffer, then this buffer is full. 8. As each terminal is processed by data conversion, DOMTBSET must build a new terminal header in the buffer	DOMTBSET	2.2.1.1.5

DIAGRAM 2.2.1.1.5

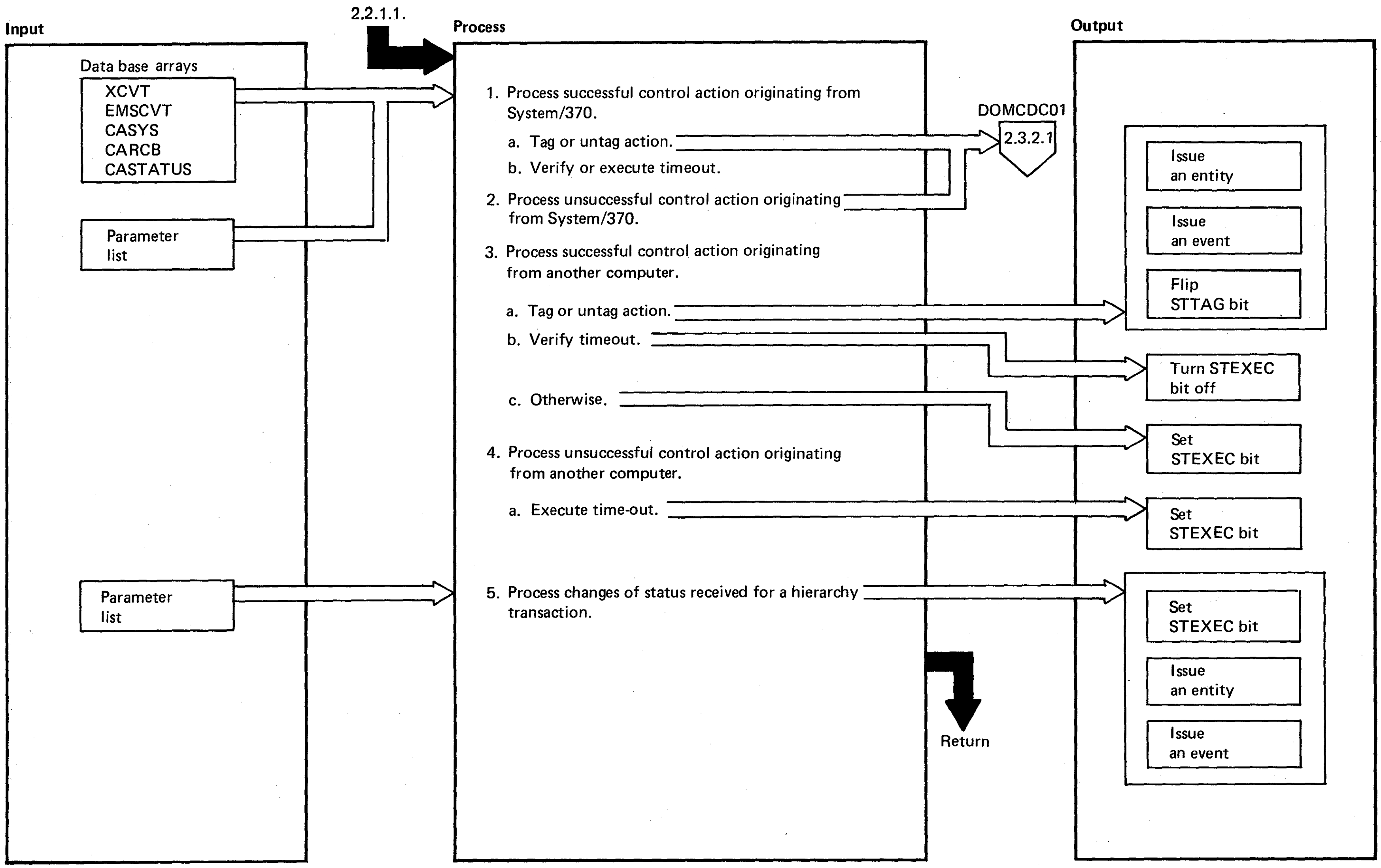


DIAGRAM 2.2.1.1.6: PDC Hierarchy Processor

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. Parameter list contains a calling ID, a pointer to the XCVT, and a pointer to the PDC portion of the Raw Data Array.	DOMTSSYN	2.2.1.1.1
1.a. Program DOMCDC01 is patched with the address of the PDC portion of the RDA	DOMCDC01	2.3.2.1
1.b. Program DOMCDC01 is patched with the address of the PDC portion of the RDA	DOMCDC01	2.3.2.1
2. Program DOMCDC01 is patched with the address of the PDC portion of the RDA	DOMCDC01	2.3.2.1
3.a. Issue an entity and an event	DOMCPDC1	2.2.1.1.6.3
5. Parameter list contains a calling ID, a pointer to the XCVT, a pointer to the status item, a pointer to the remote control block	DOMTCSES	2.2.1.1.3
5. Issue an entity and an event	DOMCPDC1	2.2.1.1.6.5

DIAGRAM 2.2.1.1.6

Input

DOMTSMTB
- Scan ID
- Point address of sum point
- Point address of points to sum

Data base
- Pulse counter and analog data base
- Sum points

2.2.1.0

Process

1. Use scan ID passed to search point summation table for a match.
2. Are there any points to sum for current scan ID?
   — NO — (11)
3. Initialize group counter.
4. Get the data for each point in this group.
5. Add all points for this group.
6. Is new sum to be added to old sum?
   — NO — (8)
7. Add new sum to old sum.
8. Perform normal processing on new sum.
9. Store sum value in data base.
10. Have all groups for this scan been processed?
    — NO — (3)
11. Return.

Output

Data base
- Sum points

DIAGRAM 2.2.1.2: Point Summation

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. The scan ID is passed as the PATCH ID 4. The address of each point is obtained from the point of summation table that was built by supervisory control initialization 7. When the point summation table is sysgened, a bit is used to indicate if the new sum is to replace the old sum or if it is to be added to the old sum. If the new sum is to be added to the old sum, additional checking is done to determine if the sum point is to be reset. This is done by checking the summation duration value in minutes, which is part of the sysgened point summation table	DOMTSSYN	2.2.1.2

DIAGRAM 2.2.1.2

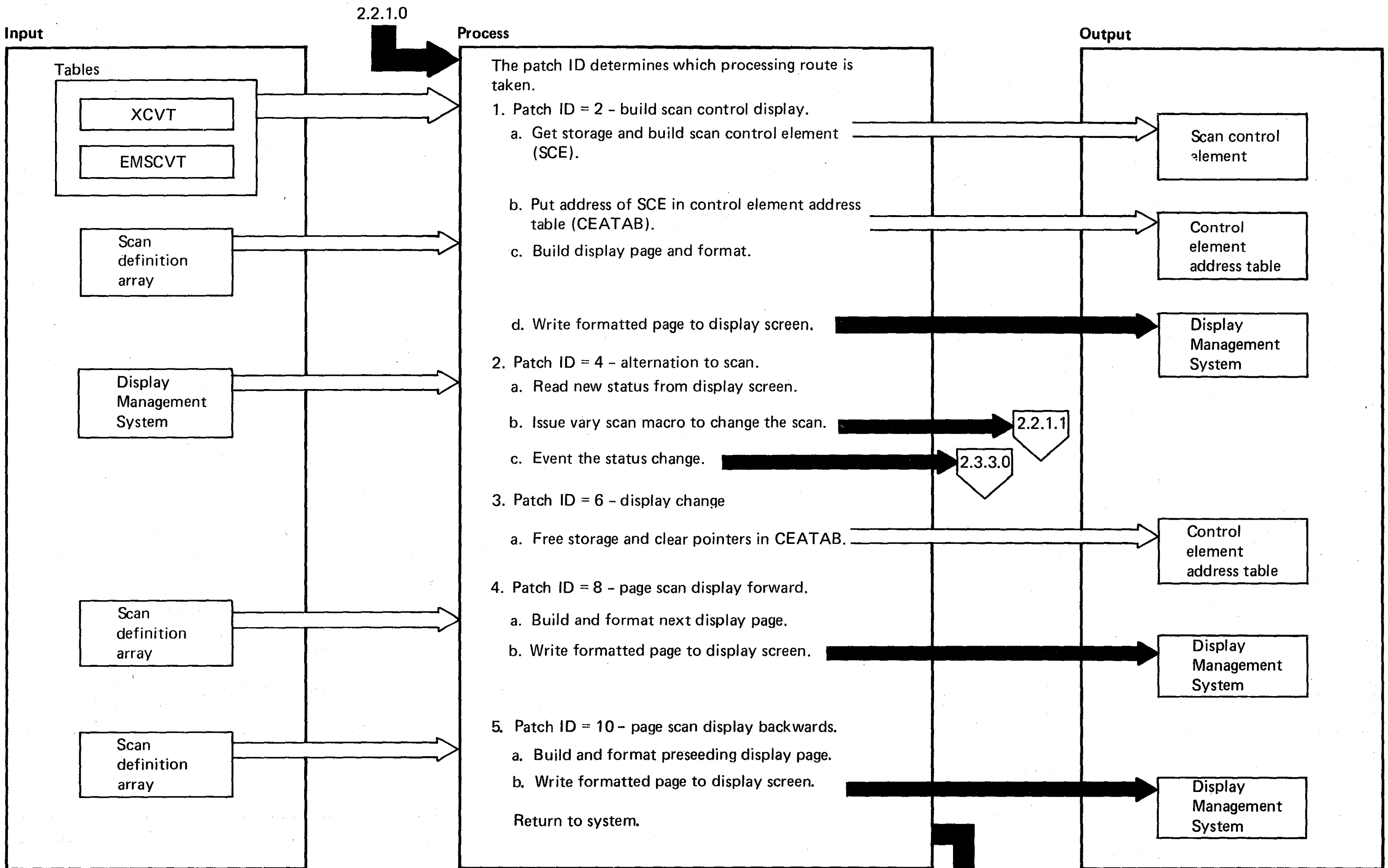


DIAGRAM 2.2.1.3: Scan Display Control

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. Control received by manual input of scan display request by the power system operator	DOMCAND1	2.2.1.3
2. Control received by manual input of new scan status by the power system operator	DOMCAND1	2.2.1.3
3. Control received by manual input of display change (display other than scan display brought to screen) by the power system operator	DOMCAND1	2.2.1.3
4. Control received by manual input of page forward request by the power system operator. If last page is being viewed the first page will be displayed	DOMCAND1	2.2.1.3
5. Control received by manual input of page backward request by the power system operator. If the first page is being viewed the last page will be displayed	DOMCAND1	2.2.1.3

DIAGRAM 2.2.1.3

Page of LY20-2226-0
Updated August 31, 1976
By TNL: LN20-3620

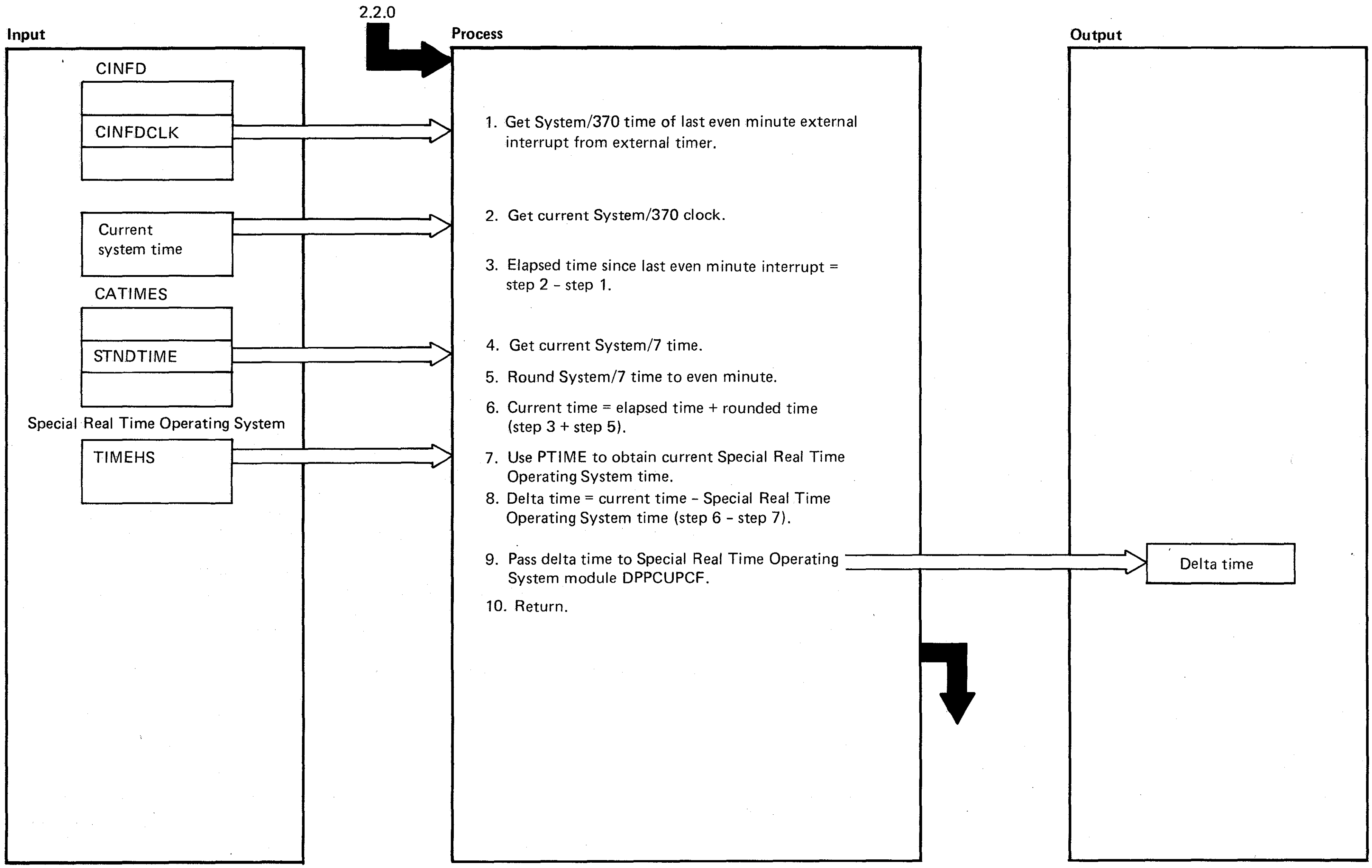


DIAGRAM 2.2.2: Time Synchronization

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. The external interrupt handler which receives an even minute pulse from the System/7 and scans the System/370 time at that time, PATCHES DOMTCLOCK after the EMSCVT has been built. DOMTCLOCK then causes itself to be PTIMED until the first initial scan is complete, thereby allowing scan processing to move the System/7 time from the RDA to the CATIMES array 6. The current time is equal to the last even minute pulse time plus the elapsed time since the last even minute pulse	DOMTCLOK	2.2.2

DIAGRAM 2.2.2

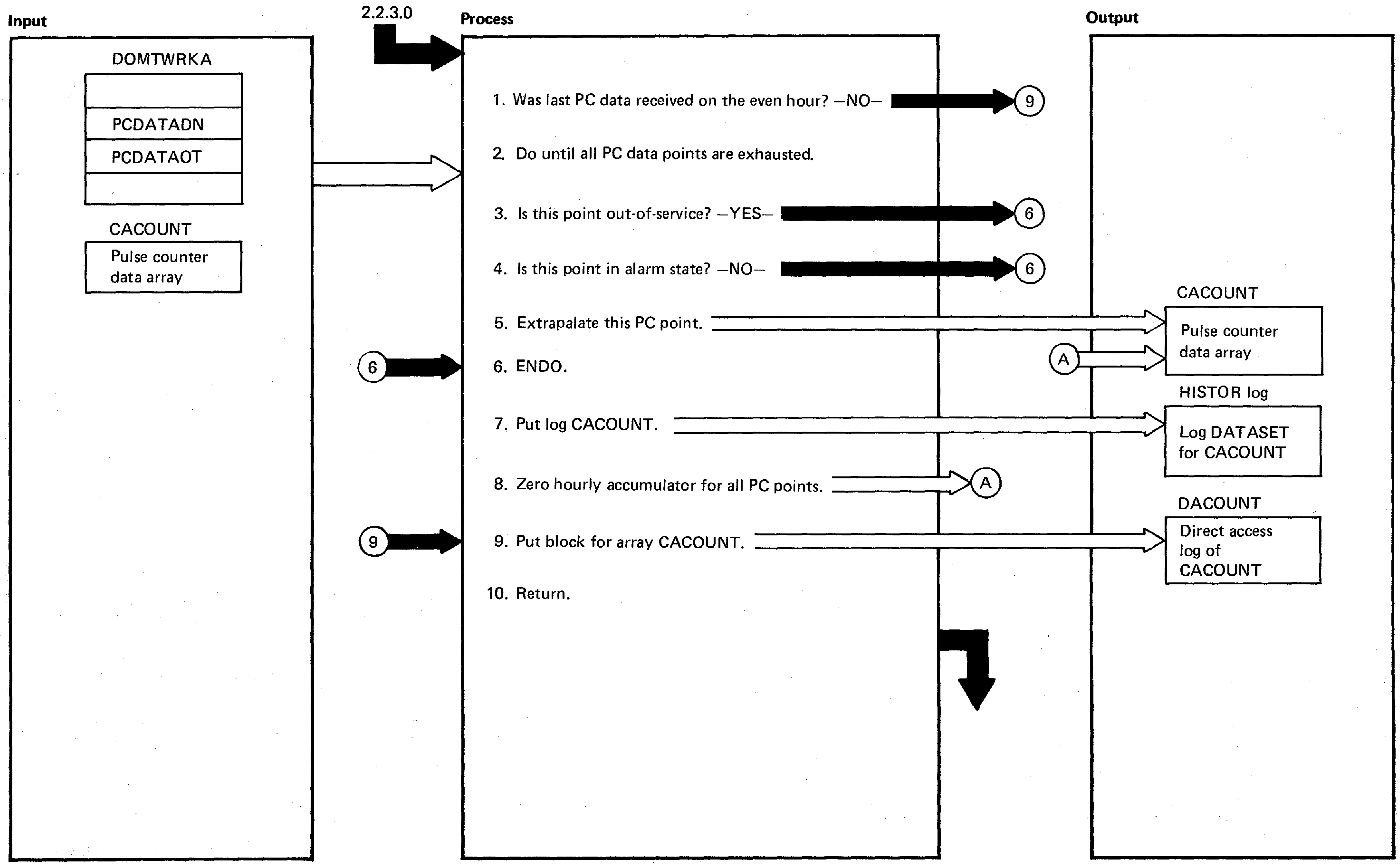


DIAGRAM 2.2.3.1: Pulse Counter Data Logging

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. Two bits in the DAQ workarea are turned on when the last PC data processing occurred on the even hour. The pulse counter data is logged on the even hour, but the last copy is kept on a direct access data set for restart purposes. 5. The data for this point is estimated by multiplying the filter value calculated when the last good data was received by the number of PC scans missed. If this is more than an hour ago, only one hours worth is estimated	DOMTPCLG	2.2.3.1

DIAGRAM 2.2.3.1

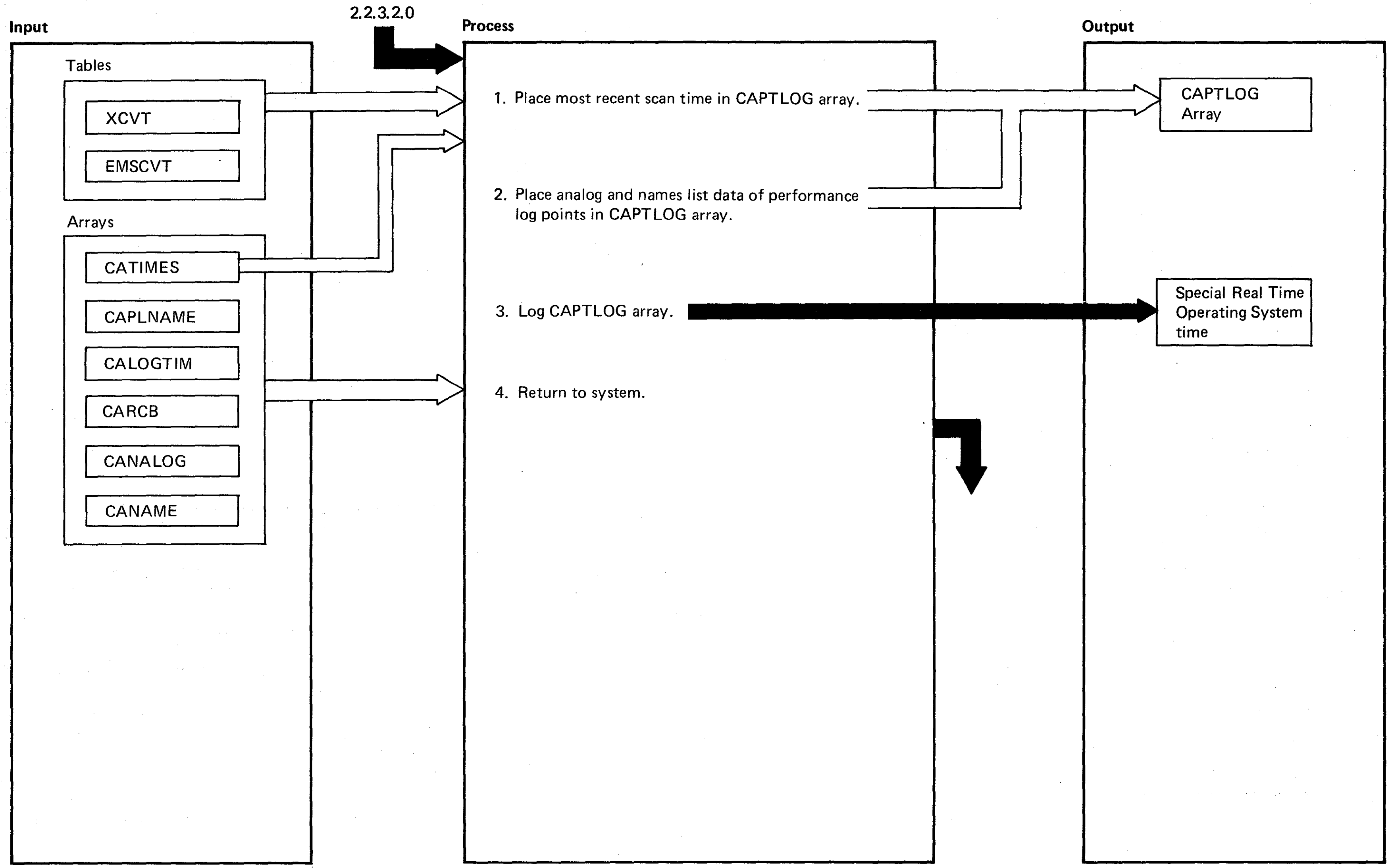


DIAGRAM 2.2.3.2.1: Performance Log Cyclic Processing

EXTENDED DESCRIPTION

Notes	Modules	Diagram
Note: All processing takes place every basic scan cycle		
1. Scan time from the CATIMES array is placed in the CAPTLOG array	DOMTAPLP	2.2.3.2.1
2. All analog point names which are to be logged during performance log cyclic processing are maintained in the CAPLNAME array	DOMTAPLP	2.2.3.2.1
3. CAPTLOG array is logged every scan cycle unless the performance log retrieval function is active and the logging of this array would cause data yet to be retrieved, to be overlayed	DOMTAPLP	2.2.3.2.2

DIAGRAM 2.2.3.2.1

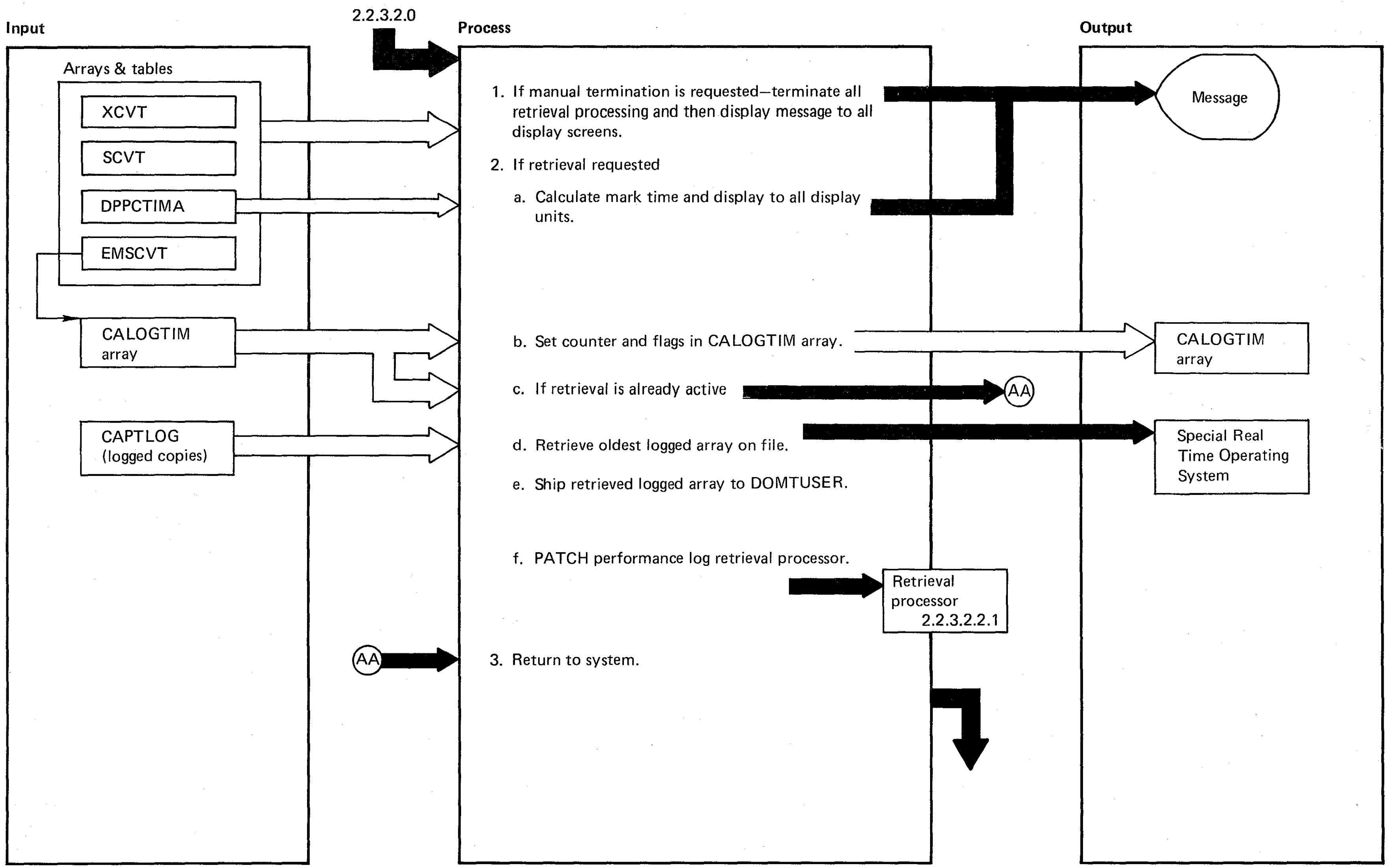


DIAGRAM 2.2.3.2.2: Performance Log Retrieval Processing

EXTENDED DESCRIPTION

<table>
<tr><th>Notes</th><th>Modules</th><th>Diagram</th></tr>
<tr><td>1. Message is displayed in the system message zone</td><td>DOMTAPLM</td><td>2.2.3.2.1</td></tr>
<tr><td>2.a. Message is displayed in the system message zone</td><td>DOMTAPLM</td><td>2.2.3.2.1</td></tr>
<tr><td>2.b. These counters and flags insure that the logged data set is retrieved completely for one full cycle prior to the mark time and for one full cycle past the mark time
Example: Logged array holds 4 copies — N = Mark Time
N
<table><tr><td>N-4</td><td>N-3</td><td>N-2</td><td>N-1</td><td>N + 1</td><td>N + 2</td><td>N + 3</td><td>N + 4</td></tr></table>Logging would overlay N-4 through N-1 with N+1 through N+4 during the retrieval process yet allow all copies N-4 through N+4 to be retrieved</td><td>DOMTAPLM</td><td>2.2.3.2.1</td></tr>
<tr><td>2.c. If retrieval is already active a new mark time is noted and retrieval continues up to the new mark time and then for one more full cycle</td><td>DOMTAPLM</td><td>2.2.3.2.1</td></tr>
<tr><td>2.e. DOMTUSER is supplied as a "STUB" with the system. A user program DOMTUSER must replace the "STUB" if the data is to be used</td><td>DOMTAPLM</td><td>2.2.3.2.1</td></tr>
<tr><td>2.f. DOMTAPLR performs the actual retrieval while DOMTAPLM maintains the CALOGTIM array</td><td>DOMTAPLM</td><td>2.2.3.2.1</td></tr>
</table>

DIAGRAM 2.2.3.2.2

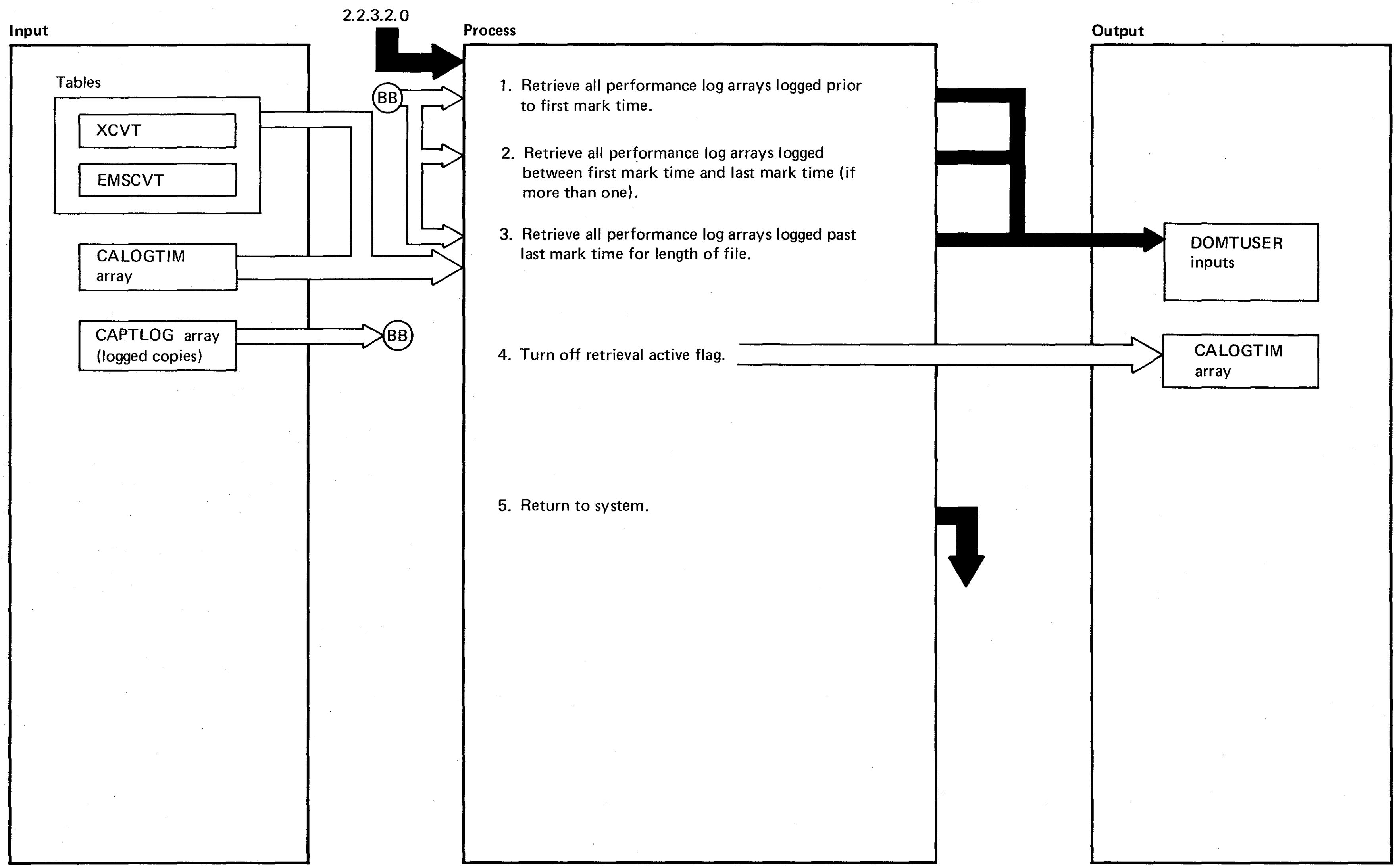


DIAGRAM 2.2.3.2.2.1: Performance Log Retrieval User Interface

EXTENDED DESCRIPTION

Notes	Modules	Diagram
Note: This function is only executed when the flag in CALOGTIM array is on to indicate that Performance Log retrieval is active		
1.a./1.b./1.c. All of the retrieved records are sent to the program DOMTUSER. DOMTUSER must be expanded by the user if further processing of the retrieved records is desired.	DOMTAPLR	2.2.3.2.2.1

DIAGRAM 2.2.3.2.2.1

Page of LY20-2226-0
Updated August 31, 1976
By TNL: LN20-3620

2.2.3.2.0

Input

Tables

- XCVT
- EMSCV

Arrays

- CAPLNAME
- CARCB
- CANALOG
- CANAME

Process

1. Process following sections based on PATCH ID.
   a. Initialize display and display page one. ID = 2
   b. Page display forward one page. ID = 4
   c. Page display backwards one page. ID = 6
   d. Refresh current display page. ID = 8
   e. Add an analog point name to CAPLNAME array and resolve analog and names list address. ID = 10
   f. Delete an entry in the CAPLNAME array. ID = 12
   g. Cancel-release unneeded buffers. ID = 14
2. Return to system.

Output

- Performance Log Display
- CAPLNAME array

DIAGRAM 2.2.3.2.3: Performance Log Display Control

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1.b. If display is on last page it will wrap around to first page	DOMTAPLD	
1.c. If display is on first page it will wrap around to last page	DOMTAPLD	
1.e. & 1.f. After the specified function is complete the current page of the display is refreshed	DOMTAPLD	

DIAGRAM 2.2.3.2.3

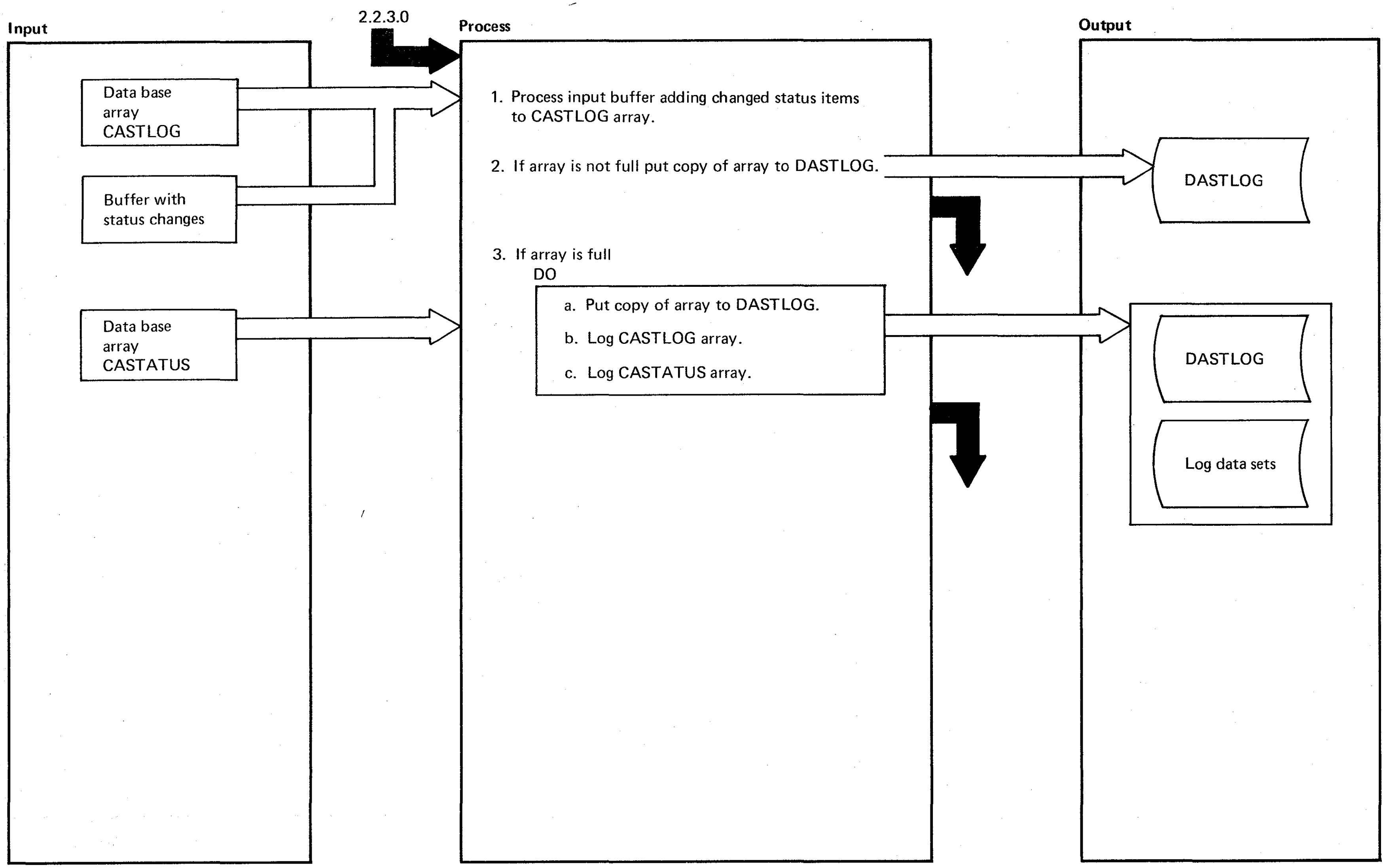


DIAGRAM 2.2.3.3: Change of Status Logging

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. Input buffers are generated by the following functions:		
a. Alarm processing	DOMCALR1	2.3.1.1
b. Alarm display	DOMCALD1	2.3.1.2
c. Device control	DOMCDC01	2.3.2.1
d. Scan processing	DOMTSSYN	2.2.1.1
2. Using PUTBLOCK macro		
3. Using PUTLOG macro		

DIAGRAM 2.2.3.3

Input

Register 1

System/7
ID mode

DOMCVT

2.2.4.0

Process

1. Is the mode initial? —YES— (5)
2. Is the mode emergency? —YES— (7)
3. Load flags for cyclic mode message to be passed to hierarchy.
4. (8)
5. (5) Load flags for initial mode to be passed to hierarchy.
6. (8)
7. (7) Load flags for emergency mode to be passed to hierarchy.
8. (8) Issue INITSCAN message to the hierarchy.
9. Return.

Output

Transaction code

0A

DIAGRAM 2.2.4.1: INITSCAN Macro

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. If the mode is not initial or emergency, it must be cyclic	DOMTABLE	2.7.2.0

DIAGRAM 2.2.4.1

Input

VARYSCAN parameter list

Data base array CASCAN

2.2.4.0

Process

1. Search scan table (CASCAN) for match or scan ID.
2. Determine scan mode desired.
   DO
   - A. Normal mode
     Set normal scan mode in message and CASCAN.
   - B. Initial mode
     Set initial scan mode in message and CASCAN.
   - C. Emergency mode
     Set emergency scan mode in message and CASCAN.
   - D. No change to mode.
3. Determine if scan to be activated or deactivated.
   DO
   - A. Deactivated.
     DO
     - (1) Set deactivate flag in message and CASCAN.
     - (2) Format remainder of message.
   - B. Activate.
     DO
     - (1) Set activate flag in message and CASCAN.
     - (2) Format remainder of message.
4. Build end of message and convert to ASCII.
5. Write message (SCM common) to System/7.

Output

Data base array CASCAN

SCM command message

DIAGRAM 2.2.4.2: Vary Scan Macro (DOMTVARY)

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. The scan ID is an input parameter from the VARYSCAN macro. The parameter list indicate what type of change requested a. Mode change Normal (standard) Initial Emergency b. Activate c. Deactivate		
4. The message (SCM command) is converted to ASCII using the ASCICONV macro		
5. The message is written to the System/7 using the S7WRITE macro. The transaction code used is X'08'	DOMTABLE	2.7.2.0

DIAGRAM 2.2.4.2

2.2.4.0

Input

Data base
S7CT
RCB list
CARCB array

Data base
S7CT arrays:
CARCB
CANALOG
CASTATUS
CACOUNT

Process

1. Determine whether to vary terminal or point.

   A. Vary a terminal.

      (1) Find RCB for terminal requested.

      (2) Check state of terminal against state requested.

      (3) PATCH vary terminal processor.

   B. Vary a point.

      (1) Find item for point requested.

      (2) Search arrays for item and determine type of point.

      (3) Check state of point agains state requested.

      (4) PATCH vary point processor.

2. Exit

Output

DIAGRAM 2.2.4.3: VARY CONF Macro Processor

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. DOMTVARY processes the inputs from the VARYCONF macro		
1.a.3 DOMTVARY patches DOMTVRPT to do the processing required to change the state of a terminal	DOMTVRPT	2.2.6.1.1
1.b.1 A GETITEM macro is issued to get the address of the point		
1.b.2 A GETARRAY macro is issued to get the addresses of the three data base arrays		
1.b.4 DOMTVARY patches DOMTVRPT to process the change of state on a point	DOMTVRPT	2.2.6.1.1
2. DOMTVARY exits to the user program returning a code indicating the success or failure of the request		

DIAGRAM 2.2.4.3

Input

Transaction code X'95' message

System/7 control table

RCB list

Data base arrays
CARCB
CANALOG
CANAMES
CASTATUS
CACOUNT

2.2.0

Process

1. Convert message text to EBCDIC and release input buffer.
2. Search data base to convert reference number to a name.
3. Identify whether point is being varied online or offline.
4. Patch vary point processor.

Output

DIAGRAM 2.2.5: Hierarchy Vary Point

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. The transaction code X'95' message is passed by the System/7 I/O processor. The message is converted using the ASCICONV macro and the input buffer is released using the RLSEBUFF macro	DOMTS7IO	2.6.3.0
4. DOMTVDB patches DOMTVRPT to process the change in the state of the point	DOMTVRPT	2.2.6.1.1

DIAGRAM 2.2.5

Input

DOMDPCC1

Configuration control element

DOMDPCC2

System/7 control table

CCE

2.2.6

Process

1. Determine whether to vary System/7 terminal or point by patch ID.

   A. Vary System/7.

   (1) Compare access area & function to power system operator.

   (2) Issue VARYS7 macro for each System/7 to be changed.

   (3) Write message to screen indicating result of request.

   B. Vary a terminal.

   (1) Compare access area & function to power system operator and terminals.

   (2) Issue VARYCONF macro.

   (3) Write message to screen indicating result of request.

Output

DOMDPCC1

DOMDPCC2

DIAGRAM 2.2.6.1: Power Configuration Control Processor

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. DOMCFGD2 is patched from one of the three PCC displays by the Display Management System routines: input from all displays consists of a PSREAD of the applicable screen areas	DOMCFGD2	2.2.6.1
1.a. Patch IDs indicating vary a System/7 are: 2 – offline 3 – backup 4 – primary copies segment DOMCFGDQ		
1.a.3 Message is written to screen using DISPLAYM macro		
1.b. Patch ID indicating vary a terminal are: 7 – in service 8 – out of service copies segment DOMCFGDR		
1.b.2 Varyconf macro	DOMTABLE	2.2.4.3
1.b.3 Message is written to screen using DISPLAYM macro		

DIAGRAM 2.2.6.1

Input

DOMDPCC3

CCE

Data base arrays

CARCB
CASTATUS
CANALOG
CACOUNT
CANAMES

Process

C. Vary a point.

(1) Check access area and function to power system operator, terminal and point.

(2) Issue VARYCONF macro.

(3) Write message to screen indicating result of request.

2. Refresh display.

Output

DOMDPCC3

DIAGRAM 2.2.6.1

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1.c. PATCH IDs indicating vary a point are: 11 — in service 12 — out of service copies segment DOMCFGDS		
1.c.2 Varyconf macro	DOMTABLE	2.2.4.3
1.c.3 Messages are written to screen using DISPLAYM macro		
2. Display is refreshed by PATCHing PCC display processor with an ID of 32	DOMCFGD1	2.2.6.2

DIAGRAM 2.2.6.1

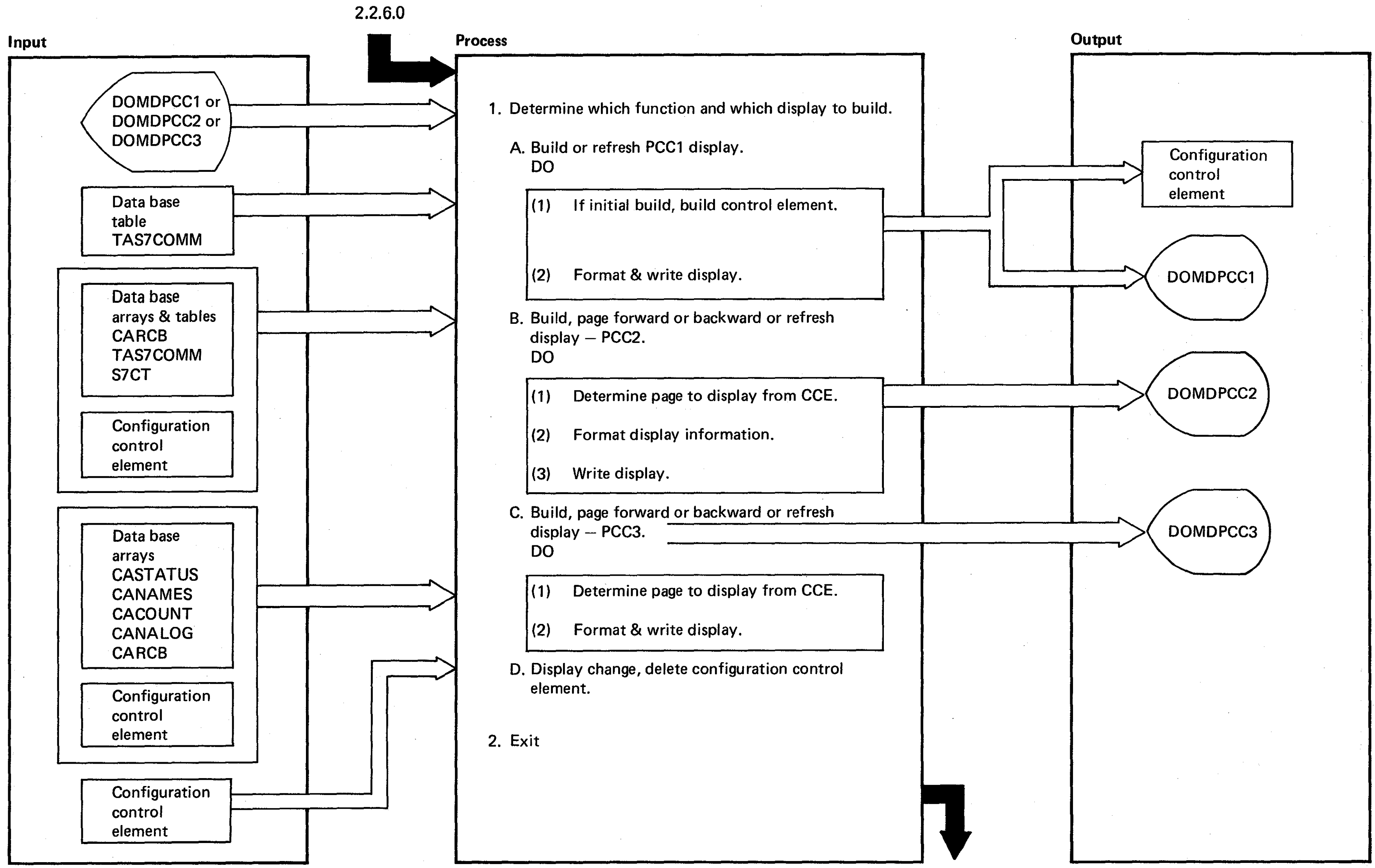


DIAGRAM 2.2.6.2: Power Configuration Control Displays

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. The control module (DOMCFGD1) is patched from the Display Management System display routines to:	DOMCFGD1	2.2.6.2
a. Build a display		
b. Refresh the display		
c. Page forward		
d. Page backward		
e. Indicate the display is no longer active		
This module consists of the following copy segments:		
DOMCFGDA DOMCFGDB DOMCFGDC DOMCFGDD DOMCFGDE		
DOMCFGDF DOMCFGDG DOMCFGDH DOMCFGDI DOMCFGDJ		
1.a. The control module calls DOMCBLD1 to format and write the display	DOMCFGD1	2.2.6.2
1.b. The control module calls DOMCBLD1 to format and write the display	DOMCFGD1	2.2.6.2
1.c. The control module calls DOMCBLD3 to format and write the display	DOMCFGD1	2.2.6.2

DIAGRAM 2.2.6.2

Intentionally Blank

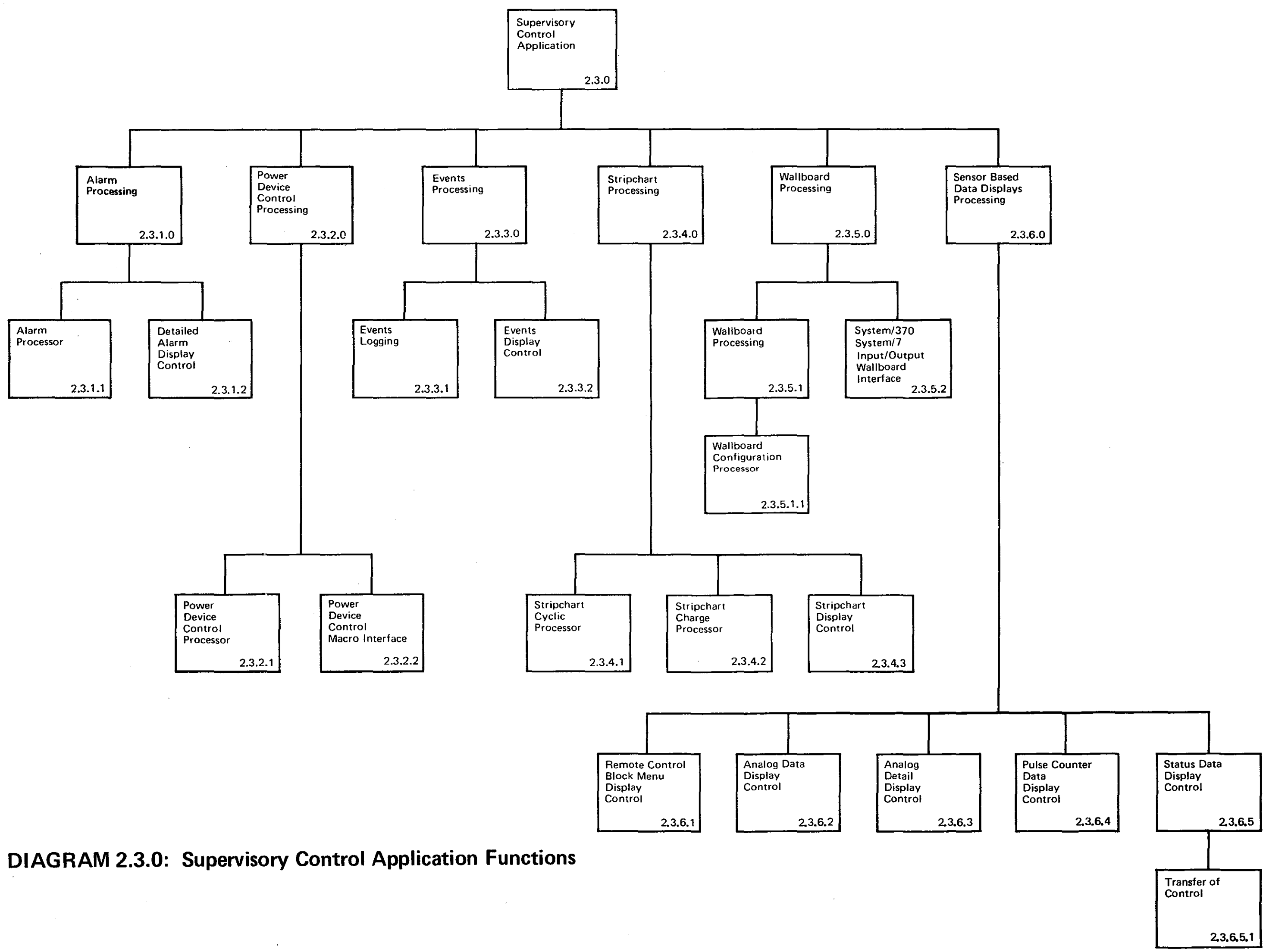


DIAGRAM 2.3.0: Supervisory Control Application Functions

Input

Data base arrays.
CAAATBL
CARCB
CASYS
CASTATUS
CANALOG
CANAMES
CACOUNT
CAAPLPT
CAAPPL

Incore
Alarm
Chain

Scan exception
table or
macro list

2.3.1.0.

Process

1. Determine whether input is scan exception table or macro list and set controls.
2. Build alarm record.
3. If status alarm
   patch change of status log processor.
4. Issue entity.
5. If wall board flag on patch wallboard processor.
6. If add alarm add to incore alarm chain, rechain and adjust controls.
7. If update
   update the alarm in the chain.
8. If delete
   delete the alarm from the chain, rechain and adjust controls.
9. If add or update
   DO
   (a) issue or reissue general alarm
   (b) send audible alarm message to system /7
10. If delete and alarm count for RCB or ACB is zero, delete general alarm.
11. issue event
12. issue typer message.
13. RETURN.

Output

Incore alarm
chain

Data base arrays
CAAATBL
CARCB
CASTATUS
CANALOG
CACOUNT
CAAPLPT
CAAPPL

DIAGRAM 2.3.1.1: Detail Alarm Processor

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. The control program (DOMCALR1) receives control from:	DOMCALR1	2.3.1.1
scan processing	DOMTSSYN	2.2.1.1
DOMCALRM macro	DOMTABLE	2.7.2.0
configuration change processor	DOMCFGD2	2.2.6.1
2. The control program calls:		
status processor	DOMCALR1	2.3.1.1
analog processor	DOMCALR1	2.3.1.1
pulse counter processor	DOMCALR1	2.3.1.1
or processes non-sensor based (text) alarms itself		
3. Change of status log processor is PATCHed with list of changes	DOMCSLOG	2.2.3.3
4. Entity is updated using DISPENT macro		
5. Module called to do action on alarm and issue messages	DOMCALR1	2.3.1.1
5. Patch wallboard processor with list of items	DOMCWBPR	2.3.5.1
9.a. General alarm is issued using DALARM macro		
9.b. Transaction code X'10' message sent using S7WRITE macro		
10. Delete general alarm using DALARM macro		
11. A list of events is passed to the event processor	DOMCEVT5	2.3.3.1
12. The message is issued to the typers using the MESSAGE macro		

DIAGRAM 2.3.1.1

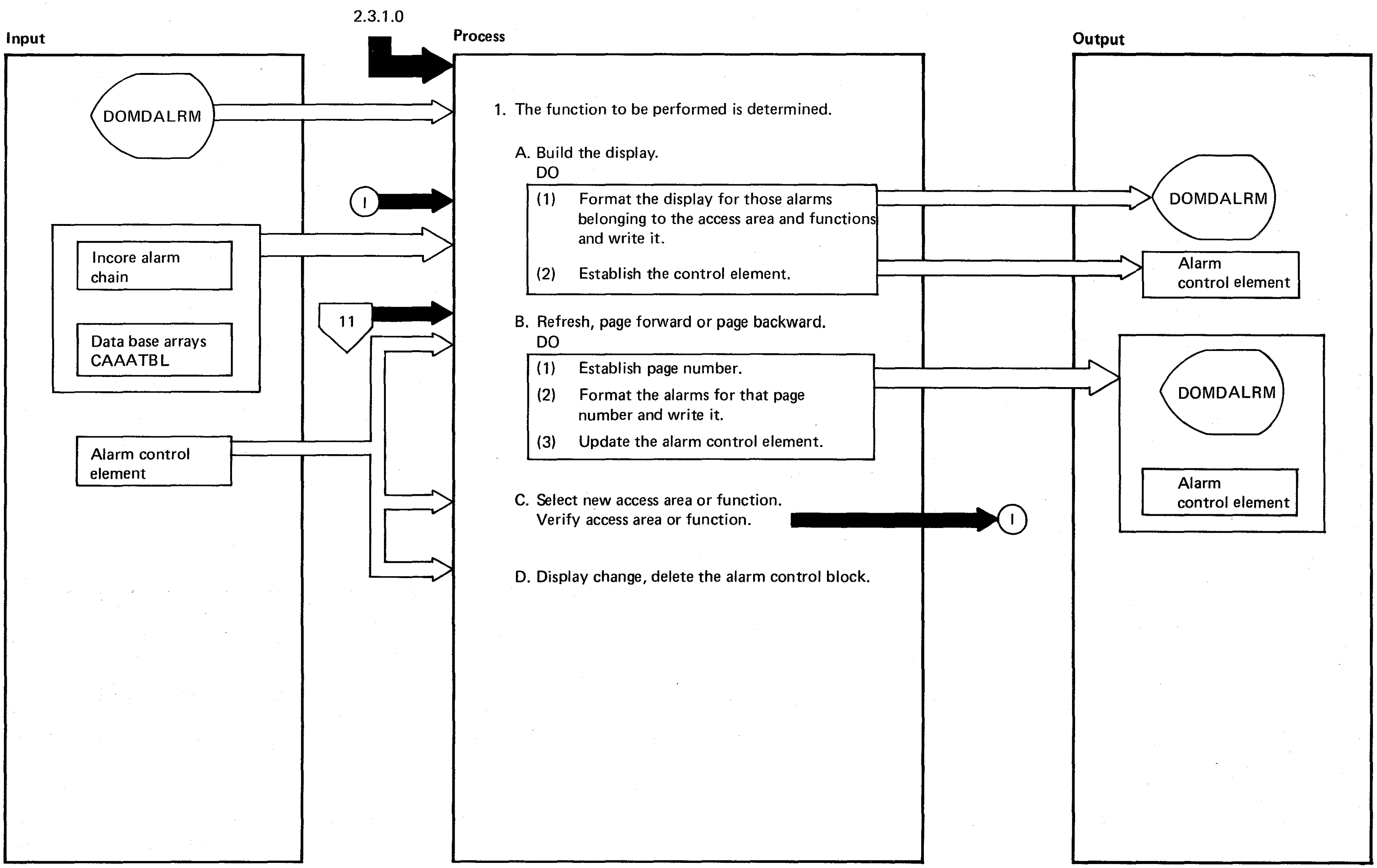


DIAGRAM 2.3.1.2

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. The display processor is PATCHed by the Display Management System display *		
1.a. The control module calls the retrieval module to find the applicable alarms; calls the display formatter to write the display	DOMCALD1	2.3.1.2
1.b.2 The control module calls the retrieval module and calls the display formatter to write the display	DOMCALD1	2.3.1.2

DIAGRAM 2.3.1.2

Input

Data base arrays
CARCB
CASTATUS
CASYS
CACOUNT
CANALOG
CANAMES
CAAPLPT
CAAPPL

Process

E. Acknowledge or delete an alarm.
DO

(1) Find alarm in incore alarm chain.

(2) If acknowledge, set acknowledge indicator.

(3) If delete,
DO

(a) If all alarms deleted for RCB or ACB, delete general alarm.

(b) Issue entity to change to non-alarmed state.

(c) Delete alarm from chain.

(4) If wallboard indicator on, patch wallboard processor.

(5) Event the action.

(6) If status point, patch status logger.

(7) Issue alarm type message.

11

Output

Data base arrays
CARCB
CASTATUS
CACOUNT
CANALOG
CAAPPLT
CAAPPL

Wallboard item

Event

Alarm type message

DIAGRAM 2.3.1.2

EXTENDED DESCRIPTION

Notes	Modules	Diagram
e.3.a Delete general alarm using DALARM macro		
b Issue entity using DISPENT macro		
e.4. Pass wallboard processor address of changed item	DOMCWBPR	2.3.5.1
e.5. Event using the SCEVENT macro		
e.6. Pass the status item to the change of status logger	DOMCSLOG	2.2.3.3
e.7. Issue the alarm typer message using MESSAGE macro		

DIAGRAM 2.3.1.1

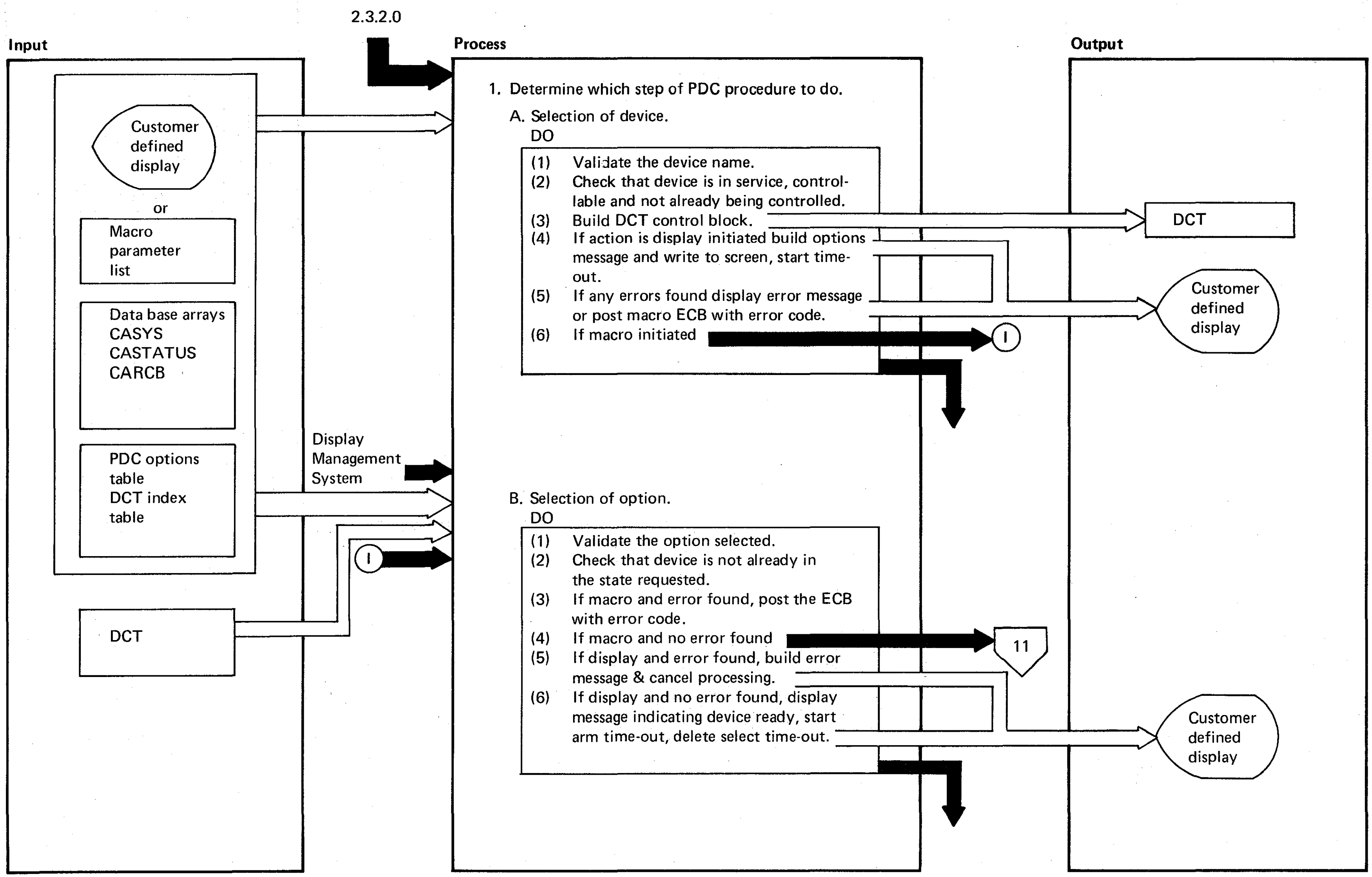


DIAGRAM 2.3.2.1: Power Device Control Processing

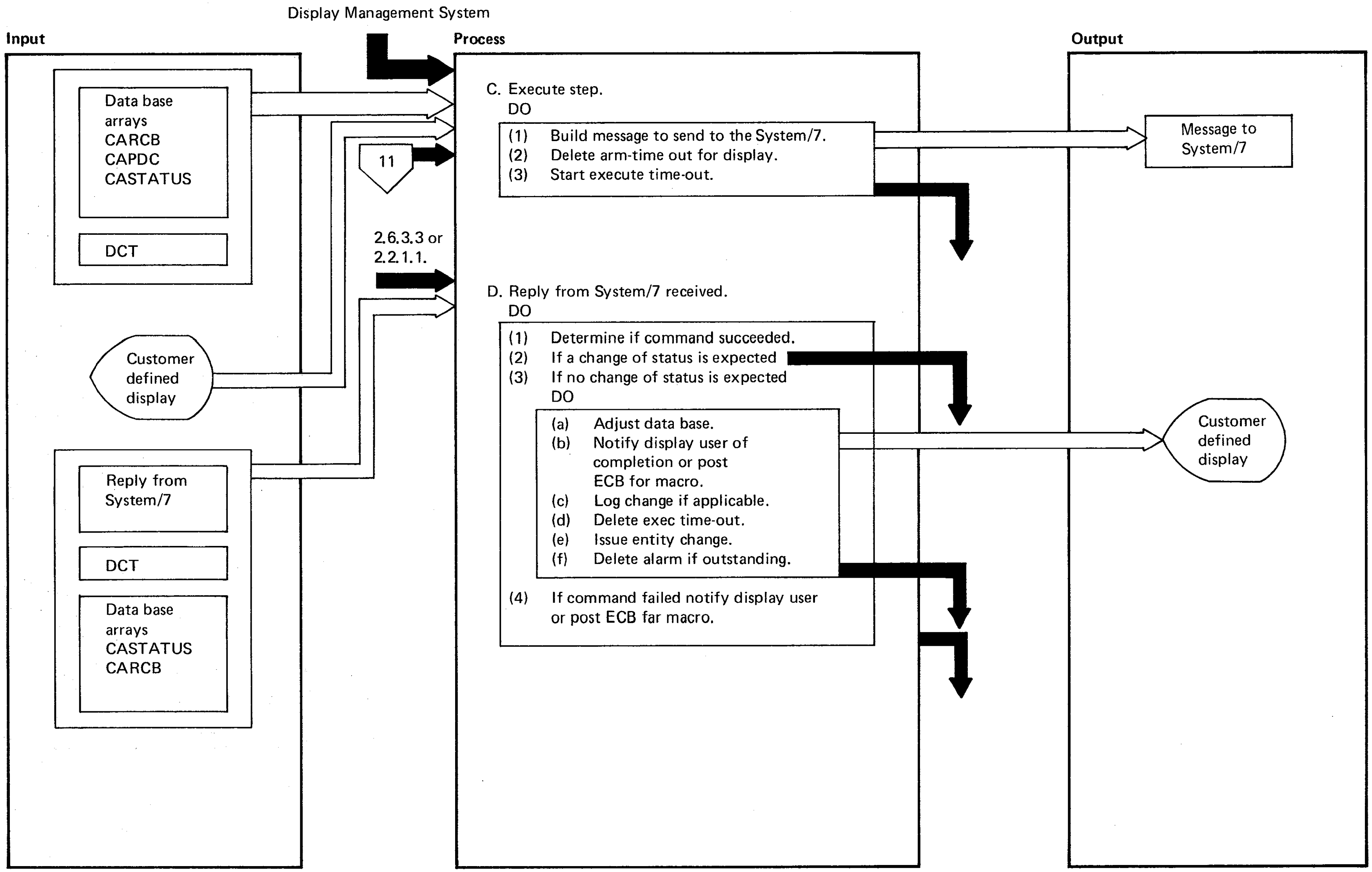


DIAGRAM 2.3.2.1

2.2.1.1.

Input

Change of status information

Time-out parameter

DCT

Process

E. Change of status received.
DO steps 1.D.(3)(b) through 1.D.(3)(f) above.
Delete execute time-out.

F. Error processing.
DO
(1) Time-out is associated with device.
(2) Display user is notified or ECB is posted for macro.

G. Completion/cancellation.
Areas reserved for the process (DCT) are released and data base flags are reset.

2. Eventing.
Events are issued for each step of the procedure using the SCEVENT macro.

Output

Customer defined display

Data base array CASTATUS

DIAGRAM 2.3.2.1

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. The PDC control processor is PATCHed from the customer defined display by the Display Management System display routines or from the PDC macro interface module	DOMCDC01 DOMCDC04	2.3.2.0 2.3.2.2
1.a. The control processor calls DOMCDC06 to validate the device name and set up the control tables	DOMCDC01	2.3.2.0
1.a.4 The selection time-out is initiated by issuing a PTIME for DOMCDC03 for 30-seconds	DOMCDC03	2.3.2.0
1.b. The control processor calls DOMCDC07 to process the selection request from the display. For the macro, DOMCDC06 calls DOMCDC11 to verify the action selected. For the display, DOMCDC07 calls DOMCDC11	DOMCDC01 DOMCDC01	2.3.2.0 2.3.2.0
1.b. If the device is defined as manual, DOMCDC11 would process the change requested and adjust the data base, and complete the control action		
1.b.6 The selection time-out is deleted using the PTIME macro. The arm-time is initiated by issuing the PTIME macro for DOMCDC03 for 30-seconds	DOMCDC03 DOMCDC03	2.3.2.0 2.3.2.0
1.c. If display initiated, DOMCDC07 is called to process the execute request This module calls DOMCDC05 to build and send the message If macro initiated, DOMCDC06 calls DOMCDC05	DOMCEC01 DOMCDC01 DOMCDC01	2.3.2.0 2.3.2.0 2.3.2.0
1.c.1 The transaction code X'06' message is sent to the System/7 using the S7WRITE macro		
1.c.2 The arm time-out is deleted using the PTIME macro	DOMCDC03	2.3.2.0
1.d. The reply from the System/7 is received from the following sources: • Transaction code X'16' – Tag/untag before scanning • Transaction code X'86' – For raise/lower commands • In the raw data array The control program calls DOMCDC08 to process the reply	 DOMTROUT DOMTROUT DOMTSSYN DOMCDC01	 2.6.3.3 2.6.3.3 2.2.1.1 2.3.2.0
1.d.3.c Change is logged by PATCHing change of status log processor	DOMCSLOG	2.2.3.3
1.d.3.d Time-out is deleted using PTIME macro	DOMCDC02	2.3.2.0
1.d.3.e Entity is changed using DISPENT macro		

DIAGRAM 2.3.2.1

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1.d.3.f Alarm is deleted using DOMCALRM macro		
1.e. The change of status information is received from scan processing	DOMTSSYN	2.2.1.1
1.f. Time-out information is received from DOMCDC02 or DOMCDC03		

DIAGRAM 2.3.2.1

Intentionally Blank

2.3.2.0

Input

Macro parameter list

Data base arrays
CARCB
CASYS

Process

1. The macro parameter list is formatted for the PDC processor.
2. The PDC processor is PATCHed with the formatted information.
3. The ECB is posted by DOMCDC01 with the results of the control action.
4. A message indicating the result of the action is retrieved and moved to the user area if requested.
5. A return code is returned to the user indicating the success or failure of the action.

Output

Parameter list

DIAGRAM 2.3.2.2: Power Device Control Macro Interface

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. The input is received via a PATCH from DOMCDC10 and is controlled from this module DOMCDC04	DOMTABLE DOMCDC04	2.7.2.0 2.3.2.2
2. DOMCDC04 PATCHes DOMCDC01 to process the PDC request	DOMCDC01	2.3.2.1
If the macro passes a list of devices to control, the above steps are followed for each device passed and the message and return code are returned when the processing is complete		

DIAGRAM 2.3.2.2

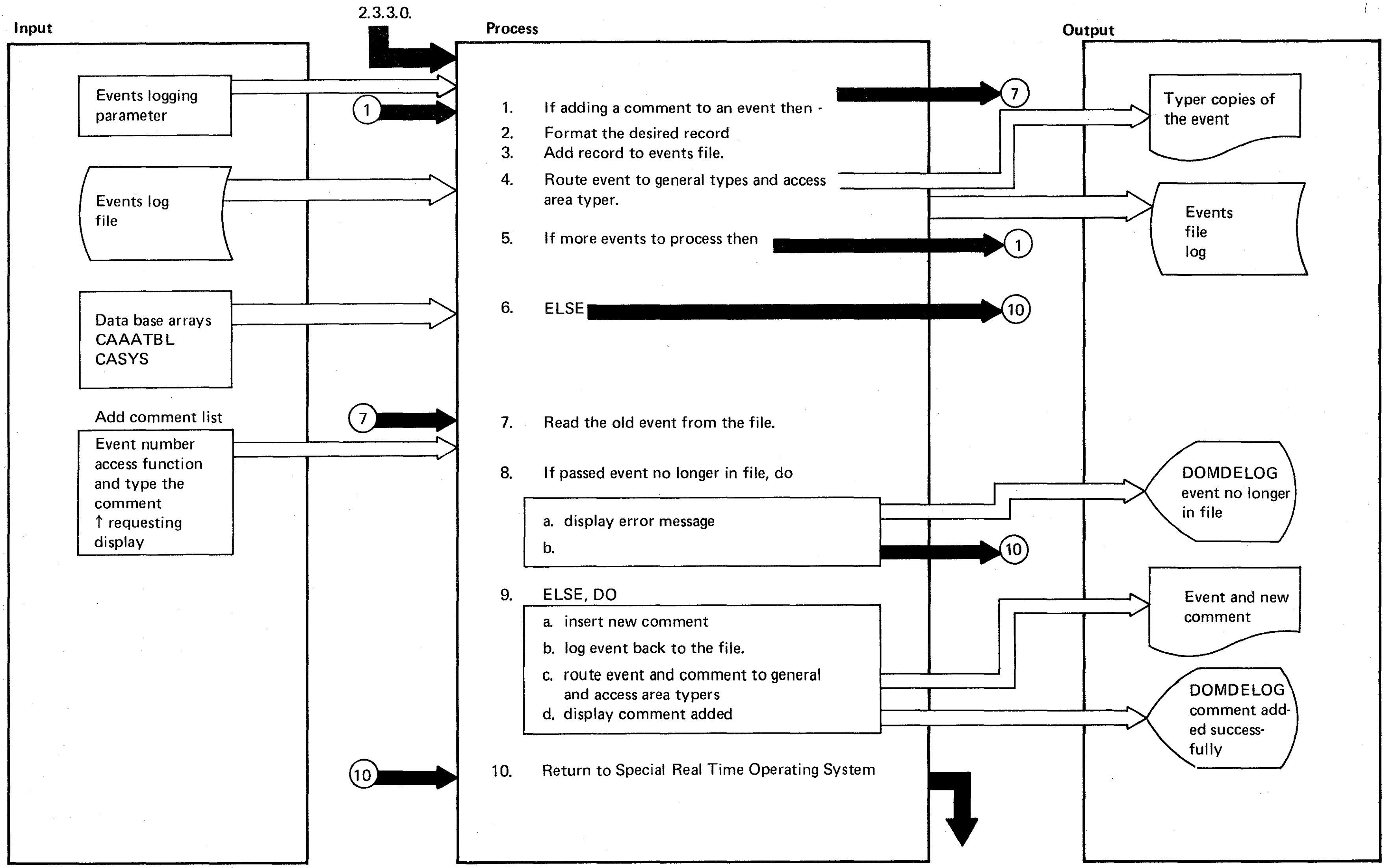


DIAGRAM 2.3.3.1: Events Logging

EXTENDED DESCRIPTION

Notes	Modules	Diagram
The address of the events logging parameter list is passed as the patch parameter from either the SCEVENT macro processor or the alarm processor 3. The events file is a closed loop, circular file, and is maintained by direct access methods 4,9.c The MESSAGE macro is used to route the event to the typers using routing codes. The general typer routing code is in the CASYS array while the access area typer is in the CAAATBLE The add comment list is passed by the events display control program	DOMCEVD4	2.3.3.2

DIAGRAM 2.3.3.1

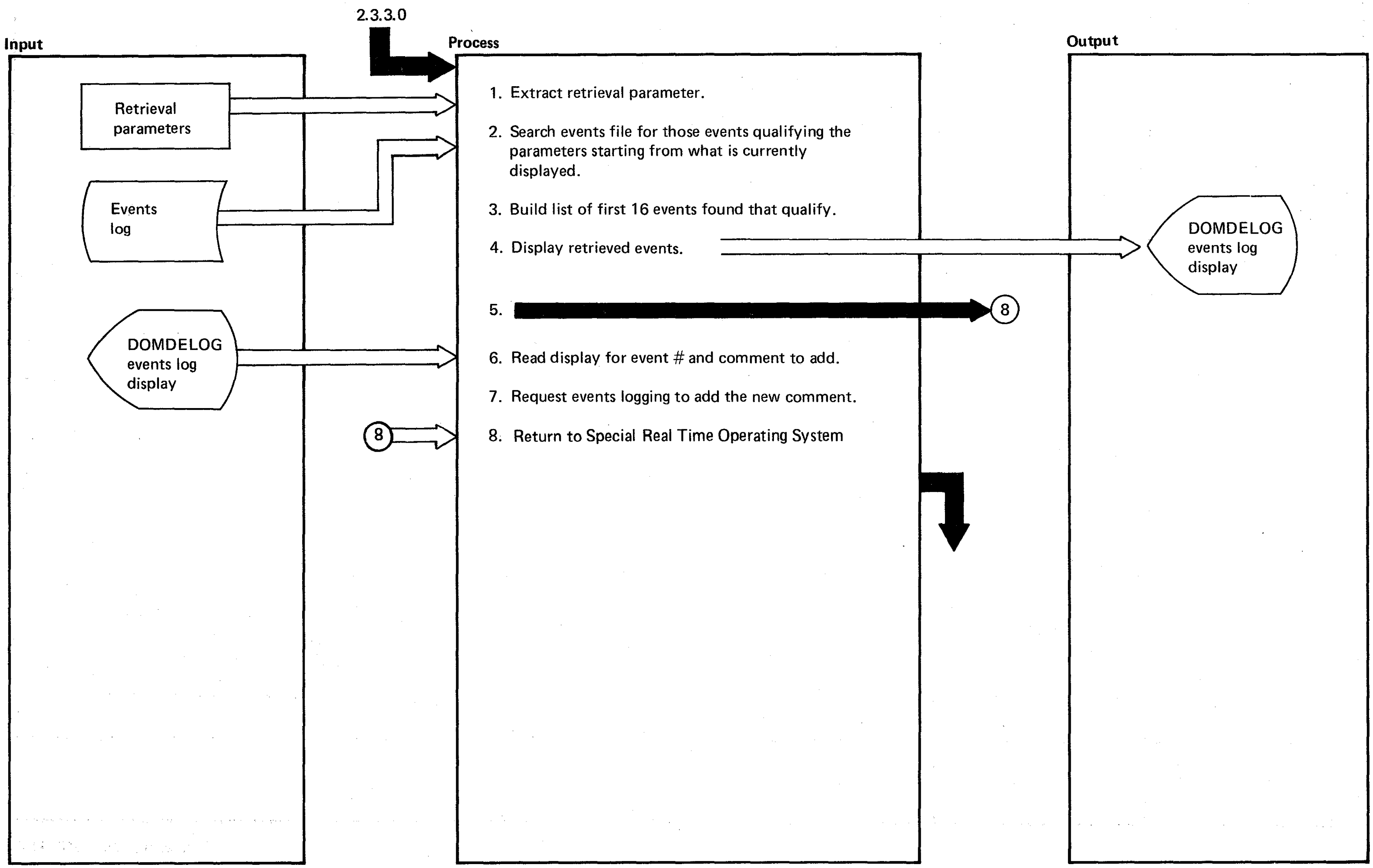


DIAGRAM 2.3.3.2: Events Display Control

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. Retrieval parameters are built by the menu display program	DOMCEVD1	2.3.3.2
4. The retrieval parameters are updated to reflect what is currently on the display		
7. Request is made via a PATCH to the events logging program	DOMCEVT1	2.3.3.1

DIAGRAM 2.3.3.2

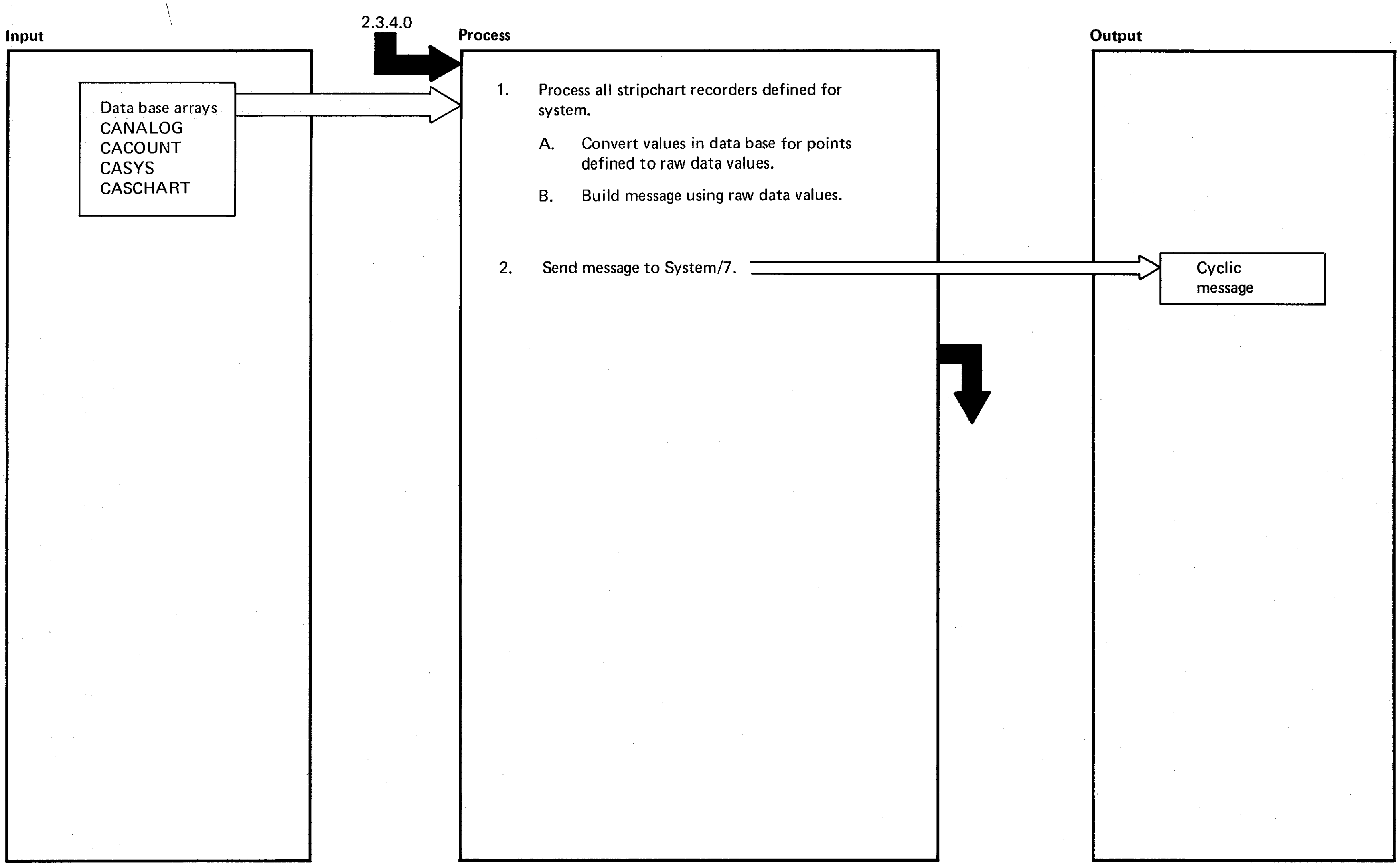


DIAGRAM 2.3.4.1: Stripchart Cyclic Processor

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. PTIME DOMCHRTC at the basic scan cycle rate whenever any recorders are active. 2. Issue System/7 message transaction code X'13' using S7WRITE macro	DOMCHRTA	2.3.4.2

DIAGRAM 2.3.4.1

2.3.4.0

Input

Data base arrays
CASCHRT
CANALOG
CANAMES
CACOUNT
CARCB
CASYS

Data base array
CASCHART

Data base array
CASCHART

Process

1. The function to be performed is determined.

   A. Change the status of the recorder.
   DO

   (1) Validate the information received.
   (2) Build message to send to System/7.
   (3) Start time-out interval.

   B. SCC reply received from System/7.
   DO

   (1) Delete the time-out.
   (2) Update the array to show new status.
   (3) If first recorder to be activated start cyclic processor.
   (4) If all recorders now off delete cyclic processor.
   (5) Log the stripchart array.

   C. Time-out
   Reset the array entry to its previous state.

2. Post macro ECB with return code indicating success or failure of request.

Output

System/7 message

Data base array
CASCHART

Log files

DIAGRAM 2.3.4.2: Stripchart Change Processor

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. This module (DOMCHRTA) receives control from the DOMCSCHT macro interface module	DOMTABLE	2.7.2.0
1.a.2 The transaction code X'08' message is sent to the System/7 using the S7WRITE macro		
1.a.3 The time-out is activated by PTIMEing itself for a 10-second interval	DOMCHRTA	2.3.4.2
1.b. The System/7 reply is a transaction code X'88'		
1.b.1 The time-out is deleted using the PTIME macro		
1.b.3 The cyclic processor (DOMCHRTC) is started using the PTIME macro at the basic scan cycle rate	DOMCHRTC	2.3.4.1
1.b.4 The cyclic processor is deleted using the PTIME macro		
1.b.5 The CASCHART array is logged using the PUTLOG macro		

DIAGRAM 2.3.4.2

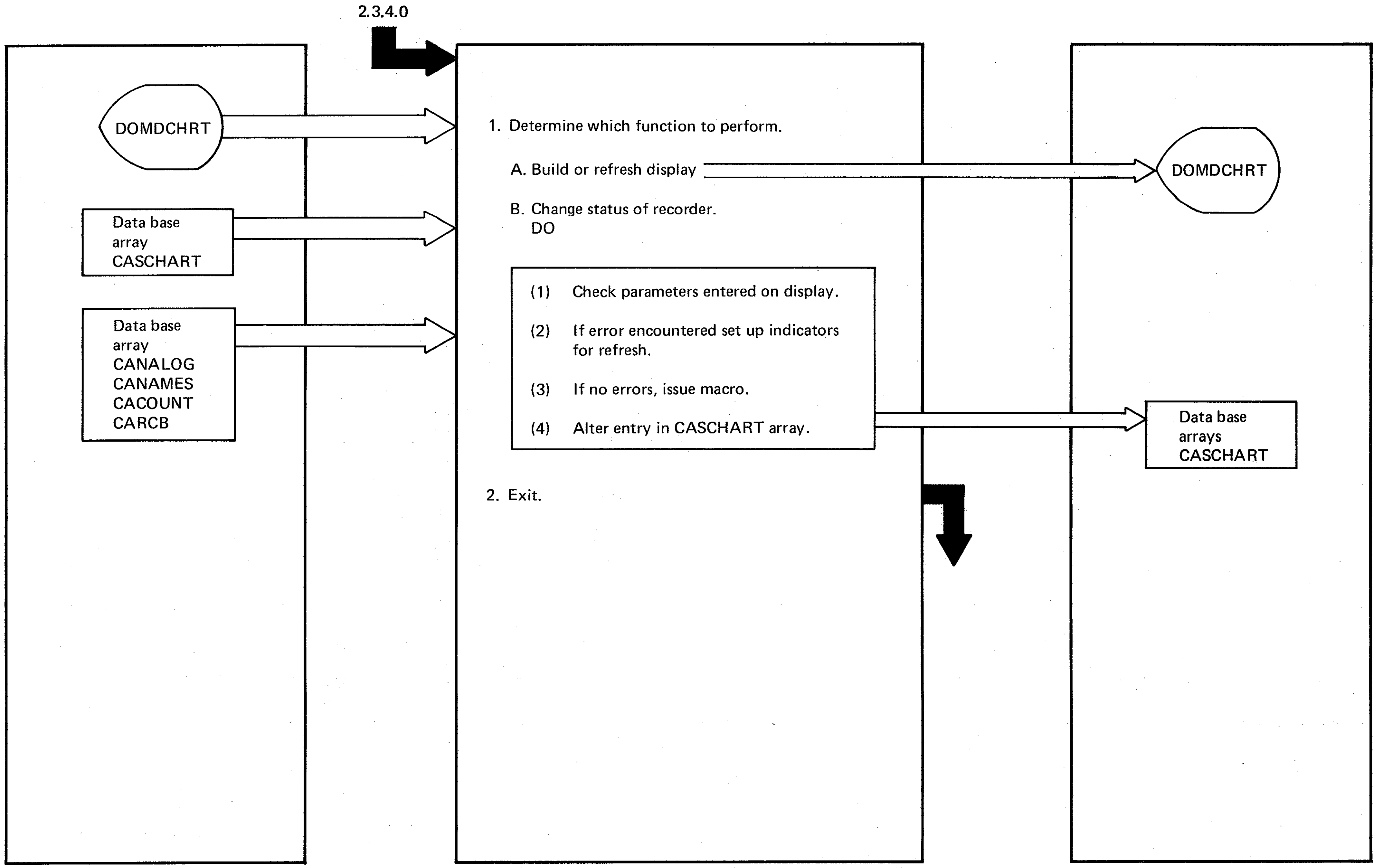


DIAGRAM 2.3.4.3: Stripchart Display Processor

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. The display processor (DOMCHART) is patched from the Display Management System display routines	DOMCHART	2.3.4.3
1.a. DOMCHRT1 is called to format and write the display	DOMCHART	2.3.4.3
1.b. DOMCHRT2 is called to process change requests. It issues the DOMCSCHT macro	DOMCHART	2.3.4.3
1.b.2 Errors encountered are included in the display when refreshed		

DIAGRAM 2.3.4.3

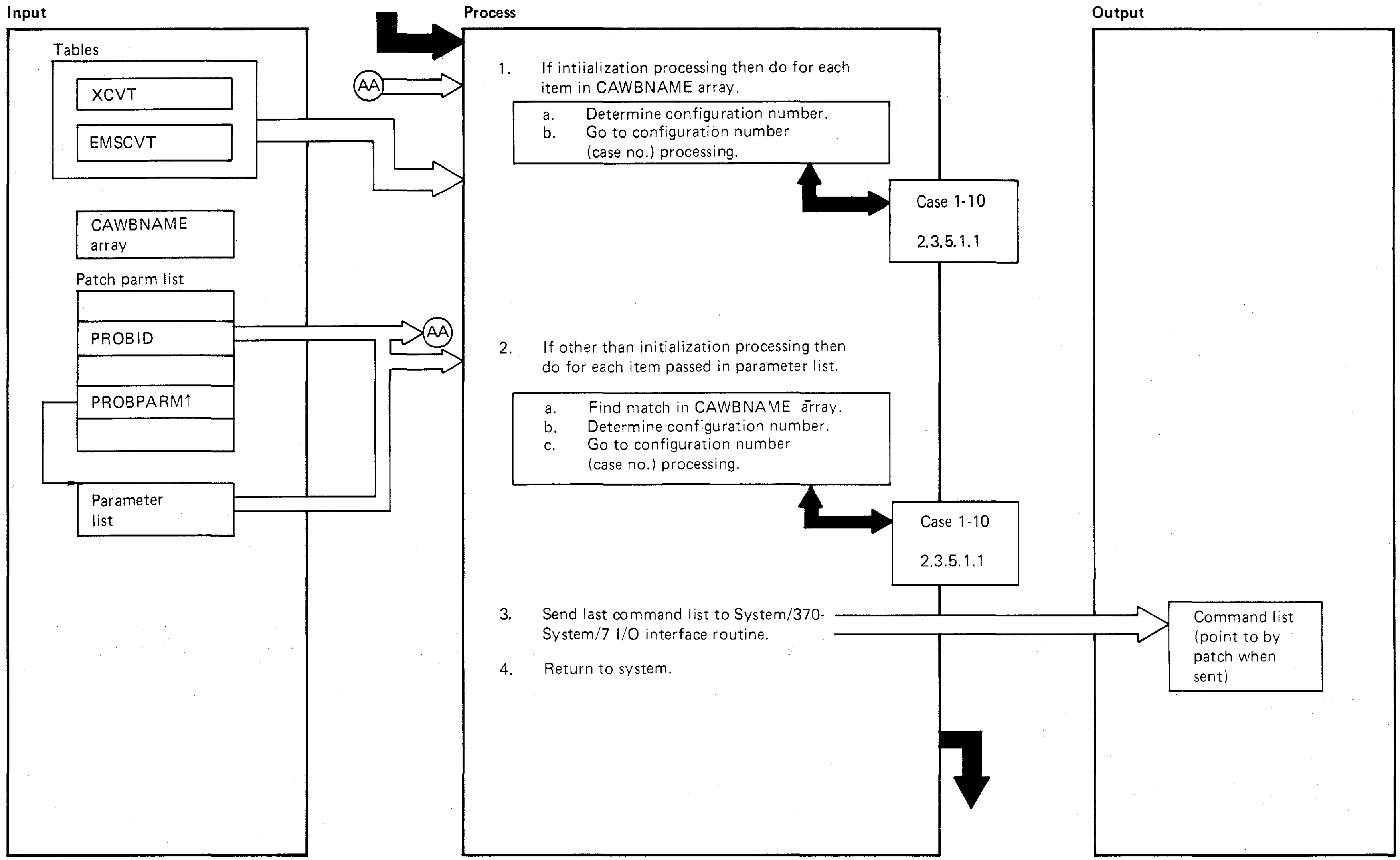


DIAGRAM 2.3.5.1: Wallboard Processing

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. Patch ID of 4 is received from data acquisition whenever initial scan received from System/7 wallboard controller or this processing takes place when the wallboard is placed in the automatic mode from the manual mode	DOMCWBPR	
1.b. CASE 1-10, Each configuration No. (Case No.) is processed separately under a subroutine	DOMCWBPR	2.3.5.1.1
2. Patch ID of 8 is received from Scan Control, Alarm Controller, Alarm Display Controller, or Power Device Control whenever an action is taken on a wallboard defined item. Wallboard processing will only be patched if a wallboard is defined with the system. The parameter list received is one or more data item addresses in the data base which has had some action take place.	DOMCWBPR	2.3.5.1
2.c. Same as 1.b.		
3. The command list that has been generated is sent as a parameter of the patch when the interface routine is patched	DOMCWBPR	

DIAGRAM 2.3.5.1

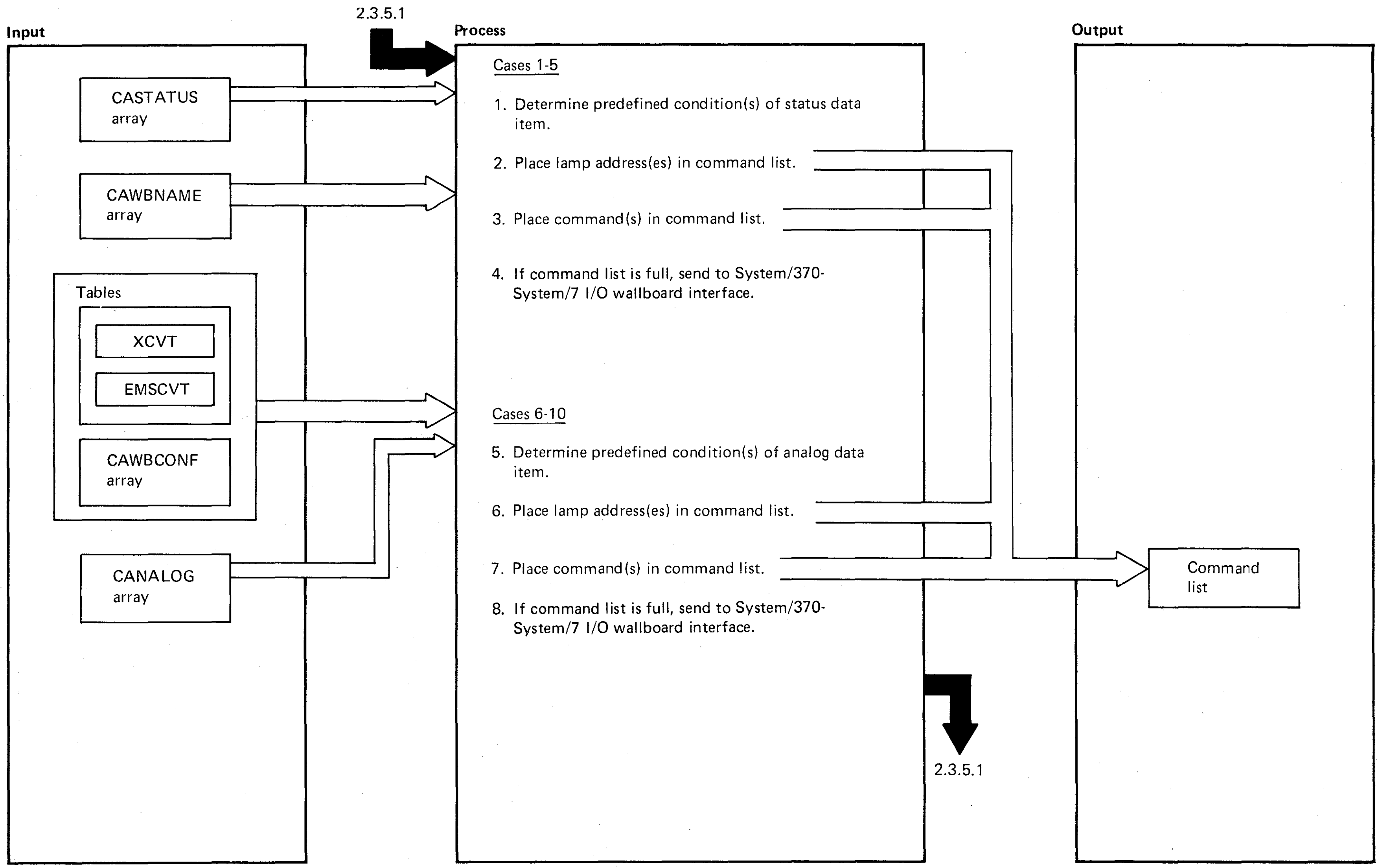


DIAGRAM 2.3.5.1.1: Wallboard Configuration Processor

EXTENDED DESCRIPTION

Notes	Modules	Diagram
Note: The command list generated is sent to the System/370-System/7 IO Wallboard Interface. A pointer to the command list is included as a parameter field in the PATCH.		
1-4. Cases 1-5 or Configuration No's 1-5 are only for DI points (status data). The configurations (No. Lamps, conditions which cause each lamp(s) to be set to a certain state, and the lamp states) are all predefined	DOMCWBPR	2.3.5.1
5-8. Cases 6-10 or Configuration No's 6-10 are only for AI points (Analog data). The configurations (No. Lamps, conditions which cause each lamp(s) to be set to a certain state, and the lamp states) are all predefined	DOMCWBPR	2.3.5.1

DIAGRAM 2.3.5.1.1

2.3.5.0

Input

Tables

XCVT

EMSCVT

S/7 Communication

Sysgen options array

Command list

Process

1. If Patch ID = 2 (new command list)

   a. If command list chain already exists then place command list address at end of chain.

   b. Else send command list to System/7 (transaction code 14).

2. If Patch ID = 4 (return from System/7 after initialization command list).

   a. Send first command list in chain to System/7 (transaction code 14).

   b. If initialization command list chain is complete then send all remaining command lists to System/7 (transaction code 14).

3. Return to system.

Output

See AA in extended description

EMSCVT (command list chain)

Command list

DIAGRAM 2.3.5.2: System/370-System/7 Input Output Wallboard Interface

EXTENDED DESCRIPTION

Notes	Modules	Diagram
Note: No processing takes place if the System/7 controlling the wallboard is not active and scanning		
1. Control is received from wallboard process or DOMCWBPR module with a patch parameter pointing to the command list	DOMCWBS7	2.3.5.2
1.a. During initialization only one command list is shipped to the seven until a return transacation code of 94 is received back (indicates the System/7 is ready to process the next command list sent)	DOMCWBS7	2.3.5.2
2. No data is received back from the System/7. A transaction code of 94 causes a patch to DOMCWBS7 with an ID of 4	DOMCWBS7	
AA. The initial command list address is kept in the EMS CVT. Each command list chained points to the next by a pointer in the first word of the command list. The transaction code 14 header is completed in the command list and the chaining address is dropped before shipping to the System/7	DOMCWBS7	

2.3.6.0

Input

Tables
- XCVT
- EMSCVT
- AID
- DCE
- CEATAB

CARCB array

Remote control element

Tables
- CEATAB
  - CEASBDT
- AID
- DCE

Process

The PATCH ID determines which processing route is taken.

1. PATCH ID = 10—build RCB menu display page one.
   a. Get storage and build remote control element (RCE) if Control Element Address (CEA) Table entry is zero for Sensor Based Data (CEASBDT).
   b. Put address of RCE in CEASBDT if CEASBDT is zero.
   c. Build display page and format in buffer.
   d. Write formatted page to display screen.
2. PATCH ID = 20—page RCB display forward one page.
   a. Build and format next page.
   b. Write formatted page to display screen.
3. PATCH ID = 25—page RCB display backward one page.
   a. Build and format preceeding page.
   b. Write formattted page to display screen.
4. PATCH ID = 40—display change.
   a. If new display is not a sensor based data display, free RCE storage and zero CEASBDT field in CEATAB.

Return to system.

Output

Remote control element

CEATAB
- CEASBDT

RCB Menu Display

RCB Display

CEATAB
- CEASBDT

DIAGRAM 2.3.6.1: Remote Control Block Menu Display Control

EXTENDED DESCRIPTION

Notes	Modules	Diagram
Note: Control is received by manual input of Sensor Based Data (Remote Control Block Menu) Display request by the power system operator	DOMCSGET	2.3.6.1
2. If display is on the last page then the first page will be built and displayed	DOMCSGET	2.3.6.1
3. If display is on the first page then the last page will be built and displayed	DOMCSGET	2.3.6.1
4. The new display name is compared against a list of all valid sensor based data display names	DOMCSGET	

DIAGRAM 2.3.6.1

Page of LY20-2226-0
Updated August 31, 1976
By TNL: LN20-3620

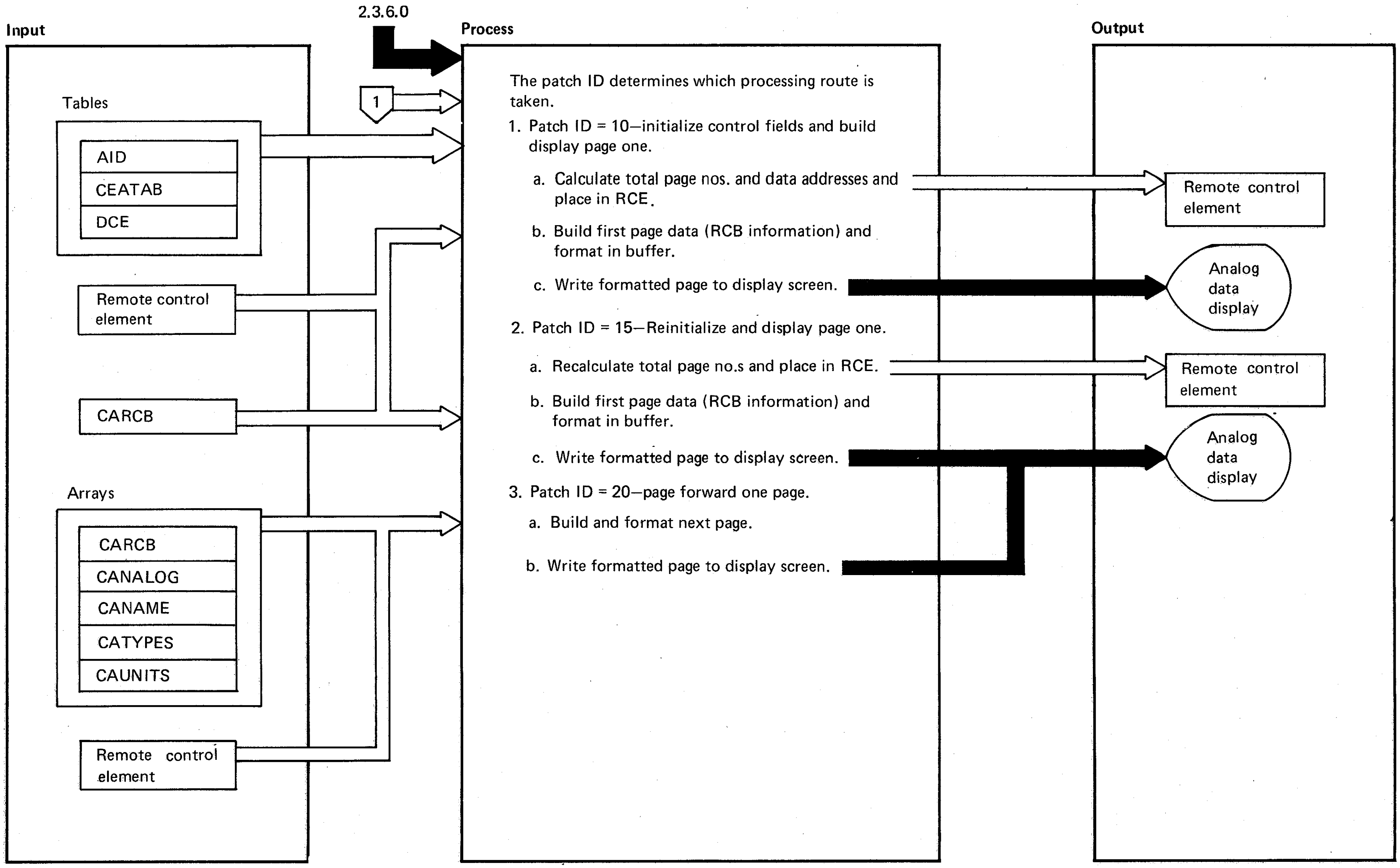


DIAGRAM 2.3.6.2: Analog Data Display Control

EXTENDED DESCRIPTION

Notes	Modules	Diagram
Note: Control is received by manual action taken by a dispatcher (i.e., depressing a program function key)		
2. This function is executed any time this function receives control (2.3.6.2) from other sensor based data display functions (2.3.6.4, 2.3.6.3, and 2.3.6.5). Display pages 1-N are the pages used to display analog data	DOMCSANA	2.3.6.2
3. If display is on the last page then the first page will be displayed	DOMCSANA	2.3.6.2

DIAGRAM 2.3.6.2

Page of LY20-2226-0
Updated August 31, 1976
By TNL: LN20-3620

Process

Tables

- AID
- DCE
- CEATAB

Arrays

- CARCB
- CANALOG
- CANAME
- CATYPES
- CAUNITS

Remote control element

Data from screen

- XCVT
- EMSCVT

Input

4. Patch ID = 25—page display backward one page.

   a. Build and format preceeding page.

   b. Write formatted page to display screen.

5. Patch ID = 30—refresh current displayed display page.

   a. Build and format current page.

   b. Write formatted page to display screen.

6. Patch ID = 35—update analog values entered from screen in data base.

   a. Check for invalid characters input from screen. Format error indicator if not OK.

   b. Check for item being offline. Format error indicator if not offline.

   c. Check to see if display unit has correct access area and function code to update this item. Write event error messages if not OK.

   d. If all three above conditions are met successfully then

1

DOMCNCNV (convert no. to floating point)

Output

Analog data display

Event(s) and system messages

DIAGRAM 2.3.6.2

EXTENDED DESCRIPTION

Notes	Modules	Diagram
4. If display is on first page then the last page will be built and displayed	DOMCSANA	2.3.6.2
5. The page currently being displayed is re-displayed using current data from the data base. This function is invalid for the first page of the display	DOMCSANA	2.3.6.2
6.d. CSECT DOMCNCNV is passed the EBCDIC form of the new value and converted to single precision floating point	DOMTABLE	2.3.6.2

DIAGRAM 2.3.6. 2

Input

Tables

- AID
- DCE
- CEATAB

Arrays

- CARCB
- CANALOG
- CANAME
- CATYPES
- CAUNITS

Remote control element

Process

e. Replace converted value in data base.

f. Build and display current display page.

1

g. Write formatted page to display screen.

7. Patch ID = 40 display change

a. If new display is not a sensor based data display, free RCE storage and zero CEASBDT field in CEATAB.

8. Return to system.

Output

CANALOG

Analog data display

CEATAB

- CEASBDT

DIAGRAM 2.3.6.2

EXTENDED DESCRIPTION

Notes	Modules	Diagram
6.e. If update is successful, the new value is placed in the data base and an event and message are generated	DOMCSANA	2.3.6.2
7.a. The new display name is compared with a list of all valid sensor based data display names	DOMCSANA	2.3.6.2

DIAGRAM 2.3.6.2

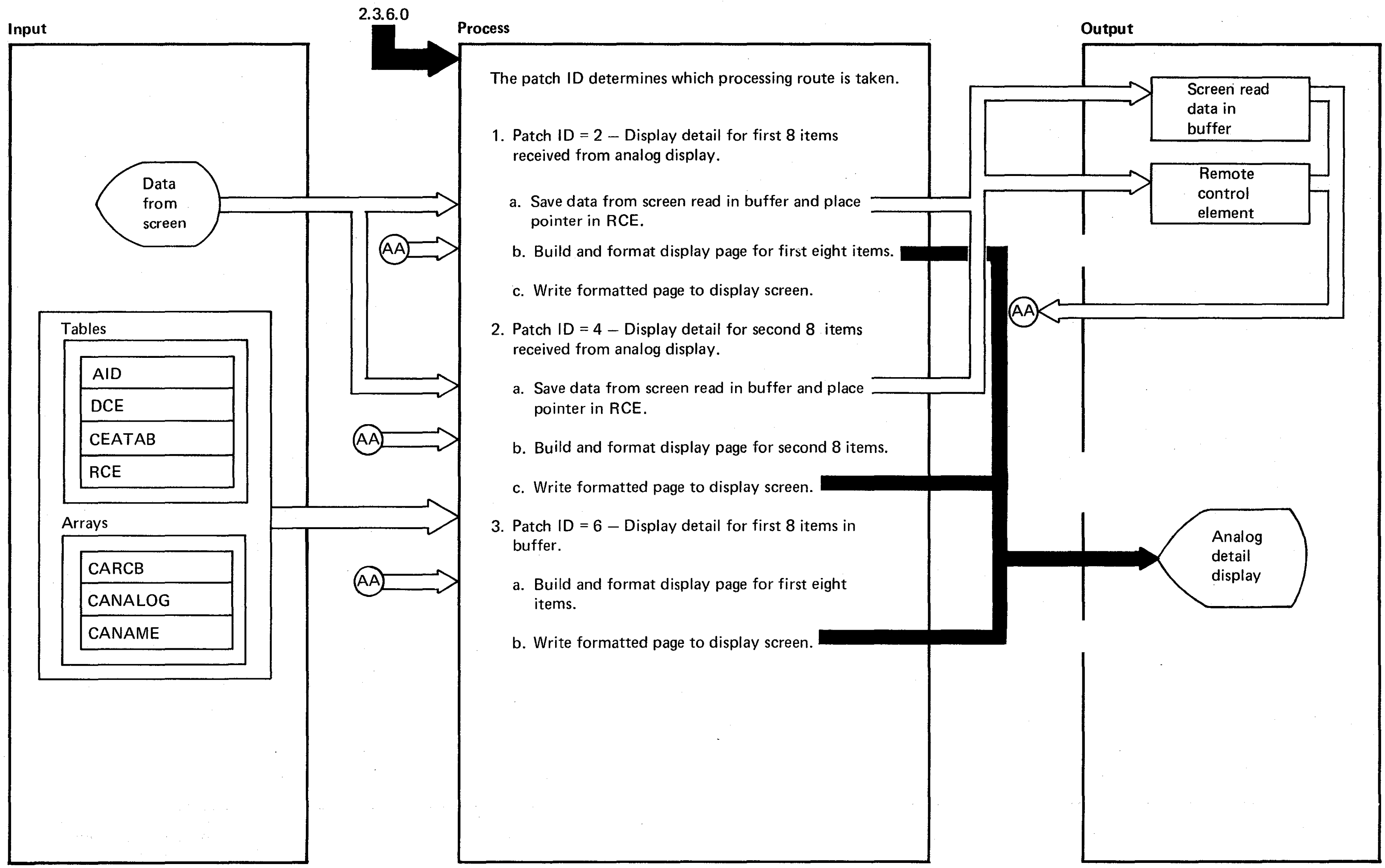


DIAGRAM 2.3.6.3: Analog Detail Display Control

EXTENDED DESCRIPTION

Notes	Modules	Diagram
Note: Control is received by manual action taken by a operator (i.e., depressing a program function key)		
1. & 2. Control is received with display screen read data from the analog data display	DOMCSANL	2.3.6.3
3. Input data for display page is from buffer generated during steps 1 or 2	DOMCSANL	2.3.6.3

DIAGRAM 2.3.6.3

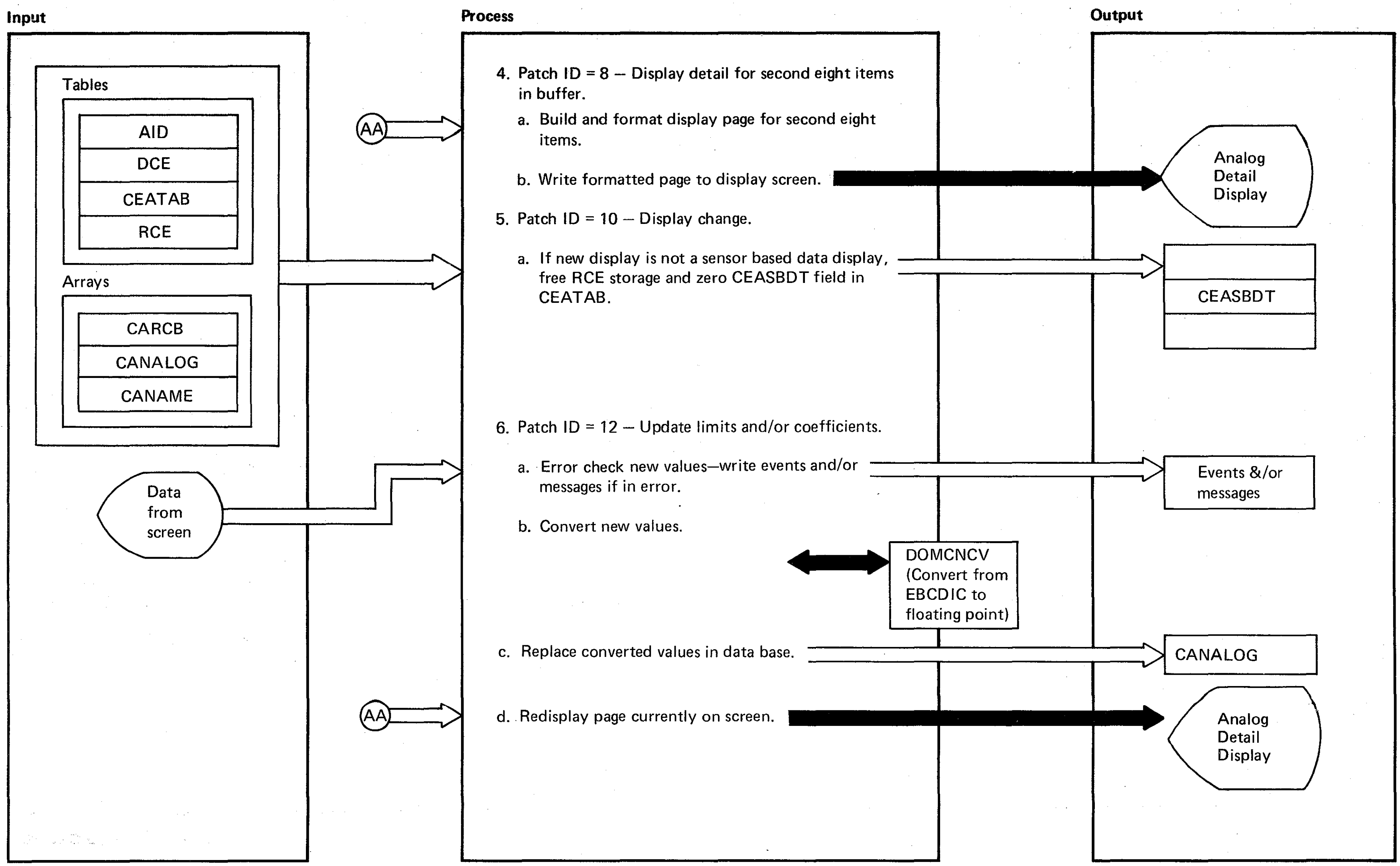


DIAGRAM 2.3.6.3

EXTENDED DESCRIPTION

Notes		Modules	Diagram
4.	Input data for display page is from buffer generated during steps 1 or 2	DOMCSANL	2.3.6.3
5.a.	The new display name is compared with a list of all valid sensor based data display names	DOMCSANL	2.3.6.3
6.a.	New values are checked for (1) valid characters, (2) if item has access area and function code as display unit, and (3) if item is off line	DOMCSANL	2.3.6.3
6.b.	Operating limits and coefficients are converted from EBCDIC to floating point and then the operating limits are converted into warning limits to be stored in data base	DOMTABLE	2.7.2.0

Input

Tables

- AID
- DCE
- CEATAB
- RCE

Arrays

- CARCB
- CANALOG
- CANAME

Screen read data in buffer

Process

7. Patch ID 14 and 16 – Change sysgen options alarmability or user conversion.

   a. If item is not offline, or display unit access areas and function codes are not correct then generate error messages and/or event(s).

   b. If checks in 7a. are OK and patch ID = 14 then reverse alarmability bit in data base and event.

   c. If checks in 7a. are OK and patch ID = 16 then reverse user conversion bit in data base and event.

   d. Redisplay page currently on screen.

8. Return to system.

Output

Events and messages

CANALOG

Events and messages

Analog detail display

DIAGRAM 2.3.6.3

EXTENDED DESCRIPTION

Notes	Modules	Diagram
7.a. Display unit may also be power system operator (can update all access area/function codes)	DOMCSANL	2.3.6.3
7.d. Page is redisplayed after successful modifications to indicate new conditions	DOMCSANL	2.3.6.3

DIAGRAM 2.3.6.3

Page of LY20-2226-0
Updated August 31, 1976
By TNL: LN20-3620

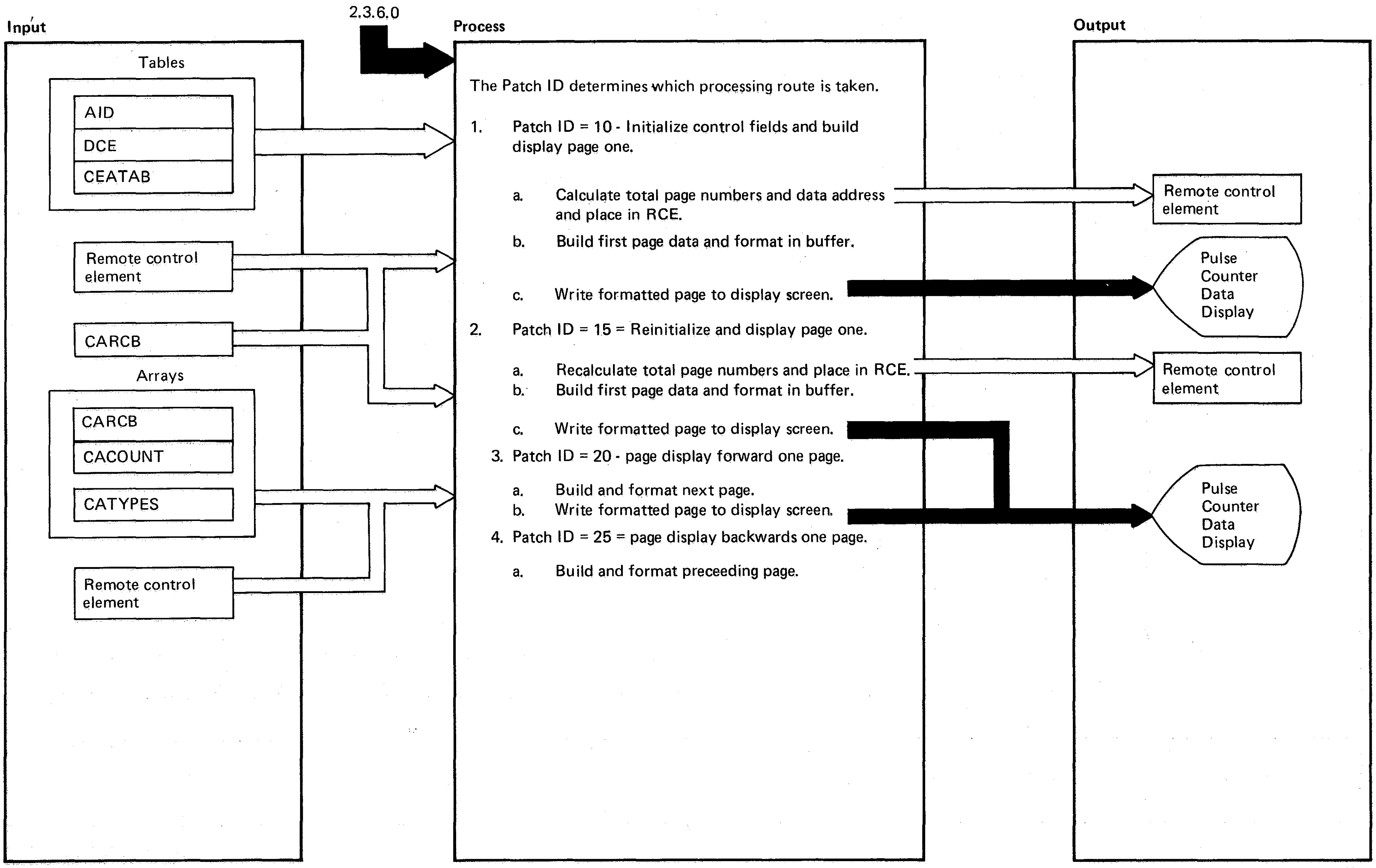


DIAGRAM 2.3.6.4: Pulse Counter Data Display Control

EXTENDED DESCRIPTION

Notes	Modules	Diagram
Note: Control is received by manual action taken by a dispatcher (i.e., depressing a program function key)		
2. This function is executed any time this function receives control (2.3.6.4) from other sensor based data display functions (2.3.6.2 and 2.3.6.5). Display pages 1-N are the pages used to display pulse counter data	DOMCSPCD	2.3.6.4
3. If display is on the last page then the first page will be displayed	DOMCSPCD	2.3.6.4

DIAGRAM 2.3.6.4

Page of LY20-2226-0
Updated August 31, 1976
By TNL: LN20-3620

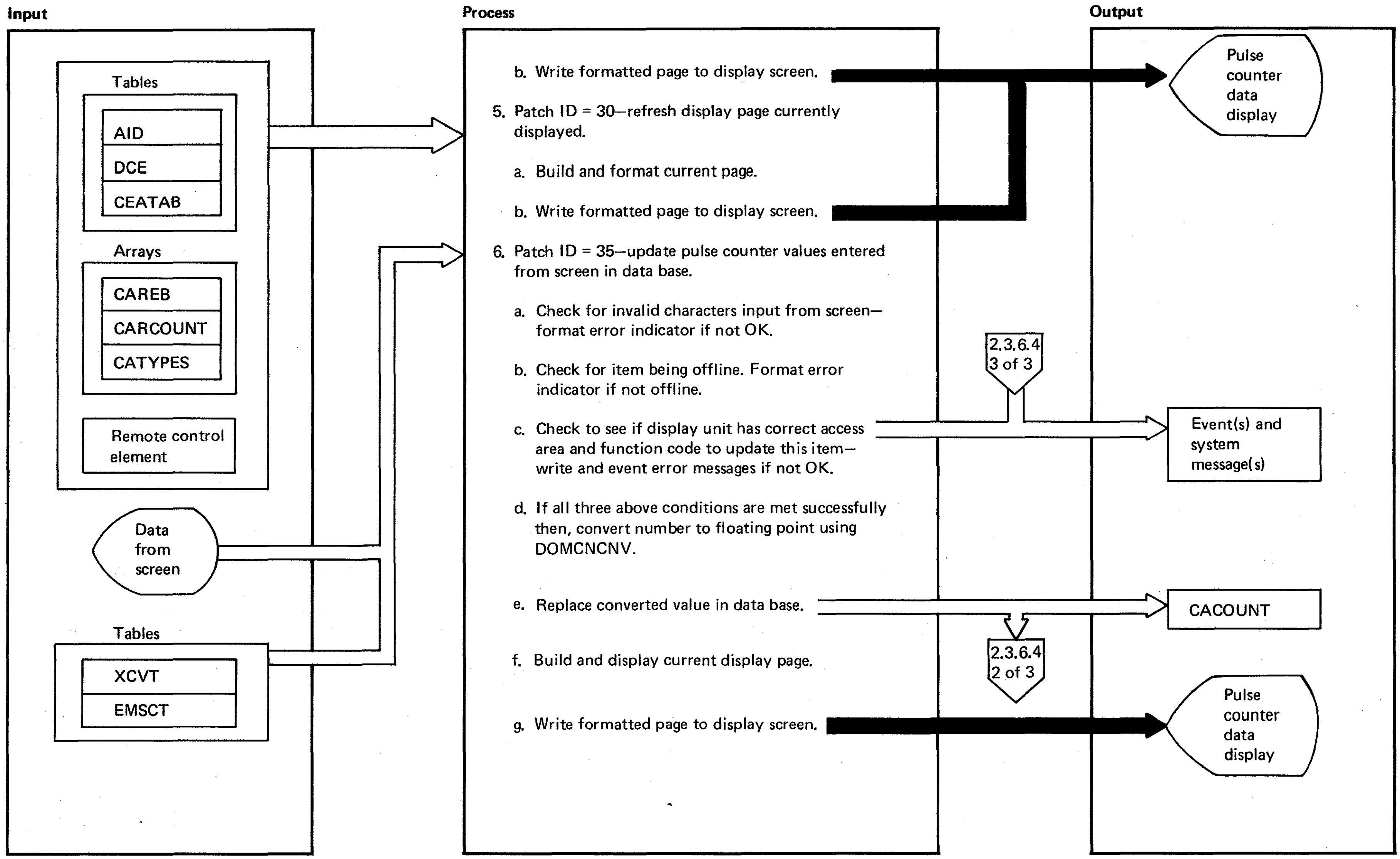


DIAGRAM 2.3.6.4

EXTENDED DESCRIPTION

Notes	Modules	Diagram
4. If display is on first page then the last page will be built and displayed	DOMCSPCD	2.3.6.4
5. The page currently being displayed is re-displayed using current data from the data base. This function is invalid for the first page of the display	DOMCSPCD	2.3.6.4
6.d. CSECT DOMCNCNV is passed the EBCDIC form of the new value and converted to single precision floating point	DOMTABLE	2.7.2.0

DIAGRAM 2.3.6.4

Input

Tables

AID

DCE

CEATAB

Arrays

CARCB

CACOUNT

CATYPES

Remote control element

Process

7. Patch ID = 40—display change.
   a. If new display is not a sensor based data display, free RCE storage and zero CEASBDT field in CEATAB.

8. Return to system.

Output

CEATAB

CEASBDT

DIAGRAM 2.3.6.4

EXTENDED DESCRIPTION

Notes	Modules	Diagram
6.e. If update is successful, the new value is placed in the data base and then an event and message are generated	DOMCSPCD	2.3.6.4
7.a. The new display name is compared with a list of all valid sensor based data display names	DOMCSPCD	2.3.6.4

DIAGRAM 2.3.6.4

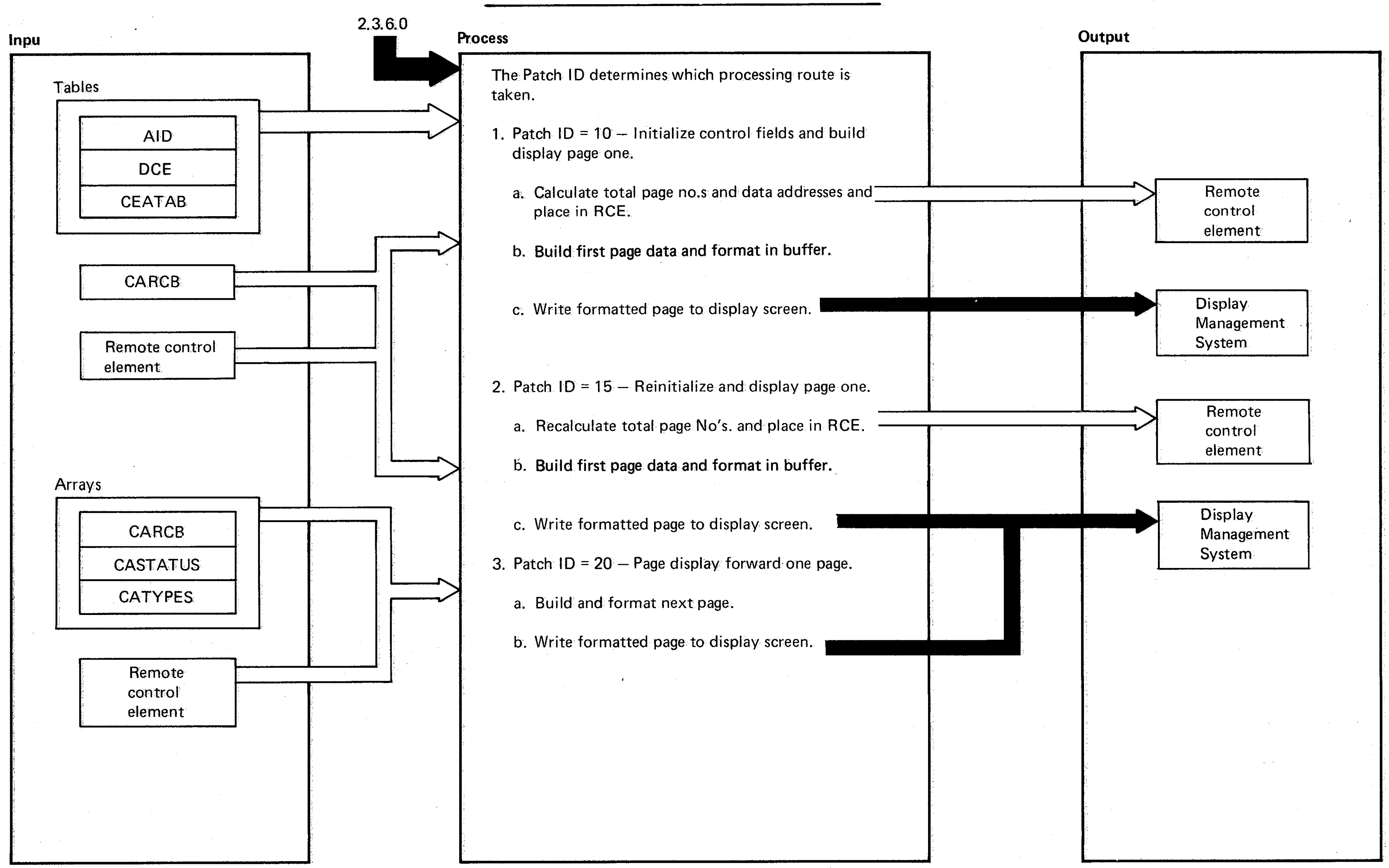


DIAGRAM 2.3.6.5: Status Data Display Control

EXTENDED DESCRIPTION

Notes	Modules	Diagram
Note: Control is received by manual action taken by an operator (i.e., depressing a program function key)		
2. This function is executed any time this function receives control (2.3.6.5) from other sensor based data display functions (2.3.6.2 and 2.3.6.4). Display pages 1-N are the pages used to display status data	DOMCSSTA	2.3.6.5
3. If display is on the last page then the first page will be displayed	DOMCSSTA	2.3.6.5

DIAGRAM 2.3.6.5

Page of LY20-2226-0
Updated August 31, 1976
By TNL: LN20-3620

Input

Tables

- AID
- DCE
- CEATAB

Arrays

- CARCB
- CACTATOS
- CATYPES

Remote Control Element

Data from screen

Tables

- XCVT
- EMSCVT

Process

4. Patch ID = 25 - page display backwards one page
   a. Build and format preceeding page
   b. Write formatted page to display section

5. Patch ID = 30 - Refresh display page currently displayed
   a. Build and format current page
   b. Write formatted page to display screen

6. Patch ID = 35—Modify Sysgen option
   a. If display unit is not master dispatcher or display unit access area and function code do not match item access area and function code then generate an event and error message
   b. If item is not offline then generate an error message
   c. If both conditions (a & b) above are met then reverse the alarmability bit setting and generate an event and message and refresh screen

AA

Output

Status data display

Event(s) and Message(s)

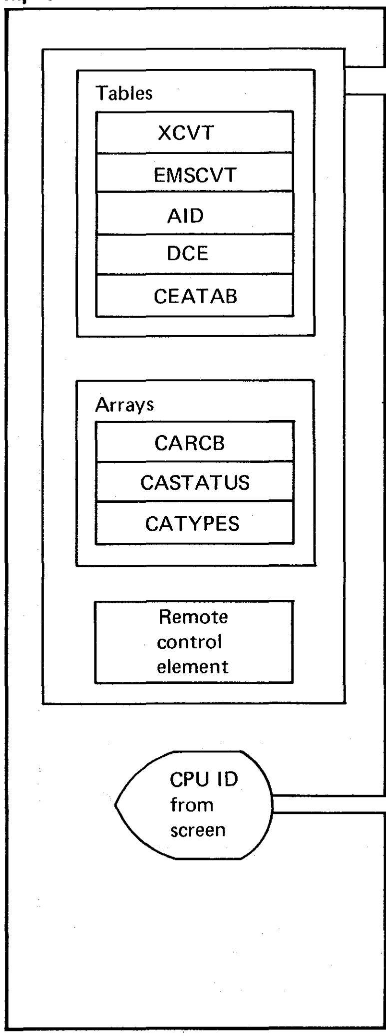
CASTATUS

DIAGRAM 2.3.6.5

EXTENDED DESCRIPTION

Notes	Modules	Diagram
4. If display is on first page then the last page will be built and displayed	DOMCSSTA	2.3.6.5
5. The page currently being displayed is re-displayed using current data from the data base. This function is invalid for the first page of the display	DOMCSSTA	2.3.6.5

Input



Process

7. If Patch ID = 36 — Generate a Transfer of Control Command.

   a. Retrieve the new CPU ID.

   b. Send a 'CTL' control command out to new CPU ID.

   c. Turn the controllability indicator off.

   d. Generate an event and message and refresh screen.

8. Patch ID = 40 — Display Change.

   a. If new display is not a Sensor Based Data display, free RCE storage and zero CEASBDT field in CEATAB.

9. Return to system.

Output

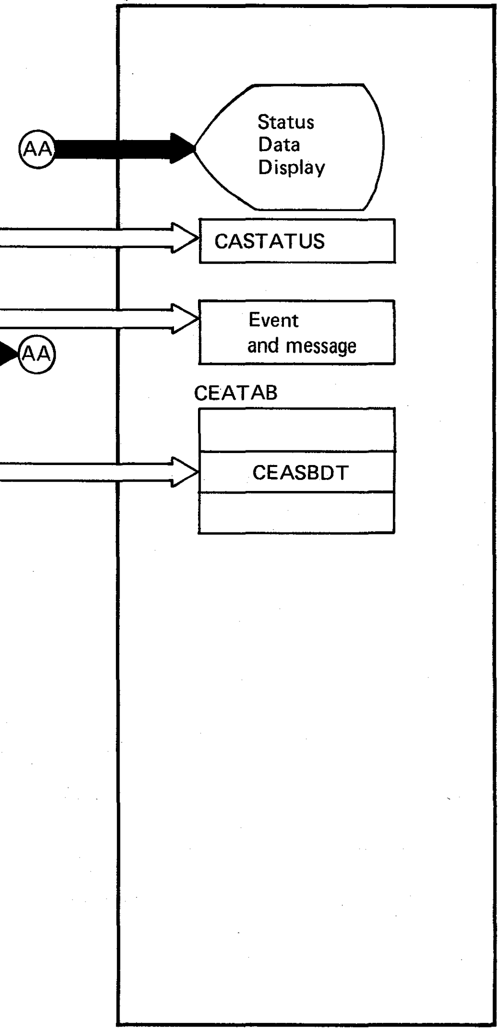


DIAGRAM 2.3.6.5

EXTENDED DESCRIPTION

Notes	Modules	Diagram
8. The new display name is compared with a list of all valid sensor based data display names	DOMCSSTA	2.3.6.5
7. This function takes place in CSECT DOMCTLCM which is branched and linked to	DOMCSSTA	2.3.6.5.1

Input

"CTL" command from System/7 or step 2 this diagram

System/7 communication array

Status array display

EMSCVT

Process

1. Process "CTL" command received from System/7.

   a. "To" CPU ID to receive control is same as this CPU—make controllable and modify PDC array entry.

   b. "To" CPU ID to receive control is not same as this CPU—make item not controllable and modify PDC table.

   c. When this CPU is at top of hierarchy route command back to destination (front end) System/7 CPU.

2. Process below if control received from status display.

   a. Generate a "CTL" command from manual input from screen.

   b. When at top of hierarchy,
      DO: AA
      ELSE:
      Route command to top of hierarchy.

3. Return.

Output

CASTATUS array

CAPDC array

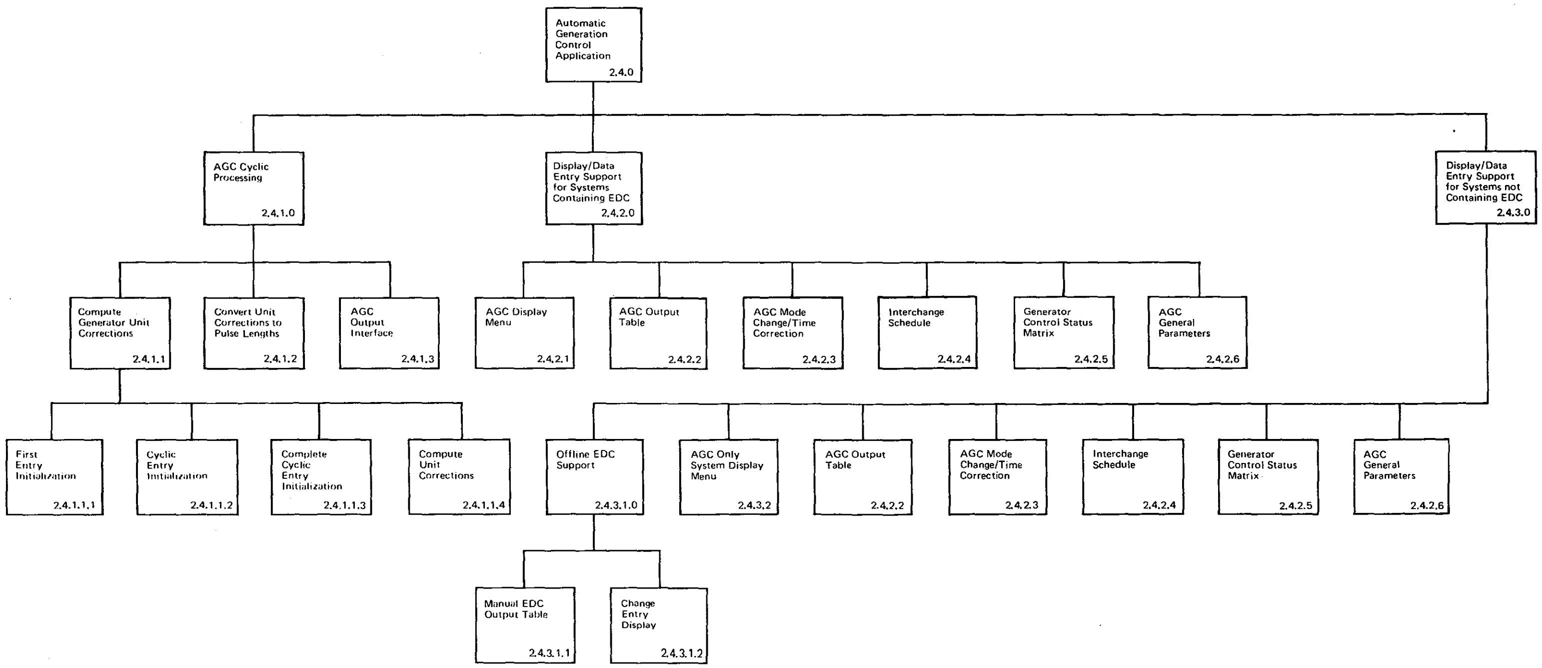
DIAGRAM 2.3.6.5.1: Transfer of Control

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. Control is received from DOMTROUT when the System/7 sends a transaction code 08 "CTL" command to the 370	DOMCSSTA	2.3.6.2
2. Control is received from the status data display control program DOMCSSTA	DOMCSSTA	2.3.6.2

DIAGRAM 2.3.6.5.1

Intentionally Blank



Automatic Generation Control Application 2.4.0
AGC Cyclic Processing 2.4.1.0
Display/Data Entry Support for Systems Containing EDC 2.4.2.0
Display/Data Entry Support for Systems not Containing EDC 2.4.3.0
Compute Generator Unit Corrections 2.4.1.1
Convert Unit Corrections to Pulse Lengths 2.4.1.2
AGC Output Interface 2.4.1.3
AGC Display Menu 2.4.2.1
AGC Output Table 2.4.2.2
AGC Mode Change/Time Correction 2.4.2.3
Interchange Schedule 2.4.2.4
Generator Control Status Matrix 2.4.2.5
AGC General Parameters 2.4.2.6
First Entry Initialization 2.4.1.1.1
Cyclic Entry Initialization 2.4.1.1.2
Complete Cyclic Entry Initialization 2.4.1.1.3
Compute Unit Corrections 2.4.1.1.4
Offline EDC Support 2.4.3.1.0
AGC Only System Display Menu 2.4.3.2
AGC Output Table 2.4.2.2
AGC Mode Change/Time Correction 2.4.2.3
Interchange Schedule 2.4.2.4
Generator Control Status Matrix 2.4.2.5
AGC General Parameters 2.4.2.6
Manual EDC Output Table 2.4.3.1.1
Change Entry Display 2.4.3.1.2

2.4.1.0

Input

The following arrays:

- AALFCODR
- AA00INSF
- AA00ICSM
- CANALOG
- AA0TGCSM
- AAACTADR
- AATTTGCS
- AA0TICSM
- AA0ICLFC
- AAPMAXEM
- AAPMINEM
- AALFCACT
- AALFCODD
- AALFCOUT
- AA0PDEDC
- AA00RHOO
- AALFCEXC
- AA0LFCRHO
- AA0PDLFC
- AAPMAXEC
- AAPMINEC
- AALFCOTT
- AA0DELTP
- AALFCUNC

Process

1. If AGC initialization not done issue system message and return.
2. If this is first entry since restart perform first entry initialization. 2.4.1.1.1
3. Begin cyclic entry initialization. 2.4.1.1.2
4. Complete cyclic entry initialization. 2.4.1.1.3
5. Compute instantaneous area requirement (IAR) and store in AALFCOTT.
6. Compute smoothed area requirement (SAR) and store in AALFCOTT.
7. Compute unit correction for each generator. 2.4.1.1.4
8. Suspend AGC if in emergency assist mode on two successive executions with sign of SAR changed between them.
9. Update AGC output table.
10. Convert unit corrections to pulses. 2.4.1.2

Return to system

Output

System msg.

The following arrays:

- AA0TICSM
- AA00ICSM
- AALFCOUT
- AATTTGCS
- AAPMINEC
- AAPMAXEC
- AAPMINEM
- AAPMAXEM
- AA00RHOO
- AA0PDEDC
- AALFCACT
- AALFCUNC
- AA0ICLFC
- AALFCOTT
- AALFCEXC
- AALFCRHO
- AA0PDLFC
- AA0DELTP
- AALFCODR

AALFCODR

AILFCSSP

DIAGRAM 2.4.1.1: Compute Generator Unit Corrections

EXTENDED DESCRIPTION

Notes	Modules	Diagram
0. DOMALFCB is patched periodically by a PTIME macro set up originally by the initialization routine DOMALFCI. The time between patches may be changed by manual intervention via the AGC General Parameters display	DOMALFCI	2.1.3
5. IAR = T (NATF-NSTF) - 10K (FREQS + FREQ0 - FREQA)	DOMALFCB	2.4.1.0
6. SAR = (1-N) SAR + N (IAR)	DOMALFCB	2.4.1.0

DIAGRAM 2.4.1.1

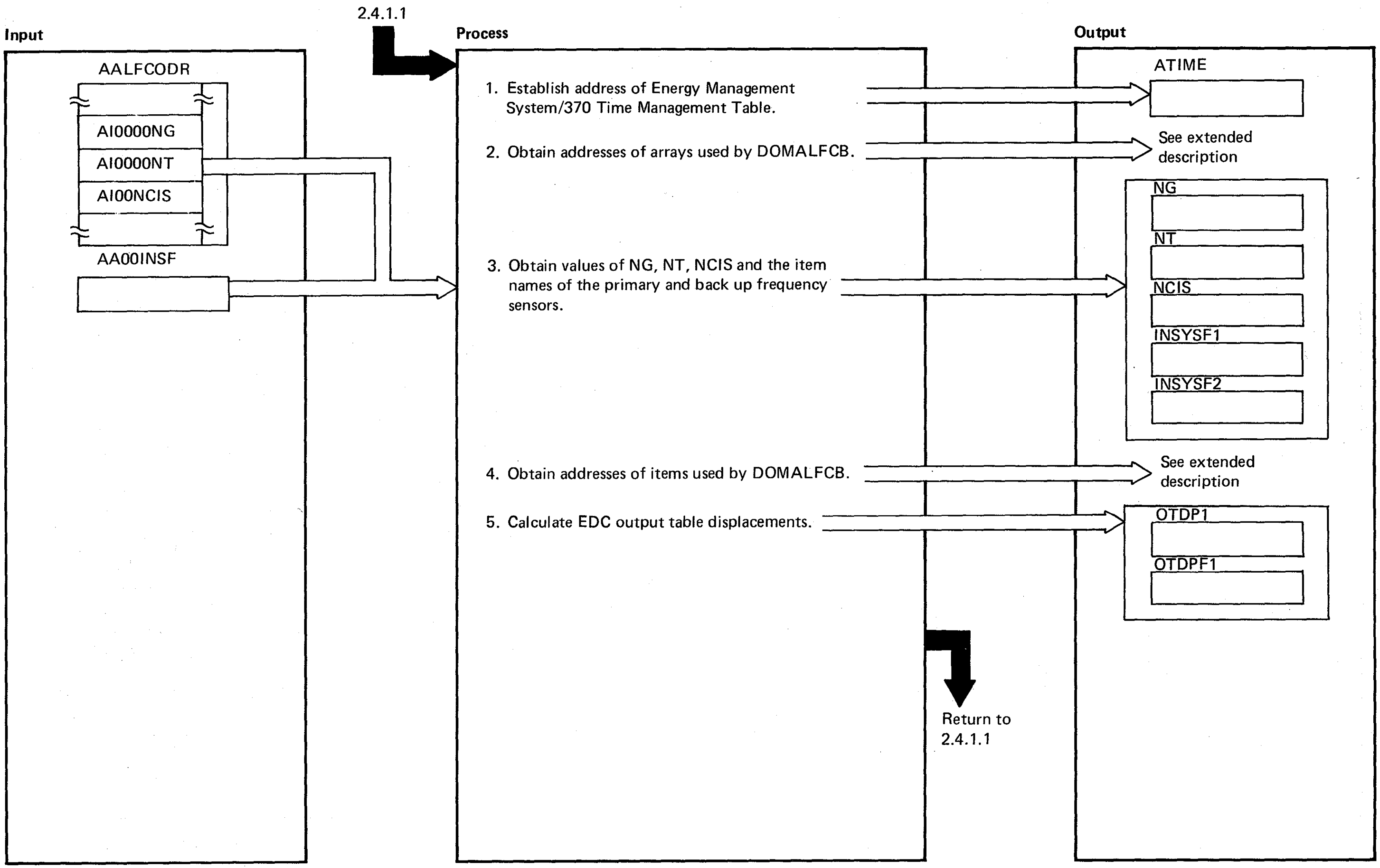


DIAGRAM 2.4.1.1.1: First Entry Initialization

EXTENDED DESCRIPTION

Notes	Modules	Diagram
2. Array names and symbolic locations of the resolved addresses are listed below:	DOMALFCI	2.1.3

Array	Address	Array	Address
AAACTADR	ACTADR	AA00RHOO	EDCRHO
AA00ICSM	ICSM	AALFCRHO	LFCRHO
AA0TICSM	TICSM	AA0PDEDC	PDEDC
AA00INTL	TIES	AA0DELTP	DELTP
AA0TGCSM	GCSM	AA0ICLFC	ICLFC
AATTTGCS	TMPGCSM	AA0PDLFC	PDLFC
AAPMAXEC	PMAXEC	AALFCACT	LFCACT
AAPMINEC	PMINEC	AALFCEXC	LFCEXC
AAPMAXEM	PMAXEM	AALFCOUT	LFCOUT
AAPMINEM	PMINEM	AALFCOTT	LFCOTT
AA0TEDCA	TEDCA	AALFCUNC	LFCUNC

4. The addresses of the primary and back up frequency readings are stored in ASYSF1 and ASYSF2. Other item names and symbolic locations of the resolved addresses are listed below:

Item	Address	Item	Address
AICMFORL	ACMFOR	LFCAL14	ADDAL14
AIPLFCGO	APLFCGO	AILFCINT	LIST
AILFCGO	ALFCGO	AILFCGO	LIST+4
AIRUNMOD	ARUNMOD	AILFCSSP	LIST+8
AILFCSSP	ALFCSSP	AIEDCINP	LIST+12
LFCAL01	ADDAL10	AILFC00K	LIST+16
LFCAL02	ADDAL02	AI0FREQO	LIST+20
LFCAL03	ADDAL03	AI0FREQS	LIST+24
LFCAL04	ADDAL04	AI0GAINA	LIST+28
LFCAL05	ADDAL05	AI0GAINB	LIST+32
LFCAL06	ADDAL06	AI0GAINC	LIST+36
LFCAL07	ADDAL07	AI0LFC00N	LIST+40
LFCAL08	ADDAL08	AILFC00T	LIST+44
LFCAL09	ADDAL09	AILFC00X	LIST+48
LFCAL10	ADDAL10	AILFC00Y	LIST+52
LFCAL11	ADDAL11	AILFCCRL	LIST+56
LFCAL12	ADDAL12	AILFCCRR	LIST+60
LFCAL13	ADDAL13	AIPLFCGO	LIST+64

DIAGRAM 2.4.1.1.1

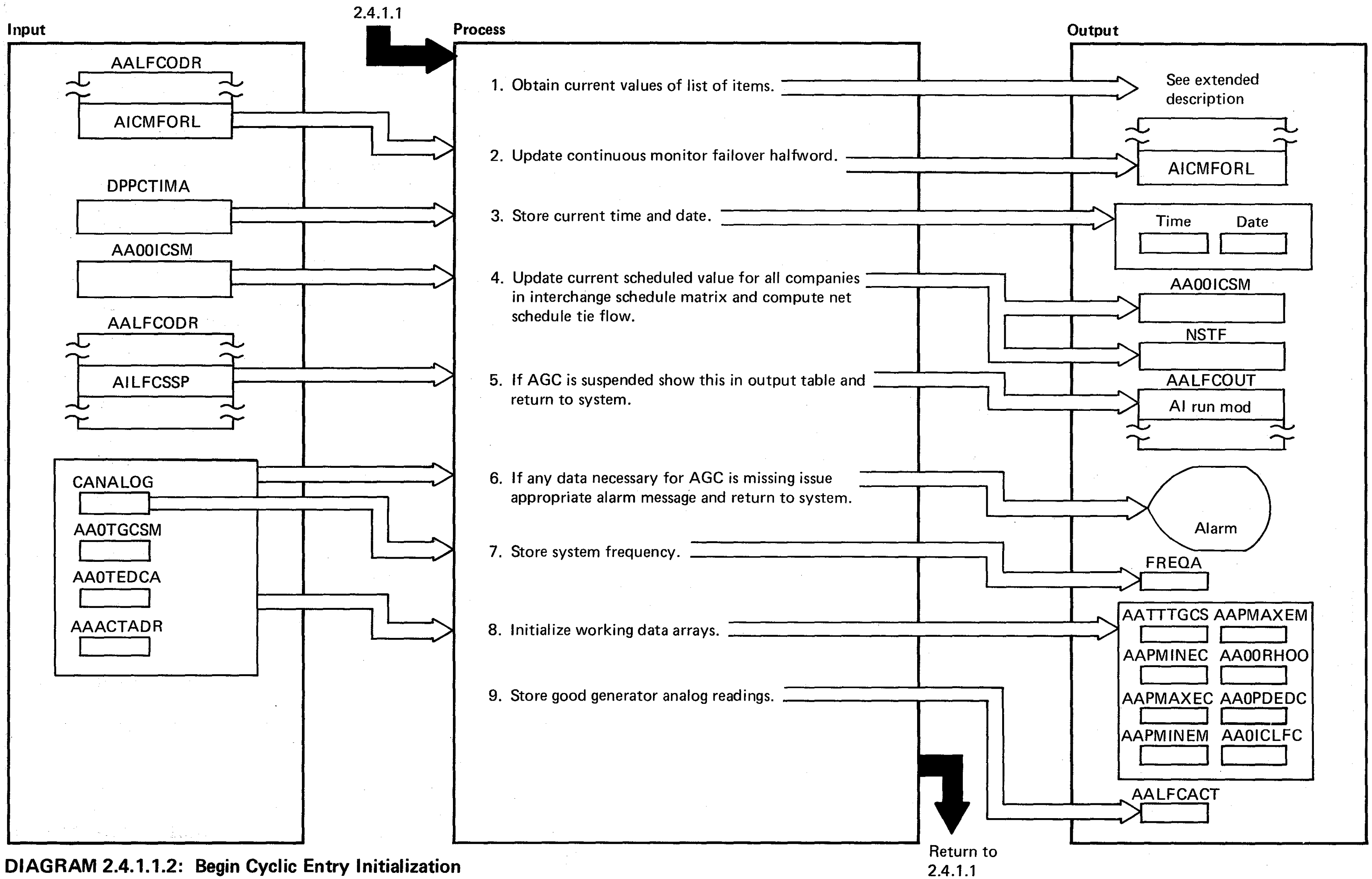


DIAGRAM 2.4.1.1.2: Begin Cyclic Entry Initialization

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. Item values are stored in internal storage as follows (values are changed in real time by indicated load modules):		

		Value Originally Set By:	May Be Changed By:
LFCINT	AGC interval	DOMALFCI	DOMAGPUP
LFCGO	Non-zero if scan complete	DOMTSSYN	DOMTSSYN
LFCSSP	Non-zero if AGC suspended	DOMALFCI	DOMAMTUP
EDCINP	Non-zero if automatic EDC output available	DOMALFCI	DOMAEDCB
K	Frequency Bias	SYSGEN	DOMAMTUP
FREQ0	Frequency offset	SYSGEN	DOMATCOR
FREQS	Scheduled frequency	SYSGEN	DOMAGPUP
GAINA	Assist mode unit correction gain control	SYSGEN	DOMAGPUP
GAINB	Normal mode and dead band mode unit correction economy gain control	SYSGEN	DOMAGPUP
GAINC	Normal mode unit correction SAR gain control	SYSGEN	DOMAGPUP
N	Proportion of instantaneous value in SAR	SYSGEN	DOMAGPUP
T	AGC Mode Adjustment	SYSGEN	DOMAMTUP
X	Dead Band Limit	SYSGEN	DOMAGPUP
Y	Normal Mode Limit	SYSGEN	DOMAGPUP
CRL	Lower Generation Capability Criterion	SYSGEN	DOMAGPUP
CRR	Raise Generation Capability Criterion	SYSGEN	DOMAGPUP
PLFCGO	Previous LFCGO value	SYSGEN	DOMALFCB

6. Causes for alarm are:
   a. No system frequency available
   b. No EDC output available
   c. Incomplete data scan (twice in a row)

8. The control element in AA0ICLFC is set to 1 for all generators on automatic control and to 0 for all others

9. If data for a generator is bad and its status code is base loaded or on automatic control the status code for this pass through DOMALFCB is changed to out of service and an alarm is issued

Input

- AATTTGCS
- AALFCACT

- AAACTADR
- AA00INTL
- CANALOG

- AA0ICLFC
- AAPMAXEM
- AAPMINEM
- AALFCACT
- AALFCODD
  - AILFCCRL
  - AILFCCRR

2.4.2.0

Process

1. Compute unit correction for all base loaded units. Sum them.
2. If on flat tie line control and data is bad from a tie that is not marked out of service self, issue an alarm message and return to the system.
3. If on tie line bias control and data from a tie is bad that is not marked out of service self, issue an alarm message, set the running mode for this execution of AGC to flat frequency control, and continue.
4. Sum net actual tie flow.
5. Check the reserve generation control capability of the AGC system.

Return to 2.4.1.1

Output

- AALFCUNC
- RAMPD
- Alarm
- Alarm
- NATF
- Alarm

DIAGRAM 2.4.1.1.3: Complete Cyclic Entry Initialization

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. Base loaded units unit correction formulation Unit correction = desired base load point - actual power Maximum allowed correction = (short term ramp rate) (AGC interval) $\vert$Unit correction$\vert$ is adjusted to be $\leqslant$ $\vert$max. allowed correction$\vert$	DOMALFCB	2.4.1.0
5. For all generators on automatic control compute the sums of the differences between the present generator outputs and the high and low emergency limits. If either sum is less than the supplied criterion issue an alarm message	DOMALFCB	2.4.1.0

DIAGRAM 2.4.1.1.3

Input

AA0ICLFC
AA0PDEDC
ALLFCACT
AA00RHOO
AALFCEXC
AALFCRHO
AA0PDLFC
AAPMAXEC
AAPMINEC
AALFCOTT
AA0DELTP
AATTTGCS
AALFCUNC
AAPMAXEM
AAPMINEN

2.4.1.1

Process

1. If|SAR| > Y then set dynamic mode to emergency assist, compute unit corrections, and return to 2.4.1.1.
2. Adjust desired power settings.
3. Compute economic correction for each generator.
4. If|SAR| ≤ X set dynamic mode to dead band, compute unit corrections, and return to 2.4.1.1.
5. Set dynamic mode to normal, compute unit corrections, and return to 2.4.1.1.

Return to 2.4.1.1

Output

AALFCEXC
AALFCRHO
AA0PDLFC
AA0DELTP
AALFCOTT
AALFCUNC

DIAGRAM 2.4.1.1.4: Compute Unit Corrections for Each Generator

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. For all generators on automatic control plus those base loaded units to be used in emergency assist mode: If SAR $<$ 0 unit correction = GAINA (max. emergency limit - actual value) If SAR $<$ 0 unit correction = GAINA (min. emergency limit - actual value)	DOMALFCB	2.4.1.0
2. For all generators on automatic control adjust desired power settings in accordance with EDC participation factors to account for the difference between total current generation and total generation at the time of the last automatic EDC. If this is not possible issue an alarm	DOMALFCB	2.4.1.0
3. Economic correction = desired power setting - actual value		
4. For each generator on automatic control: unit correction = GAINB (economic correction) If for any generator |unit correction| $>$ (AGC interval) (max. short term ramp rate) adjust its correction to the maximum allowed value	DOMALFCB	
5. USAR = SAR - RAMPD, i.e., usable area requirement = smoothed area req. - unit corrections applied to base loaded units For each generator on automatic control: - GAINC (USAR)(participation factor) + GAINB (economic correction) Adjust unit corrections so as not to violate maximum short term ramp rate or minimum allowed correction	DOMALFCB	2.4.1.0

DIAGRAM 2.4.1.1.4

Input

- AALFCODR
- AALFCOUT
- AATLFCIC
- ALLFCPDA
- AALFCPDB
- AALFCPDP

2.4.1.1

Process

1. If this is the first entry

   a. Obtain array addresses.

   b. Obtain address of lock control block for array AALFCPDO.

   c. Obtain value of number of generators, NG.

2. Compute pulse lengths in .1 seconds.

3. Increment AGC integrated command array by amount of commands issued.

4. Store pulse commands for each generator.

5. Transmit commands to generator controllers. 2.4.1.3

Return to 2.4.1.1

Output

- AALFCPDP
- AATLFCIC
- AALFCPDO

DIAGRAM 2.4.1.2: Convert Unit Corrections to Pulses

EXTENDED DESCRIPTION

Notes	Modules	Diagram
0. DOMALFCO is called by DOMALFCB when it has unit corrections to be transmitted to the generator controllers		
1.a. Array names and symbolic locations of the resolved addresses: **Array** **Address** AALFCPDO LFCPDO AALFCPDA LFCPDA AALFCPDB LFCPDB AALFCPDP LFCPDP AALFCOUT LFCOUT AATLFCIC LFCIC		
1.b. The lock is used jointly by DOMALFCO and DOMCGENO	DOMALFCB	2.4.1.0
2. Pulse length in units of .1 seconds = A(unit correction in NW) + B Truncate number of pulses to an integer between -255 and +255	DOMALFCB	2.4.1.0
3. Command in MW = (Pulse length -B) /A	DOMALFCB	2.4.1.0
5. DOMALFCO calls DOMCGENO, the supervisory control AGC output interface processor, which initiates the transmission of the specified raise or lower pulses to the generator controllers		

2.4.1.0

Input

- SCWKAREA
- Array AALFCPDO
- DOMCGENT

Process

1. Obtain count of entries to process.

2. Lock array AALFCPDO
   DO

   A. Find match on generator name
      DO

      1. Pulse duration is zero
         do nothing

      2. Pulse duration is positive
         raise using SCDEVICE macro

      3. Pulse duration is negative
         lower using SCDEVICE macro

   B. Point to next entry

3. Unlock AALFCPDO array.

Output

DIAGRAM 2.4.1.3: (DOMCGENO)

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. The address and item count are resolved at initialization time and stored		
2.a. If all pulse duration values are zero, no data is sent to the System/7		
2.a.2 and 3 The SCDEVICE macro branches to the SCDEVICE macro processor which patches the PDC macro interface program	DOMTABLE DOMCDC04	2.7.2.0 2.3.2.2

DIAGRAM 2.4.1.3

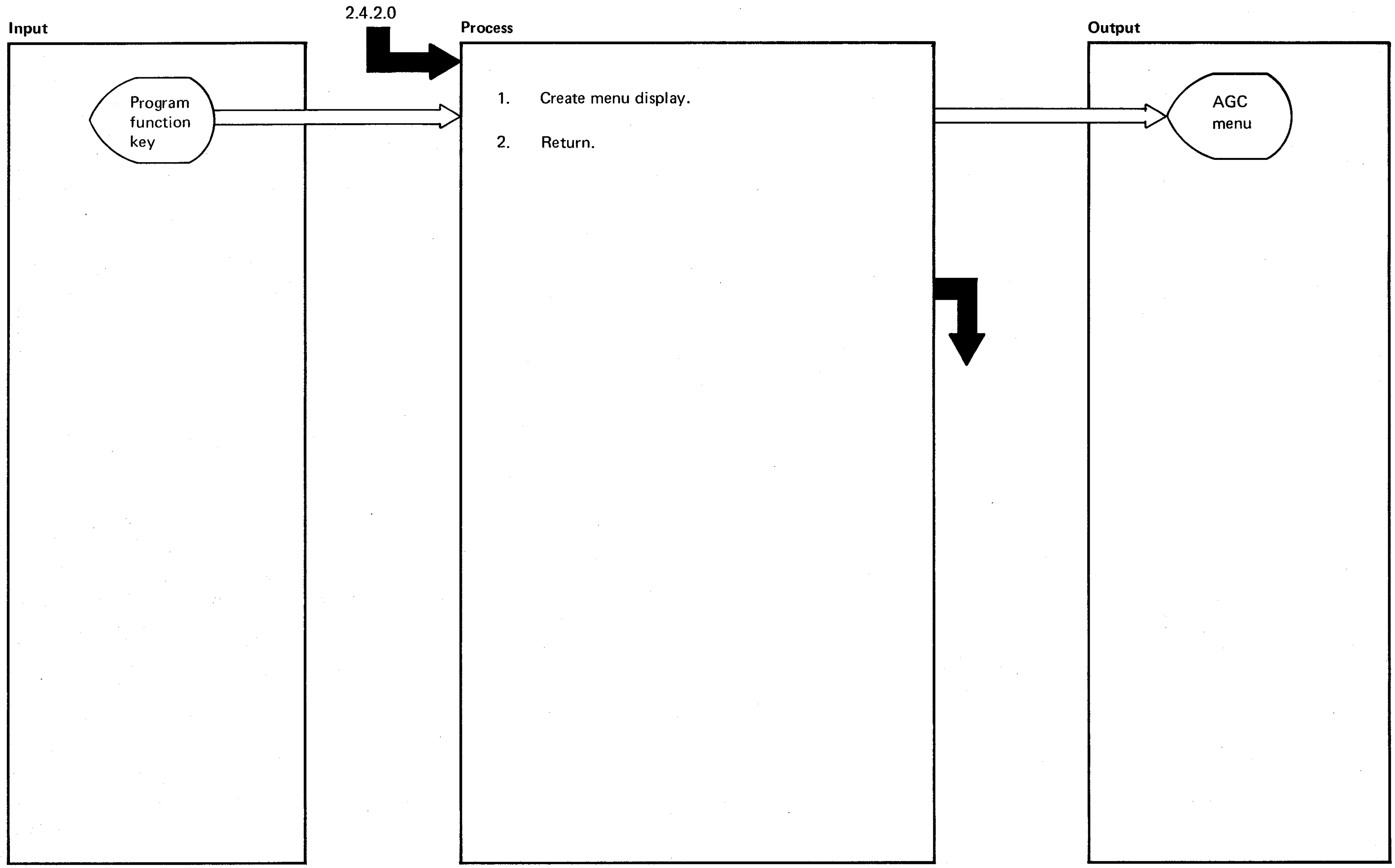


DIAGRAM 2.4.2.1: AGC Display Menu

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. Display Management System PATCHes DOMALMUP with a PACH ID of 1 when the system function key for the AGC menu or EDC menu is pressed. Display Management System provides a parameter list which contains the display name, requesting unit ID, and the format control list ID.	DOMALMUP	2.4.2.1

DIAGRAM 2.4.2.1

Input

Data base

- AALFCOUT
- AAPLANTS
- AI0000NP
- AI00GENC

CEATAB

Display Management System

2.4.2.0

Process

1. Generate first page AGC output.
   Patch ID = 1
   a. Determine data base addresses if first execution.
   b. Determine no. pages required for output.
   c. Save page no.
   d. Create AGC output table display first page.
2. Remove display.
   Patch ID = 2
   a. Delete paging pointers for requesting unit.
3. Page forward.
   Patch ID = 3
   a. Increment page number in CEATAB for requesting unit.
   b. Determine data dddresses for next page of output.
   c. Create AGC output table display (next page).
4. Page backward.
   Patch ID = 4
   a. Decrement page number in CEATAB for requesting unit.
   b. Determine data addresses for previous page.
   c. Create AGC output table display (previous page).
5. Return.

Output

CEATAB

AGC Output Table Display

CEATAB

DIAGRAM 2.4.2.2: AGC Output Table

EXTENDED DESCRIPTION

Notes		Modules	Diagram
1.a., 3.c., & 4.c.	The AGC Output Table display is created by providing Display Management System with the addresses of the data to be output, the number of items to output, and the unit ID to be output to, using DINFO macros, and then using DISPUP macros	DOMALTPG	2.4.2.2

DIAGRAM 2.4.2.2

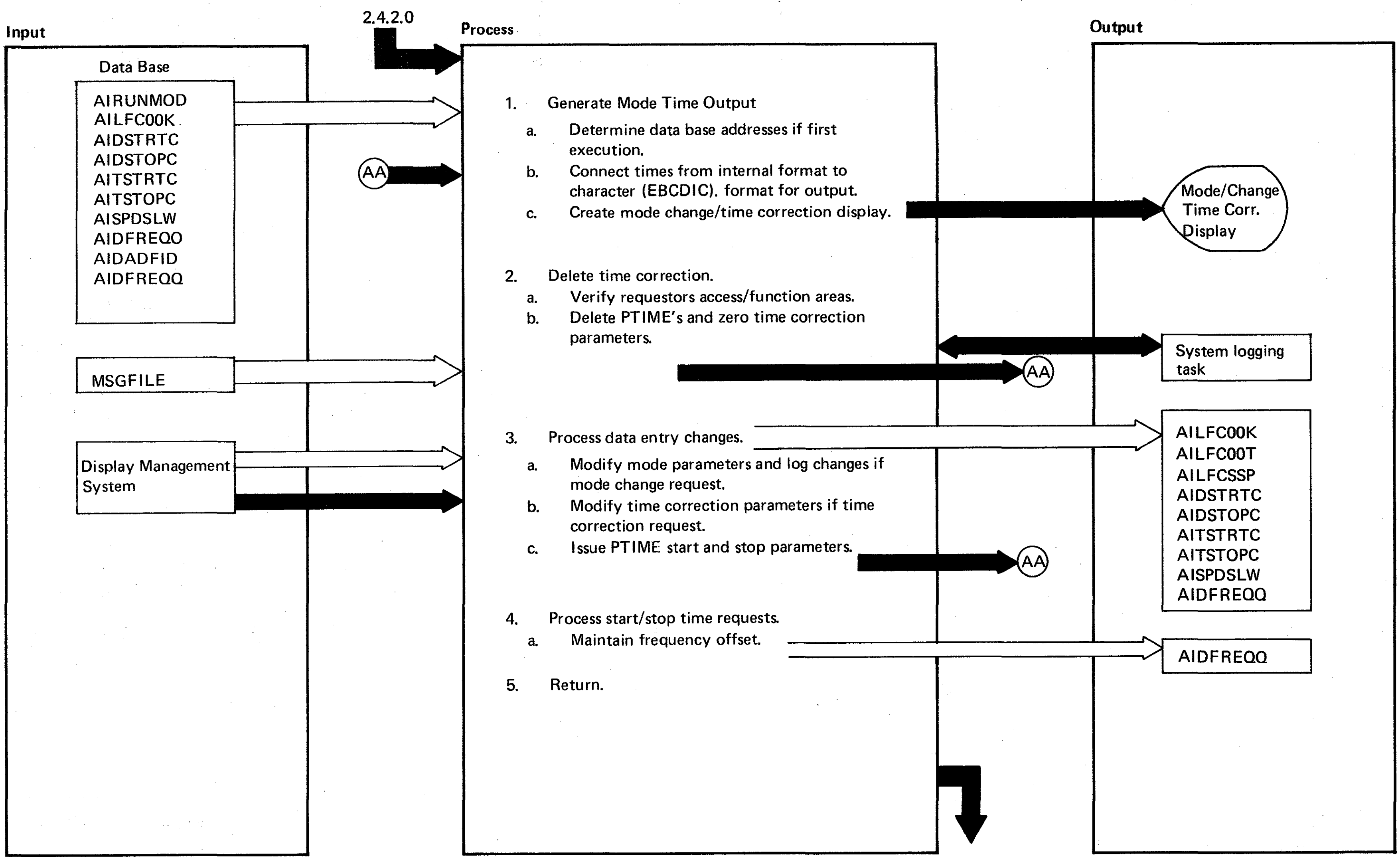


DIAGRAM 2.4.2.3: AGC Mode Change/Time Correction

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. Patch ID = 1	DOMAMTUP	2.4.2.3
1.b. Times are in fixed point 1/100 sec. internally		
1.c. Display Management System is provided with addresses VIA the DINFO macro and DISPUP macros are used to create the display		
2. Patch ID = 5	DOMAMTUP	2.4.2.3
2.a. Valid requestors are determined by comparing the requesting Access/function area IDs to SYSGENed IDs in the data base		
3. Patch ID = 7	DOMAMTUP	2.4.2.3
3.a. Valid mode change requests are as follows: • Suspended mode – may be selected anytime • Flat Tie Line – may be selected as long as at least one generator is on auto control • Tie line bias – same as flat tie line plus frequency bias must be non-zero (currently or as entered) • Flat frequency – same as tie line bias Mode Parameters are set as follows: Suspended mode – AILFCSSP(1) Flat Tie Line – AILFCSSP(0) AILFC00K(0) AILFC00T(1) Tie Line Bias – AILFCSSP(0) AILFC00T(1) AILFC00K (centered value) Flat Frequency – AILFCSSP(0) AILFC00T(0) AILFC00K(centered value)		
3.b. Valid times are current time $<$ start time $<$ stop time $\leqslant$ 24 hours. Valid dates are today and tomorrow The following time correction parameters are stored in the data base: AIDSTRTC – Date to start time correction AITSTRTC – Time to start time correction AISPDSLW – Speed up or slow down time AIDSTOPC – Date to stop time correction AITSTOPC – Time to stop time correction AIDFREQO – Frequency offset for time correction		

DIAGRAM 2.4.2.3

EXTENDED DESCRIPTION

Notes	Modules	Diagram
3.c. PTIME patches are issued to DOMATCOR to occur at the start and stop time for correction		
4. Save frequency offsets passed by patch	DOMATCOR	2.4.2.3

Intentionally Blank

2.4.2.0.

Input

Data Base

- AA00ICSM
- AI00NCIS
- AIRUNMOD
- AI00YESC
- AI000NOC
- AI00UNKC
- AIDADFID

CEATAB

MSGFILE

Data Base

- AIRONMOD
- AIDADFID

Process

1. Generate first page of display patch ID = 100
   a. Build paging pointers block for requesting unit.
2. Remove display - patch ID = 49.
   a. Delete paging pointers block for requesting unit.
3. Page forward to next page
   Patch ID = 51
   a. Update paging pointers block.
4. Page backward to previous page - patch ID = 50.
   a. Update paging pointers block.
5. Direct page selection - patch ID = 5.
   a. Update paging pointer block.
6. Process data entry changes - patch ID = 52.
   a. Set paging pointers to cause refresh of current page.
7. Delete Interchange Schedule
   Patch ID = 98 (upper schedule)
   Patch ID - 99 (lower schedule)
8. Generate display page
9. Return.

Output

CEATAB

Data Base

AA00ICSM

Interchange Schedule Display

DIAGRAM 2.4.2.4: Interchange Schedule Matrix

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1.a. The CEATAB contains a pointer to control blocks which maintain the current page number of the display being viewed (interchange by company matrix)	DOMATLPG	2.4.2.4
6. Valid data that may be entered is as follows: a. $-9999.999 \leqslant$ current schedule value $\leqslant +9999.999$ if AGC current running mode is suspended. If not in suspended mode, no change is allowed to current schedule value b. $-9999.999 \leqslant$ desired schedule value $\leqslant +9999.999$ (either schedule) c. Dates must be today's or tomorrow's date. Times must be $\leqslant$ 24 hours. Start time must be $<$ stop time (either schedule) d. (Time to stop change - time to start change) + maximum rate of change (MRC) $\leqslant$ |desired schedule value - current schedule value| either schedule e. $0 \leqslant$ MRC $\leqslant +999.999$ (either schedule) Values entered in character that need conversion are: a. Dates are converted to Julian b. Times are converted to fixed point 1/100 sec. All entered changes accepted are stored in AA00ICSM. MRC values are scaled from MW/MIN to MW/SEC	DOMATLPG	2.4.2.4
7. A check is made for valid access/function area before zeroing either schedule	DOMATLPG	
8. Company names and associated poke points are output via DINFO and DISPUP macros based on the number of companies. Display Management System is provided with the addresses of the output data via DAUPDAT and DINFO macros and DISPUP macros are used to output all the data associated with a particular Tie Line	DOMATLPG	2.4.2.4

DIAGRAM 2.4.2.4

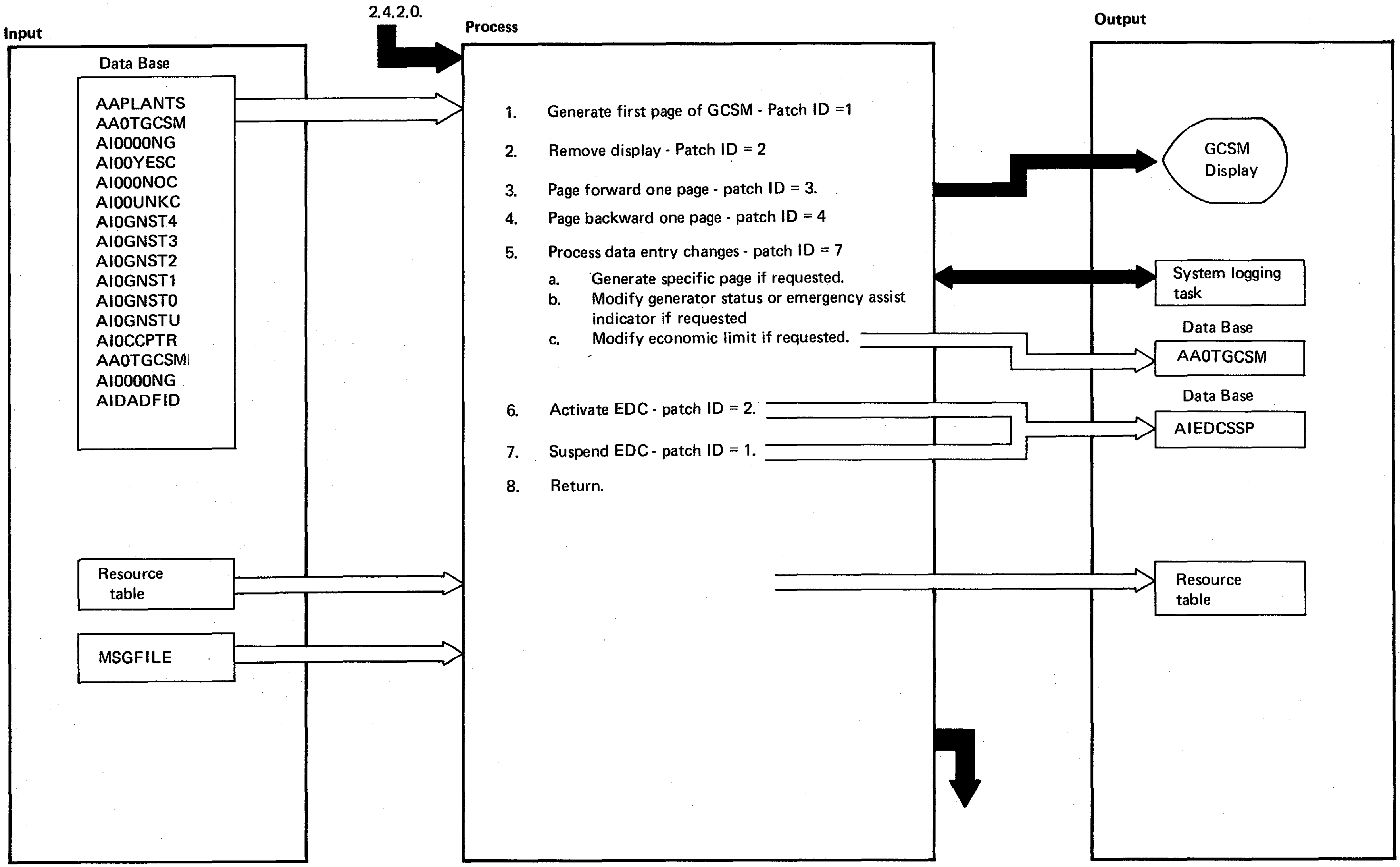


DIAGRAM 2.4.2.5: Generator Control Status Matrix

EXTENDED DESCRIPTION

Notes	Modules	Diagram
5. • Display Management System patches DOMAGSPG with a patch ID of 7 passing the address of the data entry change list in the patch parameter list • Valid data that can be entered is as follows: a. Generator Status 1. Out of service 2. Off control 3. Base loaded 4. ECON variable 5. Auto control b. Emergency Assist Indicator yes or no c. $0 \leqslant$ Ramp Rate $\leqslant 999.999$ d. $0 <$ base load point $\leqslant 999.999$ e. Economic Power Limits (High/Low) and Economic Incremental Cost Limits (High/Low) must be on current cost curve f. Emergency Limits – $0 <$ low $<$ high $\leqslant 999.999$ g. Ramp Rates – $0 <$ short $<$ long $\leqslant 999.999$ h. $0 <$ Minimum Usable Rate $\leqslant 999.999$ i. $0 <$ fuel cost $\leqslant 999.999$ • Acceptable entered values are stored in AA0TGCSM in the data base	DOMAGSPG	2.4.2.5
6-7. Valid requestors are determined by comparing the requesting access/function IDs to SYSGENEd IDs in the data base	DOMAESUS	2.4.2.5

2.4.2.1

Input

Data base

AI0FREQS
AI0GAINA
AI0GAINB
AI0GAINC
AILFC00N
AILFC00X
AILFC00Y
AILFCCRL
AILFCCRR
AILFCINT

Display Management System

Process

1. Generate general parameter output display. Patch ID = 1.
2. Process data entry changes. Patch ID = 7.
   a. When AGC interval change request—convert interval to internal format.
   b. Modify PTIME patch interval for AGC to execute at requested interval.
   c. Store all changes in data base and log array AALFCODD.
3. Return.

Output

AGC General Parameters Display

Data base

AI0FREQS
AI0GAINA
AI0GAINB
AI0GAINC
AILFC00N
AILFC00X
AILFC00Y
AILFCCRL
AILFCCRR
AILFCINT

DIAGRAM 2.4.2.6: AGC General Parameters

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. The AGC interval is in fixed point 1/100 seconds internally	DOMAGPUP	2.4.2.6
2. Valid data entered is as follows: a. 45.0 ≤ schedule frequency ≤ 75.0 b. 0 ≤ assist mode unit correction gain ≤ 1 c. 0 ≤ normal economic gain ≤ 1 d. 0 ≤ normal smoothed area requirement (SAR) gain ≤ 1 e. 0 ≤ percent instantaneous area requirement in SAR ≤ 1 f. 0 ≤ dead band mode limit ≤ 999.999 g. 0 ≤ normal mode limit ≤ 999.999 h. 0 ≤ system response range/lower generation capability criterion ≤ 999.999 i. 0 ≤ system response range/raise generation capability criterion ≤ 999.999 j. 0 ≤ AGC interval ≤ 6000	DOMAGPUP	2.4.2.6
2.c. AGC interval is stored in AILFCINT. Data stored as follows: AI0FREQS – Schedule Frequency AI0GAINA – Assist Mode U.C. Gain AI0GAINB – Normal Economic Gain AI0GAINC – Normal SAR Gain AILFC00N – Percent IAR in SAR AILFC00X – Dead Band Mode Limit AILFC00Y – Normal Mode Limit AILFCCRL – SRR/LGCC AILFCRRL – SRR/RGCC		

DIAGRAM 2.4.2.6

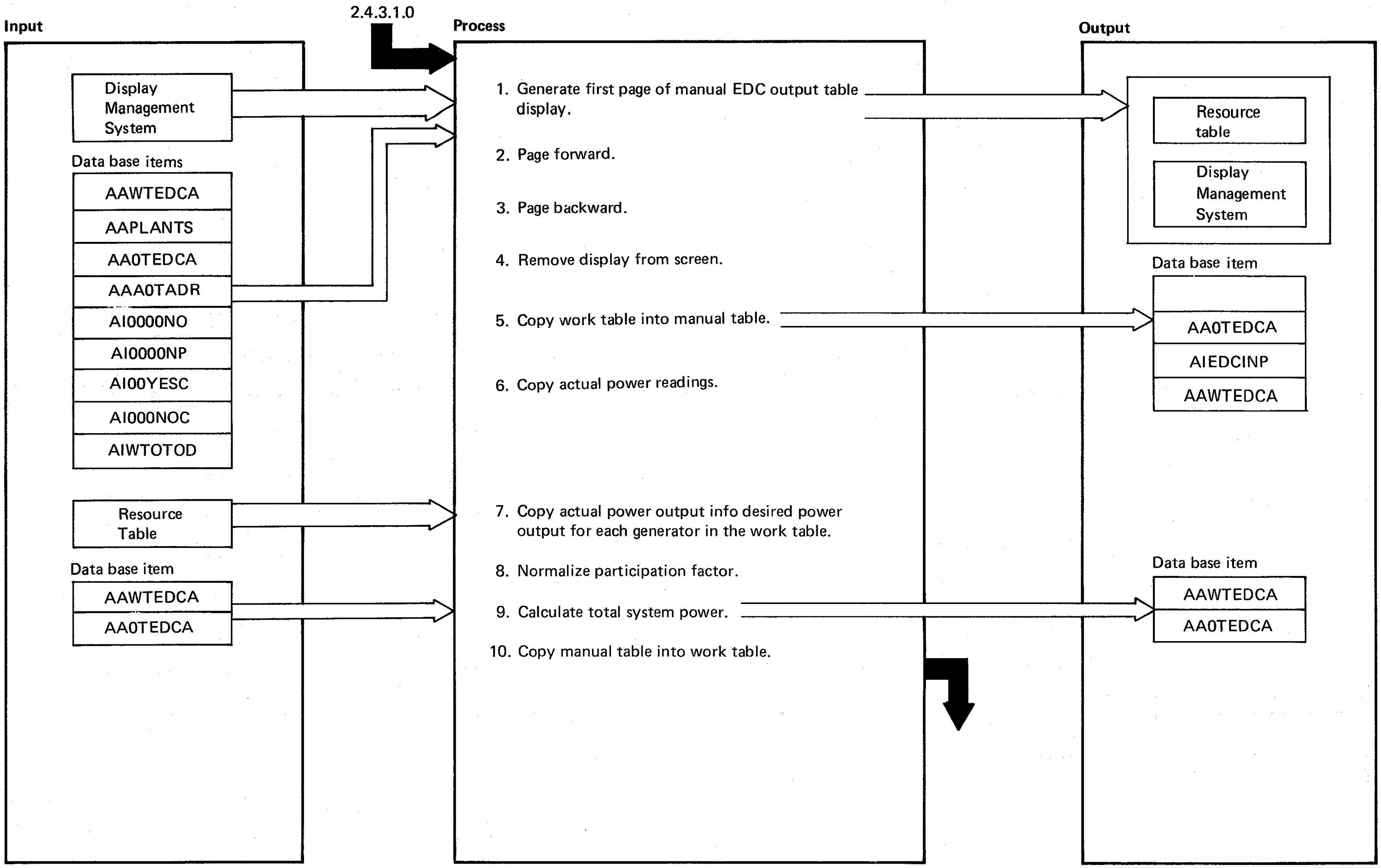


DIAGRAM 2.4.3.1.1: Manual FDC Output Table

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. Patch ID of 1 indicates this function is to be performed	DOMAMTPG	2.4.3.1.1.1
2. Patch ID os 3 indicates this function is to be performed	DOMAMTPG	2.4.3.1.1.2
3. Patch ID of 4 indicates this function is to be performed	DOMAMTPG	2.4.3.1.1.3
4. Patch ID of 2 indicates this function is to be performed	DOMAMTPG	2.4.3.1.1.4
5. Patch ID of 5 indicates this function is to be performed		
6. Patch ID of 6 indicates this function is to be performed		
7. Patch ID of 7 indicates this function is to be performed	DOMAMTPG	2.4.3.1.1.7
8. Patch ID of 8 indicates this function is to be performed	DOMAMTPG	2.4.3.1.1.8
9. Patch ID of 9 indicates this function is to be performed	DOMAMTPG	2.4.3.1.1.9
10. Patch ID of 10 indicates this function is to be performed	DOMAMTPG	2.4.3.1.1.10

DIAGRAM 2.4.3.1.1

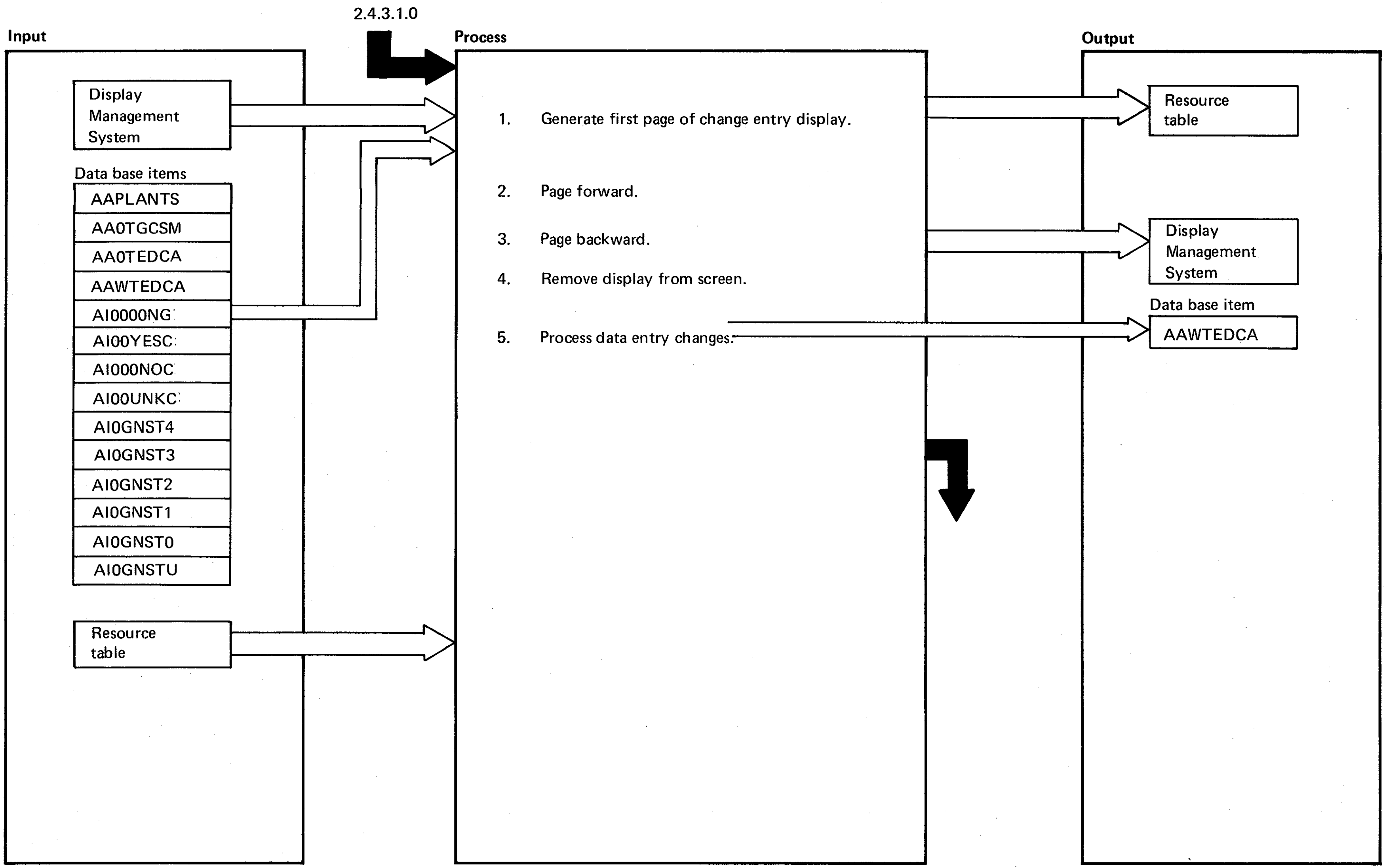


DIAGRAM 2.4.3.1.2: Change Entry Display

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. Patch ID of 1 indicates this function is to be performed	DOMACEPG	2.4.3.1.2.1
2. Patch ID of 3 indicates this function is to be performed	DOMACEPG	2.4.3.1.2.2
3. Patch ID of 4 indicates this function is to be performed	DOMACEPG	2.4.3.1.2.3
4. Patch ID of 2 indicates this function is to be performed	DOMACEPG	2.4.3.1.2.4
5. Patch ID of 7 indicates this function is to be performed	DOMACEPG	2.4.3.1.2.5

DIAGRAM 2.4.3.1.2

2.4.3.0

Input

Process

1. Create menu display for AGC only display.

2. Return.

Output

AGC Menu Display

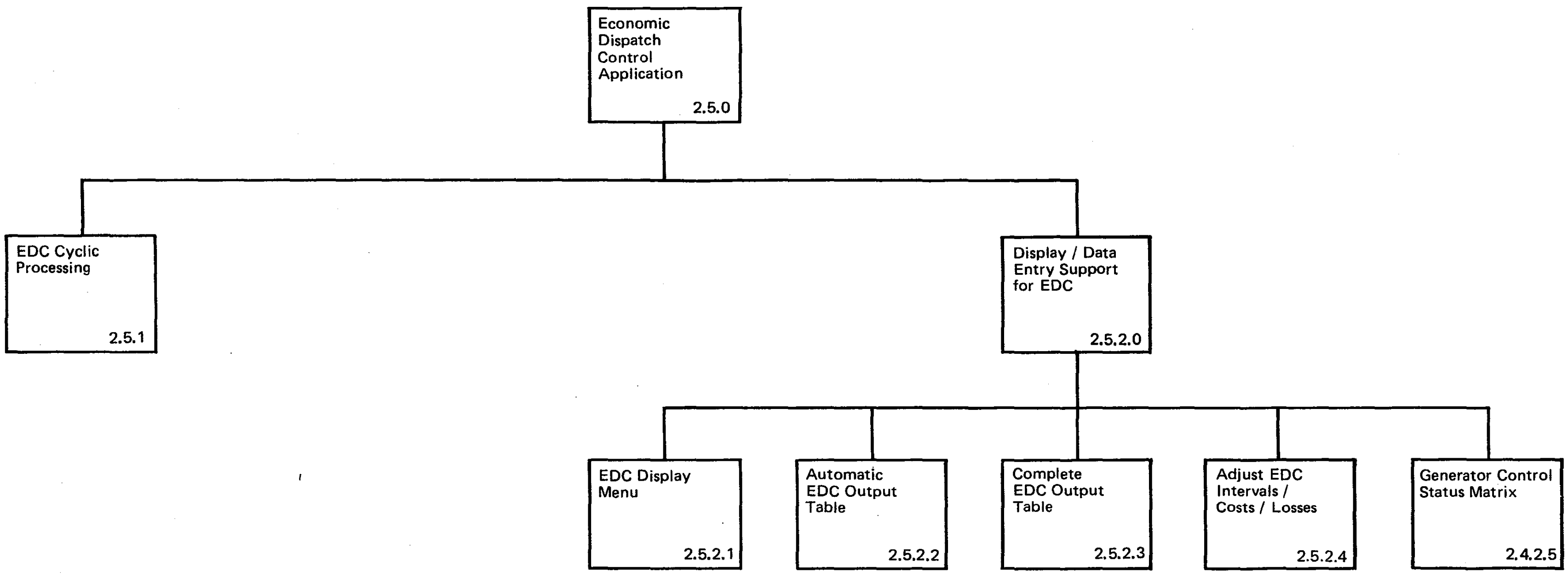
DIAGRAM 2.4.3.2: AGC Only System Menu Display

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. Display Management System patches DOMALMUP with a patch ID of one when the system function key for the AGC menu is depressed. Display Management provides a parameter list which contains the display name, requesting display unit ID, and the FCLID	DOMALMUP	2.4.3.2

DIAGRAM 2.4.3.2

Intentionally Blank


Economic Dispatch Control Application 2.5.0
EDC Cyclic Processing 2.5.1
Display / Data Entry Support for EDC 2.5.2.0
EDC Display Menu 2.5.2.1
Automatic EDC Output Table 2.5.2.2
Complete EDC Output Table 2.5.2.3
Adjust EDC Intervals / Costs / Losses 2.5.2.4
Generator Control Status Matrix 2.4.2.5

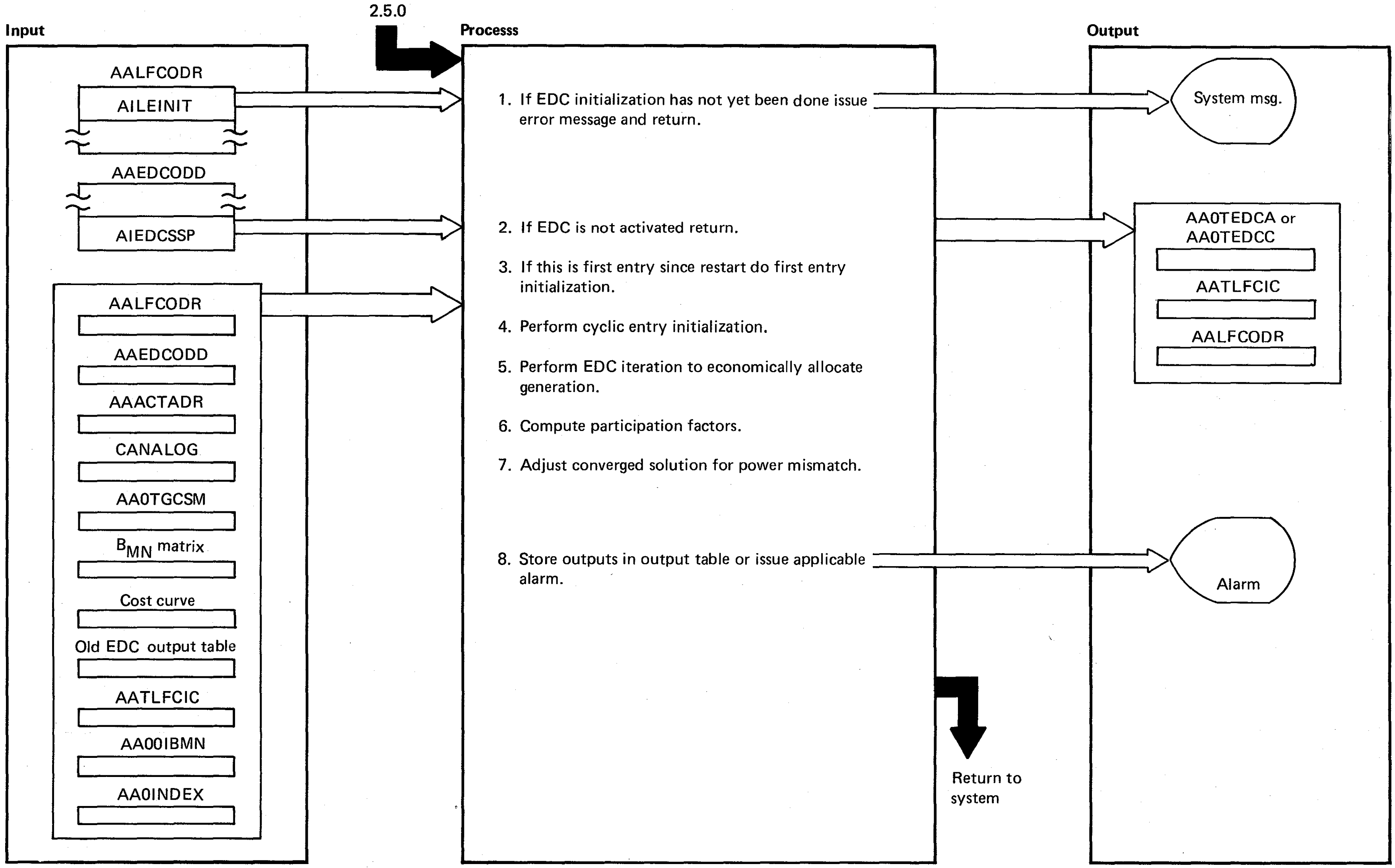


DIAGRAM 2.5.1: EDC Cyclic Processing

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. DOMAEDCB is patched with a patch ID of 1 when an automatic dispatch is to be performed. This means that only those generators whose status in AA0TGCSM is "on automatic control" will be dispatched. DOMAEDCB is patched with a patch ID of 2 when a complete dispatch is to be performed. In this case generators "on automatic control" or "economically variable" are both included		
Cyclic patches for both automatic and complete dispatches are initiated in DOMAEDCI	DOMAEDCI	2.1.4
3. EDC control segment	DOMAEDCB	2.5.1
General initialization	DOMAEDCB	2.5.1
4. EDC control segment	DOMAEDCB	2.5.1
Automatic entry initialization	DOMAEDCB	2.5.1
Generator compliance checks	DOMAEDCB	2.5.1
Complete entry initialization	DOMAEDCB	2.5.1
5. EDC control segment	DOMAEDCB	2.5.1
Inverse penalty factors computation	DOMAEDCB	2.5.1
Incremental cost, slope of cost curve, cost difference computation	DOMAEDCB	2.5.1
Demand difference computation	DOMAEDCB	2.5.1
Generator exclusion adjustment	DOMAEDCB	2.5.1
$\Delta\lambda$ Computation	DOMAEDCB	2.5.1
Desired power readings computation	DOMAEDCB	2.5.1
6. Participation factors computation	DOMAEDCB	2.5.1
7. Mismatch adjustment	DOMAEDCB	2.5.1
8. Post processor	DOMAEDCB	2.5.1
Output routine	DOMAEDCB	2.5.1
The following arrays are used for working storage:		
AAJTGCSM AA0CURVE AA00PACT AA000BMN AA00PMIN		
AA00PMAX AA00PDES AA00ICEDC AA00000T AA00000H		
AA0GAMMA AA00000D AA00000A AA00000M AA000EXC		
AA00PTHI AA00PTLO AA000DDP AA000RHO		

DIAGRAM 2.5.1

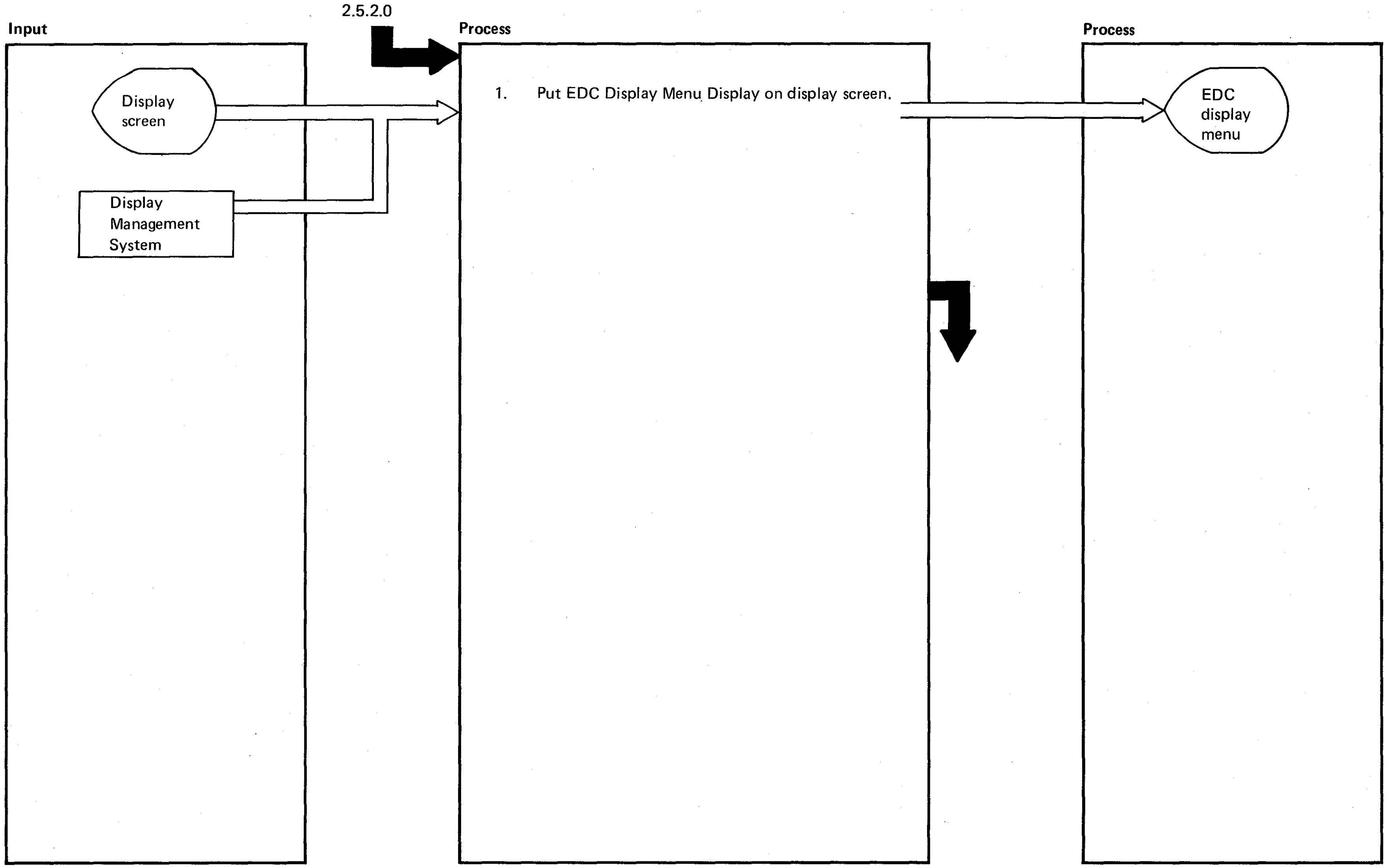


DIAGRAM 2.5.2.1: Displays EDC Display Menu

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. EDC Menu Display is shown on the display screen based on input parameters	DOMALMUP	

DIAGRAM 2.5.2.1

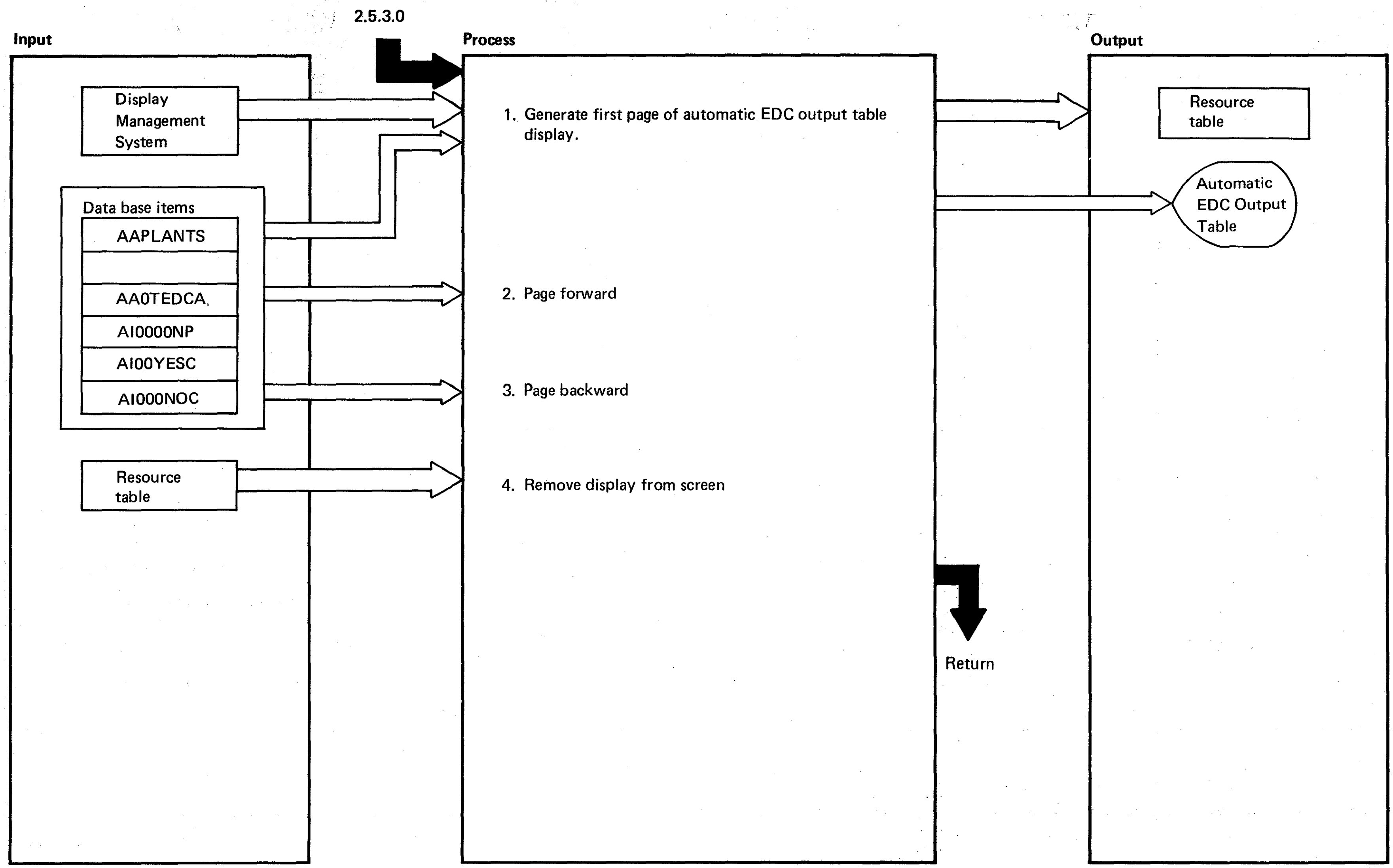


DIAGRAM 2.5.3.2: Automatic EDC Output Table

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. A patch ID of 1 indicates this function is to be performed	DOMAATPG	2.5.3.2.1
2. A patch ID of 3 indicates this function is to be performed	DOMAATPG	2.5.3.2.2
3. A patch ID of 4 indicates this function is to be performed	DOMAATPG	2.5.3.2.3
4. A patch ID of 2 indicates this function is to be performed	DOMAATPG	2.5.3.2.4

DIAGRAM 2.5.3.2

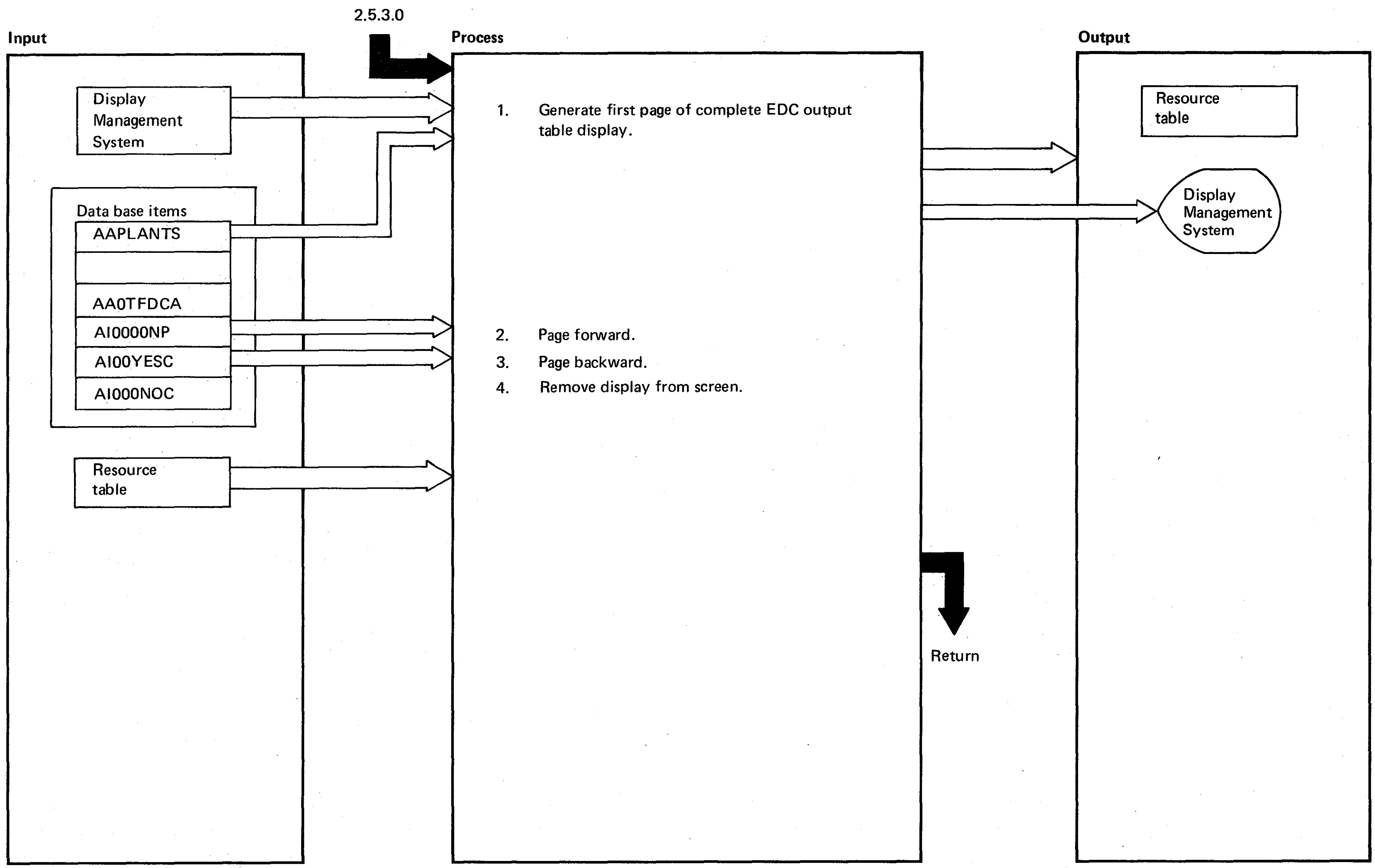


DIAGRAM 2.5.3.3: Complete EDC Output Table

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. A patch ID of 1 indicates this function is to be performed	DOMAATPG	2.5.3.3.1
2. A patch ID of 3 indicates this function is to be performed	DOMAATPG	2.5.3.3.3
3. A patch ID of 4 indicates this function is to be performed	DOMAATPG	2.5.3.3.4
4. A patch ID of 2 indicates this function is to be performed	DOMAATPG	2.5.3.3.2

DIAGRAM 2.5.3.3

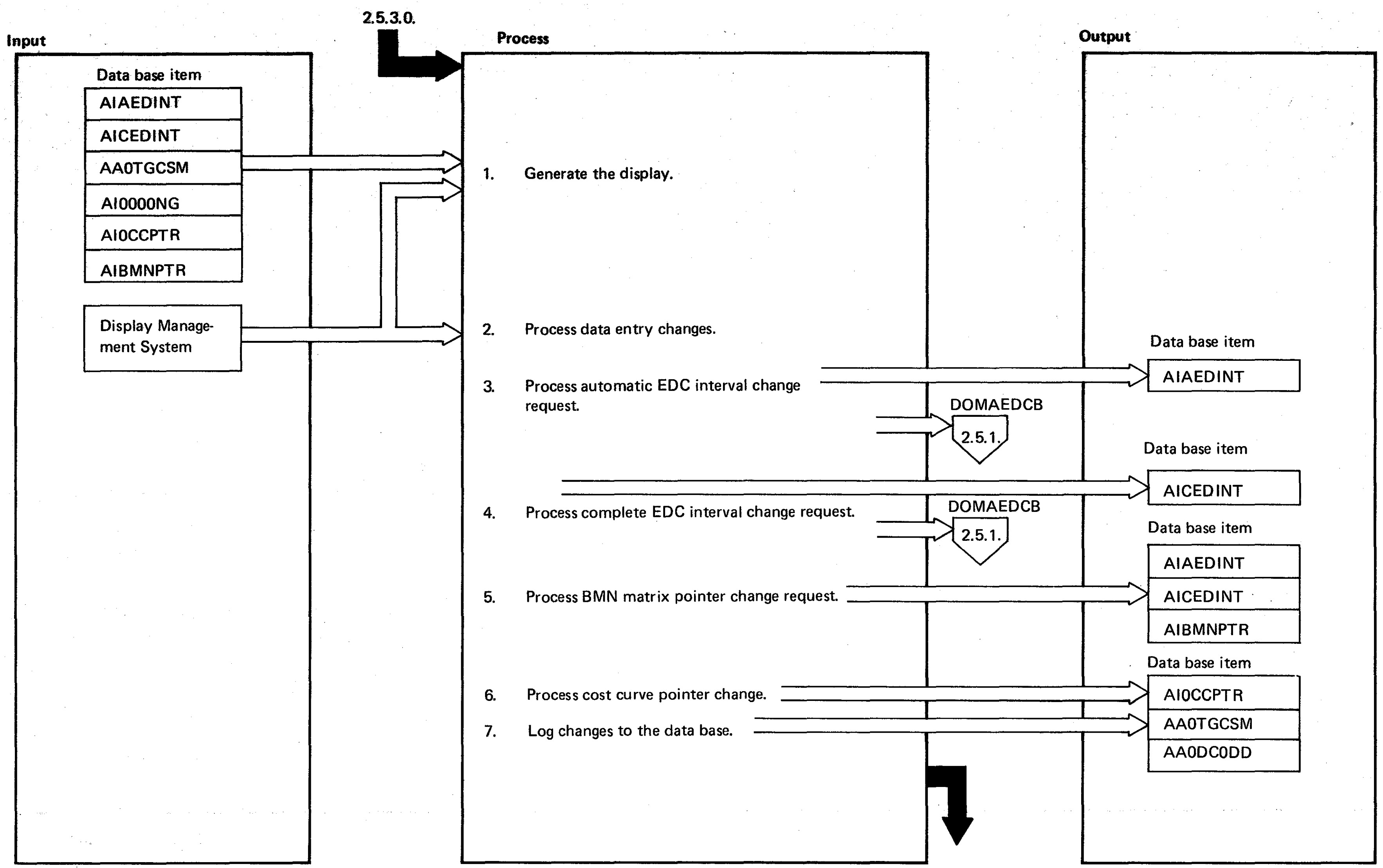


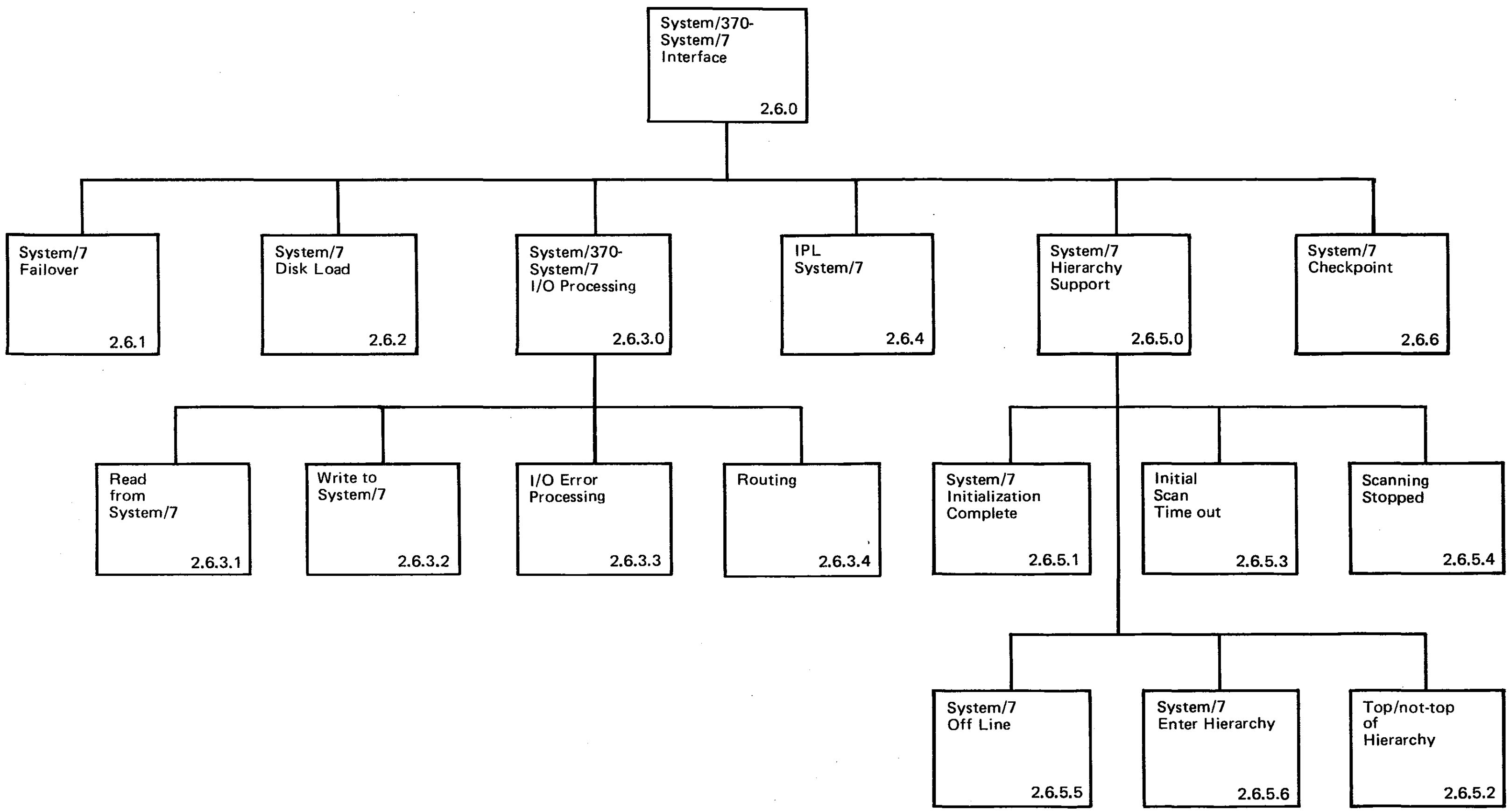
DIAGRAM 2.5.3.4: EDC Intervals/Costs/Losses

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. A patch ID of 1 indicates this function is to be performed	DOMAICUP	2.5.3.4.1
2. A patch ID of 7 indicates this function is to be performed	DOMAICUP	2.5.3.4.2
3. PTIME patch interval is modified, patch ID is 1	DOMAEDCB	2.5.1
4. PTIME patch interval is modified, patch ID is 2	DOMAEDCB	2.5.1
6. New cost curve pointer name is stored in AIOCCPTR	DOMAICUP	2.5.3.4.6
7. Item AA0TGCSM is logged only if the cost curve changes, otherwise AA0DC0DD is logged	DOMAICUP	2.5.3.4.7

DIAGRAM 2.5.3.4

Intentionally Blank


System/370-System/7 Interface 2.6.0
System/7 Failover 2.6.1
System/7 Disk Load 2.6.2
System/370-System/7 I/O Processing 2.6.3.0
IPL System/7 2.6.4
System/7 Hierarchy Support 2.6.5.0
System/7 Checkpoint 2.6.6
Read from System/7 2.6.3.1
Write to System/7 2.6.3.2
I/O Error Processing 2.6.3.3
Routing 2.6.3.4
System/7 Initialization Complete 2.6.5.1
Initial Scan Time out 2.6.5.3
Scanning Stopped 2.6.5.4
System/7 Off Line 2.6.5.5
System/7 Enter Hierarchy 2.6.5.6
Top/not-top of Hierarchy 2.6.5.2

2.6.0

Input

Data base
arrays
CASYS
TAS7COMM

Process

1. If failure is of a primary System/7 then vary all terminals out of service.
2. Purge I/O to the System/7.
3. Update TAS7COMM of failure.
4. Event the failure.
5. Alarm the failure.
6. Vary a backup into service.
7. Return to Special Real Time Operating System.

Output

TAS7COMM
data base
array

DIAGRAM 2.6.1: System/7 Failover

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. Varying terminals is done by PATCHing DOMTVRPT with ID = 5	DOMTVRPT	2.2.4
3. Issue UPD7COMM macro	DOMTABLE	2.7.2.0
4. SCEVENT macro	DOMTABLE	2.7.2.0
5. DOMCALRM macro	DOMTABLE	2.7.2.0
6. VARYS7 macro	DOMTABLE	2.7.2.0

DIAGRAM 2.6.1

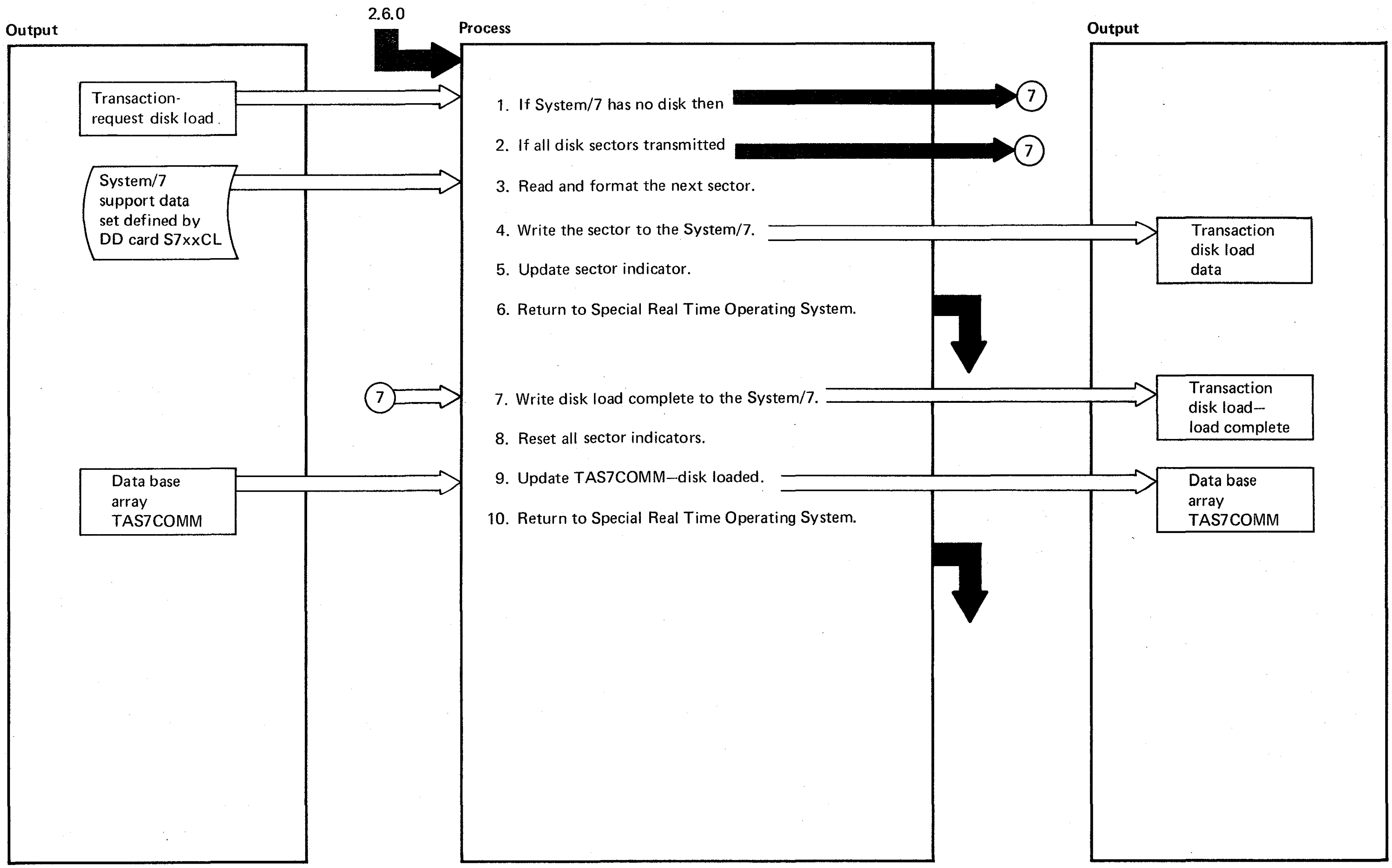


DIAGRAM 2.6.2: System/7 Disk Load

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. Input transaction code is X'92'		
3. Call to DOMTPDSG to read a logical record. Records are FORMAT/7 format	DOMTDISK	2.6.2
4. Output transaction code is X'12'		
Data written to System/7's done through S7WRITE macro	DOMTABLE	2.7.2.0
9. UPD7COMM macro	DOMTABLE	2.7.2.0

DIAGRAM 2.6.2

Input

Data from System/7

2.6.3.0

Process

1. Read data from System/7.
2. If no I/O error route the input.

DOMTPUNT 2.6.3.4

3. Else I/O error.

DOMTIOER 2.6.3.3

4. Allocate input buffer.
5. Read next input.

Output

DIAGRAM 2.6.3.1: Read From System/7

EXTENDED DESCRIPTION

Notes	Modules	Diagram
4. Buffers are allocated here but they must be released by issuing the RLSEBUFF macro which invokes the subroutine DOMTRLBF. This subroutine flags the buffers as again available		

DIAGRAM 2.6.3.1

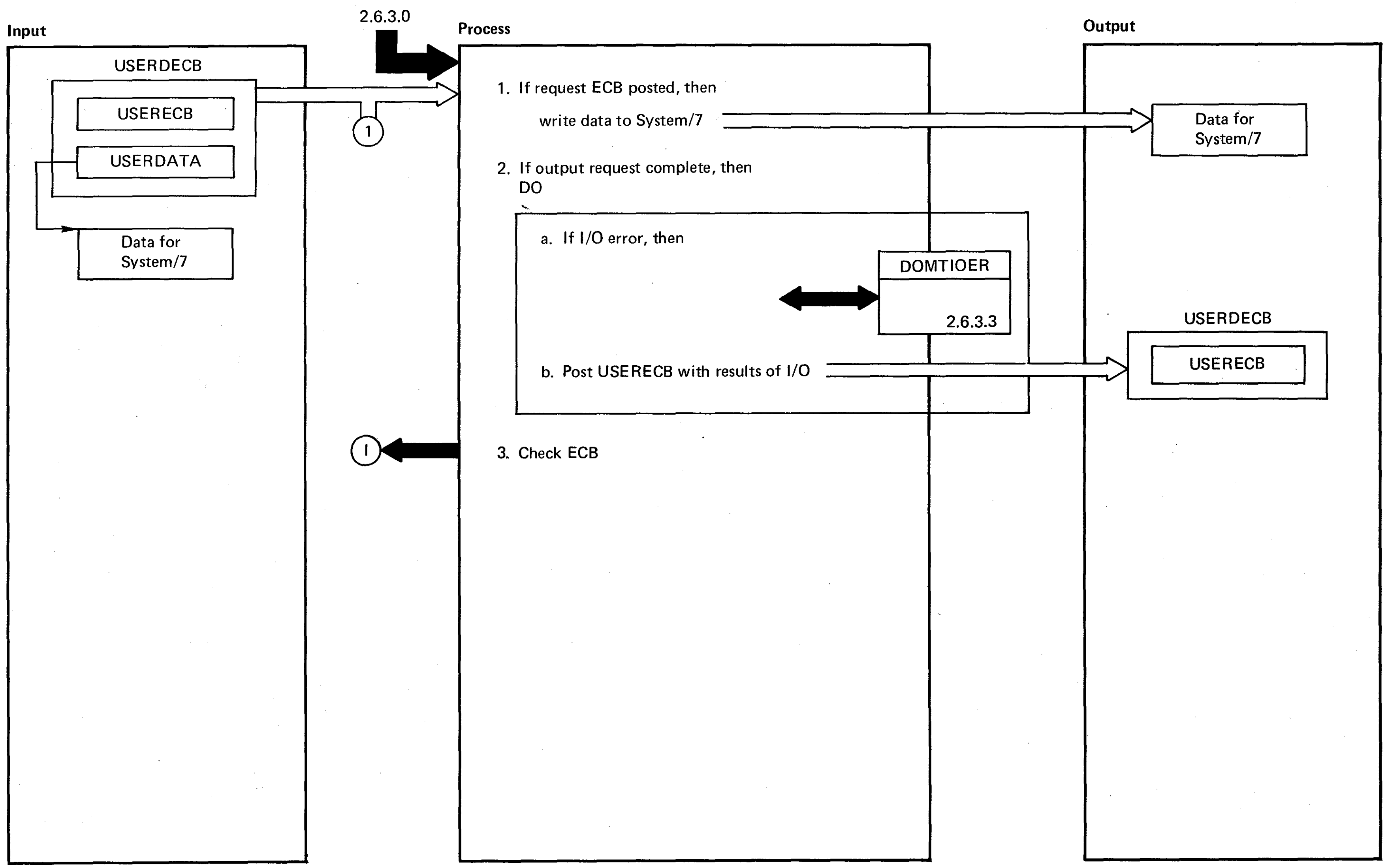


DIAGRAM 2.6.3.2: Write to System/7

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. The user deck is built by the output interface subroutine DOMTWRIT		

DIAGRAM 2.6.3.2

2.6.3.0

Input

Process

1. Purge all I/O to the System/7 in error.
2. Invoke System/7 failover processing.
3. Return to caller.

EXIT

Output

DIAGRAM 2.6.3.3: System/7 I/O Error Processing

EXTENDED DESCRIPTION

Notes	Modules	Diagram
2. Patch DOMTFAIL	DOMTFAIL	2.6.1

DIAGRAM 2.6.3.3

2.6.3.0

Input

Input transaction

Transaction routing table

Process

1. Search routing table for input transaction code.
2. If transaction code found, then route to processing program.
3. ELSE-DO
   a. Issue message that invalid transaction code.
   b. Release input buffer.
4. Return to caller.

Output

Input transaction

Error message

DIAGRAM 2.6.3.4: System/7 Input Routing

EXTENDED DESCRIPTION

Notes	Modules	Diagram
2. Routing is done by PATCHing the processing program with the address of the input buffer as the parameter		

DIAGRAM 2.6.3.4

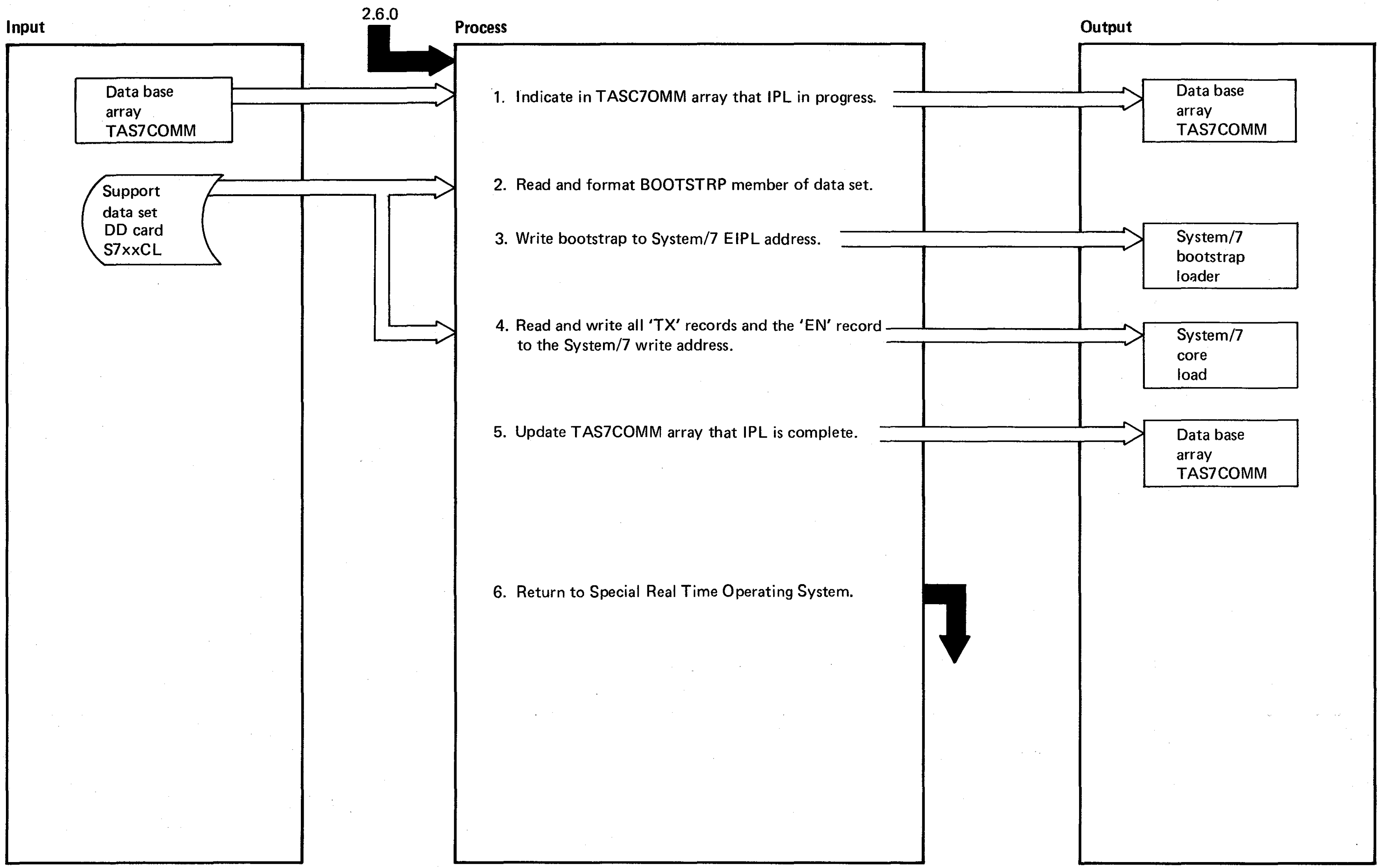


DIAGRAM 2.6.4: IPL System/7

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1,5. DOMTUPDT is called to perform the update		
2,4. DOMTPDSG is called to read a logical record. The records are FORMAT/7 records	DOMTEIPL	2.6.4
The xx in S7xxCL DD card defines the logical ID of the system for which the data was gened		
S7IPL macro used to write data to System/7	DOMTABLE	2.7.2.0

DIAGRAM 2.6.4

2.6.5.0

Input

System/7 initialization complete transaction

Process

1. If System/7 desired state is backup then
   a. Flag System/7 now in desired state.
   b. Event-backup operational.
   c. System Message operational.
   d. Go to step 4.
2. Change System/7 input buffers to scan buffers.
3. Command System/7 to perform primary initialization functions.
4. Release input buffer.
5. Return to Special Real Time Operating System.

Output

System Message

Assume primary transaction

DIAGRAM 2.6.5.1: System/7 Initialization Complete

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. Input transaction code is X'83' 3. Output transaction code is X'04' 4. RLSEBUFF macro calls DOMTRLBF		

DIAGRAM 2.6.5.1

Input

- Top/not-top of hierarchy transaction
- Data base arrays CASYS TAS7COMM

Process

1. If change in position in hierarchy, then DO
   - a. Update new position.
   - b. Event new position in hierarchy.
2. If transaction for local System/7 initializing, then DO
   - a. Vary all terminals online this leg.
   - b. If no operator action required, then DO
     - (1) Flag node is up.
     - (2) If at the top, then command leg to stop scanning.
     - (3) Else inform top that node is up.
   - c. Else, notify operator that note is ready to enter hierarchy.

Output

- TAS7COMM array
- Stop scanning transaction
- Top/not-top transaction (node up below)
- System message to power system operator

DIAGRAM 2.6.5.2: Top/Not-Top of Hierarchy

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. Input transaction code is X'84', Informs position (top/not-top) in hierarchy, lines up and down 1.b. System type event, SCEVENT macro 2.a. PATCH DOMTVRPT ID = 6 2.b.1 Issue UPD7COMM macro 2.b.2 Output transaction is X'00'stop' immediately S7WRITE macro 2.b.3 Output transaction is X'84' with node is up, S7WRITE macro 2.c. Display system message to power system operator access and function area	DOMTVRPT	2.2.4

DIAGRAM 2.6.5.2

Input

Process

3. ELSE IF NODE IS UP, THEN DO

a. IF NODE IS UP FROM BELOW' THEN DO

(1) Event the node is up

(2) If local System/7 not waiting for dispatcher action, then command leg to stop scanning.

4. ELSE IF DOWN NODE IS BELOW, THEN EVENT NODE DOWN.

5. Release input buffer.

6. Return to Special Real Time Operating System.

Output

DIAGRAM 2.6.5.2

EXTENDED DESCRIPTION

Notes	Modules	Diagram
3.a.1 System type event, SCEVENT macro 3.a.2 Output transaction is X'00', stop immediately, S7WRITE macro 4. System type event, SCEVENT macro 5. RLSEBUFF macro		

DIAGRAM 2.6.5.2

2.6.5.0

Input

Initial scan timeout

Process

1. Event initial scan timeout.
2. Notify operator of condition.
3. Release input buffer
4. Return to Special Real Time Operating System.

Output

Initial scan timeout, restart scanning

DIAGRAM 2.6.5.3: Initial Scan Timeout

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. SCEVENT system type 2. System message to power system operator access and function areas 3. RLSEBUFF macro		

DIAGRAM 2.6.5.3

2.6.5.0

Input

Scanning has stopped transaction

Data base arrays TAS7COMM

Process

1. Update TAS7COMM array that scanning has stopped.
2. If System/370 at the top of hierarchy, then, DO
   a. If stop to change modes, then DO
      (1) Broadcast scanning mode down leg.
      (2) Restart scanning.
   b. Else DO
      (1) Write current date and time to System/7.
      (2) Write status of all cyclic scans and current mode.
      (3) Command leg to start scanning.
      (4) Update TAS7COMM array that the mode is up.
3. Release the input buffer.
4. Return to Special Real Time Operating System.

Output

TAS7COMM array

Current scan mode

Current year, day, time

Scan ID status

TAS7COMM array

DIAGRAM 2.6.5.4: Scanning Has Stopped

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. Issue UPD7COMM macro 2.a.1 Transaction code X'08' 2.a.2 INITSCAN macro type cyclic 2.b.1 Transaction code X'0E' 2.b.2 Transaction code X'08' 2.b.3 INITSCAN macro type initial 2.b.4 UPD7COMM macro 3. RLSEBUFF macro		

DIAGRAM 2.6.5.4

2.6.5.0

Input

System/7 offline transaction

Data base arrays TAS7COMM

Process

1. Vary all terminals on this leg out of service.
2. Update TAS7COMM array that System/7 is offline.
3. Event System/7 going offline.
4. Release the input buffer.
5. Return to Special Real Time Operating System.

Output

TAS7COMM array

DIAGRAM 2.6.5.5: System/7 Offline

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. PATCH DOMTVRPT with ID of 5 2. Issue UPD7COMM macro 3. SCEVENT macro TYPE = SYSTEM 4. RLSEBUFF	DOMTVRPT	2.2.4

DIAGRAM 2.6.5.5

2.6.5.0

Input

DOMDPCC1 power config. control display (logical id)

Process

1. IF LOGICAL ID WAITING for Power System Operator ACTION, THEN DO

   a. Display request accepted.

   b. IF SYSTEM/370 AT TOP THEN COMMAND LEG TO STOP SCANNING

   c. ELSE DO

      (1) Sent top/not-top, mode is up, to the top.

      (2) Update TAS7COMM array node is up.

2. ELSE DISPLAY ERROR MESSAGE.

3. Return to Special Real Time Operating System.

Output

DOMDPCC1 request accepted

Stop scanning immediately

Top/not-top transaction to the top

DOMDPCC1 logical id not controllable

DIAGRAM 2.6.5.6: System/7 Enter Hierarchy

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1.b. Output using S7WRITE macro transaction code, X'00' 1.c.1 Transaction code, X'84'		

2.6.0

Input

Request to log transaction

System/7 checkpoint data set—DDname S7xxCKPT

Request to retrieve transaction

Process

1. Read the requested array id and sector.
2. Update the sector incore and write sector back to data set.
3. Return to Special Real Time Operating System.
4. Read the requested array id and sectors.
5. Read the requested sector(s) from check point data set.

   Write the sector(s) to the System/7

   Return to Special Real Time Operating System.

Output

Updated, System/7 checkpoint data set DDname S7xxCKPT

Requested logged data

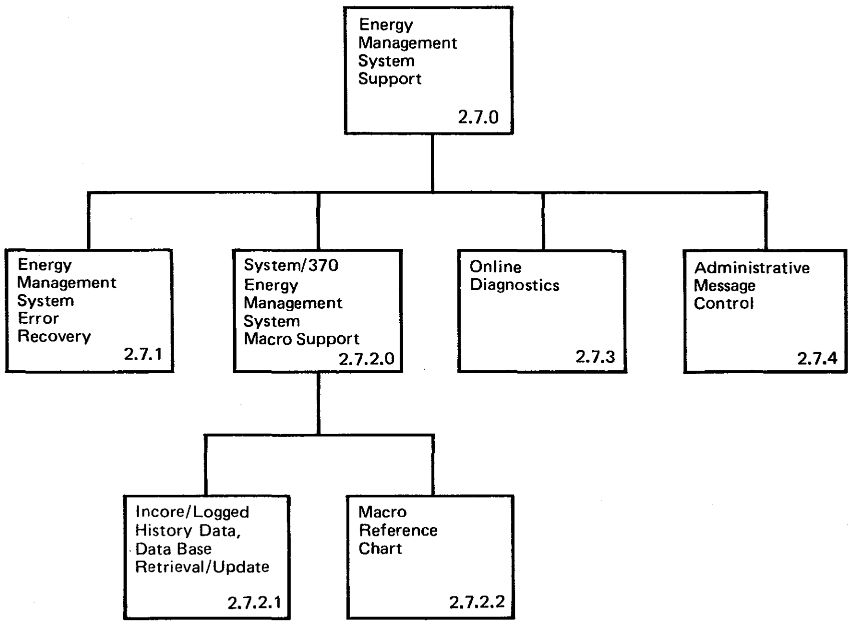
DIAGRAM 2.6.6: System/7 Checkpoint

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. Input transaction code is X'98' 4. Input transaction code is X'99' 5. Output transaction code is X'19' Data written to System/7 using S7WRITE macro		

DIAGRAM 2.6.6

Intentionally Blank


Energy Management System Support 2.7.0
Energy Management System Error Recovery 2.7.1
System/370 Energy Management System Macro Support 2.7.2.0
Online Diagnostics 2.7.3
Administrative Message Control 2.7.4
Incore/Logged History Data, Data Base Retrieval/Update 2.7.2.1
Macro Reference Chart 2.7.2.2

Input

OS abnormal termination and STAE input parameters

Process

1. Search Task Control Block chain for abending task and module names.

2. Is parameter list provided by operating system?

   YES → 4

3. Is information provided via registers?

   NO → 6

4. (4) Write message telling abending module name and displacement into module.

5. Release system resources used by abending task.

6. (6) Re-establish task via DOMCTCBI.

Return

Output

DIAGRAM 2.7.1

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. The TCB chain will provide information to the current task and entry point name and address of the abending module. Displacement into module can be computed 2. STAE parameter list may be provided by OS if space is available. If not, registers will point to input. Parameter format is doc ımented with OS STAE documentation 4. Message output will give name of abending module, displacement from entry point, and abend condition code 5. The STAE parameter list passed by the abending module will determine what system resources are to be released. The following requests are currently being handled: FREEMAIN, FREEWA, zero an area, re-initialize and area, and the alteration of flag bits 6. Patch to entry point DOMCTCBI under the abending task name to re-establish the task		

DIAGRAM 2.7.1

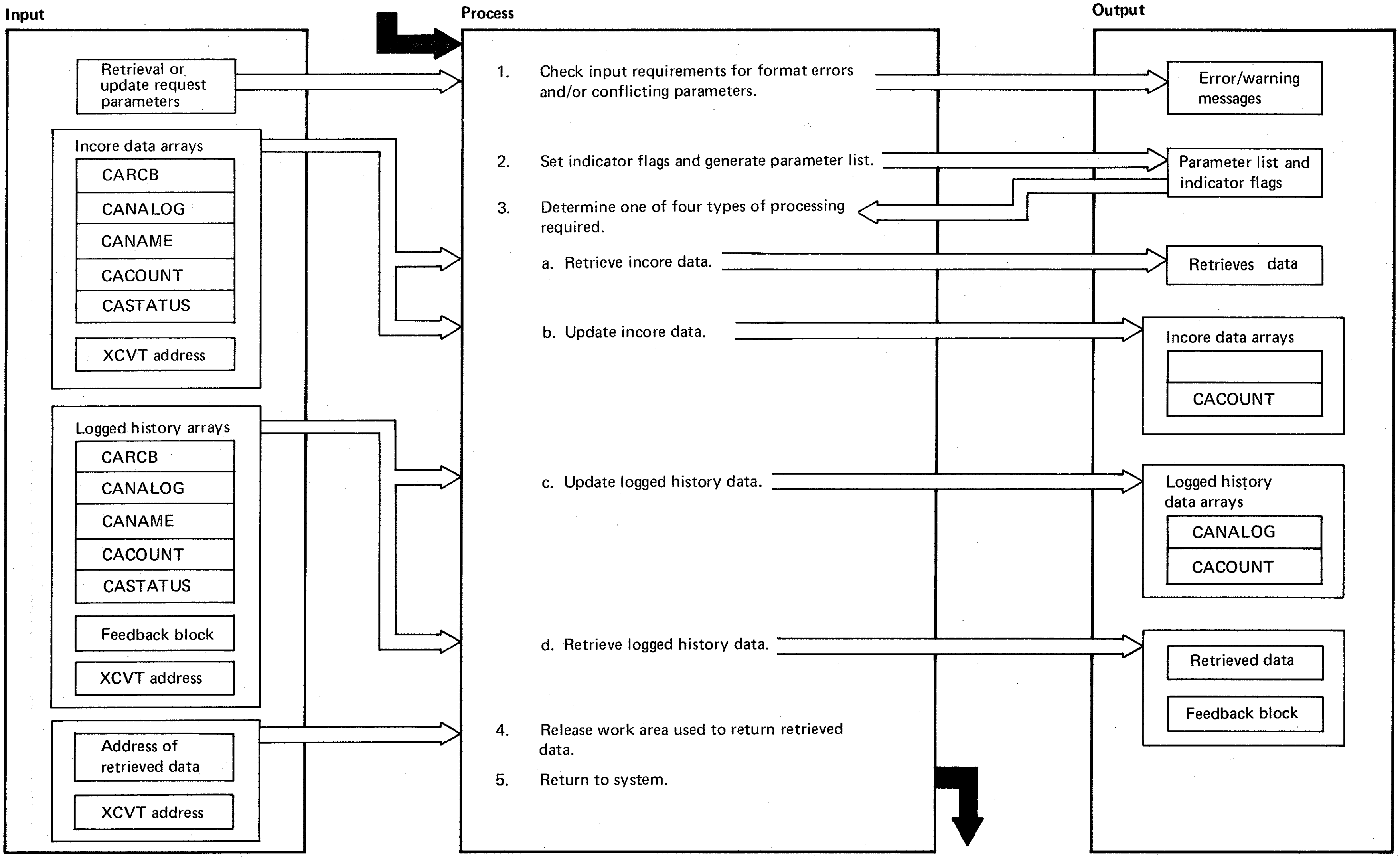


DIAGRAM 2.7.2.1: Incore/Logged History Data Base Retrieval/Update

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. Entry is from a user module which issues a DOMCLGET or DOMCLPUT macro	User	
2. The macro sets up a parameter list of addresses and flags which is passed to the control program	User	
3. One of four types of processing will take place based on four high order bits of the passed macro ID ID = 0000XXXX – 3.a. ID = 0100XXXX – 3.b. ID = 0110XXXX – 3.c. ID = 0010XXXX – 3.d.	DOMTABLE	2.7.2.0
Note for 3.a. and 3.b.: Incore data arrays use the remote control block (RCB) array to retrieve pointers into the analog, names list, pulse counter, and/or status arrays. Data is only retrieved from one of the five arrays or data is only updated in the analog or pulse counter array during a single execution of this function. The only exception to the above is the names list and analog arrays from which data is retrieved at the same time for a request for analog data. Names list data cannot be requested separately		
3.a. Register one will point to the retrieved data	DOMTABLE	2.7.2.0
3.b. Only analog data and pulse counter data may be updated	DOMTABLE	2.7.2.0
Note for 3.c. and 3.d.: Logged history data arrays use the remote control block (RCB) array to access the other four arrays: analog, names list, pulse counter, and/or status array. Data is only retrieved or updated in one array during a single execution of this function. The only exception to the above is the analog and names list arrays, from which data is retrieved in both arrays at the same time.		
3.c. Only analog data and pulse counter data may be updated	DOMTABLE	2.7.2.0
3.d. Register one will point to the retrieved data	DOMTABLE	2.7.2.0
4. This function is entered by a user module issuing a DOMCFREE macro which releases main core which has been used to pass retrieved data	DOMTABLE	2.7.2.0

DIAGRAM 2.7.2.1

Input

Process

This area intentionally left blank.

See Extended Description below:

Output

DIAGRAM 2.7.2.2: Macro Reference Chart

EXTENDED DESCRIPTION

Notes	Modules	Diagram
These macros are System/370 Energy Management macros which may be used by the user. These macros have control programs which are referenced in the CSECT column. The Ref. column refers to other HIPO diagrams where these CSECT will be referenced.		
Macro		
DOMCLGET	DOMTABLE	2.7.2.1
DOMCLPUT	DOMTABLE	2.7.2.1
DOMCFREE	DOMTABLE	2.7.2.1
DOMCALRM	DOMTABLE	2.3.1.0
ASCICONV	DOMTABLE	2.7.4
SCEVENT	DOMTABLE	2.3.3.0
SCDEVICE	DOMTABLE	2.3.2.0
S7WRITE	DOMTABLE	2.6.3.1
VARYCONF	DOMTABLE	2.2.4
VARYSCAN	DOMTABLE	2.2.1.0
RLSEBUFF	DOMTABLE	2.6.3.3
DOMCSCHT	DOMTABLE	2.3.4.0
VARYS7	DOMTABLE	2.6.1

DIAGRAM 2.7.2.2

2.7.0

Input

DOMDDIGF access, typer function, routing code and pattern

Data base arrays CAAATBC CASYS

Process

1. Read display input fields.
2. If any errors on input data, then DO
   a. Issue error message.
   b. Return to Special Real Time Operating System.
3. Determine the typer, routing, code, and pattern is to be written.
4. Write pattern using routing code.
5. Display destination routing code.
6. Return to Special Real Time Operating System.

Output

Error message DOMDDIGF

Pattern to target typer

DOMDDIGF routing code

DIAGRAM 2.7.3: Online Diagnostics

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. Input consists of access, function, typer (G, E or A), routing code		
2. Invalid parameter error messages are issued by calling DISPLAYP	DOMCDIG1	2.7.3
3. If routing code specified, then skip to step 4 if typer = G user general typer routing code from CASYS array and go to step 4. Using access and function search CAAATBL for routing code for desired access/function and typer		
4. Use MESSAGE macro		

DIAGRAM 2.7.3

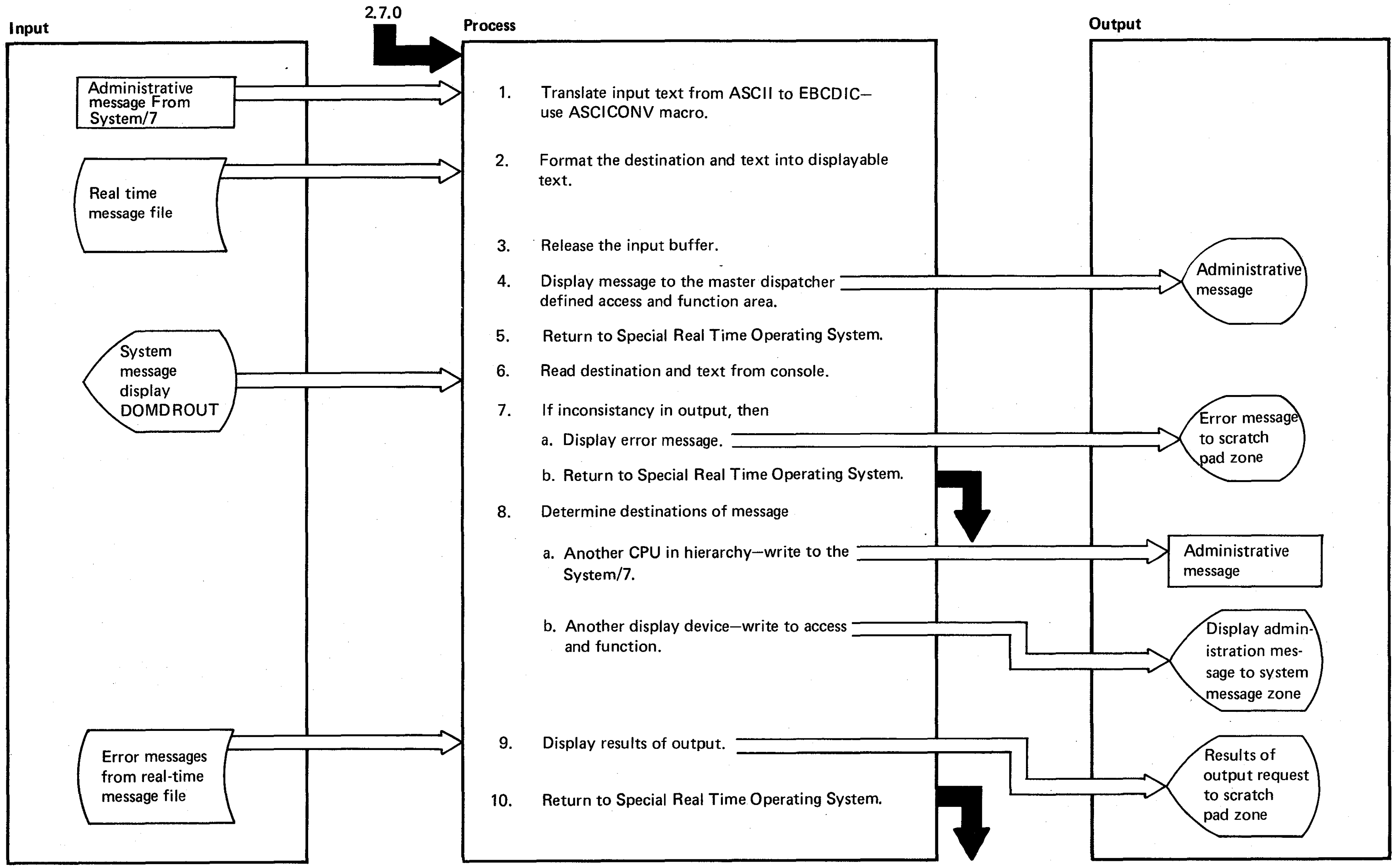


DIAGRAM 2.7.4: Administrative Message Control

EXTENDED DESCRIPTION

Notes	Modules	Diagram
1. ASCICONV macro executes a translate instruction from the CSECT DOMTCODE which is linkedited in DOMTABLE 4. A DWZONE to the system message zone is used to route the message Input transaction code is X'89' 3. RLSEBUFF macro 7. The input may have a CPU ID or access and function specified but not both; text must be supplied 8. CPU or display routing. The macros issued validate the input a. Use the S7WRITE macro Output transaction code X'89' b. Use Display Management System DWZONE, ZONE = SM	DOMTABLE	2.7.2.0

DIAGRAM 2.7.4

CHAPTER 3. PROGRAM ORGANIZATION

The System/370 Energy Management System is organized as several subsystems, each with its own function. The subsystems are logically separated from each other, but each uses services or data provided by the other subsystems, or provides data or services for the other subsystems. These various online subsystems communicate with each other through the Special Real Time Operating System Programming RPQ (enhanced OS/VS1). Communication with the IBM 5985 Color Display is provided by the Display Management System Programing RPQ.

The online System/370 Energy Management System executes as a job step under OS/VS1. Within this job step, each subsystem has an independent task substructure. For each subsystem responsible for the tables used by the subsystem there is an independent task that is the subtask of the job step task. The system initialization routines create these tasks at system startup. From that time they remain in existence to maintain the resources required for the subsystem to operate.

The System/370 Energy Management System is basically table-driven. The power network paramaters are input to the offline utility to build the data set that initializes the online tables. These tables are used to monitor and control the power network. The power system operator controls the System/370 Energy Management System processing by modifying the table content or selecting the tables and initiating or suppressing certain functions.

The system is designed to fulfill the needs of any electric utility from a medium-sized one which may utilize an IBM System/370 Model 135 and a single IBM System/7 to a large utility which may utilize either a single or multiple duplex large scale System/370s and multiple System/7s. The System/370 Energy Management System may be generated with different options and configurations selected according to the needs of the user. The System/370 Energy Management System can be generated on a standard OS/VS system which has been system generated by standard procedures, but it must execute in conjunction with the Special Real Time Operating System and the Display Management System.

The data acquisition subsystem provides routines that retrieve data from the power network through the System/7 interface to that network, convert the data, and store it into the data base according to the parameters specified during system generation. Commands which control the power network are output to the appropriate devices by this subsystem through the System/7. This transfer of data is under the control of the supervisory control subsystem.

The supervisory control subsystem is the central point of status determination and network control. This subsystem interfaces with the power system operators through the displays to accept network control commands and to present the status of the network. Alarm conditions that occur in the network are presented on the displays, typers and/or online printers according to the specifications established during system generation. The supervisory control subsystem causes the network data to be scanned at the rate selected by the power system operators by sending the appropriate data to the data acquisition subsystem. Any network alarm conditions that are detected by the data acquisition subsystem are forwarded to supervisory control for appropriate action. Control actions (opening or closing circuit breakers, etc.) are commanded by the power system operator to supervisory control. Supervisory control then validates the commands and forwards them to the appropriate System/7 for execution.

Portions of the data base are selected for restart/switchover considerations and for later analysis. These data base portions are then recorded on direct access devices on a cyclic basis by supervisory control. The arrays to be recorded and the time recording interval for each array are selected at system generation.

The application processing subsystem performs the two functions of Automatic Generation Control (AGC) and Economic Dispatch Control (EDC). They use the data which has been retrieved from the network in conjunction with user supplied parameters to determine that the proper amount of power is being supplied to the network. From these computations, power output from selected sources is increased or decreased to meet the needs of the network by the most economical means.

The following sections describe the program logic of each subsystem of the System/370 Energy Management System. The basic functions of each subsystem are as follows:

<u>Supervisory Control</u>

- Initialization
- Alarm management
- Alarm display
- Device control
- Display of sensor based data
- Events processing
- Scan control display
- Stripchart processing
- Power configuration control
- Incore/logged data base retrieval/update
- Online diagnostics

<u>Data Acquisition</u>

- Initialization
- System/7 intercommunications (channel attached)
- Scan processing
- Scan data conversion
- Realtime changes
- Time synchronization
- System/7 failover (channel attached)
- Point summation
- Data logging
- AGC/EDC initialization
- Customer interface control

<u>Application Processing</u>

- Automatic Generation Control
  Data Base Structure
  Initialization
  Cyclic Processor
  Output Interface Processor
  Operator Interface Processing
- Economic Dispatch Control
  Data Base Structure
  Initialization
  Cyclic Processor
  Operator Interface Processing

SUPERVISORY CONTROL

This section describes the use of the supervisory control functions. These functions are:

- Initialization
- Alarm management
- Alarm display
- Device control
- Sensor based data display
- Events processing
- Scan control display
- AGC outputinterface
- Stripchart processing
- Power configuration control
- Incore/logged data base retrieval/updata
- Online diagnostics

Each function is described through the description of the programs which accomplish the function and through a Program Design Language (PDL) listing of each program.

INITIALIZATION

The EXEC card of the job step specifies the program name of the Special Real Time Operating System initialization load module. This program initializes the control blocks necessary to accept service requests by other subsystems and initializes the data base. The control program reads a series of card images from a pre-defined data set or the input stream. This data defines the sequence of events which is to occur during system initialization. It consists of PATCH, WAIT and other Special Real Time Operating System control statements which are executed in the sequence in which they appear on the input. The control statements and their functions are as follows:

PATCH causes a program to be executed or queued for execution. Most parameters that are allowed on the PATCH macro are allowed here. Data parameter values may be specified as character data (EBCDIC), hexadecimal data, or full words specified as decimal integers. A label may be coded (card columns 1-8) and referenced in succeeding WAIT control statements.

WAIT causes the initialization procedure to wait until a specific task created as the result of a PATCH control statement completing execution before processing the next control statement.

RESTART WRITE causes a restart data set to be written. The program waits for all preceding PATCHes to be completed before the data set is written. When a restart occurs, processing resumes at the point in the initialization sequence where this statement appears. PATCH and WAIT statements that follow are processed both during the initial startup and after a restart. A flag is set to indicate to the PATCHed program that an initial startup or restart is occurring. As the result of the control statement processing, the required subsystem control task is created. The subsystems are then responsible for any table or device initialization which is required by that subsystem. In most cases, the subsystem requires an initialization PATCH before the restart data set is written. At this time its task and table structures are built. Another PATCH is required when data sets are OPENed after the restart set is written. Operation actually begins now.

System/370 Energy Management System initialization results from the following card images:

```
A1  PATCH    EP=DOMTRESI ,PARAM=(C,{EVSAVE|EVINIT},C' {CPSAVE|CPINIT} ' )

    WAIT     A1

    RESTART  WRITE

A2  PATCH    EP=DOMTRESI

    WAIT     A2
```

The System/370 Energy Management System initialization program, DOMTRESI, receives two PATCHes, one prior to RESTART WRITE to accomplish the initialization required to go into a checkpoint of the restart data set and a second to accomplish initialization required on a warm start. Supervisory control initialization is initiated by a PATCH from DOMTRESI to DOMCINIT.

DOMCINIT

This processor performs the pre-system functions required by the supervisory control system prior to entering realtime. A PATCH ID of 1 is processed as a cold start and a PATCH ID of 2 is processed as a warm start.

On a cold start (PATCH ID of 1), DOMCINIT performs the following functions. The address of the Energy Management System/370 processor, DOMCSTAE, is obtained through use of the LOAD macro. This address is placed in the ECVTSTAE field of EMSCVT. By using the DOBTAIN Display Management System macro, the power system operator access area and function code IDs are obtained. These IDs are placed in the ECVTMDAA and ECVTMDFC fields of EMSCVT, respectively.

Each remote control block (RCB) is updated with the addresses of its associated analog data, pulse counter data, status data, and names list data. The Special Real Time Operating System GETARRAY NAMELST option is used to retrieve the addresses of the arrays CANALOG,CACOUNT, CASTATUS, and CANAME. If the return code is 8, an S/C message DPPZ52I is output with CC=2 indicating that the GETARRAY failed. If the return code is 4, each address in the address list returned is checked. For each zero address in the list, an S/C message DPPZ52I is output with CC = 3, 4, 5, or 6 indicating failure for array CANAME, CANALOG, CACOUNT, or CASTATUS, respectively. The first RCB for the first System/7 in the System/7 Control Table (S7CT) is updated with the addresses of the above four arrays. The next data array addresses are obtained by multiplying the number of points defined for each in this RCB by the length of each point and adding these values to the previous addresses. These addresses are then placed in the next RCB and this process continues until all RCBs for this System/7 have been updated. The next RCB address list is obtained from the next System/7s entry in the S7CT and processing continues as above with addresses being processed in the same order as they appear in the S7CT.

The Special Real Time Operating System DEFLOCK macro defines the pulse duration output lock, APDO, and the alarm lock, ALOK. Their addresses are placed in the ECVTLPDO and ECVTALOK fields of the EMSCVT, respectively. For non-zero return codes from the DEFLOCK macro an S/C message is output with CC=14 or 10 for APDO or ALOK, respectively, indicating a DEFLOCK failure.

The point summation table, DOMTSMTB, is used by the point summation processor, DOMTPSUM, during realtime processing. This table is built from the system generation point summation table, DOMTPTSM, which is built as a load module with DCs, only. Supervisory control initialization uses the LOAD macro to obtain the system generated copy of DOMTPTSM. The ECVTPTSM field of the EMSCVT contains the address of a partial GETMAIN area large enough to contain the point summation table to be built. As data is obtained from DOMTPTSM, it is processed and placed in the area pointed to by the above address. The scan ID and the length of the first group are placed in table DOMTSMTB first. The addresses of the sum point and points to sum are resolved through use of the Special Real Time Operating System GETITEM macro. If the return code is 8, an S/C message DPP252I is output with CC=11 indicating the GETITEM failed due to invalid options; if the return code is 4, an S/C message DPP252I is output with CC=12 indicating a bad item name. The valid addresses are then placed in the table DOMTSMTB. Processing continues as above for all items in this group. The next group is then processed as above until all scan groups have been completed (X'FF' indicates the end of DOMTPTSM table). A X'FF' is then moved in to indicate the end of the point summation table, DOMTSMTB.

DOMCSENT is called to issue entities based on the sysgened status of the items in the CASTATUS array.

The event log file can either be initialized as a complete new file or it can be reinitialized so events from previous runs are not lost. DOMTRESI uses the PARAM field input on the PATCH card to determine if the file is to be initialized or reinitialized. For reinitialization, the high order bit of the ECVTEVNT field of the EMSCVT is set to one. For initializing the event log file, DOMCINIT creates the event log file data set as a direct data set by opening the data set and initializing all records to X'FFs'. For reinitializing the event log file, DOMCINIT reads all records in the file. A counter is incremented as each record is written or read to obtain the number of records possible on the data set. This information is used to create the event logging control table ECTABLE which is pointed to by the ECVTEVNT field of the EMSCVT. The event lock EVNT is defined and its address placed in the ECLOCKB field of the ECTABLE. For a non-zero return code, an S/C message is output with CC=9 indicating a DEFLOCK failure. Message DPP359I is obtained using the Special Real Time Operating System MESSAGE macro and the initialized ECTYPE field of the ECTABLE. For a non-zero return code, an S/C message is output with CC=15 indicating a MESSAGE failure. For reinitialization, after the event logging control table is built, the last record is located in the event log file as is done during warm start processing. The device control index table pointed to by the ECVTDCIX field of the EMSCVT is cleared for the length of the table (12 * number of entries in the DOMDCTS field of the system generation options array CASYS). The analog performance log initialization routine, DOMTAPLI, is CALLed by DOMCINIT to perform its initialization processing.

Upon receiving control, DOMTAPLI performs the following functions. The addresses for arrays CAPLNAME, CAPTLOG, and CALOGTIM are resolved and stored in the EMSCVT. Each eight character (seven characters plus one blank) analog point name in the CAPLNAME array is resolved to two address pointers. The first is the analog data array pointer and the second address is the pointer into the names list array. Each analog item pointer address is used to search the remote control blocks to find the RCB to which the analog point is attached. When the RCB is found, the pointer into the names list array is then calculated. All entries in the CAPLNAME array which are not valid analog point names are zeroed. A control field that contains the number of valid entries and the number of void entries is maintained as the first word of the CAPLNAME array. This completes analog performance log initialization and control is returned to DOMCINIT.

On a warm start (PATCH ID of 2), DOMCINIT performs the following functions: If this is an initial program load, the Special Real Time Operating System PUTBLOCK macro is used to move one copy of the CACOUNT array to the DACOUNT array to guarantee a good copy of pulse counter data in the direct access array DACOUNT. The Special Real Time Operating System PUTLOG macro is used to log one copy of the RCB array CARCB. This guarantees at least one logged copy on a non-initial warm start initial program load; therefore, when the data base is refreshed, CARCB will not be initialized with initial data.

For non-initial warm start initial program load, the device control index table is cleared again as on a cold start. The event log file is reinitialized to point to the end of the oldest record in the log file. The CSECT member, DOMCIALM, is called to generate alarms for all points that have the alarm outstanding bit on. A loop is set up to process all points for all System/7s. Alarm conditions are placed in a series of buffers which are passed to the alarm processor, DOMCALR1, after all points have been checked. The pulse counter data initialization routine, DOMTPCNT, is called to perform pulse counter data initialization.

Upon receiving control, DOMTPCNT uses the filter value calculated for each good data point during normal realtime processing to estimate each hours worth of accumulated pulse counter data. As each hour of down time is calculated, the pulse counter data array CACOUNT is logged until all pulse counter data points have been estimated up to the last full hour. This makes it possible to have the normal hourly pulse counter data log for the time the system was down.

For both initial and non-initial warm start initial program loads, the CSECT, DOMTAGCI, is called to queue AGC and EDC initialization. Upon receiving control, DOMTAGCI PATCHes the AGC initialization routine, DOMAAGCI, if AGC has been system generated and the EDC initialization routine, DOMAEDCI, if EDC has been system generated. The flag bits, DOMAGC and DOMEDC, in the system generation options array DOMSYSG are checked to determine if AGC and EDC have been system generated, respectively. (Bit on indicates yes.) DOMTAGCI waits for each PATCH to finish in order to guarantee that AGC initialization is complete before PATCHing EDC initialization and that both PATCHes are complete before returning to DOMCINIT.

For both initial and non-initial warm start initial program loads, the CSECT DOMCWSTA is called to update the CASTATUS array to the current status when the system went down.

Sign On Display Initialization

This sign on display is defined in system generation as the display brought to the screen upon Display Management initialization. Display DOMDSGON is an introduction to the System/370 Energy Management System. Another display, DOMDSONM, further defines and explains system defined program function keys and acts as a menu display for functions which do not have system key assignments. These two displays are related in that DOMDSGON is page one of two and DOMDSONM is the second page of the sign-on display.

Display DOMDSONM explains those program function keys which are system defined (keys 1, 2, 3, 4, 5, 6, 9, 10, 11, 12, 13, 24, and 32), those program function keys which have assigned functional responsibilities but are display defined (keys 7, 8, 14, 15, and 16), and special program function keys which are defined for display DOMDSONM menu actions (keys 17 and 18). Information is also displayed on how to call a display to the screen by name and how to generate a PATCH to a program.

DOMCSOMD: This module is PATCHed when display DOMDSGON is brought to the screen. The purpose of this CSECT is to do initialization processing, if required, for the display unit. Backlights for the program function keys are turned on. Keys 1 through 8, 11 through 16, 24 and 32 are lighted. Keys 9 and 10 are lighted if the Economic DisPATCH Control and Automatic Generation Control functions are supported.

A control element address table (CEATAB) is generated for the display unit if one does not already exist. The address pointer is placed in the DCEUSER field of the display control element (DCE) table. Initially the DCEUSER field is zero. The control element address table is zeroed upon generation. The first full word (4 bytes) of the CEATAB, pointed to by the DCEUSER field in the DCE, is reserved for use by electric utility industry users. The remaining fields are used by separate display functions, which are independent tasks for the purpose of maintaining pointers to the remote control element (RCE) for each task. Each task is responsible for the control and use of the four byte area assigned to it in the control element table (CEATAB).

DOMCSOMD runs as an independent task and is re-enterable and reusable.

ALARM MANAGEMENT

Alarm management controls the issuing, acknowledgment, deletion and viewing of alarms in System/370 Energy Management System. By using the DOMCALRM macro, the user may enter an alarm into the system causing a general alarm and a detail alarm to be generated. Using Display Management, the general alarm is added to the general alarm table and displayed on the display units for the applicable access area. The alarm is also sent to the alarm typer(s), if any. The detail alarm becomes part of the incore alarm list. The alarm handler module, DOMCALR1, also handles alarms included in the scan exception table which are created by DOMTCONV, the scan conversion processor.

The modules which perform these functions are as follows:

DOMCALR1

DOMCALR1, running under the DOMXALRM task, is PATCHed by DOMCALM2 or DOMTCONV. The parameters passed to it are alarm information generated by using the DOMCALRM macro or the scan exception table (SET) from scan processing. The DOMCALR1 module calls DOMCALR3, DOMCALR4, or DOMCALR5 to format the status, analog, or pulse counter detail alarm record. The non-sensor based data (message) type of detail alarm record is built by DOMCALR1. DOMCALR6 is then called to add, update or delete the alarm according to the indicators passed, and to issue the typer messages. DOMCALR1 PATCHes the events processor (DOMCEVT5), the wallboard processor (DOMCWBPR), and the change of status logger (DOMCSLOG) if applicable. Using the DISPENT macro, the list of entity changes is passed to the Display Management System entity function.

DOMCALM2

DOMCALM2 is a module linked to by the macro which generates a parameter list resembling the SET. DOMCALM2 PATCHes DOMCALR1 and passes it the parameter list for the actual processing of the alarm.

DOMCALR3

DOMCALR3 is a module called by DOMCALR1. It formats the detail alarm record for status data. It also determines what the alarm condition code is, based on the type of status point and the current status of the point. It adds an entry to the entity list indicating whether the entity should be added or deleted. It also adds an entry to the change of status list to be passed on to the status log processor. If a point has the wallboard flag set on, the address of the point is added to the wallboard list.

DOMCALR4

DOMCALR4 is a module called by DOMCALR1. It formats the detail alarm record for analog data. It adds an entry to the entity list indicating whether the entity is to be added or deleted. If the wallboard flag is on, the address of the point is added to the wallboard list.

DOMCALR5

DOMCALR5 is a module called by DOMCALR1. It formats the detail alarm record for pulse counter data. It adds an entry to the entity list indicating whether the entity is to be added or deleted.

DOMCALR6

DOMCALR6 is a module called by DOMCALR1 to add, update or delete an alarm record from the incore alarm chain. For both alarmable and not alarmable points an alarm typer message is issued and an event is added to the event buffer. It also sends a message (transaction code x'10') to the System/7 for the audible alarm using the S7WRITE macro.

If the general alarm is not outstanding or if it has been acknowledged, it issues the general alarm using the DALARM macro. If all alarms for a remote control block RCB or an alarm control block ACB have been deleted, the general alarm is deleted using the DALARM macro. The module locks on the incore alarm chain and either adds, updates, or deletes the alarm. The count of alarms in the RCB and the access area table are adjusted and the chain pointers in the access area table are adjusted if applicable.

If the point is not alarmable or out of service and is a status point, an event is added to the event buffer indicating that a change of status occurred.

ALARM DISPLAY

The power system operator may view the existing alarms for any access area and function through the Detail Alarm Display. Initially, the primary access area and function are used in building the display. The power system operator may key in a different access area and/or function. The alarms are displayed in the sequence in which they occurred and he may page forward and backward to view alarms if there is more than one page of alarms. The power system operator may acknowledge or delete an alarm which has the same access area and function as his display unit. The power system operator may refresh the page he is viewing. The current page is automatically refreshed when an acknowledgement or deletion is made.

The modules which accomplish these functions are as follows:

DOMCALD1

The viewing of alarms by the power system operator is handled by DOMCALD1, which runs under the DOMXEMD2 task. It is PATCHed by Display Management to view the detail alarms for the access area and function chosen. Through MIAT entries, the power system operator may acknowledge or delete alarms which pertain to the access area and function code assigned to his display unit. He may also page backward and forward and refresh the current page being viewed. The various functions of DOMCALD1 are indicated by PATCH IDs as follows:

- A PATCH ID of 1 indicates the initial phase of the display processing. The access area table is searched to find the number of alarms outstanding for the access area and function chosen. If there are none, a message is displayed indicating no alarms. Otherwise, an alarm control element (ACE) is built and the first page of alarms is displayed. DOMCALD3 is called to retrieve the alarms from the incore alarm chain and DOMCALD2 is called to format and display the alarms.

- A PATCH ID of 2 indicates alarm acknowledgment. The acknowledgment indicator in the detail alarm record is set and the current page is updated to reflect the change in attributes of the acknowledged alarm. The acknowledgement is also evented. DOMCALD4 is called to do the acknowledgement. DOMCALD3 is called to retrieve the alarms and DOMCALD2 is called to format and display the updated alarms.

- A PATCH ID of 3 indicates alarm deletion. If the access area and function code are valid, the detail alarm record is deleted, and the deletion is evented. The RCB alarm count is decreased. If the RCB count goes to zero, the general alarm is deleted, using the DALARM macro. The data base item is updated, if applicable. The current page being viewed is updated to reflect the deletion. If the deletion was not allowed, a message to that effect is displayed. DOMCALD4 is called to do the deletion. DOMCALD3 is called to retrieve the alarms and DOMCALD2 is called to update the page.

- A PATCH ID of 4 indicates forward paging. DOMCALD3 and DOMCALD2 are called to retrieve the most current alarms and display the next page.

- A PATCH ID of 5 indicates backward paging. DOMCALD3 and DOMCALD2 are called to retrieve the most current alarms and display the previous page.

- A PATCH ID of 6 indicates refresh of the current page being viewed. Any acknowledgments or deletions from a different display unit are reflected. DOMCALD3 and DOMCALD2 are called as above.

- A PATCH ID of 7 is used when a new display is requested. This is a cleanup function which frees any areas reserved by a new previous functions.

- A PATCH ID of 8 indicates that the power system operator has chosen a different access area and/or function. The display screen is cleared, the ACE is updated, and the alarms are retrieved and displayed as in the initial PATCH.

DOMCALD2

This module is called by DOMCALD1 to format the alarms and write them to the screen using the DINFO and DISPUP macro.

DOMCALD3

This module is called to retrieve the alarms from the incore alarm chain. It enqueues on the chain using the LOCK macro and reads the alarms into a save area. It then releases the chain for further use using the LOCK macro.

DOMCALD4

This module is called by DOMCALD1 to acknowledge or delete an alarm. If the alarm is still active it either acknowledges the alarm or deletes it if the alarm condition has not changed. Deletion causes the rechaining of the alarm chain. If an alarm is no longer active or has changed condition, a message is displayed on the screen. The alarm chain is locked using the LOCK macro while it is being utilized. The data base indicators are reset when a deletion is done; and, if no other alarms exist for a terminal, the general alarm is deleted, using the DALARM macro. If the alarm has already been acknowledged no processing occurs.

DEVICE CONTROL

The device control programs provide the user a means of controlling power network devices and units in the power environment, using the display unit or software. DOMCDC01 is the controlling module and calls DOMCDC06, DOMCDC07, DOMCDC08, and DOMCDC09 to handle the various phases of device control processing. These, in turn, call DOMCDC11 to validate the control action selected and call DOMCDC05 to send a message to the System/7. DOMCDC01 interfaces with the power system operator at the display unit to select, arm, and execute or cancel the control action. Communication with the power system operator is accomplished through messages generated at various times in the control action indicating the success or failure of the select, arm, and execute or cancel phases. When the control action is initiated by use of the SCDEVICE macro, the user ECB is posted with a return code indicating the success or failure of the device control command. An alarm is issued if the execution phase fails or times-out. The various phases of the control action are recorded in the event log data set as are any errors.

There are three time-outs which may occur during a device control action. The first, selection time-out, is only applicable to display-initiated actions. When the device is selected, DOMCDC03 is PTIMEd to time-out in 30 seconds. If the power system operator does not select an action within 30 seconds, a time-out occurs and the action is canceled and evented.

The second time-out is for the arm phase. It is also only applicable to display initiated actions. When the command is sent to the System/7, DOMCDC03 is PTIMEd to time-out in 30 seconds. If the power system operator does not execute or cancel the command within the 30-second period, a time-out occurs and the action is canceled and evented.

The third time-out is for the execute phase when a change of status is expected. DOMCDC02 is PTIMEd to time-out in the period indicated by the user at system generation. If the change of status is not received within the period, a time-out occurs, the action is canceled, and an event and an alarm are issued.

DOMCDC04 and DOMCDC10 are macro processors interfacing between the SCDEVICE macro user and DOMCDC01, the main control module.

DOMCDC01

DOMCDC01, running under the DOMXPDC task, is PATCHed by either Display Management System or DOMCDC04 to initiate the device control action. It is the control module for device control and is also PATCHed by Data Acquisition when communications are received from the System/7. The PATCH ID determines which phase of the control action is to be handled.

A PATCH ID of 2 is used to indicate initialization from a display unit. DOMCDC06 is called to initialize the control areas needed and do error checking. DOMCDC01 displays the options message or an error message returned by DOMCDC06.

A PATCH ID of 4 is used to indicate initialization of the action through the SCDEVICE macro. DOMCDC06 is called to initialize the control areas needed and do error checking. If the device selected is manual and the control action is complete, the user ECB is posted with the completion code.

A PATCH ID of 8 indicates that a PDC reply message has been received from the System/7 through data acquisition. DOMCDC08 is called to process the message. If the return code from DOMCDC08 indicates that the action is complete, the status item in the data base is updated. If the action was initiated from a display, a message indicating the success or failure of the step is written to the screen. The success or failure of the step is evented.

A PATCH ID of 9 indicates that the PDC information has been received in the RDA from the System/7. DOMCDC08 is called as above (PATCH ID of 8). The only difference in the input for processing by DOMCDC08 is the format of the message processed.

A PATCH ID of 10 indicates that the power system operator is ready to execute the command. If the device is ready, DOMCDC07 is called to set up the execute command. Otherwise, the power system operator is informed that the device is not yet ready. If the control action is complete, a message is displayed and evented and the device control table address is cleared and the device control table released.

A PATCH ID of 12 indicates that a change of status was received in a scan. DOMCDC09 is called to process the change of status. The completion of the action is evented and a message is displayed to the power system operator if the action was display-initiated.

A PATCH ID of 14 indicates that the power system operator has canceled the action or that a display change has occurred. DOMCDC09 is called to clear the device control table address and release the device control table.

A PATCH ID of 16 indicates an execution time-out. DOMCDC09 is called to process the time-out. It is evented and displayed if the action was display-initiated.

A PATCH ID of 18 indicates a selection time-out or an arm time-out. DOMCDC09 is called to process the time-out. It is evented and displayed if the action was display-initiated.

PATCH IDs 19 through 24 indicate which control action command the power system operator has chosen. The IDs are as follows:

19 - AUTOMATIC
20 - OPEN/TRIP/RAISE
21 - CLOSE/LOWER
22 - MANUAL
23 - TAG
24 - UNTAG

DOMCDC01 checks that a control action is in progress and calls DOMCDC07 to process the control action command. If the device has executed, the power system operator is informed by a message written to his display unit and the completion is evented. The device control table area is freed and the pointer to the device control table is cleared for display initiated requests.

DOMCDC01 also processes errors which are encountered by the other modules which are called. The return codes from the modules indicate which error has occurred and DOMCDC01 events, alarms, and cleans up the control indicators and areas based on the return code. It also either displays a message on display initiated requests or POSTs the user ECB to indicate which error occurred on macro initiated requests.

DOMCDC02

DOMOCDC02, running under the DOMXPDC task, is PATCHed by DOMCDC01 using the PTIME facility when a change of status is expected from the control action. If the module is activated (on execution time-out), DOMCDC01 is PATCHed with a PATCH ID of 16.

DOMCDC03

DOMCDC03, running under the DOMXPDC task, is PATCHed by DOMCDC01 using the PTIME facility to control the 30-second time-out of the arming phase and the 30-second time-out of the selection phase. If the module is activated (on arm time-out or on selection time-out) DOMCDC01 is PATCHed with a PATCH ID of 18.

DOMCDC04

DOMCDC04, running under the DOMXPCC task, is the interface between the application and the device control programs. The module determines whether there is a single device or unit to be controlled or several. In the latter case, the control actions are passed on to DOMCDC01 one at a time and the module waits until the ECB has been posted before passing on the next action. If the user has specified that the list is dependent and if one of the control actions fails, processing terminates with the failing control action, and the user ECB is posted with a return code indicating that the list was only partially executed. Entries up to the failing entry in the list are processed. If the list is independent, all entries in the list are processed regardless of the success or failure of the previous ones. If any of the entries fail to execute successfully, the user's ECB is posted with a return code indicating that one or more entries failed to execute. Regardless of the dependency of the list, the individual return codes associated with each action are incorporated into the list.

DOMCDC05

DOMCDC05 is called by DOMCDC06, DOMCDC07, and DOMCDC09 to send messages to the System/7 using the S7WRITE macro. If the action is display-initiated, it cancels the PTIME of DOMCDC03 for the arming phase if a response is not received to the arm request in a time period determined by the user.

The module does a GETITEM on the PDC options table (CAPDC) array to get the control information needed for the message. It then constructs the transaction code x'06' message to be sent to the System/7. The command or the verify or execution time-out information is contained in this message. The applicable part of the message is converted to ASCII using the ASCICONV macro and sent to the System/7. DOMCDC02 is PTIMEd at the sysgened rate - contained in the system generation options array (CASYS) - for the execution time-out interval. The executing indicator is turned on in the status item and the DCT item.

DOMCDC06

DOMCDC06 is called by DOMCDC01 to do the initial processing and error checking for both display and macro initiated control actions. For a display initiated action, the program checks the access area and function codes of the device against those of the display unit for a match. The control areas needed are initialized and the device is checked to ensure that it is not already being controlled, that it is controllable, and that it is in service. The options message is formatted to be displayed showing only those control action options which apply at the time. A selection time-out is also set, giving the power system operator 30 seconds to select a control action.

For a macro-initiated action, the same error checking is done except that the access area and function codes check is not applicable. In addition, DOMCDC11 is called to validate the control action. If the device is in a controllable state and the command is valid, DOMCDC05 is called to send a message to the System/7.

The selection of the device is evented in both display and macro initiated cases. DOMCDC06 also does some error processing. It returns to DOMCDC01 with a code indicating whether further processing is needed or not.

DOMCDC07

DOMCDC07 is called by DOMCDC01 to process the option selected by the power system operator or the execute request from the power system operator.

In the former case, it cancels the selection time-out PTIME of DOMCDC03 and checks that the option selected is valid for the type of device with which the power system operator is working. It then calls DOMCDC11 to further validate the option selected.

For the execute request, it calls DOMCDC05 to format and send the execute command to the System/7.

A return code is passed back to DOMCDC01 indicating whether any error processing is to be done.

DOMCDC08

DOMCDC08 is called by DOMCDC01 to process System/7 replies and the PDC information received in the RDA. The module has three possible inputs as follows:

1. transaction code x'16' - tag or untag command reply when the System/7 is not scanning.

2. transaction code x'86' - command reply for those devices which do not generate a change of status i.e. raise or lower command.

3. RDA information - command reply for all other types of commands and for verify or execute time-outs.

If the command reply received is for a device which is being controlled from a different CPU, the data base is updated with the new status of the device if the device is known in the data base. An entity is also issued or deleted when applicable using the DISPENT macro.

If the reply applies to a device being controlled from this CPU, the PTIME of DOMCDC02 for the execution time-out is deleted. DOMCDC11 is called to log the change of status. If an alarm is outstanding on the point being controlled, it is deleted using the DOMCALRM macro. If the point has the wallboard indicator on, DOMCWBPR is called to process the change. The entity is issued using the DISPENT macro.

If the reply indicates that the command failed or that it timed-out, an alarm is issued on the point.

If the command is macro-initiated, the user ECB is posted with a return code and a message number indicating the success or failure of the command.

A message indicating the result of the command is retrieved from the message file using the MESSAGE macro and passed back to DOMCDC01 along with a return code indicating what further processing is necessary.

DOMCDC09

DOMCDC09 is called by DOMCDC01 to process change of status received, cancel or display change requests, and all time-outs.

When the change of status is received, the PTIME of DOMCDC02 - execution time-out - is cancelled. The proper entity is issued using the DISPENT macro and, if an alarm is outstanding, it is deleted using the DOMCALRM macro. The wallboard processor, DOMCWBPR, is PATCHed if applicable. DOMCDC11 is called to PATCH the status change logger.

When the control action is cancelled or the display is changed, if the PTIME for DOMCDC03 - select or arm time-out - is outstanding, it is deleted. Indicators in the data base item are reset.

The execution time-out processing differentiates between a execute time-out and a verify time-out. The former indicates no change of status is expected and the latter indicates a change of status is expected. DOMCDC05 is called to send a message to the System/7.

The arm and select time-outs result in a message being retrieved using the MESSAGE macro which is passed back to DOMCDC01 for display.

A message and a return code are returned to DOMCDC01 indicating the results of the processing in this module.

DOMCDC10

DOMCDC10 is the routine to which the SCDEVICE macro links. A PATCH to DOMCDC04 is issued and the address of the parameter list generated by the macro is passed.

DOMCDC11

DOMCDC11 is called by DOMCDC06, DOMCDC07, DOMCDC08, and DOMCDC09 to validate the control action selected or to PATCH the change of status log processor.

If the flags indicate the control action is complete, a parameter list including the status item and the date and time of occurrence is created. DOMCSLOG, the status log processor, is then PATCHed passing the parameter list to it.

If the control action is not complete, the option selected is validated. The option is checked against the state of the device to insure the device is not already in the state requested. A type 3 tap-changing-transformer is checked to insure that it is in the manual state if any other option is selected. The device is checked for tag status; if tagged, the only valid option is untag. The device is verified to be controllable and in service.

Once the option is validated, the device is checked to see if it is a manual device. If it is, the control action is completed by altering the data base, issuing the appropriate entity using the DISPENT macro, deleting any outstanding alarm using the DOMCALRM macro, logging the status change, and PATCHing the wallboard processor (DCMCWBPR) if applicable.

If the device is not manual, the arm time-out is started by issuing a PTIME for DOMCDC03 at a 30-second interval. The armed indicator is set on in the DCT.

DOMCPDC1

DOMCPDC1 is a program that is called by one of two programs. One program is DOMTSSYN and the other program is DOMTCSES. Program DOMTCSES calls DOMCPDC1 in order that the proper status item bits may be set when changes of status occur and also that the appropriate entity will be changed and events issued. DOMCPDC1 is called by program DOMTSSYN to process the PDC portion of the Raw Data Array.

DOMCPDC1 is called by program DOMTCSES for hierarchy changes of status. The device type for which the Power Device Control (PDC) action is requested is determined. After the device has been determined, the appropriate status item executing bit is reset, the entity is changed, and an event is issued. The function of DOMCPDC1 is to analyze the PDC information in the raw data and take the appropriate action.

Program DOMCDC01 is PATCHed for control actions originating from the System/370 which are successful and the requested action is a tag/untag or an execute or verify timeout. Unsuccessful control action originating from the System/370 also causes program DOMCDC01 to be PATCHed with the address of the PDC information in the Raw Data Array.

For tag/untag successful actions where the controlling CPU is not the System/370, the status item tag bit is flipped, the entity is changed and an event is issued. Verify timeouts for non System/370 controlling CPU successful actions cause the status item executing indicator to be turned off. In all other cases when the PDC action is successful and the controlling CPU is not the System/370, the status item executing indicator is set.

There is one final case processed by program DOMCPDC1. Whenever an unsuccessful control action originates from a controlling CPU that is not the System/370 for an execute time out in the controlling CPU, the status item execute indicator is turned off.

DISPLAY OF SENSOR BASED DATA

Sensor-based data can be defined for local and remote control stations in the system. A separate display for each of the sensor based data types, pulse counter (PC), analog, and status is available for viewing upon request through a menu type display. This menu type display contains the remote control block information needed in order to request either of the three types of data for a desired control station.

After the desired data is selected for a specified station, the data is displayed as page one. Data can be changed and the incore data updated provided that both the display console and the RCB have the same access area ID and the display console and the data item have the same function code and the data item is out of service.

DOMDSRMT (Display of Remote Control Block)

This menu type display is provided for retrieval and display of sensor based data. The user can display page 1 of the desired data by placing the cursor on the cursor sensitive character on the line corresponding to the desired IBM 3707 and System/7 IDs of a remote control station and then pressing the cursor entry key.

When the cursor entry key is pressed, the data retrieval program receives a queue through a PATCH. By using the information read in from the display and the ID passed, these programs cause retrieval of the desired data from the resident data base and cause this data to be formatted and displayed on the appropriate display. Pulse Counter (PC) data, analog data, and status data for the desired remote control station are displayed using displays DOMDSPC2, DOMDSANL, DOMDSAN2, and DOMDSST2, respectively.

DOMCSGET

DOMCSGET, which is the control program for the RCB display DOMDSRMT, receives control through a PATCH from Display Management System. Upon receiving control, DOMCSGET performs different processing depending on the PATCH ID passed.

For a PATCH ID of 10, DOMCSGET initializes the display DOMDSRMT and its associated remote control element (RCE). Page one of the display is then built and displayed.

For a PATCH ID of 20 (page forward), DOMCSGET updates the display's remote control element pointers by searching forward through the list of RCB pointers to the desired remote control element pointer. The next page of the display is then built and displayed.

For a PATCH ID of 25 (page backward), DOMCSGET updates the remote control element pointers by searching backward through the list of RCB pointers to the desired remote control element pointer. The previous page of the display is then built and displayed.

For a PATCH ID of 40 (delete display), DOMCSGET frees the buffer area for the remote control element (RCE), and zeros the RCE address in the control element address table (CEATAB) if the new display is not a data base display.

DOMDSPC2 (Display of Pulse Counter Data)

These displays are provided for displaying retrieved current resident pulse counter data. They are initiated by the user selecting the pulse counter display from DOMDSRMT. Pages 1-N are displayed using display DOMDSPC2.

To update the resident pulse counter data base, the user takes the pulse counter point out of service, enters the new value(s) on the display, presses the DATA ENTRY display function key, and puts the pulse counter point back in service. The new values start at the tab that precedes the original displayed data. Display function keys or cursor positions are also provided to page the display forward and backward, to refresh a page currently being displayed, to call displays DOMDSST2 or DOMDSAN2, or to return to display DOMDSRMT.

DOMCSPCD (Control Program for Display DOMDSPC2)

This program formats and controls the display of all retrieved resident pulse counter sensor based data. It receives a queue through a PATCH from the counter data display DOMDSPC2, or the remote control block menu display DOMDSRMT. Upon receiving control, DOMCSPCD performs different processing depending on the PATCH ID passes.

For a PATCH ID of 10, DOMCSPCD initializes the display DOMDSPC2 and its associated remote control element, and page one is displayed.

For a PATCH ID of 15, DOMCSPCD initializes the display DOMDSPC2. The PATCH is received from display DOMDSAN2 or DOMDSST2, and page one is displayed.

For a PATCH ID of 20 (page forward), DOMCSPCD updates the display remote control element pointers based on the requested page number entered on the screen or on the calculated current page number. The requested page of the display is then built and displayed.

For a PATCH ID of 25 (page backward), DOMCSPCD updates the display remote control element pointers based on the requested page number entered on the screen or on the calculated current page number. The requested page of the display is then built and displayed.

When receiving a PATCH ID of 30 (refresh display data), DOMCSPCD gets the current incore counter data for the data currently being displayed. The display buffer is updated, and the display is rewritten.

For a PATCH ID of 35 (read display data), DOMCSPCD scans the display tabs. For any display tab that has been overlaid, the new data is read from the display into a storage area. If the access area ID(s) of the display console and the access area ID of the displayed RCB are the same and the RCB service bits are off and the function code of the display console and the function code of the data item agree, the pulse counter data base array is updated with the new values. When the display unit is designated as a power system operator unit only the service bits are evaluated before updating takes place. The display buffer is then updated and displayed. Module DOMCNCNV is branched and linked to in order to convert EBCDIC numbers entered on the screen into a single precision floating point number.

For a PATCH ID of 40 (delete display), DOMCSPCD frees the buffer area for the remote control element and zeros the control element address in the control element address table if the new display is not a data base display.

DOMDSAN2 and DOMDSANL (Display of Analog Data)

These displays are provided for displaying retrieved current resident analog data. They are initiated by the user placing the cursor on the cursor sensitive location provided on display DOMDSRMT.

Pages 1-N of the display contain the retrieved resident analog data (DOMDSAN2). Display DOMDSANL provides further detail of data displayed on the DOMDSAN2 display page and allows the user to update displayed values and to modify certain system generated options.

To update the resident data, the user places the point out of service, puts in the new value on the display, presses the DATA ENTRY display function key, and puts the point back in service. The new values must start at the tab that precedes the original display data. Display function keys and/or cursor positions are also provided to page the display forward and backward, to refresh the page currently being displayed, to call displays DOMDSPC2 or DOMDSST2, or return to display DOMDSRMT. A direct paging capability is provided that allows the user to enter the page number desired and then proceed with the normal paging entry methods to have the requested page displayed.

DOMCSANA (Control Program for Display DOMDSAN2)

This program formats and controls the display of all resident retrieved analog sensor based data. It receives a queue through a PATCH from either the sensor based data menu display (DOMDSRMT), from the analog data display DOMDSAN2, from status data display DOMDSST2, or from the pulse counter data display DOMDSPC2. Upon receiving control, DOMCSANA performs different processing depending on the PATCH ID passed.

For a PATCH ID of 10, DOMCSANA initializes the display DOMDSAN2 and its associated remote control element, concerning RCB data and the analog data is built and displayed.

For a PATCH ID of 15, DOMCSANA initializes the display DOMDSAN2. The PATCH is received from display DOMDSPC2 or DOMDSST2, and page one of the display is displayed.

For a PATCH ID of 20 (page forward), DOMCSANA updates the display remote control element pointers based on the calculated current page number or from the page number entered on the screen. The requested page of the display is then built and displayed.

For a PATCH ID of 25 (page backward), DOMCSANA updates the display remote control element pointers based on the calculated current page number or from the page number entered on the screen. The requested page of the display is then built and displayed.

When receiving a PATCH ID of 30 (refresh display data), DOMCSANA gets the current resident analog data and updates the display buffer with this data. The display buffer is then displayed.

For a PATCH ID of 35 (read display data), DOMCSANA scans the display tabs. For any display tab that has been overlaid, the new data is read from the display into a storage area. If the access area ID(s) of the display console, the access area ID of the displayed RCB are the same, the RCB service bits are off, the function code of the item being updated is the same as one of the display function codes, the data base from where the original values came will be updated with the new values. The display buffer will then be updated and displayed. Only numeric data will be used to update the data base. When the display unit is designated as a power system operator's terminal only the service bits are evaluated before updating takes place. Module DOMCVCNV is branched and linked to in order to convert EBCDIC numbers entered on the screen into a single precision floating point number.

For a PATCH ID of 40 (delete display), DOMCSANA frees the GETMAINed area for the display buffer and its control element, and zeros the control element address in the Control Element Address Table if the new display is not a data base display.

DOMCSANL (Control Program for Display DOMDSANL)

This program formats and controls data displayed on display DOMDSAN1, detail of the data items on display DOMDSAN2. A queue is received through a PATCH from analog data display DOMDSAN2 or DOMDSANL.

The PATCHes to program DOMCSANL are received from two sources. One source is display DOMDSAN2 and the second source is display DOMDSANL. PATCH ID's of 2 and 4 are received from display DOMDSAN2 and are requests to display detail of the first or second eight analog data items from the display DOMDSAN2. Display DOMDSANL may also request the displaying of detail for the first or second eight analog items, on the previously viewed analog display DOMDSAN2, with PATCH ID's of 6 and 8 respectively. Upon receiving control, DOMCSANL performs different processing depending on the PATCH ID passed (see Figure 1).

For a PATCH ID of 2 or 4 (PATCHes from DOMDSAN2 display), the item numbers and item names are stored in main storage as they are received from the partial screen read.

For a PATCH ID of 2 or 6 (display detail for first eight items from DOMDSAN2 display), DOMCSANL retrieves the detail analog data from the data base. A page of the display is then built and displayed.

For a PATCH ID of 4 or 8 (display detail for the second eight items from the DOMDSAN2 display), DOMCSANL retrieves the detail analog data from the data base. A page of the display is then built and displayed.

ID	Action	Source of PATCH
2	Display detail of first eight items from display DOMDSAN2	Display DOMDSAN2
4	Display detail of second eight items from display DOMDSAN2	Display DOMDSANL
6	Display detail of first eight items from display DOMDSAN2	Display DOMDSANL
8	Display detail of second eight items from display DOMDSAN2	Display DOMDSANL
10	Display DOMDSANL replaced on screen-cancel processing	Display Management
12	Update data base from manually entered values on the screen	Display DOMDSANL
14	Modify system generation defined option of alarmable/not alarmable	Display DOMDSANL
16	Modify system generation defined option of user conversion/new user conversion	Display DOMDSANL

Figure 1. DOMCSANL PATCH IDs

For a PATCH ID of 10 (delete display), DOMCSANL frees the GETMAINed area for the remote control element and zeros the control element address in the control element address table if the new display is not a data base display.

For a PATCH ID of 12 (read display data), DOMCSANL scans the display tabs. For any display tab that has been overlaid, the new data is read from the display into a storage area. If the access area ID(s) of the display console, the access area ID of the displayed RCB are the same, the RCB service bits are off, the function code of the display console and the function code of the data item agree, the analog data base array is updated with the new values. The display buffer is then updated and displayed.

For a PATCH ID of 14 or 16 (update alarmable and user conversion status conditions, respectively), DOMCSANL updates the data base by inverting the indicated condition and then redisplays the updated status conditions. Before an update is allowed, the access area of the displayed RCB must agree with one of the access areas of the display console, the function code of the item being modified must agree with one of the function codes of the display console, and the item must be either out of service self or other. When the display unit is a master power system operator unit only the out of service bits are checked before updating takes place.

DOMDSST2 (Display of Status Data)

These displays are provided for displaying retrieved current resident status data. They are initiated by the user selecting the status display from menu display DOMDSRMT.

The user has the capability to page the display forward and backward, to refresh the page currently being displayed, to modify certain system generated options, to call displays DOMDSAN2 or DOMDSPC2, or to return to display DOMDSRMT. The user also has the ability to page directly to any given page of status data.

DOMCSSTA (Control Program for Display DOMDSST2)

This program formats and controls the display of all resident status sensor based data. It receives a queue through a PATCH from either the sensor based data menu display (DOMDSRMT), from the pulse counter data display DOMDSPC2, or from the analog data display, DOMDSAN2. Upon receiving control, DOMCSSTA processes depending on the PATCH ID passed.

For a PATCH ID of 10, DOMCSSTA initializes the display DOMDSST2 and its associated control element, and page one of the display is built and displayed.

For a PATCH ID of 15, DOMCSSTA initializes the display DOMDSSTA. The PATCH is received from display DOMDSAN2 or DOMDSPC2, then page one of the display is displayed.

For a PATCH ID of 20 (page forward), DOMCSSTA updates the display's control element pointers based on the current page number which has been calculated. The next page of the display is then built and displayed.

For a PATCH ID of 25 (page backward), DOMCSSTA updates the display's control element pointers based on the current page number which has been calculated. The previous page of the display is then built and displayed.

When receiving a PATCH ID of 30 (refresh display data), DOMCSSTA gets the current incore RCB and status data and updates the display buffer with this data. The display is then rewritten.

For a PATCH ID of 35 (reverse alarmability bit setting) DOMCSSTA updates the data base by inverting the indicated condition and then redisplays the updated status conditions. Before an update is allowed, the access area of the displayed RCB must agree with one of the access areas of the display console, the function code of the item being modified must agree with one of the function codes of the display console, and the item must be either out of service self or other. When the display unit is a master power system operator unit only the out of service bits are checked before updating takes place.

For a PATCH ID of 36, control is passed to the program DOMCTLCM, which builds a CTL (transfer of control) command. When control is returned, the display refreshed and any scratch pad messages generated by the DOMCTLCM program are displayed.

For a PATCH ID of 40 (delete display), DOMCSSTA frees the GETMAINed area for the display buffer and its control element, and zeros the control element address in the control element address table if the new display is not a data base display.

DOMCTLCM

This CSECT receives control from one of three sources. The first is the routing module for System/7 transaction codes received, the second is from the status data display (DOMDSST2) control program, and the third is from itself (DOMCTLCM). The program has two major processing sections. The first section processes transaction code 08, transfer of control (CTL) messages received from the System/7 (PATCH ID 2), and simulates CTL commands built by the second section of the program (PATCH ID 1). The second major processing section builds CTL commands from manual inputs entered on the status data display (DOMDSST2).

Processing of CTL transaction code messages is done by comparing the "to" CPU ID in the command to the front end System/7 CPU ID for the System/370. When the two CPU IDs are the same, control is transfered to the device specified in the command. When the IDs are different, the not controllable bit is turned on if it is not already on. Either occurrence causes a data event to be generated. After the above processing takes place the command is routed back to the destination CPU, if the System/370 is at the top of the hierarchy.

The building of CTL commands is initiated from the status data display. The operator enters the device name being transferred, the CPU ID to receive control and the CPU ID (System/7) to which the device is attached. A CTL command is generated only when the display unit attempting to transfer control is at the top of the hierarchy or if the device (item) currently has control and the access area/function codes of the display unit agree with the device access area/function codes. If transfer of control is allowed from the requesting display unit a command is built and shipped. When the System/370 is at the top of the hierarchy, the CSECT DOMCTLCM is PATCHed with an ID of 1, and the command is passed as a PATCH parameter. The command is routed via the System/7(s) to the top of the hierarchy when the System/370 is not at the top.

DOMCNCNV (Convert EBCDIC Numbers to Floating Point or Fixed Point)

This program receives control with general purpose registers zero and one containing information required for processing. Register zero is expected to contain the length of the EBCDIC number to be converted in the high order byte. The low order three bytes are expected to contain the address of the area to receive the converted number. Register one is expected to contain a flag in the high order byte indicating the type of conversion desired. A hexadecimal '04' indicates the number is to be converted to a single precision floating point number and a hexadecimal '00' indicates the number is to be converted to a fixed point number. The low order three bytes of register one contain the address of the number to be converted.

Register fifteen contains a return code upon completion of the conversion program. A zero indicates successful completion; a return code of four indicates that the EBDCIC number was larger than the maximum single precision floating point number; and a return code of eight indicates that the EBCDIC number contained invalid characters.

When the conversion program is used to convert a number to floating point, a double word storage area is required to receive the converted number.

EVENTS PROCESSING

Events Logging

Events logging is handled by a set of modules which allow the user to event a system action by using the SCEVENT macro. Events are written to the event log file, to a typer assigned the same access area and function code, and to a general event typer.

DOMCEVT1: The event logging module (DOMCEVT1) executes under the independent task, DOMXALRM, PATCHed by DOMCEVNT with a PATCH ID of 2 to record a new event.

The current logging position is retrieved from the events logging control table, and the record is added sequentially to the event log file. The oldest event record is overlaid with a new event. When the physical end of the data set is reached, the current logging position is set back to the beginning of the data set.

The new event is logged to the event printer assigned to the same access area and function code and to the general event printer using the message handler facility. If, in the specified access area, there is no printer for the function code, the event is logged to a default printer assigned at system generation. If the access area is not specified, the event is logged only to the general printer.

DOMCEVT5: This module has the same function as DOMCEVNT and DOMCEVT1 but events more than one event each time it is PATCHed. DOMCEVTS is PATCHed each time the alarm macro is issued and when the scan processor encounters alarms. DOMCEVT5 is PATCHed by the alarm program with a parameter list with all the alarms and access area/function codes and they are evented in a single cycle of the program. This is a special program and should not be PATCHed by the user.

Events Log Display

The power system operator can view any event that occurs within the system. When the power system operator requests the events log display, the program DOMCEVD1 is PATCHed to obtain the retrieval parameters and request retrieval.

DOMCEVD1: This load module determines if the events log display is currently being viewed on the requesting unit. If the events log display is currently being viewed and it is automatically updating, then the automatic update is stopped. The old event control element (ECE) address from the CEATAB is saved. A new ECE and page save area, PSAR, are obtained and chained. If the events log display is not currently being viewed then the default parameters are assumed, also the parameters are read and verified. When all parameters are verified the events log display is requested, if not currently being viewed, and the events log display program, DOMCEVD2, is PATCHed with an initial request.

DOMCEVD2: This load module processes initial, paging and refresh requests for the events log display. The input PATCH ID determines the function of the work queue.

A PATCH ID of 1 indicates an initial request. If the retrieval time is blank a PTIME is issued to automatically update the display at the sysgen defined rate. The time an event is logged, record key, the time used to retrieve events. If a time was specified in the retrieval request causes the determination of the starting point for retrieval by attempting to read a record with the requested time. If no such record is found a binary search is done to determine the record with the key closest to requested time. The key of the starting point simulates the last event on the page. A page forward is indicated and the change page segment is called.

If no time was specified in the retrieval request then the oldest record in the file simulates the only event on the page. A page backward is indicated and the change page segment is called.

A PATCH ID of 2 is used to indicate a page forward mode. If a PTIME is outstanding, it is cancelled. The indicator in the ECE is set to page forward, and the change page routine is called.

A PATCH ID of 3 is used to indicate a page backward mode. If a PTIME is outstanding, it is cancelled. The indicator in the ECE is set to page backward and the change routine is called.

A PATCH ID of 4 is used when the display is changed. An outstanding PTIME is cancelled and the ECE and page save areas freed. The pointer in the CEATAB is also cleared.

A PATCH ID greater than 4 is used by the PTIME so that each different PTIME will have a unique PATCH ID for cancel purposes. The ECE contains the PTIME PATCH ID being used.

The change page routine will read the file either forward or backward as requested. If paging backwards and the first event on the page is the newest event in the file, then start the search with the newest event, else back up one event and start the search. The index records are searched for access and function area and type to determine if the record qualifies. A list of up to 16 events that satisfy the access, function and type requirements is generated in the ECE.

The records that fulfill the power system operator's selections are displayed and the relevant information is stored in the page save area; the remainder of the events display zone is blanked. The first and last event in this page field in the ECE are updated.

If a match for an event is not found before the logical end of the file is reached, a message is displayed and a wraparound occurs on the file. If a match does not exist, a message is displayed after all the logical file is searched.

The logical file is defined as those events in the file which do not fall in the dead zone. The dead zone size is defined at sysgen time to be x number of event records. The dead zone is defined as the x oldest events in the file. Events in the dead zone are never displayed, new events are added here.

DOMCEVD4: When the power system operator enters a comment for a particular event, the MIAT entry causes DOMCEVD4 to be PATCHed under the independent task DOMXEMD1. If a PTIME is outstanding it is deleted and the workqueue purged of any queued PTIME elements. If the access and function areas are valid for updating events then the event number and comment are read. The event number is checked to insure that the event is currently displayed on the screen. The event file is then locked, the requested event read and verified that it has not been overlayed. The comment is then updated to the file and the file unlocked. The access area and general printers are then updated and the comment written to the display. All requests result in some message to the power system operator, either an error message or a successful update message. If the update was successful, then the comment entry line is cleared else it remains for the dispatcher to determine the error.

SCAN CONTROL DISPLAY

The scan control module (DOMCAND1) provides the means of altering the status of predefined scans and viewing the entire scan list on a display unit. The scan control display lists the scans defined for the system and displays the scan ID, the frequency, the offset (if any), the status (active or inactive), and the type (initial, emergency and normal) for each scan. The display also lists the present scanning mode and the basic scan cycle. The status of one or more scans may be changed by entering the new active/inactive status and depressing the data entry function key from the power system operator's terminal at the top of the hierarchy. The scanning mode may also be changed by entering the new mode and depressing the data entry function key. The scan control display processor then notifies data acquisition of the change(s) through the VARYSCAN macro and data acquisition communicates the change(s) to the appropriate System/7s.

DOMCAND1

DOMCAND1 is PATCHed by an independent task, DOMXEMD3, which builds the scan control display and reads the input from the screen to determine which scans are to be changed. If the information to be displayed exceeds the capacity of the screen, forward and backward paging through the list is handled. The various functions of the scan control module are identified by PATCH ID as follows:

- A PATCH ID of 2 initializes scan control. Using the GETWA facility, areas for the scan control element (SCE) and the display buffer are allocated. The address of the SCE is saved in the control element address table (CEATAB) associated with the display unit. The scan control display is built in the buffer and the first and last scan IDs in the buffer are saved in the SCE for paging control. The display is then written from the buffer.

- A PATCH ID of 4 indicates that the user has entered a change of status for one or more scans. The change information is read from the display buffer to determine if a scanning mode change is required and to determine which scans are to be altered. The VARYSCAN macro is used to inform data acquisition of the change(s).

- A PATCH ID of 6 indicates a display change. The SCE and display buffer areas are freed, and the SCE address in CEATAB is set to zero.

- A PATCH ID of 8 indicates forward paging. Using the last scan ID in the SCE, the next page of the list is built in the buffer and the new first and last scan IDs are saved in the SCE. The display is then updated.

- A PATCH ID of 10 indicates backward paging. Using the first scan ID in the SCE, the previous page of the scan list is built in the buffer and the new scan IDs are saved in the SCE. The display is then updated.

- A PATCH ID of 12 indicates a return communication message from the System/7 as a result of a VARYSCAN command. The buffer containing the scan change information is both saved and released. The event that indicates the mode or scan change as well as the success or failure or the action is formulated and is issued. The change to the scan control array is logged.

AGC OUTPUT INTERFACE

At the conclusion of each execution of the AGC program, the AGC module DOMAAGCO converts unit corrections for each generator from MW to pulse duration. It locks array AAAGCPDO and fills it with an entry of the following form for each generator. Each entry contains the device name of the generator followed by a signed pulse length in 100 millisecond units, followed by the destination code of the System/7 associated with the generator. After filling in the array, DOMAAGCO unlocks the array and calls the AGC output interface processor DOMCGENO.

Through use of the S7WRITE macro, DOMCGENO transfers these pulse durations to the appropriate System/7 to control the generators. For each name and value in AAAGCPDO, DOMCGENO groups all entries with the same System/7 ID and places these in a buffer to be sent to the proper System/7.

The capability to terminate or activate output from DOMCGENO is provided by PATCHing the routine DOMCAGCK with the correct PATCH ID.

A PATCH ID of 1 causes the output to be terminated and a PATCH ID of 2 causes the output to be activated. This program can be queued by whatever means desired as long as the input is set up as the PATCH issued through the Display Management functions.

Upon receiving control, DOMCAGCK sets a bit to terminate or activate AGC output. In addition to this, a system event is issued and a message is written to the system message zone of the input unit ID indicating what action has been taken.

STRIPCHART PROCESSING

The user may define stripchart recorders which are controllable by stripchart processing in the System/370. The recorders may be controlled from the power system operator terminal or they may be controlled by using the stripchart macro (DOMCSCHT) via user provided programs.

The stripchart processor - DOMCHRTA - receives information from the display and from the macro, analyzes the information, and notifies the System/7 of the request to turn a recorder on or off. It then notifies the requestor of the success or failure of the stripchart command. While any stripchart recorders are active, a cyclic program - DOMCHRTC - sends the most current raw data values for those points which are being charted to the System/7. If all recorders are inactive, this cyclic processing terminates.

The stripchart display processor controls the building, refresh and data entry functions of the display.

The programs which accomplish these functions are as follows:

DOMCHRTA

DOMCHRTA, running under the DOMXUTIL task, is PATCHed by DOMTCHRT - the macro interface. Its function is to analyze the macro parameter list and notify the System/7 of the change. It also receives the System/7 reply and handles the time-out if one occurs. The PATCH ID determines what processing is to be done as follows:

PATCH ID of 1: This is a change request. The program analyzes the input. It checks that the recorder ID is a valid one, that the action is appropriate to the current status of the recorder, that the point name belongs to a valid analog or pulse counter point, that the A scale factor and the B scale factor are within the allowed range (1 to 32,767 and 0 to 32,767 respectively), and that the time mark option is valid.

After the input is validated, a stripchart command message is built (transaction code 08) and sent to the System/7 using the S7WRITE macro. A PTIME is then issued for ten (10) seconds with a PATCH ID of 3 or 4 on itself to time-out the reply from the System/7.

If errors are encountered in processing, the macro ECB is posted with an appropriate return code.

PATCH ID of 2: This System/7 reply (transaction code - 88) was received. The PTIME for the time-out is deleted and the reply is analyzed. If the reply indicates the command to turn on the recorder was successful, the active indicator is set on in the CASCHART array entry for the appropriate recorder. If the cyclic program is not being PTIMEd, DOMCHRTC is PTIMEd at the basic scan cycle rate. If the command was to turn off the recorder and was successful, the entry in the array is zeroed; and, if no other recorders are active, the cyclic program is deactivated using the PTIME macro. The macro ECB is posted with a zero return code. If the reply indicates the command was not successful, the macro ECB is posted with an appropriate return code.

The reply is evented whether it was successful or not.

PATCH IDs of 3 and 4: A time-out occurred because the System/7 reply was not received within ten seconds. The macro ECB is posted and an event is issued.

DOMCHRTC

DOMCHRTC, is a cyclic program running under the DOMXPDC task. It is PTIMEd from DOMCHRTA to activate and deactivate it. It runs at the basic scan cycle rate which is defined by the user at system generation.

This program uses the CASCHART array to determine which recorders are active. The most current values for those points which are being recorded are converted back to raw data values and included in a table of sixteen entries. This table is sent to the System/7 as part of the transaction code x'13' message and is used by the System/7 to update the DARS370 table used in the stripchart process.

DOMTCHRT

DOMTCHRT, a part of the DOMTABLE load module, is the macro interface processor for the stripchart macro. It is called by the DOMCSCHT macro. It PATCHes DOMCHRTA with a PATCH ID of 1 passing the macro parameter list. It waits on the PATCH ECB. If the return code is zero, it waits on a second ECB for processing to be complete. It returns a return code to the macro user in register 15 indicating the success or failure of the stripchart request.

DOMCHART

DOMCHART, running under the DOMXEMD3 display task, controls the stripchart display. It is PATCHed by Display Management System when the master power system operator requests the stripchart display. DOMCHRT1 is called to build and update the display and DOMCHRT2 is called to process the changes requested. The PATCH ID determines the processing to be done.

PATCH IDs of 1 and 2. These are initial build and refresh respectively. If initial, the DLITES macro is used to turn the applicable function key back lights on and the others off. DOMCHRT1 is called to format and update the display.

PATCH ID of 3 - Data Entry. The access area and function of the terminal at the top of the hierarchy are compared to those of the display unit. If a match is not found, an error message is displayed in the scratch pad zone. Otherwise, DOMCHRT2 is called to process the change requests. Then, DOMCHRT1 is called to refresh the display.

DOMCHRT1

DOMCHRT1 is called by DOMCHART to format and update the stripchart display. The program uses the CASCHART array to format the display. If the PATCH ID is 3 (data entry), it also updates an area indicating whether or not any errors were encountered in processing.

DOMCHRT2

DOMCHRT2 is called by DOMCHART to process the stripchart change requests. The data entry area of the display was read and passed to this program. This information is analyzed to determine if the data entered is valid. The DOMCSCHT macro is issued for each change that is valid. If invalid inputs are found, flags are set so that the errors will be properly displayed when the display is refreshed.

WALLBOARD PROCESSING

The wallboard processing programs provide automatic wallboard support. DOMCWBIN initializes arrays and resolves addresses for use by the wallboard processing program DOMCWBPR. The wallboard processing program generates commands that are used by the System/7 to indicate the current status of selected points in the data base on the wallboard. A final System/370/System/7 input/output interface program, DOMCWBS7, communicates with the System/7 controlling the wallboard. Command lists are sent to the System/7 at regulated intervals. If the wallboard controller System/7 ID in the sysgen options array is zero it indicates that no wallboard exists and programs DOMCWBIN, DOMCWBPR, and DOMCWBS7 will not be link edited into the system.

DOMCWBIN

DOMCWBIN, running as a dependent task, executes from a call by DOMTRESI during phase II initialization. Addresses for arrays CAWBNAME and CAWBCONF are resolved and placed in the EMSCVT. If a manual/automatic wallboard switch has been defined through system generation, the name is resolved to an address and stored in the CAWBCONF array. Item names in the CAWBNAME array are resolved to addresses and stored in the CAWBNAME array. Error messages indicate if the arrays are not locatable. Any item name which cannot be located successfully is deleted from the array and a message is generated indicating the item name(s) that could not be located.

DOMCWBPR

DOMCWBPR runs under the independent task name DOMXPCC. Control is received via a PATCH from data acquisition, alarm control, alarm display control, power device control, or scan control. Data acquisition will PATCH DOMCWBPR with an ID of four when an initial scan is received from the System/7 controlling the wallboard. A PATCH ID of four indicates that initialization processing is to be performed. If the wallboard has been placed from manual mode, into automatic mode then initialization processing will also take place.

All other sources of control generate a PATCH with an ID of 8 and include a parameter list consisting of data base item addresses which represent wallboard defined points that have undergone a change indicating that wallboard processing is required. Status changes, alarms, alarm acknowledgements, and alarm deletions are conditions which cause the wallboard processor to receive control from the alarm processor, alarm display controller, scan control, or power device control. No updates are made to the wallboard for a point(s) that is offline.

Processing continues only if the wallboard is in the automatic update mode and the System/7 which controls the wallboard is active and scanning. Initialization processing involves branching to a subroutine to process the configuration number in the CAWBNAME array through one of the ten predefined configurations. The configuration number is associated with each item in the CAWBNAME array and is used to determine which subroutine will be branched to. Each item in the CAWBNAME array is processed in this manner until all entries have been processed. Completion of the last entry causes the final command list generated in the configuration subroutines to be sent to the System/370 wallboard input output interface program DOMCWBIN.

Cyclic processing (not initialization) involves receiving a parameter list with a PATCH ID of 8 which contains the address of items in the data base that are wallboard items. These wallboard items have a new condition that is to be reflected on the wallboard. A search is made of the CAWBNAME array to find a match for the item address in the paramater list. After an item is matched the configuration number associated with the array entry determines which of the ten subroutines will process the item into a command list for the System/7. When the last item in the parameter list has been processed the command list is sent to the System/370-System/7 wallboard input/output interface program DOMCWBS7.

The ten subroutines for processing the ten predefined configurations are divided into two major groups. The first group, configurations one through five are used to process status points (digital input - DI) while the second group is used to process analog points (analog input - AI). Each subroutine is internal to DOMCWBPR. Processing for each subroutine involves placing the wallboard lamp addresses in a temporary hold area, determining the state of the item and then adding the correct predefined command states to the lamp addresses, and then calling the command list processor subroutine. The command states are maintained in array CAWBCONF.

The command list processor subroutine retrieves a buffer to hold the command list and transaction code header. Header information and flag bytes are inserted in the header and command list text. The command(s) and lamp address(es) are placed in the command list buffer. The command list buffer is sent to the System/370 System/7 input output wallboard interface program DOMCWBS7 when the buffer is full and cannot accept any new commands. Otherwise processing continues on the command list.

Program DOMCWBPR outputs a message and an event whenever the wallboard is placed in the manual mode (DPP471I) or in the automatic mode (DPP470I).

DOMCWBS7

This program provides input and output interface for wallboard processing between the System/370 and the System/7 controlling the wallboard. Command lists are received from the wallboard processor DOMCWBPR and sent to the System/7 upon demand of the System/7 if the command lists are initialization command lists. Otherwise the command lists are sent directly upon receipt to the System/7. All command lists sent to the System/7 have a transaction code of fourteen. During the initialization of the wallboard all messages are chained together and after the first transaction code X'14' message is shipped to the System/7 a return message of transaction code X'94' must be received prior to the next transaction code fourteen message being sent.

A message (DPP378I) is written and an event is generated upon completion of the initialization of the wallboard. No transaction code X'14' command list messages are sent to the system seven unless the System/7 controlling the wallboard is active and scanning. Before each transaction code X'14' message is sent to the System/7, the transaction code number, flag byte, destination System/7 ID, origin System/370 ID and an end flag are inserted in the buffer. When the transaction code X'14' message being shipped to the System/7 is finished, the command list buffer is freed. Program DOMCWBS7 runs under the DOMXUTIL task.

POWER CONFIGURATION CONTROL

The power configuration control modules, DOMCFGD1 and DOMCFGD2, provide a user with a means of viewing the power configuration structure of the system and of varying the status of the System/7s, IBM 3707s, and data points. Through the power configuration control displays, the user enters the information to change the status of the system configuration. Supervisory control analyzes the information entered and interfaces with data acquisition to pass the information to the appropriate System/7.

There are three displays generated for the power configuration control function: Power Configuration Control I, II, and III. The first display DOMDPCC1, lists the System/7s and provides information about their status. The user keys in the changes to the System/7s or requests the second display DOMDPCC2.

PCC II lists the IBM 3707s which are associated with each System/7 logical ID and their status. The user may enter changes to the status or request to see the previous display, PCC I, or the third display DOMDPCC3.

PCC III lists the data points which are associated with a specific local, manual, summation or IBM 3707 and their status. The user may change the status of a data point and may also return to either PCC I or PCC II or another display.

Both PCC II and PCC III have a forward and backward paging feature to allow for the number of IBM 3707s or the number of data points exceeding the capacity of the common zone of the display. Another feature is that the page being viewed can be refreshed. PCC III also has a direct paging feature which allows the user to page forward or backward directly to the desired page rather than one page at a time.

DOMCFGD1

DOMCFGD1 handles the creation of the displays, the paging, and the refresh features. It runs under the DOMXEMD3 task and is PATCHed by Display Management. The PATCH IDs determine which function the user is requesting, and are as follows:

- A PATCH ID of 4 causes the first display (DOMDPCC1) to be built. If a configuration control element (CCE) does not previously exist, one is built, and a buffer area is acquired. Using the System/7 communications table, the display is created and written to the screen. The address of the CCE is saved in the control element address table (CEATAB). DOMCBLD1 is called to format the display and write it to the screen.

- A PATCH ID of 8 causes the second display (DOMDPCC2) to be built. A buffer area is acquired. Using the System/7 control table, System/7 communications table, and the remote control blocks (RCB), the display is created and written to the screen. DOMCBLD2 is called to format the display and write it to the screen.

- A PATCH ID of 12 causes the third display (DOMDPCC3) to be built. A buffer area is acquired. Using the RCB and the status, pulse counter, and analog data points, the display is created and written to the screen. DOMCBLD3 is called to format the display and write it to the screen.

- A PATCH ID of 16 indicates a page forward request for PCC II or PCC III. The next IBM 3707 or data point is found and the next page of the display is written to the screen. If the last page of the display is up, the program displays the first page. DOMCBLD2 or DOMCBLD3 is called to format the display and write it to the screen. If direct paging is indicated, the selected page is written to the screen for DOMDPCC3.

- A PATCH ID of 20 indicates a backward paging request for PCC II or PCC III. The program determines the start of the previous page's data and writes the previous page to the screen. If the first page of the data is up, the program displays the last page. DOMCBLD2 or DOMCBLD3 is called to format the display and write it to the screen. If direct paging is indicated, the selected page is written to the screen for DOMDPCC3.

- A PATCH ID of 24 indicates a refresh request for one of the three displays. The same page is rebuilt and displayed. DOMCBLD1, DOMCBLD2, or DOMCBLD3 is called to format the display and write it.

- A PATCH ID of 28 indicates a display change. A flag in the CCE is checked to determine whether the power system operator is viewing a different display in the PCC group or has changed to a different function. In the latter case, the areas used by the PCC displays are freed. In the former case, none of the areas are freed.

DOMCBLD1

DOMCBLD1 is called by DOMCFGD1 to format and write the DOMDPCC1 display. The System/7 Communications table is used to pick up the information. This module handles refresh of the PCC I display as well as the initial writing of it.

DOMCBLD2

DOMCBLD2 is called by DOMCFGD1 to format and write the DOMDPCC2 display. The System/7 Control table, the System/7 Communications table and the PCB array are used. Refresh and forward and backward paging are also handled by this module.

DOMCBLD3

DOMCBLD3 is called by DOMCFGD1 to format and write the DOMDPCC3 display. The applicable RCB item and the analog, pulse counter, and status arrays are used. Refresh and forward and backward paging are also handled by this module.

DOMCFGD2

DOMCFGD2 is a dependent task PATCHed by Display Management System when the user enters a status change request from PCC I, PCC II or PCC III. The PATCH ID determines from which display the request originated.

- A PATCH ID of 2, 3 or 4 indicates a change request to a System/7 from PCC I. The unit number and System/7 ID are passed to the program by Display Management System in the parameter list. The VARYS7 macro is issued to vary the System/7 to offline, backup, or primary according to the PATCH ID used (2, 3 or 4, respectively). This macro notifies data acquisition of the requested change.

- A PATCH ID of 7 or 8 indicates a change request to an IBM 3707 from PCC II. The IBM 3707 name passed to the program by Display Management is used to issue a VARYCONF macro that will vary the IBM 3707 in service or out of service according to the PATCH ID used (7 or 8, respectively). A message indicating the success or failure of the request is displayed.

- A PATCH ID of 11 or 12 indicates a change request to a data point from PCC III. The VARYCONF macro is issued to vary the data point in-service or out-of-service according to the PATCH ID (11 or 12, respectively). A message is displayed indicating the success or failure of the request.

DOMCTIME

Program DOMCTIME is PATCHed when program function key 21 is pressed and the Power Configuration Control I (PCC 1) Display is on the display screen. DOMCTIME insures that the requesting system only has one primary System/7. DOMCTIME also verifies that the requestor is at the top of the hierarchy and that the request is from a display unit with either the primary, secondary or tertiary power system operator access area and function code. An error message is written to the requesting display unit scratch pad zone if any of the above conditions is not satisfied.

Program DOMCTIME next checks the values input in the increment and decrement fields on the PCC I display. A value must only be entered in one of the fields. The value entered in the increment or decrement field must be between 0 and 99. If the value input to DOMCTIME fails to satisfy the above conditions, an error message is written to the scratch pad zone of the requesting display unit.

A transaction code '08' is formatted by DOMCTIME for valid time correction requests. Once the transaction has been formatted by DOMCTIME, the transaction is converted to ASCII by the macro ASCICONV. The transaction code '08' is then sent to the System/7 using the S7WRITE macro in order to satisfy the time correction request.

INCORE/LOGGED DATA BASE RETRIEVAL/UPDATE

The data base is used to monitor and control alarms, devices, scans, and the power configuration. The logged data base is also used to initialize the incore data base during a warm start to the System/370 Energy Management System. It houses all sensor based conversions and limit data for the entire system. It is most likely to be used by user application programs. It is for this reason that supervisory control provides an access method that supports retrieval of control block data (remote control block (RCB), analog, names list, pulse counter, and status) from the incore or logged history data bases, and update of analog data and pulse counter data in the incore or logged history data base.

The data base access method receives requests through the DOMCLGET and DOMCLPUT macros. These macros are available in standard, execute, and list forms. The DOMCFREE macro is used by the data base access method to free a buffer area retrieved through the DOMCLGET macro. It is available in standard form only.

The DOMCLGET macro is the vehicle through which data can be retrieved. DOMCLGET provides the user with the option of specifying a request for data from the most current incore data base to any time frame existing within the logged data base. A feedback (FEEDBCK) block is constructed in addition to retrieving data for logged history retrievals. It allows DOMCLGET to store locator and array header identification information for a request which is to be used by the DOMCLPUT macro in locating and replacing data on the logged history data base.

DOMCLPUT is the vehicle through which data, retrieved through DOMCLGET, is replaced in the incore or logged data base after being updated. This macro requires that the feedback (FEEDBCK) block returned by DOMCLGET for logged history data base data be provided to DOMCLPUT. The data block being replaced must be an exact duplicate in size, position, and type as the replacing data block.

DOMCFREE is the vehicle through which main storage, used to pass retrieved data, can be released. It is the user's responsibility to release the main storage area upon completion of its use by issuing the DOMCFREE macro.

DOMCLRMT

DOMCLRMT is a control program linked to when the DOMCLGET or DOMCLPUT macros are executed. The program is divided into a retrieval and an update section both of which are further divided into incore and logged history retrieval and update section. Each retrieval section (incore and logged history) is divided further into four subsections for retrieving remote control block data, analog and names list data, pulse counter data, and status data. Each update section (incore and logged history) is subdivided into two subsections. One is analog data update and the second is the pulse counter data update. The macro ID passed to this control program DOMCLRMT determines which routine is to be branched to. The macro ID is one byte and the bit settings as indicated below represent all of the valid combinations.

0000	Incore retrieval
0010	Logged history retrieval
0100	Incore update
0110	Logged history update
.... 0000	RCB data
.... 0010	Analog data/names list data
.... 0100	Pulse counter data
.... 0110	Status data

DOMCLFRE

DOMCLFRE is a control program linked to when the DOMCFREE macro is executed. The program frees main storage based on the address passed in general register one from the DOMCFREE macro. A return code is returned in general register 15 to the calling program indicating the success or failure of freeing the main storage area.

ONLINE DIAGNOSTICS

The online diagnostics display is for use by the power system operator who wishes to test the operability of any of the system typers. The power system operator selects the desired typer(s) specifying either routing code, or access and function areas plus the type of typer desired, events alarms or both.

The online diagnostics display is called to the screen by requesting display name DOMDDIGF using the sign on display, or by using the appropriate function key specified on the System/370 Menu Display.

DOMCDIG1

This program is PATCHed through Display Management after the operator chooses the options to be used in the online diagnostics display.

If a routing code is specified the message is sent to this routing code regardless of what the other fields contain.

If general typer was specified, the message is sent to both the general typers (Event and Alarm).

If event or alarm typers are chosen, the message is sent to the event or alarm typer of the access area/function code specified and using the routing code in the access area table. If the access area or function code were not specified, the display primaries will be used.

All the options are written back to the display as well as the defaults used and the routing code used. (Two routing codes if general typer was specified.)

A message indicates the successful dispatching of the message.

The user may modify the pattern to be sent by just keying in his data over the pattern on the display. The default pattern may be modified in the background of the DOMDDIGB display.

ADMINISTRATIVE MESSAGE CONTROL

There are two sources of administrative messages, the System/7 and the local power system operator. One program, DOMCROUT processes all administrative messages.

DOMCROUT

The input PATCH ID determines the source of the administrative message. A PATCH ID of one indicates the message is from the System/7. The text of the transaction and the logical ID are extracted and routed to the power system operator access and function area system message zone.

PATCH ID of two indicates that the power system operator wishes to issue an administrative message. The program reads the display and determines if the message is to be routed to another CPU or a local display device. The text is extracted from the text line, ignoring leading and trailing blanks and underscores. An S7WRITE macro is used to write the message to the indicated CPU while the DWZONE is used to write the message to the system message zone of the access and function area supplied. No error checking of the access, function and CPU id is performed in this program, this will be done by the subroutines for DWZONE and S7WRITE. The power system operator will be notified of the results of this request by the message written in the requestor's system message zone.

DATA ACQUISITION

The data acquisition (DAQ) subsystem provides the power system operator, through supervisory control, with convenient methods of communicating data and controls with System/7 Energy Management System and the IBM 3707 remote data acquisition and control stations. Data acquisition handles all scan and status input data from the time of receipt until the storing of data in the appropriate data base arrays in a form usable by EDC/AGC and the display subsystem.

The data acquisition subsystem performs the following functions:

- Initialization
- System/7 intercommunications
- Scan processing
- Scan data conversion
- Realtime changes
- Time synchronization initialization
- System/7 failover
- Point summation
- Data logging
- AGC/EDC initialization
- Customer interface control

INITIALIZATION

Before data acquisition can be performed, the data acquisition system must be initialized. All available System/7s must be initial program loaded and scan processing initiated.

The following input stream initiates this initialization.

```
                                   (EVSAVE)      (CPSAVE)
A1   PATCH    EP=DOMTRESI,PARM=(C' {      } C'  {      }')
                                   (EVINIT)      (CPINIT)

     WAIT     A1

     RESTART  WRITE

A2   PATCH    EP=DOMTRESI

     WAIT     A2
```

Certain functions prior to data acquisition initialization must be performed by DOMTRESI.

Data acquisition initialization occurs in two steps. The first step occurs before the checkpoint request. It consists of building and initializing tables, LOADing service routines, initializing CVT pointers, OPENing System/7 load module data sets, and DCBs.

The second step occurs after the checkpoint request or after a System/370 restart. It consists of initial program loading all available System/7s and initiating System/7 scan processing (initial and cyclic scans).

DOMTRESI - System Initialization

DOMTRESI builds the System/370 Energy Management System task structure and passes input parameters along to the initialization processors.

The Energy Management System task structure is then established by PATCHing all the tasks as defined by the load module DOMTASKS. This load module details the task names, priorities, queue lengths, and entry points for all Energy Management System Tasks, including the Display Management System initialization task.

The input parameters are read and the requested actions scheduled. EVSAVE and EVINIT are events logging parameters. EVINIT specifies that the events log file is to be initialized as a new file. EVSAVE, the default, specifies that the events log file has been initialized and that the data currently contained is to be preserved.

CPSAVE and CPINIT are System/7 checkpoint parameters. CPINIT specifies that the System/7 checkpoint data sets are to be initialized as new data sets. CPSAVE, the default, specifies that the checkpoint data sets have been initialized and that the data currently contained is to be preserved.

The phase one data acquisition initialization program, DOMTP1IN, is then PATCHed. If the return code from DOMTP1IN was zero, then DOMTINFO is PATCHed under the task DOMXS7IO. A PATCH ID of zero will be used if the System/7 checkpoint file is to be initialized or an id of one will be used if the System/7 checkpoint file is to be saved.

Upon return, the events file option is set. The high order bit of the ECVTEVNT field of the EMSCVT is set to zero if the events file is to be initialized and one if it is to be retained. DOMCINIT is then PATCHed. Upon return from COMCINIT processing of the restart data set is next.

After the restart data set is written, selected arrays, load modules, and control blocks are page fixed using the Special Real Time Operating System page fix array, DPPXFIX. Phase two initialization of display management is then PATCHed. Phase two of supervisory control initialization, DOMCINIT, is PATCHed followed by a PATCH to DOMTP2IN.

<u>DOMTP1IN</u> - <u>Pre-checkpoint</u> <u>Initialization</u>

DOMTP1IN is the pre-checkpoint initialization processor for data acquisition. It builds and initializes data acquisition and supervisory control tables. The data acquisition and supervisory control table values for restart are the same as those for pre-check point initialization.

Sufficient main storage is obtained for the Energy Management System communication vector table (EMSCVT), System/7 control table (S7CT), and other System/370 highly used data and work areas and this area inserted into DPPXFIX array. The address of the EMSCVT is stored in the Special Real Time Operating System communication vector table (DPPXCVT). The S7CT and data base address pointers are stored in the EMSCVT. The S7CT is built with each entry (there is one entry for each channel attached logical ID) containing pointers to the raw data array map (RDAM) and remote control block (RCB) address list. The RDAM is a data base array. The remote control block address list is a generated list of all remote control block addresses for the given logical 3707 ID. Figure 2 illustrates the vector and address scheme utilized.

For scan processing, all flags and pointers in the work area used by scan synchronization are initialized. The address of the scan performance array CASPA and data base indicator array CADBIND are resolved and placed in this work area. The address of the special limit checking array CASPECLM and times table array CATIMES are resolved and placed in the EMSCVT. The realtime data lock and consistent data lock are defined and the addresses of their control blocks placed in the scan synchronization work area also.

DOMCSENT is called to check the items in the status array and issue an entity macro DISPENT to insure that the proper entity is displayed for each status item.

The macro services subroutines load module DOMTABLE and the System/7 routing table DOMTROUT are loaded into main storage and chained together. Control is then returned to DOMTRESI.

DOMTABLE is a reentrant load module consisting of 13 reentrant system macro processor subroutines and a directory module. The directory module consists of a table of the addresses of each included subroutine. The address of the directory module is stored in the EMSCVT. A macro is used to generate this table of addresses. The system macro(s) using these subroutines obtain the address of the directory module from the EMSCVT.

The following table lists each module name and the macro(s) it services.

Module Name	Invoking Macro
DOMTABLE	
DOMTWRIT	S7WRITE,S7IPL
DOMTSCAN	VARYSCAN
DOMTVS7	VARYS7
DOMCLRMT	DOMCALRM
DOMCEVNT	SCEVENT
DOMCALM2	DOMCALRM
DOMCDC10	SCDEVICE
DOMCLFRE	DOMCFREE
DOMTCODE	ASCICONV
DOMCNCNV	CNCNVLNK
DOMTUPD7	UPD7COMM
DOMTCHRT	DOMCSCHT
DOMTRLBF	RLSEBUFF

DOMTROUT is a table containing information necessary to route a transaction from the System/7 to a processing program. The format of this table is defined in the tables subsection of the Data Areas section.

DOMTINFO

This program must execute under the DOMXS7IO task. DOMTINFO is responsible for building the System/7 Support Control table FCVT, the System/7 initialization/failover support control table, opening DCBs and properly initializing the System/7 checkpoint data sets.

The required space for the FCVT and associated control blocks is calculated and obtained. The FCVT is then chained to the EMSCVT and all fields and control blocks initialized, including opening the System/7 support data sets identified by the DD name of S7XXCL where XX is the logical ID of the channel attached System/7.

All System/7 device address DCBs are opened and the required locks defined to insure that the DCB will not be modified. The System/7 checkpoint data set control block space is calculated and allocated on a page boundary. The PATCH ID is then checked to determine the disposition of the System/7 checkpoint data sets. ID of zero indicates to save the data sets; ID of one specifies initialize the data sets. When the System/7 checkpoint data sets are properly initialized the program then exits.

DOMTP2IN Post-checkpoint Initialization

A PATCH is made to DOMTS7IO to allow System/7 I/O processing to begin. The System/7s are then initial program loaded. The System/7 communication table (TAS7COMM) is used to determine which System/7s are initial program loaded as primaries and which are initial program loaded as backups. The initial program load is conducted through the VARYS7 macro.

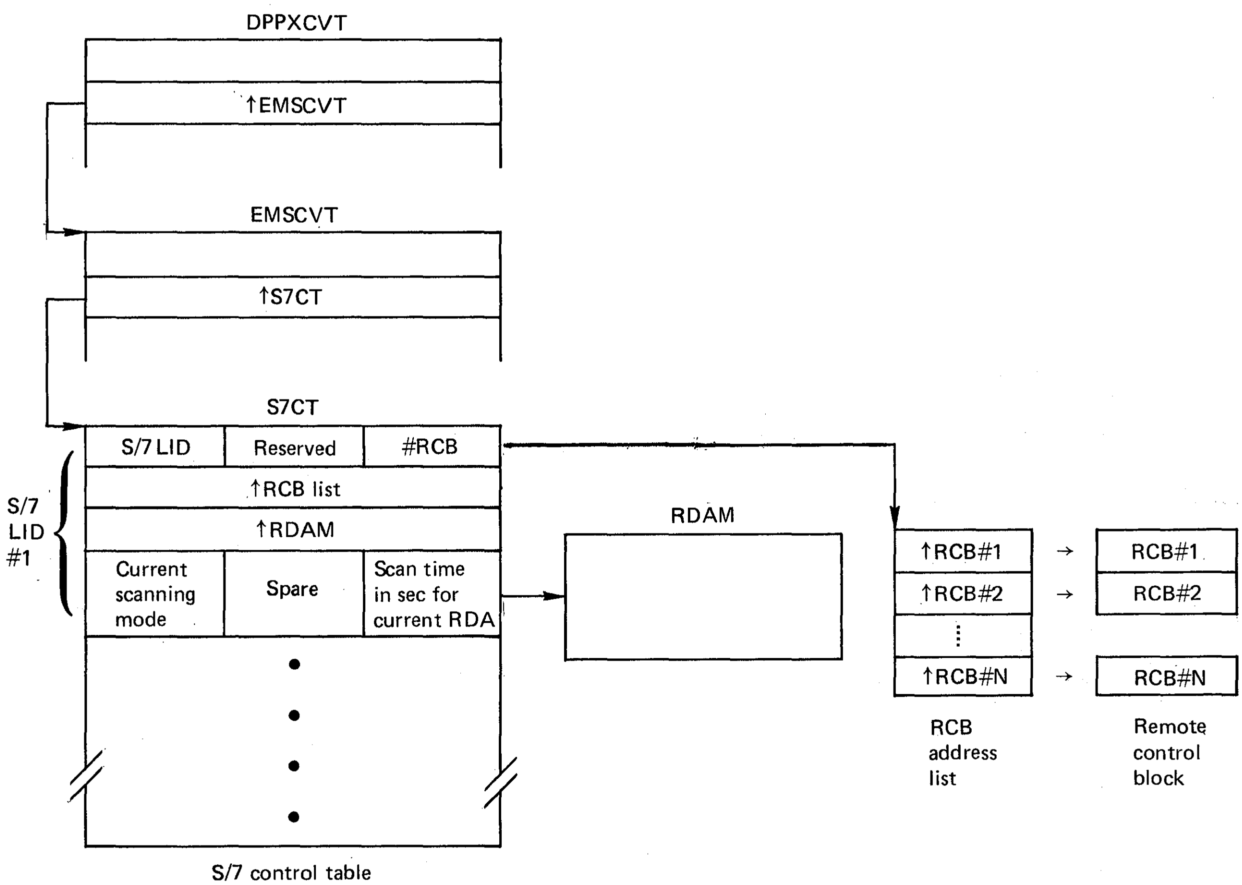


Figure 2. System/7 control table

SYSTEM/7 INTERCOMMUNICATION

System/370 Energy Management System is designed to operate as the host processor in a hierarchy of System/370 and System/7 processors or in a standalone environment. System/370 Energy Management System communicates with IBM 3707 Remote Data Acquisition and Control Stations through IBM System/7s. Data acquisition and power device control functions are performed by the System/7 on command from the host. Support is provided for System/7 attachment through a byte multiplexer, block multiplexer, or selector channel by means of the channel attachment.

The System/7 computer performs all communication to and from each IBM 3707. All communication is initiated by the System/370. System/370 to System/7 communication is supported by the EXCP access method. All I/O commands are issued from one module, DOMTS7IO, which operates as an independent task. This module accepts requests for I/O and executes them through one of the three unit addresses assigned to each System/7, the initial program load, read and write unit addresses. The following sections describe the method of requesting I/O and the method of processing those requests.

Read System/7 Data

There are two types of input buffers maintained by DOMTS7IO, utility and scan buffers. Utility buffers may contain any data read from a System/7 except scan data. Backup System/7s write "I'm OK" messages directly into these buffers. Once a utility buffer is allocated, it may not be used again until released by the processing program. If a utility buffer is desired and one is not available, a GETWA is issued to obtain a temporary buffer. If the GETWA is unsuccessful then no read is issued and data may be lost. A message is written to indicate such a condition.

Scan buffers are used only by primary System/7s. If any data is received other than scan data, an attempt is made to allocate a utility buffer and move the data into it. If a utility buffer is not available, the data is lost and a message issued to that effect. Data is read into scan buffers continuously even if the data acquisition processor has not completed processing the data previously read into it. It is the responsibility of the data acquisition processor to determine if any unprocessed data may have been overlayed. Once the data has been read control is given to DOMTPUNT to route the input.

Routing Input From System/7 - DOMTPUNT

Control is passed to DOMTPUNT, which accesses the transaction code and searches the System/7 routing table to determine which application program is to service the transaction. To minimize system overhead, the System/7 routing table consists of the necessary list forms of the PATCH so that the PATCH may be issued directly on the routing table. The only modification required is the address of the input buffer. Once the application program has finished processing the buffer of information, it issues a RLSEBUFF macro which frees the buffer.

Upon completion of the read to the System/7, the System/370 receives a channel-end indication without the device-end indication. By delaying the device-end interrupt until the next I/O is ready from the System/7 and by DOMTS7IO issuing the next read at channel-end time, an interrupt capability is established, allowing the System/7 to send data whenever it is ready.

Buffer Control

Buffers are allocated by DOMTS7IO (System/7 I/O processor) and released by using the RLSEBUFF macro. The RLSEBUFF macro calls the subroutine DOMTRLBF. This subroutine will determine if the buffer is a dynamic GETWA buffer or a permanent buffer. GETWA buffers will be FREEWAed and permanent buffers will be flagged as available.

Write Data to System/7 - DOMTWRIT

Data acquisition supports the passing of messages to the System/7 as required. This is done through the S7WRITE macro supported by the subroutine DOMTWRIT operating under the caller's task. The subroutine checks the System/7 communications array to determine the appropriate System/7 to route the message to notify the System/7 communication task of the request. DOMTWRIT waits until it is notified of the event completion and then returns to the caller.

The write function is performed by DOMTS7IO. Upon receipt of a write request DOMTS7IO builds the channel program and issues the EXCP. When the I/O is complete the requestor is posted of the completion of the output operation.

A user exit is supplied for all transactions issued through an S7WRITE macro. To use this facility, the user at system generation time must supply a program to process the transaction. The name of this program must be supplied in the USERMOD field of the TMS (Terminal Management System) macro.

The program must be a re-entrant program. The program is loaded at system initialization, phase one, time. It is entered immediately prior to posting the System/7 I/O supervisor of the output request. Register one points to the parameter list USERDECB, and the contents of register zero is unpredictable. The user exit must return a condition code in register 15. If register 15 does not contain zero, the data is not written and control is returned to the caller. For the return codes that register 15 can contain, see writeup for S7WRITE macro in Appendix A.

System/7 Error Detection and Recovery

PROCESS PERMANENT I/O ERRORS - DOMTIOER: If any I/O is posted complete for a System/7 with any condition other than successful, then a permanent error is assumed and the I/O error processing subroutine, DOMTIOER, is entered. The System/7 failover program DOMTFAIL is then PATCHed to remove the System/7 from service. To assist in the detection of an I/O error, two appendages are supplied. The load module names are specified at system generation time by the user.

START I/O APPENDAGE - DOMTSIOA: This program turns on the DEBRSIOA bit in the DEBFLGS1 byte of the DEB upon entry to the routine. This causes the appendage to be entered on every start I/O operation as opposed to only on the first one. This is done to be able to detect the control unit end-control unit busy loop. The number of consecutive times this appendage may be entered is defined by the user at sysgen time. When this limit is exceeded the I/O is terminated with a permanent error condition of X'61'.

ABNORMAL END APPENDAGE - DOMTXEND: If this appendage is entered and it is the first detection of an error condition and there is a unit check-intervention required condition, then the UCB is set not ready and the appendage is exited to retry the I/O. If the appendage is entered with any other condition, the normal exit from the appendage is taken.

DETECTING LACK OF SYSTEM/7 RESPONSE: All I/O operations are given a fixed time interval in which to complete. Output operations must complete within one second from the time of issuance. If the I/O is not posted complete at this time, then the I/O is purged and the System/7 failover program, DOMTFAIL, is PATCHed.

There are three intervals for input I/O. If a System/7 is scanning then it must reply with scan data within two times the user defined basic scan cycle. If a System/7 is not scanning then it must reply within ten seconds. After the initial scan is received the first cyclic scan must be received within six basic scan cycles. If the timeout condition occurs then the System/7 is assumed non-functional. The I/O is purged and the System/7 failover program, DOMTFAIL, is PATCHed to remove the System/7 from service. The backup System/7 is brought online if one exists.

SCAN PROCESSING

Scan Synchronization

Within an Energy Management System the responsibility for initiating the scan cycle rests in each System/7 with attached IBM 3707 Remote Data Acquisition and Control Stations. The function of System/370 scan synchronization is to indicate to application programs when a complete new set of data has been sent from all System/7s. Scan synchronization provides three indicators which are used to coordinate the realtime data base. These indicators are in the data base array CADBIND. The expected sequence of events is shown in Figure 3 which uses three System/7s as an example.

The data base closed indicator is set on when the processing of the first buffer begins. It is turned off when the last buffer is complete, or a poll overload message arrives from System/7. A poll overload condition occurs when a scan takes more time than available in one basic scan cycle to complete. Because AGC requires a complete new set of data before processing, the AGC indicator is turned on after three new buffers arrive, and it is turned off by AGC when it completes its processing cycle. (Reference Figure 4.) The data base closed indicator groups data by scan cycle. When the poll overload message is received from System/7, the cycle is defined complete (even though only two buffers have arrived), and the late buffer is regarded as part of the next scan. The third indicator is provided for a user application with similar data use requirements to AGC.

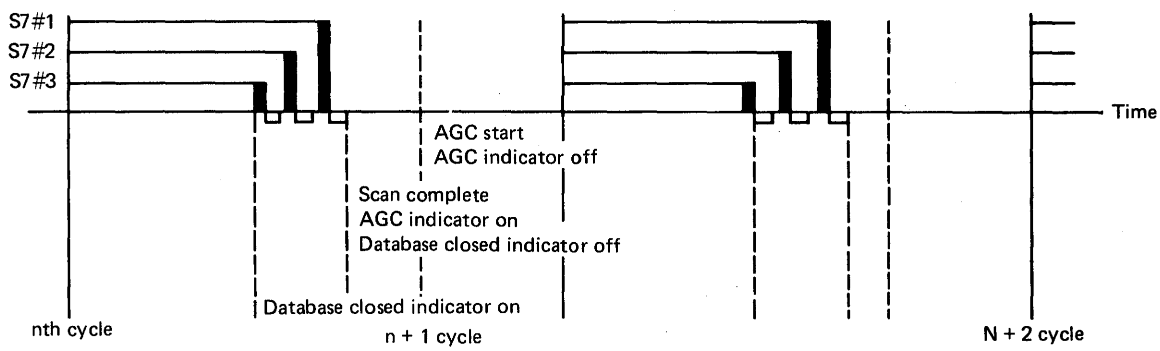


Figure 3. Normal scan processing

Data Base Synchronization Indicators

All indicators are halfwords. The names given are the item names in the data base.

- AIAGCGO - This indicator is set to 1 when scan synchronization determines that refreshed data is available for the AGC application program. AGC sets the indicator to 0 as soon as it has retrieved the AGC data.

- USERGO - This indicator is set to 1 at the same time as AIAGCGO. Another application with similar data requirements as AGC can use this indicator.

- DBCLOSED - This indicator is set to 1 when a scan is complete and 0 otherwise. The duration and setting of this indicator is discussed in the section on Scan Synchronization.

Scan synchronization also updates the scan performance array, CADSPA. This array contains the time the last scan arrived for all System/7s, and for each scan defined on a System/7, the time that scan last arrived.

Scan Performance Array

This data base array (CADSPA) is intended to provide information on scan performance (see Figure 4).

For each scan defined on each System/7, two parameters are maintained.

1. A timeword (in Special Real Time Operating System format, HHMMSSth) which is the last time this scan was used to retrieve a data sample. The time is the Special Real Time Operating System clock, sampled at the last basic scan cycle boundary before the scan arrived. The Special Real Time Operating System clock may optionally be set by an external time standard.

2. A duration (in milliseconds) which is the period from the time the data was expected to be sampled until the data was available in the data base. This is maintained as a damped average. This value is not calculated if the indicator is zero. If the indicator is non-zero, two additional values are calculated. For the "scan complete time" (see Section entitled "Scan Synchronization" for definition), a duration (in milliseconds) from the beginning of the basic scan cycle until the last expected System/7 buffer arrives. A deviation estimator for the scan complete duration is also calculated as a damped average of absolute deviation.

Discussion of Symbols

I	If zero, only S, S(I,J) are calculated If non-zero, all values calculated. This has item name CADSPAON to assist changing it
$\overline{D}$	Calculated using the recursion formula D = 1/256 (255 D + D) where D = 1500 (milliseconds)
$\overline{M}$	M + 1/256 (255 M + D - D) where M = 100 (milliseconds)
$\overline{D}$(I,J)	The duration of the J-th scan on the I-th System/7. D (I,J) = 1/8 (7 D (I,J) + D (I,J))

where D = 1500

S,S(I,J) — Time when scan was scheduled, copied from the Special Real Time Operating System clock. The th part of HHMMSSth should be zero. Non-zero implies the data acquisition task was longer in the PTIME queue than expected. For a discussion of the Special Real Time Operating System clock, see the Special Real Time Operating System documentation. Scans taking longer than one basic scan cycle will have offset timewords.

Note:

1. The duration is biased by several factors including time deviation between System/370 and System/7 clocks, time in Special Real Time Operating System PTIME queue, a few program instructions, time in System/7 scheduler queue, page faults and other System/370 task contention.

2. D, M can be used to determine what time AGC should be scheduled.

3. The scan timeword is based on the System/370 clock, scans are scheduled by the System/7 clock. The section entitled "Time Synchronization" explains how these clocks are synchronized.

4. No adjustment is made when a scan takes longer than one basic scan cycle. The values calculated will be modulo the basic scan cycle.

For these reasons, the scan performance array should be treated as a guide.

Dec	Field		Symbol	Item Name
0	Indicator		I	CADSPAON
2	Scan complete duration		D	
4	Damped mean absolute deviation		M	
6 8	Last basic scan cycle boundary when any scan arrived		S	CADLSCAN
10	S/7 ID-1	Scan ID-1		
12	Duration of this scan		D(1,1)	CADIIJJJ*
14 16	Last time this scan was used		S(1,1)	
18	S/7 ID-1	Scan ID-2		
20 22			D(1,2)	
24			S(1,2)	CADIIJJJ*
26	...			
	S/7 ID-N	Scan ID-M		
			D(N,M)	
			S(N,M)	CADIIJJJ*

This table is mapped by DSECT macro DOMTSPAD.

*where II is logical id of i-th S/7
JJJ is scan id of j-th scan on i-th S/7

Figure 4. Scan performance

Task Structure

DOMTSSYN is the program executing under the data acquisition task control block (TCB) named DOMXDAQ. The PATCH queuing capability is used to queue raw data array (RDA) buffers allowing overlap of data transfer from System/7 and data conversion and storage.

Entry Conventions and Processing Description

The system initialization begins with the "dummy" PATCH ID=255 for task creation. The scan buffer PATCH (RDA) from DOMTS7IO determines whether this is the first buffer this cycle, and, if so, turns the data base closed indicator on. It then calls the data conversion segment DOMTCONV. On return from DOMTCONV, the scan performance array is updated. DOMTSSYN then determines whether the data base closed indicator can be turned off or the AGC data refreshed indicator turned on. If so, the indicators are set and the synchronization indicators adjusted for the next entry.

If a System/7 generated synchronization message is passed from DOMTS7IO, DOMTSSYN determines whether it is a poll overload. For a poll overload, an alarm is generated and the data base closed buffer arrival indicator is set. DOMTSSYN then determines whether the data base closed indicator can be turned off or the AGC data refreshed indicator on, and, if so, sets the indicators and adjusts the synchronization indicators.

After all System/7s have reported in for this scan cycle, DOMTSSYN calls the point summation processor, DOMTPSUM, and PATCHes the user cyclic processor, DOMUSERC, passing each the current scan ID. The analog performance log routine DOMTAPLP is PATCHed to operate under the task DOMXUSER In addition to this, the pulse counter data logging routine DOMTPCLG, is PATCHed if pulse counter data was included in this scan cycle.

SCAN DATA CONVERSION

DOMTCONV

The scan data conversion program, DOMTCONV, has several functions:

1. To reformat the data supplied by System/7 since the RDA and RCB have different formats.
2. To convert analog data values from System/7 format to engineering units as System/370 floating point numbers.
3. To perform limit checks on values in RDA.
4. To inform supervisory control of unusual conditions.

ENTRY AND RETURN CONVENTIONS: DOMTCONV executes as a subroutine of DOMTSSYN under the same TCB.

DOMTCONV uses the Special Real Time Operating System LOCK function to maintain data base integrity for users. DOMTCONV stores directly into the data base.

Entry parameters are the address of the transaction code '8A' scan buffer from the System/7 and a raw data array (RDA) map (array CANNRDAM) for each System/7 which is contructed during system generation for use by DOMTCONV.

On completion of processing a scan exception table (SET) is passed to supervisory control (EP=DOMCALR1), and the scan buffer is released to the message handler, DOMTREAD.

CHANGE OF STATUS DATA (DI) PROCESSING: Status data appears in the RDA as a card address (remote 3707) of sequence number (System/7 local I/O) and a group word. DOMTCONV calls DOMTCSES to process the data. DOMTCSES locates the status group in the data base by finding a match on the card address or sequence number in the RCBs status data. When a match is found, the group word is examined to determine which status points have changed. Those status points which have changed are then further examined to determine the reason for the change. If the change was a result of a device control action, DOMCDC01 is PATCHed with an ID of 12 to indicate that change of status has been received. All other changes are considered alarms, and are passed on to DOMTBSET to be included in the scan exception table (SET), which is passed to the alarm processor when complete. If the change is a result of PDC processing in another CPU, DOMCPDC1 is called to process the change. If the wallboard indicator for the status point is on, the address of the item is added to the wallboard list for the next wallboard update cycle. These changes are also logged by PATCHing the change of status log processor - DOMCSLOG.

PULSE COUNTER DATA PROCESSING: DOMTCPC receives control from DOMTCONV through a CALL. The parameters are passed as addresses in registers. A list of the parameters is as follows: number of pulse counter points, addresses of start of the pulse counter data, terminal header, RDA, XCVT, RCB, and save areas.

DOMTCPC obtains the size of the pulse counter invalid data table (IDT) bits from the terminal header. The address of the IDT bits is obtained by indexing from the start of pulse counter data past the pulse counter size. The first pulse counter entry is located by indexing the IDT bits address with the IDT bits size. The IDT bits are used to indicate bad pulse counter data due to transfer overrun. The IDT bit is turned on to indicate the bad data and the pulse counter data base is not updated. Each IDT bit corresponds with a pulse counter entry. If the IDT bit is on, a check is made for missing data and BCH errors. If either error is found, a flag is set in the pulse counter data base, and an alarm is issued by passing control to DOMTBSET with the address of a flag table in register 1. The overrun error indicator is bit 15 of the pulse counter data from the RDA. If bit 15 is on, an overrun error has occurred. The pulse counter data base is updated, and an alarm is issued through DOMTBSET. If the pulse counter input is valid, the raw data is converted to floating point. A check for pulse counter rollover is performed by comparing the current value with the previous counter reading. If the current value is smaller, a rollover has occurred and an alarm will be issued through DOMTBSET. The previous counter value is subtracted from the current counter value to determine the pulse counter increment. This increment is compared with a maximum pulse line increment and an alarm is issued if it is greater than the maximum increment. If the converted value is a good value, the pulse counter data base is updated with the last counter reading, the accumulated pulse counter value, the increment since the last reading, and the time of the last good pulse counter reading.

A filter value used to estimate pulse counter data in case of an invalid pulse counter value and for pulse counter data initialization during a warm restart is calculated by using the following formula:

Filter Value = A * (PC) +(1-A) * Previous filter value

where: PC = current
A = constant defined
options array CASYS.

This filter value is used by the pulse counter logging routine DOMTPCLG to estimate pulse counter readings for data points with an outstanding alarm.

This process is continued until all pulse counters have been processed for this System/7 RDA. Control is then returned to DOMTCONV.

ANALOG DATA PROCESSING: DOMTCONV calls DOMTCACS to process the analog data. DOMTCACS uses the indexing tables to find the start of analog data in the RDA, and the AICOUNT field of the RDA to determine the number of analog points to be transferred.

Each analog value is checked for missing data, offscale (analog to digital conversion and transduucer), operating limits, and warning limits (within system generated percent of operating limit range). If one of the conditions occurs, a SET entry is constructed to be passed to the alarm processor.

The value is then converted to a fullword fixed point engineering unit value, using Ax+B calculation for an RDA value. The fixed point value is converted to floating point and stored in the data base. The engineering unit conversion can be alternatively performed by the user segment DOMUCONV. DOMUCONV is called by DOMTCACS, a CSECT of DOMTSSYN. DOMUCONV is a BR 14 in the delivered system. The user is expected to furnish his own DOMUCONV routine if user conversion of analog data is desired. DOMUCONV will receive two pointers: (1) register 9 will contain the address of the floating point word to contain the converted value, and (2) register 6 will contain the address of the half-word raw data array (RDA) value received from the data scan.

If the value is within limits, it is checked to see whether it is the third such consecutive cycle after an alarm. If so, a clear alarm entry is constructed in the SET.

If the point is not alarmable, or if alarm condition detected in this cycle is already marked in the RCB, an entry is not made in the SET.

CONFIGURATION CHANGES: DOMTCONV checks the offline bit in the data base (RCB) for the System/7 and the IBM 3707 and the analog point. If the offline bit is set on, then processing of the point or IBM 3707 or System/7 is bypassed.

HIERARCHY: The data conversion remains valid in a hierarchy. The higher level CPU will not be aware of other points defined on lower levels that are not shipped up the hierarchy. For example, suppose an IBM 3707 has points 1-10 wired and transferring data to the first level System/370. Suppose points 1, 2, 6, 7, 8 are marked for logging and are transferred to the second level. These will appear to the second level as points 1-5. At this level, the IBM 3707 will appear to have only 5 points wired and transferring data.

REALTIME CHANGES TO DATA ACQUISITION

Introduction

This section describes the programs used to place IBM 3707s in and out of service, place points in and out of service, place channel attached System/7s in and out of service, activate and deactivate scans, and change scan modes between initial, normal, and emergency modes. This allows pre-planned changes to the data acquisition configuration without requiring another System/370 system generation. Data acquisition configuration changes originate from System/370 programs (predominantly supervisory control), which issue VARYCONF, VARYS7, VARYSCAN, or INITSCAN macros.

Module DOMTVARY, which processes inputs from the VARYCONF and VARYSCAN macros, changes configurations by placing IBM 3707s or points in and out of service by PATCHing the module, DOMTVRPT, or by placing a scan active or inactive. The transaction codes from the configuration changes and scan changes are processed by DOMTVDB. Inputs to the VARYS7 macro to place a System/7 in or out of service are processed by module DOMTVS7. The module DOMTSCAN, which processes inputs from the INITSCAN macro, changes scans between initial, normal, and emergency modes. All the above modules are executed under the caller's task.

DOMTVARY has the function when processing a VARYSCAN to generate general System/7 commands to update System/7 tables.

DOMTVARY has two subfunction services when processing VARYCONF:

1. To verify the consistency of the macro parameters and return an error return code in case of any discrepancy.

2. To PATCH DOMTVRPT passing the address of the S7CT, the RCB and, in case of a point, the point address, name and type.

DOMTVRPT has three subfunction services:

1. To vary a point online and offline and notify the System/7

2. To vary an IBM 3707 online and offline and notify the System/7

3. To vary the IBM 3707 and points only on the varied 3707 under a System/7 online or offline when PATCHed directly by DOMTVS7

Placing an IBM 3707 In or Out of Service

The VARYCONF macro is used in the following manner to place an IBM 3707 in or out of service:

VARYCONF SYS=ID, TERM=ID, STAT=NEW

This macro is issued by supervisory control Control passes to DOMTVARY, which builds an address table and PATCHes DOMTVRPT. The System/7 command, built by DOMTVRPT, is then sent to the appropriate System/7.

Placing Points In or Out of Service

The VARYCONF macro is used in the following manner to place a point in or out of service:

VARYCONF SYS=ID, POINT=ID, STAT=NEW

This macro is issued by supervisory control. DOMTVARY calls DOMTVRPT, which finds the point (analog, pulse counter, status) and flags it in service or out of service. The System/7 command is built and sent to the System/7. The data base of the System/370 is updated after receiving a good return from the System/7.

Placing Channel Attached System/7 In or Out of Service

The VARYS7 macro is used in the following manner to place a System/7 in or out of service:

    VARYS7 TYPE = type, LID = logical ID, UNIT = System/7 CPU ID

This macro is issued by DOMTP2IN, DOMTFAIL, or by supervisory control. It is used to place a System/7 in one of three modes: 1) Primary, 2) Backup, or 3) Offline.

1. Primary - If the System/7 requested is not currently initial program loaded for the indicated logical ID, it is initial program loaded. If the System/7 requested is initial program loaded and no other System/7 is primary for this logical ID, a perform primary initialization transaction is sent to the System/7.

2. Backup - The offline System/7 is initial program loaded into a backup mode unless it is already IPLed as a backup. In this case nothing happens except a message to written to the requestor's scratch pad zone.

3. Offline - If the System/7 is primary a stop scan transaction code is sent to the System/7. If the System/7 is a backup a stop communication transaction is sent to the System/7. DOMTVS7 will PATCH DOMTVRPT passing parameters to vary all terminals and points in-service or out-of-service for this System/7.

Changing a Scan to Active or Inactive

The VARYSCAN macro can be utilized to alter a scan definition during realtime processing by changing status to active or inactive. No changes are allowed for the scan defined as initial. To perform this function the VARYSCAN macro can be issued as follows:

    VARYSCAN SCAN=ID,SYS=(System/7 ID),STAT=(desired status)

Upon receiving control from the macro call, DOMTVARY builds the System/7 command which is sent to the appropriate System/7. The System/370 data base array CASCAN is updated to reflect the scan change.

Initial, Normal, and Emergency Scanning Modes

    INITSCAN SYS=ID, MODE=NEW

This macro change System/7 scanning mode between initial, normal and emergency modes. Module DOMTSCAN, executing under the user task control block, constructs the System/7 transaction code message and sends this coded message to the System/7.

If the INITSCAN macro contains the parameter SYS=ALL, then each System/7 listed in the System/7 control table (S7CT) has a start scanning message sent with a S7WRITE. When the System/7 ID is given, the start scanning transaction code is written only to the given System/7 by the S7WRITE.

The start scanning transaction code, X'0A', sends the message X'DDDD' for normal scan, X'EEEE' for initial scan, and X'EEEE' for emergency scan. The emergency scan also sets a flag in the S7CT and scan status array (CASCAN).

The VARYSCAN macro can be utilized to change the scanning mode from normal to emergency or from emergency to normal. To change the mode to emergency the macro is issued as follows:

```
VARYSCAN MODE=E,STAT=MODE
```

To change the mode to normal, the macro is issued as follows:

```
VARYSCAN MODE=N,STAT=MODE
```

When a scan mode change request is encountered, DOMTVARY wil set a bit in the System/7 communication table (TAS7COMM) to indicate that scans are going to be stopped in order to change modes. DOMTVARY will then issue an S7WRITE macro to each channel attached System/7 which is active and has its mode active in the hierarchy. As each System/7 reports back that it has stopped scanning, the System/7 I/O processor starts scans in the new mode.

TIME SYNCHRONIZATION

Time synchronization, which must be maintained throughout the hierarchy, is supported by the System/7. It is the responsibility of the System/370 to provide the correct time to Special Real Time Operating System at system initialization and to maintain a central timetable for use by various applications.

Full-minute interrupts from an external timer are received by each locally attached System/7 and the System/370. Each locally attached System/7 also has the capability to read time from the external timer. This time, which is passed to the System/370 every basic scan cycle, is used by a System/370 time routine DOMTCLOK, to correct the Special Real Time Operating System time. This time correction function is performed only once at the conclusion of the first basic scan cycle.

The System/370 external interrupt handler, which stores current System/370 time on every full-minute pulse, causes DOMTCLOK to be PATCHed after the restart data set has been written. DOMTCLOK waits until the System/7 passes its time to the System/370 and the data conversion routine stores this time in the data base. Using the System/370 time stored above and the current System/370 time, DOMTCLOK determines the amount of time elapsed since the last full-minute interrupt. The System/7 time, which has been stored in the System/370 data base, is rounded to the last full-minute. DOMTCLOK determines the correct current time-of-day by adding the elapsed time and the rounded data base time mentioned above. The difference (delta) between this time and the Special Real Time Operating System current time is calculated and passed to Special Real Time Operating System module DPPCUPCF for their time correction. After this, the external interrupt handler supports Special Real Tire Operating System time correction as needed.

A central time table is maintained by the System/370 Energy Management System as an array CATIMES. This array is made up of seven entries as follows: day of the year, current System/7 time (standard time), current power system time (systerr time), time deviation, system frequency, fast-time correction frequency, and slow-time correction frequency. The last three values are specified by the user during system generation. The day of the year is updated by scan processing from the day field of the time in the raw data array. The standard time, which is updated by scan processing every basic scan cycle with the time passed from the System/7 is used to determine power system time. Standard time minus time deviation equals power system time. The time deviation, which is initialized to zero, is updated by user inputs or from the time deviaticn in the raw data array (RDA).

SYSTEM/7 FAILOVER

If, for any reason, a System/7 fails to respond, either during initialization or in the normal operating environment, recovery procedures are initiated to switch to a backup System/7. The System/7 failover program is queued from the System/7 I/O task, DOMTS7IO, when a System/7 ceases to transmit, or when an I/O error occurs for a System/7. These are accomplished by PATCHing task DOMXPCC entry point DOMTFAIL.

Upon entry to DOMTFAIL, the System/7 communication table TAS7COMM is updated and logged. This is done by branching to a program DOMTUPD7 which is link-edited as part of DOMTABLE. DOMTUPD7 flags the System/7 as not ready and, if it is either a primary or backup, resets those indicators to reflect the condition. An alarm is issued that the System/7 has failed. Data acquisition is lost for the logical ID, and a search for a backup is initiated. If a System/7 is available, then a VARYS7 macro is issued for the recovery. If a System/7 is not available, the System/7 must be commanded to be initial program loaded again manually through the power configuration control display.

<u>System/7 Initial Program Load</u>

The initial program loading of the channel attached System/7s is performed by DOMTEIPL, which is PATCHed from the VARYS7 macro Processor DOMTVS7. Initial program load of each System/7 is supported by a partitioned data set. There is such a data set for each locally attached logical System/7. Each data set is identified by a JCL card with a DDNAME of S7XXCL, where the XX is the associated logical ID. Each data set contains members BOOTSTRP, DOTHCKPT, S7LOAD, the first initial program load, and, if the logical System/7 has a disk, a member DOTHCORR plus those members listed in the DOTHCORR member. The DOTHCORR member is a directory of the disk load records for the associated System/7 disk. The relationship of the DOTHCORR member to the other members in the partitioned data set is shown in Figure 5.

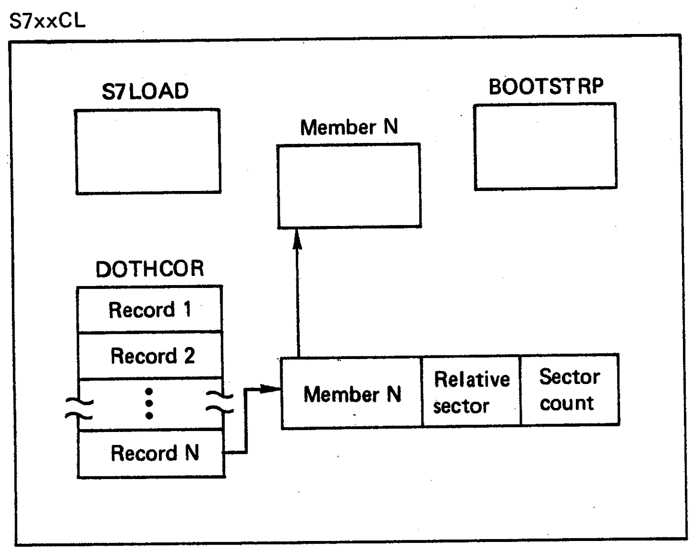


Figure 5. Relationship of DOTHCORR to other members

The data acquisition initialization routine OPENs the System/7 unit data control blocks (DCBs) and partitioned data sets. DOMTEIPL retrieves the BOOTSTRP member for the desired logical ID and writes the text to the System/7 using IPL channel address. Upon completion of the initial program load, DOMTEIPL writes all records from the S7LOAD member to the System/7 to complete the System/7 initial program load procedures. DOMTEIPL then exits to allow the processing of the disk requests from the System/7 if it has a disk module. The System/7 failover routine DOMTFAIL is PATCHed if either of the members BOOTSTRP or S7LOAD cannot be located, or of a System/7 disk support error occurs in reading or writing to the disk.

System/7 Disk Load Request

This loading of the System/7 disk is supervised by DOMTDISK.

The System/7 completes its initialization after its initial program load by sending a disk load request transaction code. Bit 14 of the transaction header indicates the success or failure of the System/7 initialization. If the System/7 initialization failed, then DOMTFAIL is PATCHed to take the System/7 out of service. The first disk load request in the sequence contains the version, mod and release level. This is checked against the version, mod and release levels in the data base. If they are not equal then the System/7 is failed over with a configuration level error. If the levels agree then processing continues. If the initialization is completed successfully, then DOMTDISK checks the System/7 communication table to see if the logical ID the System/7 is assigned to uses disk. If so, the DOTHCORR member of the System/7 partitioned data set is read to determine what disk record to send. It is formatted and sent to the System/7 requesting it, primary or backup, through the S7WRITE macro. DOMTDISK is reentered for every request for disk load. The success or failure is noted in flag bit 14. DOMTFAIL is PATCHed if any load is unsuccessful.

When the System/7 requests a disk record and all records have been sent a unique disk load transaction is sent to the System/7 informing it that its basic initialization is complete.

System/7 Hierarchial Initialization Support

In response to the last disk record transaction record, the System/7 responds with an initialization completed transaction to DOMTS7HS. At this point the System/370 determines what action is to follow. If the System/7 is to be a backup, message DPP249 is evented and a system message displayed to the access and function area of the terminal which controls communications and configuration. If the System/7 is to be primary, a transaction is sent to the System/7 to perform primary initialization functions.

When all primary initialization functions are complete in the System/7, the System/7 returns a top/not top of the hierarchy transaction. This transaction indicates the success or failure of the initialization and the status of this node in the hierarchy. The power system operator performs any power system changes necessary for the leg of the hierarchy under control.

At the conclusion of all power system changes, the power system operator at the top of the hierarchy, through the display, initiates the sequence to enter the System/7 into the hierarchy. If the System/370 is at the top of the hierarchy, then the System/7 is commanded to stop scanning down its leg of the hierarchy. If the System/370 is not at the top of the hierarchy, then the top/not top transaction is sent to the top of the hierarchy. All scanning is controlled from the top of the hierarchy.

Upon receipt of a scan stop transaction from the System/7, a transaction is sent to the System/7 giving it the date and time. The status of all scans and the current scanning mode are then broadcast down the leg of the hierarchy. This is followed by the start scan command.

POINT SUMMATION

The user is allowed to specify a number of analog points or pulse counters to be summed and limit checked as one entity. As many as 225 points may be summed together and the user may define an unlimited number of summations for a system.

During system generation, the user defines the points to be summed through the System/370 system generation procedure. The summation point name, scan ID, and names of points to be summed are used to create the point summation table module DOMTPTSM. A group identifier is also provided. Supervisory control initialization loads DOMTPTSM and builds the point summation table DOMTSMTB.

Upon completion of normal data acquisition routines, the point summation routine DOMTPSUM is called. The point summation table is searched by scan ID, and when the current scan ID matches the scan ID in the table, the array items are accessed and summed. The summation duration value in the point summation table is used to determine if the value is to be added to the previous sum, or if it is to replace the previous sum. When the scan time in seconds is divisible by the summation duration, the accumulated value is reset to zero, and the accummulation is started anew. The sum value is then placed in the data base. The location of the sum values is pointed to by entries in the point summation table as well as pointers to the points to sum.

DOMTPSUM - Point Summation Processor

This program, which receives a queue through a call from DOMTSSYN, is passed the current scan ID. DOMTPSUM searches the point summation table, DOMTSMTB, for scan IDs equal to the scan ID passed. Upon finding a match, DOMTPSUM uses the addresses of the points to be summed to obtain each data point and adds it to the total until all points for this group have been processed. DOMTPSUM uses the sum point summation duration in the point summation table to determine whether the new sum is to replace the old sum or if it is to be added to the old sum. The sum value is then placed in the appropriate entry in the data base. Each group of points with a scan ID equal to the current scan ID is processed as above until all groups have been processed.

DATA LOGGING

Pulse Counter Data Logging

The pulse counter data array CACOUNT is logged once an hour on the hour. In order to maintain the latest copy of this array for restart purposes and to log the pulse counter array every hour, the pulse counter data logging routine DOMTPCLG is PATCHed by scan processing after each pulse counter data scan.

If the last pulse counter data scan was on the hour, the Special Real Time Operating System PUTLOG function is used to log the array CACOUNT. Before the data is logged, each point is checked for an alarm condition. If an alarm condition is present, the filter value calculated during pulse counter data normal processing is used to adjust the accumulated value according to the time of the last good reading. If this time is greater than an hour ago, an hour period of data is adjusted. Otherwise the data is adjusted to bring this pulse counter data point up to the full hour.

The Special Real Time Operating System PUTBLOCK function is used to move CACOUNT to the direct access array DACOUNT. This is done for every queue to DOMTPCLG to maintain the integrity of the pulse counter data. Each new copy replaces the previous copy. During re-initialization DACOUNT is read back into the data base array CACOUNT by the pulse counter initialization routine which is part of the supervisory control initialization routine, DOMCINIT.

Status Data Logging

The status log program is PATCHed by DOMTCSES, DOMCALR1, and DOMCDCL1 to record changes of status. The status change is added to the incore CASTLOG array with the time it occurred. The array is then put to the DASTLOG array on disk using the PUTBLOCK macro. When the CASTLOG array is full, it is logged to the status log using the PUTLOG macro. Status is logged only when a status item changes. The status array CASTATUS is logged when the CASTLOG is logged. The module used in status logging is DOMCSLOG.

Performance Logging and Retrieval

Performance logging and retrieval provides the capability to log selected analog points and to retrieve the logged copies upon request.

Six programs provide the performance logging and retrieval functions. DOMTAPLI provides initialization processing during supervisory control initialization processing. DOMTAPLP controls the realtime logging of the selected analog points. DOMTAPLD gives a realtime capability to add or delete analog points from those selected at system generation time. DOMTAPLM provides performance log retrieval control while DOMTAPLR performs the actual retrieval. Finally a program is provided which has no processing capability, DOMTUSER. Program DOMTUSER must be expanded by the user to incorporate any functions desired to process the retrieved performance log data.

DOMTAPLP - PERFORMANCE LOGGING PROCESSING: Program DOMTAPLP receives control through a PATCH with an ID of 2 from DOMTSSYN. This PATCH is executed at the completion of each scan cycle by data acquisition. The purpose of the processing program is to place the data for the selected analog points in CAPTLOG array and demand log the array to the history log file.

Each valid entry in the CAPLNAME array has two address pointers. The first pointer is the address of analog data in the CANALOG incore array. The second pointer is the address of the names list data for the selected analog point in the CAPLNAME array. The data for the length of each DSECT, ANALOG, and ANAMELST respectively is moved into the CAPTLOG array for each analog point in the CAPLNAME array. Zeros are placed in the array for void entries. The first word of the CAPTLOG is the time of the last data acquisition cycle.

Before the demand log is executed the CALOGTIM array is accessed to determine if performance log retrieval processing is in progress. If it is in progress, two counters are compared. The first counter is incremented by one for each logged array that is retrieved. The second counter is incremented by one for each demand log of the CAPTLOG array by this program when performance log retrieval processing is active. When the demand log count is equal to or greater than the retrieval counter, performance logging is suspended for that cycle, and an event is generated to indicate the same. Control is returned to the system by use of the Special Real Time Operating System EXIT macro.

DOMTAPLD - PERFORMANCE LOGGING DISPLAY CONTROLLER: Program DOMTAPLD receives control from a PATCH generated as an IBM 5985 display unit program function key. This program function key defined for that unit and is valid only for those display units for which it is defined. Other means of entry into the program are from PATCHes generated from the performance log display DOMDAPLD. Each PATCH ID initiates a separate functional processing routine.

All PATCH entries into the program are processed by a housekeeping section of the program. This housekeeping section saves the registers, zeros the control element address table (CEATAB) entry if the task is new, and sets up base registers. Background lights are turned on for the display unit program function keys that are active. The PATCH ID is then examined and a branch is taken to the section indicated by the PATCH ID. When control is returned to the housekeeping section a common exit routine is provided which provides standard linkage and returns control to the operating system.

For a PATCH ID of 2, the initialization section is executed. This section calculates the total number of pages for the display and sets up basic search parameters for the remote control block search subroutine. The page build subroutine is then branched to, which builds page one and displays it. When control is returned from the page build subroutine, the housekeeping section is then given control.

A PATCH ID of 4 initiates control in the page forward section. This section advances the current page number by one or, if the last page is already displayed, then the current page number is set equal to one. Control is then passed to the page build subroutine which builds and displays the requested page and then returns control to the page forward section which in turn returns control to the housekeeping section.

A PATCH ID of 6 causes the page backward section to be executed. The current page number is decremented by one unless the current page number equals one in which case the current page number is set equal to the last page number. The page build subroutine is then given control. The page build subroutine returns control to the page backward section which returns control to the housekeeping section.

A PATCH ID of 10 passes control to the add section. This section provides the capability of adding an additional analog point name to the performance log file array CAPLNAME. The new name is read from the display screen, but before the new name is added to the file, several validity checks are made. The CAPLNAME array is checked to ensure that there is space to add the new item. The item is verified as being a valid analog point name and a search is executed to ensure that the name is not already in the CAPLNAME array. If any of these conditions are not met, a message is displayed on the screen which indicates the problem. A valid item name is entered into the array and the analog (CANALOG) data and the names list (CANAME) data addresses are resolved and entered in the CAPLNAME array. A demand log is then executed for the CAPLNAME array to ensure that all modifications are present during a warm start of the system. The number of pages for the display are recalculated and modified if necessary and the page currently being displayed is redisplayed. Each successful addition is evented to the primary access code and function code of the IBM 5985 Display Unit on which the entry is made, and a message is displayed to the screen indicating the successful addition. Control is returned to the housekeeping section.

A PATCH ID of 12 causes the deletion processing section to receive control. This PATCH is generated when the cursor is over any of the screen positions which the name field normally occupies and the cursor entry program function key is depressed. The name field is read from the screen on the same line as the cursor and passed to DOMTAPLD with the PATCH. If the input characters match a name in the CAPLNAME array that item is deleted and all other items are moved up one place to avoid blank entries in the table. Each successful deletion is evented to the primary function code and access code of the display unit and a message is written to the screen notifying the power system operator of the deletion. The CAPLNAME array is logged through a demand log to ensure that all modifications will be present during a warm start of the system. The display page is then refreshed to display the data as it is currently represented in the data base. Control is returned to the housekeeping section.

A PATCH ID of 14 causes the cancel section to receive control. One is subtracted from the task counter. If the task counter is zero and the display on the screen is not DOMDAPLD then a DPATCH macro is issued to release the task if the work queue is empty. Control is returned to the housekeeping section.

Several subroutines are used in processing by the program, DOMTAPLD. The first subroutine is the page build and display subroutine. Each item name in the CAPLNAME array is displayed with a maximum of 26 names per page. With each name is displayed the remote control block/substation name and the 16-character types field. The last item on the last page generates an "end" message which appears directly below the last displayed item name. For each remote control block/substation name which is displayed, a search subroutine is branched to find the remote control block for which the analog point is associated.

After the display page is built in a buffer, the display background is brought to the screen if it is not already displayed. Each time the display background is brought to the screen, the task counter is incremented by one. This task counter is used to control the task, and the task is kept active as long as the task counter remains positive. The task counter is zero upon activating the task initially. After the display background is on the screen, the foreground items are put to the screen including the page numbers, analog item names, remote control block/substation name, and the type field. Control is returned to the section which called this subroutine.

A remote control block search subroutine searches the remote control blocks until the analog item address equals the analog item address in a remote control block, is greater than the remote control block analog item address, and is less than the next remote control block analog item address. When either of these conditions is met, the address of the currently selected remote control block is returned to the calling subroutine.

Another subroutine searches the CAPLNAME array to verify whether an array item is present in the file.

An error processing subroutine outputs error messages to the display or typer for error conditions which may arise.

DOMTAPLM - PERFORMANCE LOGGING RETRIEVAL CONTROL PROGRAM: Program DOMTAPLM receives control from a IBM 5985 color display unit keyboard with a PATCH ID of 2. This program activates performance log retrieval if it is not active and/or maintains control functions through array CALOGTIM if performance log retrieval is active.

This program GETMAINs a buffer to hold a retrieved CAPTLOG array record with the first four bytes being the length of the GETMAINed area. The time the PATCH is received is used to issue a GETLOG macro and retrieve the performance log array logged nearest the time of the PATCH. This retrieved time from the performance log array is stored in the CALOGTIM array as a mark time from which the retrieval function operates. A maximum of ten times can be stored, each being separate and the basis for calculating the amount of data which will be retrieved. A calculation is made to determine how many records will be retrieved prior to the first mark time and how many records will be retrieved following the last mark time in the CALOGTIM array. The total number of logged copies is stored in the CALOGTIM array with the number of records retrieved following the last mark record. The number of retrieved copies prior to the first mark record is the total number of logged copies minus three copies or ten seconds worth of logged data whichever is greater.

If performance log retrieval is not active, a GETLOG is issued for the first record. After calculating which logged record is to be retrieved first, it is placed in the buffer and then shipped to the DOMTUSER program. Several flag settings and counters are set at this time in the CALOGTIM array.

If the retrieval is active, the above action of retrieval and shipping of data does not take place; otherwise, the retrieval active flag is set on and the counters for logging and retrieval are set to zero, respectively. A flag known as the minus mode flag is always set on and is an indicator to the retrieval program that data is still being retrieved before the last mark record is reached.

All mark times that are accepted, evented to function code zero and access area code zero, and cause a message to be displayed to the system message zone of all IBM 5985 Color Display Units. The CALOGTIM array contains a flag to indicate an update to the CALOGTIM array is in progress. This flag is turned on at the start of this program and turned off at the end of this program.

When performance log retrieval is not active at the start of this program, a PATCH is made to program DOMTAPLR which continues retrieval until all the necessary records have been retrieved. Control is returned to the system at the completion of program DOMTAPLM.

A PATCH ID of 6 is received when a request has been entered via the sign on menu display to terminate performance logging retrieval prior to normal completion. Termination is acheived by zeroing the CALOGTIM array. When program DOHTAPLR checks the retrieval active flag and it is off all retrieval is terminated and message DPP377I is printed to the system message zone of all screens and an event is generated.

DOMTAPLR - PERFORMANCE LOGGING RETRIEVAL PROGRAM: Program DOMTAPLR receives control from program DOMTAPLM through a PATCH with an ID of 2. Program DOMTAPLR retrieves logged performance log arrays.

Initially, a buffer which holds one logged performance log array plus the logged header plus a four byte length field is retrieved. A loop, is then entered, which retrieves records until a retrieval counter equals the end counter, a cancel flag is set, or a PTIME active flag is set on. These checks are explained below. The retrieval plus counter is initially set at zero and is incremented by one for each post mark time record which is retrieved. The end counter is the total number of logged copies of the performance log array. The cancel flag is set on when retrieval of logged copies is unsuccessful. An event is also generated to notify the power power system operator(s) that the performance log retrieval is unsuccessful, and has been canceled. The PTIME flag bit is set to indicate that the retrieval of arrays is about to overtake logging of arrays and that this program is PTIMEing itself with an ID of 10 to start retrieving six seconds later.

As each logged array is retrieved, a branch to one of two sections is made based on whether the minus mode or plus mode flag is on. The plus mode flag is turned on when the last mark record has been retrieved. Minus mode processing compares the time of the retrieved array against the mark record, and, when the times agree, either moves all the mark records up one or initiates plus mode processing if no more mark records exist. The retrieved array is then sent to the program DOMTUSER.

Plus mode processing retrieves records until the plus mode retrieval count equals the end count. Upon processing the final record, all F-Fs are moved to the first word of the buffer and the buffer is shipped to program DOMTUSER. All flags in array CALOGTIM are set to zero.

Exit processing frees the buffer and then returns control to the system.

Before any modifications are made to the array CALOGTIM, the update flag is checked. If the flag is on, program DOMTAPLR PATCHes itself with an ID of 2 and plus mode processing is again executed. When the update flag is off, it is turned on until all updates to array CALOGTIM are complete; at which time, the update flag is turned off.

DOMTUSER - RETRIEVED PERFORMANCE LOG RECORDS PROCESSOR: This program structure consists only of an entry and an exit. No processing capability is provided. The user must provide a module by this name to process the performance log retrieved records. The module must have an entry point name of DOMTUSER.

AGC/EDC INITIALIZATION

In order to queue AGC/EDC initialization, the routine DOMTAGCI is PATCHed after all System/7s have responded with their initial scan. Upon receiving control, DOMTAGCI PATCHes the AGC initialization routine DOMAAGCI (if AGC processing has been included) and the EDC initialization routine DOMAEDCI (if EDC processing has been included). It waits for the first PATCH to finish in order to guarantee that AGC initialization is complete before PATCHing EDC initialization.

USER INTERFACE CONTROL

User Initialization Processing

A user supplied initialization DOMUSERI, is PATCHed after all active System/7s have reported in with their initial scan data. This program structure provided with the system contains only an entry and an exit. Any processing required by this routine must be supplied by the user.

Customer Cyclic Processing

A user supplied routine DOMUSERC, is PATCHed after all active System/7s have responded the current scan cycle. The current scan ID is passed as the PATCH ID to DOMUSERC. This program structure provided with the system contains only an entry and an exit. Any processing required by this routine must be supplied by the user.

Fortran - PL/I Interface Program (DOMCIF)

This program allows the Fortran and PL/I users to execute System/370 Energy Management System macros. The program accepts standard or optimizing compiler inputs. DOMCIF operates as a subroutine of the main program.

The caller passes only one parameter to DOMCIF. This parameter, however, is the first byte of a structured list of parameters. The first four bytes of which are the halfword macro ID and the halfword return code field. An invalid macro ID results in a minus one return code.

DOMCIF verifies that the macro ID is a valid one. Then it executes that section of the program which handles the requested macro service. The segment which processes the request either uses the remaining parameter list to pass to the macro service, or builds, in its own work area, a new parameter list based on the parameters supplied by the caller before invoking the desired service.

The STAE is overlaid and the SPIE canceled upon entry to DOMCIF. If an ABEND occurs, the DOMCIF STAE exit routine saves the ABEND location, if available, and the completion code and exits with the retry option.

The STAE retry routine issues a message indicating the type and location of abend, FREEMAINs the workarea if supplied, FREEMAINs the DOMCIF save area, and reissues the abend for the compiler STAE routine.

APPLICATION PROCESSING

Automatic Generation Control (AGC) regulates generator power output within a prescribed area in response to changes in system frequency and tie-line loading so as to maintain scheduled system frequency and/or established interchange with other areas. AGC may be suspended or run in any of three modes at the discretion of the power utility power system operator. These are flat frequency control, flat tie-line control, and tie-line bias control. Under flat frequency control, all control effort seeks to maintain the scheduled frequency and the power flow over the tie-lines is ignored. Under flat tie-line control, all control effort seeks to maintain the net scheduled tie-line flow. Any deviation of frequency from its scheduled value is ignored. Under tie-line bias control, the AGC program responds to deviations from both scheduled frequency and the net scheduled tie-line flow.

The integral of the frequency error is kept within reasonable bounds by means of a system time correction. This is accomplished by offsetting the scheduled system frequency upon request by the power system operator.

EDC outputs are used by the AGC program to economically allocate generation among the individual generators of the control area.

The basic function in the Economic Dispatch Control (EDC) of a power system is to minimize the cost of meeting the energy requirements of the system over some appropriate period of time and in a manner consistent with reliable service. The System/370 Energy Management System programs assume that all generators to be considered are online and running. No startup or shutdown costs are considered. The system is designed for fossil-fuel fired steam generating units.

The EDC programs use the principle of minimizing the total cost of running a set of generators by equalizing the incremental cost of delivered real power from these generators at an assumed system load center. Both fuel costs and operation and maintenance (O&M) costs are taken into account. Line losses are considered using Bmn coefficients. Other factors that are taken into account in the calculations include the economic operating limits and the control status of each generating unit.

There are two distinct sets of generators that are covered by the EDC programs. The programs recognize two types of entries. The first of these is a request for an economic dispatch to be run for all generators whose output is currently being automatically controlled by the automatic generation control program. These units are said to be on automatic control. The second type of entry to the economic dispatch programs calls for an economic dispatch to be run on all units under automatic control plus those units whose outputs are said to be economically variable but are not presently being controlled by the automatic generation control program. A dispatch is run for units on automatic control every three minutes nominally (or upon manual request or request by another program), whereas a complete dispatch for all units either on automatic control or economically variable is run every twenty minutes nominally. The three and twenty-minute intervals are the default values at the time of system generation. It is possible to SYSGEN other values for the intervals. It is also possible to change the intervals through data entry services during realtime operation.

Data Base Structure

The data base necessary to operate the AGC programs is established through the use of both the offline utility program, DPPXUTIL, and the system generation procedure. (Refer to the Energy Management System Generation Manual and to the System/370 Energy Management System Program Reference Manual.)

1. Arrays established through system generation:

   AATAGCIC, AA0INGEN, AA00INTL, AA0INNCL, AA00INSF, AA0TEDCA, AAACTADR, AAAGCOUT, AAAGCODD, AA00ICSM, AA0TICSM,AATTTGCS, AAPMAXEC, AAPMINEC, AAPMAXEM, AAPMINEM, AA00RHOO, AAAGCRHO, AAOPDEDC, AA0PDAGC, AA0DELTP, AA0ICAGC, AAAGCACT, AAAGCEXC, AAAGCOTT, AAAGCUNC, AAAGCPDO, AAAGCPDP, AAPLANTS, AATIMCOR, AAWTEDCA, AAOTEXTA, AAAGCODR, AIAGCODD, ALLOOICSM, AAAGCPDA, AAAGCPDB

2. Arrays established or modified through DPPXUTIL:

   AA0TGCSM, AAAGCPDA, AAAGCPDB. (The user supplies these arrays, initialized to their proper values as specified in the System Definition Section of the System/370 Energy Management System Program Reference Manual (SH20-1742). The system generation procedure may be used to invoke the offline utility DPPXUTIL to define the arrays.)

Details of the formats and functions of these arrays are contained in the DATA AREAS section of this manual.

The message definitions necessary for running AGC are contained in member AGCMSGDB of the data set S370EMS.MSGFILE program listings.

Many of these messages are used to provide the necessary text as input to the alarm macro DOMCALRM. They are listed below. For a more detailed explanation of the messages, reference the message by number in Appendix C of the System/370 Energy Management System Program Reference Manual (SH20-1742).

```
DPP750I HH MM SS.T DD/MMM/YY UNABLE TO DO AGC:  NO SYSTEM FREQUENCY
DPP751I HH MM SS.T DD/MMM/YY UNABLE TO DO AGC:  NO EDC OUTPUT
DPP752I HH MM SS.T DD/MMM/YY UNABLE TO DO AGC:  INCOMPLETE DATA SCAN
DPP753I HH MM SS.T DD/MMM/YY UNABLE TO DO AGC:  LACK OF TIE LINE DATA
DPP754I HH MM SS.T DD/MMM/YY GEN. CCCCCCCC OUT OF SERVICE-MISSING DATA
DPP755I HH MM SS.T DD/MMM/YY MULTIPLE AGC INITIALIZATIONS ATTEMPTED
DPP756I HH MM SS.T DD/MMM/YY TIE LINE CCCCCCCC DATA BAD-ON FLAT FREQ.
DPP757I HH MM SS.T DD/MMM/YY AGC CAPABILITY TO RAISE GEN < CRITERION
DPP758I HH MM SS.T DD/MMM/YY AGC CAPABILITY TO LOWER GEN < CRITERION
DPP759I HH MM SS.T DD/MMM/YY AGC COULD NOT CORRECT FOR ECONOMICS
DPP760I HH MM SS.T DD/MMM/YY AGC UNABLE TO CORRECT SAR AT HIGH LIMIT
DPP761I HH MM SS.T DD/MMM/YY AGC UNABLE TO CORRECT SAR AT LOW LIMIT
DPP762I HH MM SS.T DD/MMM/YY UNDEFINED RUNNING MODE IN AGC
DPP763I HH MM SS.T DD/MMM/YY AGC HAS SUSPENDED ITSELF
DPP764I HH MM SS.T DD/MMM/YY AGC CYCLIC PROCESSING NOT INITIALIZED
```

AGC Initialization (DOMALFCI)

For any system containing the AGC application programs, DOMALFCI during system initialization processing.

If control word AILEINIT is set to zero, initialization proceeds and it is set to 1. Otherwise, system message DPP755I is issued and control is returned to the system.

Each entry in array AATLFCIC is initialized to zero. AATLFCIC is an array used to accumulate the sums of the commands issued to the individual generators by the AGC cyclic processor DOMALFCO.

Each entry in array AALFCOUT (AGC output table) (see Figure 6) is initialized to zero.

NPE, the number of power elements in the actual power vector, is calculated as the sum of NG (the number of generators in the system), NT (the number of tie lines), and NNCL (the number of nonconforming loads). The result is placed in AI000NPE, an item of array AAAGCODR. The values for NG, NT, and NNCL were previously placed in items AI0000NG, AI0000NT, and AI00NNCL by system generation.

A list of item names is formed by picking up the NG item names associated with generator power output readings from array AA0INGEN, the NT item names associated with tie line power readings from array AA00INTL, and the NNCL item names associated with readings of power being supplied to non-conforming loads from array AA0INNCL. This list is used as input to the GETITEM macro to provide a list of addresses in array AAACTADR, which is used by the cyclic application programs to pick up the latest actual megawatt readings from the data base.

AA0TEDCA, the EDC output table (see Figure 9) used on an automatic control type entry by systems containing AGC, is next initialized. This table must be initialized for all systems as it is used by AGC regardless of whether or not EDC exists in the system. The first NG items of the table are set to zero to indicate that no generators were included in the last EDC. Items NG+1 through 2NG and items 2NG+1 through 3NG are set to the actual megawatt readings of the generators. The system Lambda (incremental cost of delivered power) is arbitrarily set to $3.00 per kilowatt hour. All other items in the table are initialized to zero.

Control item AIEDCINP is set to zero to indicate to AGC that EDC input values necessary for the running of AGC are not yet available. This item is set non-zero by the EDC programs after a successful automatic type EDC has been performed. It may also be set non-zero by a manual input processing routine in a system that has no EDC program. In this case, the manual input processing routine must first initialize the EDC output table to the values that AGC is to use.

Control item AILFCSSP is set non-zero to suspend AGC. The power system operator must make a manual input request for running AGC in either flat frequency control mode, flat tie-line control mode, or tie-line bias control mode to have AGC begin to function. He may request suspension of AGC at any time.

The prospective frequency bias in AIPROS0K is initialized to be equal to the current frequency bias found in AILFC00K.

The desired interval between occurrences of AGC cyclic processing is picked up from AILFCINT. The PTIME macro is used to cause the AGC cyclic processor, DOMALFCB, to be queued at this user-specified interval. The user has the ability to specify the interval at the time of system generation (the default value is two seconds) and to manually change it during real time operation via supplied displays.

Upon completion of its initialization functions, DOMALFCI completes processing.

AGC Cyclic Processor (DOMALFCB)

The cyclic processor first determines whether the current entry to the program is the first one by checking an indicator fullword, ENT located in the program DOMALFCB. If this is the first entry, the special processing described in the remainder of this paragraph is performed. If control word AILEINIT is set non-zero, processing continues. Otherwise, error message DPP764I is issued stating that AGC processing has not been initialized. The indicator word (ENT) is set to show that the first entry to DOMALFCB has been made. The address of the System/370 time management table is determined and stored away for future use in ATIME. The addresses of various arrays that are needed on this and future entries to DOMALFCB are determined and stored away. The values of NG (the number of generators in the system) NT (the number of tie lines), and NCIS (the number of company interchange schedules) are determined and stored. The addresses of the primary and backup frequency sensor data are computed and stored. The addresses of various data base items that will be needed on this and future entries to DOMALFCB are determined and stored for future use. EDC output table displacements are calculated at this time for use in future cyclic calculations. This completes the special processing required on the first entry to DOMALFCB.

The current settings of the AGC mode adjustment, the AGC frequency bias, frequency offset for time correction, scheduled frequency, proportion of instantaneous area requirement to be included in smoothed area requirement, the dead band limit, the normal mode limit, gain controls, and AGC interval are all recorded in internal storage for use on this first pass through the AGC program. The current time and date are picked up from the data base and stored. The length of the AGC interval is converted to floating point seconds and stored for future use.

The second byte of the continuous monitor failover halfword in AICMFORL is updated by adding one to it and maintaining a count in it modulo 250.

The processor next checks to determine whether there are any entries in the interchange schedule matrix, and, if so, the net scheduled tie flow is computed. Before the computation actually is performed, the schedule for each company in the interchange schedule matrix, AA00ICSM, (see Figure 7) is checked to see if it is time for the scheduled value for that company to be updated. Updating is performed when necessary. The net scheduled tie flow is computed by summing the scheduled tie flow for all companies.

If control word AILFCSSP indicates that AGC has been suspended by the power system operator, the running mode in AALFCOUT is set to zero and control is returned to the system.

The processor then checks to see whether a data scan has been completed since the last time it was entered, and, if not, AGC cyclic processing is not performed. An alarm is activated utilizing message DPP752I to alert the power system operator that the AGC could not be performed due to an incomplete data scan if the data scan was not completed for two consecutive entries to DOMALFCB. Control is then returned to the system.

If control word AIEDCINP indicates that the EDC outputs necessary to run AGC are not yet available, an alarm is activated utilizing message DPP751I and control is returned to the system.

The system frequency is next picked up using either the item name provided in AINSYSF1 or AINSYSF2 and the GETITEM macro. If the data flags indicate that neither the primary nor the backup system frequency is available and we are not on flat tie line control, it is impossible to perform the desired AGC; an alarm is activated utilizing message DPP750I and control is returned to the system.

If the above mentioned checks have been completed in a satisfactory manner, the AGC is performed. The actual frequency is stored in internal storage for future use. A copy of the generator control status table for use on this pass through AGC is stored in a temporary array (AATTTGCS).

A copy of the high economic power limits is stored in array AAPMAXEC. A copy of the high emergency power limits is stored in array AAPMAXEM. A copy of the low economic power limits is stored in array AAPMINEC. A copy of the low emergency power limits is stored in array AAPMINEM. The participation factors computed on the last automatic type EDC entry are copied from array AA0TEDCA (see Figure 8) into array AA00RHOO. The desired economic power settings are copied from array AA0TEDCA into array AA0PDEDC.

The actual data for the generator readings is next obtained. The addresses in array AAACTADR and the GETITEM macro are utilized in accessing the data base. If it happens that the actual data value from a generator whose status code indicates that it is to be given a command by AGC is bad or missing, the power system operator is alerted to this by alarm message DPP754I and the generator status code in array AATTTGCS is changed to out-of-service for this pass through AGC.

The control array AA0ICLFC is initialized based on the generator status codes in array AATTTGCS. The appropriate element is set to 1 for each generator under AGC. The other generators are set to zero. Unit corrections are set to zero in array AALFCUNC for all generators except those that are base loaded. For these units, corrections are computed that drive them to their base points. A maximum size of the allowable unit correction for this type of correction is heeded. The sum of these corrections is stored in RAMPD to be used as an offset to the correction for smoothed area requirement in the main AGC processor (DOMALFCB).

If any tie-lines exist and AGC is not in the flat-frequency control mode, the net actual tie line flow is computed. The addresses in array AAACTADR are used to pick up the actual MW readings from the tie-lines. If an actual value is bad or missing for a tie line that is not marked out of service-self and if AGC is not on flat tie-line control, an alarm is activated utilizing message DPP756I and the program switches to the flat-frequency control mode. Otherwise, the sum of all of the actual tie line values is computed and saved for use by the main AGC processor.

If there were no bad or missing tie line values or if the program is not on flat tie line control, processing continues. If an unusable tie-line value occurs when on flat tie-line control and that tie-line is not marked out of service-self, an alarm is activated using message DPP753I and the program returns control to the system.

The reserve ranges of the generators under automatic generation control are calculated by finding the differences between their present settings and their emergency limits and summing the differences. If the reserve range in either direction is less than a prescribed criterion, an alarm is activated using message DPP757I or DPP758I to warn the power system operator.

The area requirement is computed using the formula

$$A\ R = T\ (\text{Net Act. Tie Flow} - \text{Net Sched. Tie Flow}) - 10.0K\ (F_S + F_O - F_A)$$

where

T = Mode adjustment
K = Frequency bias constant
F_S = Scheduled frequency
F_O = Frequency offset for time corrections
F_A = Actual frequency

The smoothed area requirement is computed using the formula

$$(SAR)_{i+1} = (1-N)(SAR)_i + N(AR)$$

where

N = Proportion of instantaneous value to be used
AR = Area requirement

The area requirement and smoothed area requirement are stored in temporary output table AALFCOTT.

If the absolute value of the smoothed area requirement is greater than the normal mode limit, the emergency assist mode is indicated in AALFCOTT and the unit corrections are set to drive all generators under automatic generation control toward their emergency limit as rapidly as the permissible short-term ramp rate will allow.

If the absolute value of the smoothed area requirement is less than the normal mode limit, the difference between the total current generation of all generation under AGC and the total generation that these same generators had at the time of the last automatic EDC is computed. Indicators which are later used to exclude generators from further calculations are initialized to zero for all generators under AGC. The participation factors to be used are first initialized to those calculated on the last automatic EDC entry. If any generator has been removed from AGC control since the last automatic EDC, the value of all other participation factors is normalized such that their sum is unity.

The EDC calculated desired economic power settings for any generators under AGC control that have not been excluded from further calculations are now adjusted to account for the difference in total generation between the time of EDC and the time of the current AGC. The participation factors are used in this calculation. In making this adjustment, high or low economic limits for each generator are considered. If a limit should be violated, the desired power setting is adjusted to the appropriate economic limit, the value of the difference in total generation is adjusted between the time of EDC and the time of AGC, the generator is excluded from further calculations, its participation factor is set to zero, and the participation factors of all other generators under AGC are normalized. If there are no generators left on which to make changes, an alarm is issued using message DPP759I stating that the program cannot correct for economics and the solely economic correction increment is set to zero for all generators. As soon as the necessary adjustments for the violation of limits by a single generator are made, the calculations described in this paragraph are repeated from the beginning. If none of the generators has violated the limits, the solely economic correction is set to the difference between the desired power setting and the actual power setting for each generator.

The smoothed area requirement is next compared to the dead band limit. If it is less than or equal to this limit, an indicator is set in temporary output table, AALFCOTT, to show that the program is operating in the dead band mode. The unit correction vector is calculated for all generators under control as the product of an operator-controlled gain factor and the solely econcmic correction. Each unit correction is checked against the maximum permissible short-term ramp rate and adjusted if necessary.

If the smoothed area requirement is greater than the dead band limit and less than or equal to the normal mode limit, the normal mode indication is stored in temporary output table, AALFCOTT. The indicators, which will later be used to exclude generators frcm further calculations, are initialized to zero for all generators under AGC. The participation factors to be used are first initialized to those calculated on the last automatic EDC entry. If any generator has been removed from AGC control since the last automatic EDC, the values of all other participaticn factors are normalized such that their sum is unity.

The amount of generation change that will actually be commanded is termed the usable area requirement. It is calculated as the sum of the smoothed area requirement and the generation required by the base loaded units.

For all generators under AGC that have not been excluded from further calculations, a unit correction is computed as the sum cf the product of a gain times the solely economic correction plus a gain times a participation factor times the usable area requirement. The desired power setting is the sum of the actual power setting and the unit correction.

A check is then made to see if the desired power setting of any generator under AGC violates its maximum or minimum economic limits. If permissive control has been selected, desired power settings are checked to see if any of them will cause a generator to be driven in a direction opposite to that of correcting smooth area requirement. Each generator's actual reading is also checked to determine whether any unit is already out of limits. AGC treats the occurrence of these conditions as violations. If a violation has occurred, the desired power setting is set to the appropriate limit (to the actual if a permissive control violation), the usable area requirement adjusted accordingly, the indicator is set to exclude this generator from further calculations, its participaticn factor is set to zero, and other participation factors are normalized to sum to unity. If no generators are left upon which to make the calculations, alarm message DPP760I or DPP761I stating that AGC could not correct for area requirement is initiated. If permissive ccntrol has been selected, unit corrections are also set to zero. If there are generators left upon which to make generation changes, the normal mode computations continue by repeating this paragraph from the beginning. If none of the generators has violated the limits, the normal mode unit correction is set to the difference between the desired power setting and the actual one. Its magnitude is checked against the maximum and minimum permissible ones and adjusted if necessary.

If the dynamic mode was determined to be emergency-assist by the AGC cyclic processor on both this and the previous pass through AGC, and the smoothed area requirement is of opposite sign on this and the previous pass through AGC, an extremely unstable situation exists, and the outputs of AGC are meaningless. The AGC program, therefore, suspends itself, and alarms the power system operator by initiating alarm message DPP763I. It then returns control to the system.

The AGC running mode is then set to AGC suspended, tie line bias control, flat-frequency control, or flat tie line control as a function of the contents of AILFCSSP and the T and K values used on the current pass through the AGC main processor.

Unit corrections are copied from array AALFCUNC into AALFCOTT. Array AALFCOTT is then copied into the AGC output table AALFCOUT (see Figure 6). The unit corrections are sent to the generator by calling program DOMALFCO. Control is then returned to the system.

Byte	Function	Type	Dimension
0	AGC running mode*	F	-
4	AGC dynamic mode**	F	-
8	Instantaneous area requirement	E	MW
12	Smoothed area requirement	E	MW
16	Unit correction for generator #1	E	MW
20	Unit correction for generator #2	E	MW
.	.	.	.
.	.	.	.
.	.	.	.
4NG+12	Unit correction for generator #NG	E	MW

*Running modes

0 = AGC suspended
1 = Tie line bias control
2 = Flat frequency control
3 = Flat tie line control

**Dynamic modes

1 = Dead band mode
2 = Normal mode
3 = Emergency assist mode

Figure 6. AGC output table, AALFCOUT

Byte	Function	Type	Dimension
0	Company name	C	-
8	Current scheduled value	E	MW
12	Desired scheduled value #1	E	MW
16	Date to start schedule change #1	P	Julian
20	Time to start schedule change #1	F	.01 sec
24	Date to stop schedule change #1	P	Julian
28	Time to stop schedule change #1	F	.01 sec
32	Max. Rate of change #1	E	MW/sec
36	Desired scheduled value #2	E	MW
40	Date to start schedule change #2	P	Julian
44	Time to start schedule change #2	F	.01 sec
48	Date to stop schedule change #2	P	Julian
52	Time to stop schedule change #2	F	.01 sec
56	Max. rate of change #2	E	MW/sec

Figure 7. Column from interchange schedule matrix, AA00ICSM

Byte	Function	Type	Dimension
0	Internal generator number	F	-
4	Generator item name	C	-
12	Generator status code*	F	-
16	Emergency-assist regulation capability indicator (base loaded units only)	F	-
20	Desired ramp rate (base loaded units only)	E	MW/sec
24	Desired base load point (base loaded units only)	E	MW
28	Low economic limit (power)	E	MW
32	High economic limit (power)	E	MW
36	Low economic limit (incremental cost)	E	$/MWH
40	High economic limit (incremental cost)	E	$/MWH
44	Low emergency limit	E	MW
48	High emergency limit	E	MW
52	Short term ramp rate	E	MW/sec
56	Long term ramp rate	E	MW/sec
60	Minimum usable rate	E	MW/sec
64	Fuel cost	E	$/MBTU

*Generator status codes

0 = Out of service
1 = Off control (operating independently of Energy Management System)
2 = Base loaded (AGC used to control to a base point)
3 = Economically variable (used for complete EDC only)
4 = On automatic control (under automatic EDC and normal AGC)

Figure 8. Column from generator control status matrix, AA0TGCSM

Byte	Function	Type	Dimension
0	Generator #1 inclusion indicator (0 if not included)	F	-
4	Generator #2 inclusion indicator (1 if included)	F	-
.	.	.	.
.	.	.	.
4(NG-1)	Generator #NG inclusion indicator	F	-
4(NG)	Actual power output of generator #1	E	MW
4(NG+1)	Actual power output of generator #2	E	MW
.	.	.	.
.	.	.	.
4(2NG-1)	Actual power output of generator #NG	E	MW
4(2NG)	Desired power output of generator #1	E	MW
4(2NG+1)	Desired power output of generator #2	E	MW
.	.	.	.
.	.	.	.

4(3NG-1)	Desired power output of generator #NG	E	MW
4(3NG)	Desired participation factor for generator #1	E	-
4(3NG+1)	Desired participation factor for generator #2	E	-
.	.	.	.
.	.	.	.
4(4NG-1)	Desired participation factor for generator #NG	E	-
16(NG)	Total system generation at time of EDC	E	MW
16NG+4	Total power dispatched by EDC program	E	MW
16NG+8	System lambda	E	$/MWH
16NG+12	Date of dispatch	P	Julian
16NG+16	Time of dispatch	F	.01 sec

Figure 9. EDC output table, AA0TEDCA

THIS PAGE INTENTIONALLY LEFT BLANK

AGC Output Interface Processor (DOMALFCO)

DOMALFCO is called by the AGC cyclic processor DOMALFCB. It contains an algorithm which converts the unit corrections for each generator from megawatts to an integer number of 100 millisecond pulses to be transmitted as a generator raise or lower command. The algorithm is:

Number of pulses = A*(UNIT CORRECTION) + B.

The user supplies the values of A and B for each generator using the offline utility program DPPXUTIL.

The program stores its outputs in array AALFCPDO and PATCHes DOMCGENO which causes the requested commands to be sent to the generators.

DOMALFCO also adds the unit correction sent to each generator to the accumulated values in array AATLFCIC.

AGC Operator Interface Processing

AGC operator interface processing provides the capability to monitor and control the AGC application processing with the aid of several displays developed to view and modify data base parameters.

AGC application displays are logically divided into AGC "only" displays and AGC/EDC displays. The AGC menu display will contain an additional entry when an AGC "only" application is desired. This additional entry represents two displays which will be used to initialize/maintain the EDC output table required for AGC processing. In an AGC/EDC system, the EDC output table is maintained by EDC application processing.

AGC MENUs (DOMALMUP): DOMALMUP is PATCHed when a system defined function key is depressed. Display processing will cause the background (static) data to be displayed and PATCH DOMALMUP. DOMALMUP will process the PATCH and update the screen with foreground (dynamic) data and then exit.

The AGC displays are selected for viewing by positioning the cursor on the associated poke points to the left of the text and depressing the system defined function key for cursor entry.

AGC OUTPUT TABLE (DOMALTPG): DOMALTPG is PATCHed when the cursor entry function key is depressed. Display processing will cause the background (static) data to be displayed and PATCH DOMALTPG with a PATCH ID of one.

PATCH ID 1 processing outputs the first page of dynamic data to the requesting unit. Output consists of unit corrections in megawatts for each generator at the first eight plants in the system. Other data output (on each page of the display) consists of running mode, dynamic mode, instantaneous area requirement, smooth area requirement, and the total number of pages required to display AGC output table data.

Paging to the next eight plants and their associated generator data can be accomplished in two ways. First, depressing a system defined function key to page forward (PATCH ID of three) or backward (PATCH ID of four) will cause a PATCH to DOMALTPG. DOMALTPG will output the next (or previous) page of data. Second, positioning the cursor to "F" (forward) or "B" (backward) and depressing the system defined function key for cursor entry will cause the same PATCHes and subsequent output to be displayed.

DOMALTPG is PATCHed with a PATCH ID of fourteen when a display defined program function key for refresh is depressed. PATCH ID fourteen processing outputs the current page's dynamic data to the requesting unit.

MODE CHANGE/TIME CORRECTION (DOMAMTUP, DOMAMTDP, DOMATCOR): DOMAMTUP is PATCHed when the cursor entry function key is depressed. Display processing will cause the background (static) data to be displayed and PATCH DOMAMTUP with a PATCH ID of one.

PATCH ID 1 processing outputs the dynamic data to the requesting unit. Output consists of running mode and frequency bias for mode change parameters. Time correction parameters output consist of date and time to start time correction, date and time to stop time correction, a time speed up or slow down indication, frequency offset, and to delete the current time correction.

DOMAMTUP is PATCHed with a PATCH ID of 7 when the display defined function key for accepting data is depressed. PATCH ID 7 processing by DOMAMTUP consists of processing the data entry change list, storing valid input in the data base, eventing changes and refreshing the display screen. Processing the data entry change list consists of determining which values have been changed and verifying that the values passed limit checking done by Data Entry Services. If not, no update to the data base is made for those values in error.

Before storing mode changes in the data base several checks are made. If any mode other than suspended is requested, there must be at least one generator on automatic control. If not, an error message is written to the requesting unit and suspended mode is selected. If frequency bias is zero (or entered as zero) and either tie line bias or flat frequency running modes are requested, an error message is written to the requesting unit and no update to the data base is made. If the running mode is accepted, an event is generated and a message is written to the requesting unit indicating the requested mode was accepted.

Before initiating a time correction and updating the data base, several checks are made. If dates are not entered, the default date will be today's date. If frequency offset is not entered, the default will be 0.02HZ. Dates and times entered are converted to internal formats and validity checked. Valid dates are today's date and tomorrow's date. Valid times are greater than current time and stop time is greater than start time. The time speed up/slow down parameter must be entered for a time correction to be initiated. If input values are invalid, an event is generated, a message is written to the requesting unit, the data base is not updated and a time correction is not initiated.

DOMATCOR is PATCHed at the date and time to start a time correction and again at the date and time to stop a time correction. When PATCHed, DOMATCOR stores the frequency offset in the data base for use by the AGC main processor. DOMATCOR zeros frequency offset at the date and time to stop time correction.

PATCH ID 5 processing by DOMAMTUP consists of deleting the current time correction by deleting any pending PATCHes to DOMATCOR and zeroing the time correction array in the data base.

SCHEDULED INTERCHANGE (DOMATLPG): DOMATLPG is PATCHed when the cursor entry function key is depressed. Display processing will cause the background (static) data to be displayed and PATCH DOMATLPG with a PATCH ID of 100.

PATCH ID 100 processing outputs the first page of dynamic data to the requesting unit. Output consists of an identifying name and associated position for each interchange company and all of the interchange schedule matrix data for the first company. This data includes company name, current scheduled value, desired schedule value, date and time to start scheduled change, date and time to stop scheduled change, and maximum rate of change for each of two schedules. Also output is the total number of pages which is equal to the total number of companies.

Paging to the next page (company) can be accomplished in two ways. First, depressing a system defined function key to page forward (PATCH ID of 51) or backward (PATCH ID of 50) will cause a PATCH to DOMATLPG. DOMATLPG will output company data for the next (or previous) page. Second, positioning the cursor to "F" (forward) or "B" (backward) and depressing the system defined function key for cursor entry will cause similar PATCHes and subsequent output to be displayed.

Also, positioning the cursor to the position preceding any company name and depressing the cursor entry system function key will cause a PATCH to DOMATLPG which will output company data for that company.

DOMATLPG is PATCHed with a PATCH ID of 52 when the function key for accepting data is depressed. PATCH ID 52 processing by DOMATLPG consists of processing the data entry change list, storing valid input in the data base, eventing changes, and refreshing the display screen. Processing the data entry change list consists of determining which values have been changed and verifying that the values passed limit checking done by Data Entry Services. If not, no update to the data base is made.

Before storing interchange schedule changes in the data base several checks are made. If a request to change the current scheduled value is made, AGC must be in suspended running mode. If not, a message is written to the requesting unit and the current scheduled value remains unchanged. If dates and times are not entered, the dates and times already in the data base are considered to be the desired dates and times. Dates and times entered are converted to internal formats and validity checked. Valid dates are today's date and tomorrow's date. Valid times are greater than current time and stop time is greater than start time. In addition, any time a change is made to either of two interchange schedules the following check is made: (time to stop change - time to start change) X maximum rate of change must be ≥ (|desired scheduled value - current scheduled value|). If any of the above checks indicate invalid data was input, an event is generated, a message will be written to the requesting unit and no schedule changes will take place.

PATCH ID 98 or 99 processing by DOMATLPG consists of zeroing the upper schedule (ID=98) or the lower schedule (ID=99). If the access/function area of the requesting unit is not valid, an error message is issued and the schedule zeroing is bypassed.

GENERATOR CONTROL STATUS MATRIX (DOMAGSPG, DOMAESUS): DOMAGSPG is PATCHed when the cursor entry function key is depressed. Display processing will cause the background (static) data to be displayed and PATCH DOMAGSPG with a PATCH ID of 1.

PATCH ID 1 processing outputs the first page of dynamic data to the requesting unit. Output consists of all of the generator control status matrix data for the first generator and two positions for activating or suspending EDC processing. Generator data includes generator number (from one to N), name, status, emergency assist indicator, ramp rate, base load point, low/high economic power limits, low/high economic incremental cost limits, low/high emergency limits, short/long term maximum ramp rates, minimum usable rate, and fuel cost. Also output is the total number of pages which is equal to the total number of generators.

Paging to the next page (generator) can be accomplished in two ways. First, depressing a system defined function key to page forward (PATCH ID of 3) or backward (PATCH ID of 4) will cause a PATCH to DOMAGSPG. DOMAGSPG will output generator data for the next (or previous) page. Second, positioning the cursor to "F" (forward) or "B" (backward) and pressing the system defined function key for cursor entry will cause similar PATCHes and subsequent output to be displayed.

DOMAGSPG is PATCHed with a PATCH ID of 7 when the function key for accepting data is depressed. PATCH ID 7 processing by DOMAGSPG consists of processing the data entry change list, eventing changes, storing valid input in the data base, refreshing the display screen, or displaying data for a particular generator requested by name and generator number. Processing the data entry change list consists of determining which values have been changed and verifying that the values passed limit checking done by Data Entry Services. If not, no update to the data base is made for the invalid parameters. If a high (or low) economic power limit is entered the corresponding high (or low) economic incremental cost limit is calculated and updated. The reverse is also true when a high (or low) economic incremental cost limit is entered. If the requested limit is not on the current cost curve, a message is written to the requesting unit. If the data entry request is for data from a particular generator, a check is made to verify that the generator number is associated with the plant requested. If not, a message is written to the requesting unit. All parameters that have been processed are evented.

DOMAESUS is PATCHed with a PATCH ID of 2 when the cursor entry function key is depressed (after positioning the cursor on the associated screen position for activating EDC). DOMAESUS is PATCHed with a PATCH ID of 1 when the cursor entry function key is depressed (after positioning the cursor on the associated screen position for suspending EDC). An event is generated and a message is written to the requesting unit indicating EDC was activated, suspended or an error condition exists.

PATCH ID 1 and 2 processing by DOMAESUS consists of verifying the access/function area of the requesting unit. A message is written to the requesting unit if the access/function area is invalid.

PATCH ID 2 processing by DOMAESUS consists of activating EDC by PATCHing immediately with a complete type entry (PATCH ID of 2) and an automatic type entry (PATCH ID of 1), and setting a parameter in the data base indicating that EDC is active. If no generators are on automatic control, then a message will be written to the requesting unit and EDC will not be activated.

PATCH ID 1 processing by DOMAESUS consists of suspending EDC by setting a parameter in the data base indicating that EDC is inactive.

DOMAGSPG is PATCHed with a PATCH ID of 2 when this display is replaced on the screen by another display. This PATCH is required to remove paging information from internal storage for this unit.

AGC GENERAL PARAMETERS (DOMAGPUP): DOMAGPUP is PATCHed when the cursor entry function key is depressed (after positioning the cursor on the associated screen position defined on the AGC menu). Display processing will cause the background (static) data to be displayed and PATCH DOMAGPUP with a PATCH ID of 1.

PATCH ID 1 processing outputs the dynamic data to the requesting unit. Output consists of AGC scheduled frequency, assist mode unit correction gain, dead band mode unit correction economic gain, normal mode unit correction economic gain, normal mode unit correction smooth area requirement gain, percent instantaneous area requirement in smooth area requirement, deadband mode limit, normal mode limit, system response range/lower generation capability criterion, system response range/raise generation capability criterion, and permissive control indicator, AGC interval.

DOMAGPUP is PATCHed with a PATCH ID of 7 when the display defined function key for accepting data is depressed. PATCH ID 7 processing by DOMAGPUP consists of processing the data entry change list, storing valid input in the data base, eventing changes and refreshing the display screen. Processing the data entry change list consists of determining which values have been changed and verifying that the values passed limit checking done by Data Entry Services. If not, no update to the data base is made for invalid parameters. The only special processing done by DOMAGPUP is to convert the AGC interval to an internal format.

MANUAL EDC OUTPUT TABLE (DOMAMTPG): DOMAMTPG is PATCHed when the cursor entry function key is depressed (after positioning the cursor on the associated screen position defined on the AGC only menu). Display processing will cause the background (static) data to be displayed and PATCH DOMAMTPG with a PATCH ID of 1.

PATCH ID 1 processing outputs the first page of dynamic data to the requesting unit. Output consists of an indication of whether the generator is included in EDC calculations, actual power output, desired power output, and desired participation factor for each generator at the first four plants in the system. Other data output (on each page of the display) consists of system lambda, date and time of disPATCH, total system power generation, total power disPATCHed by EDC, total number of pages required to display manual EDC output table (or work table), and several display screen positions and associated text which is described below.

Paging to the next four plants and their associated generator data can be accomplished in two ways. First, depressing a system defined function key to page forward (PATCH ID of 3) or backward (PATCH ID of 4) will cause a PATCH to DOMAMTPG. DOMAMTPG will output the next (or previous) page of data. Second, positioning the cursor to "F" (forward) or "B" (backward) and depressing the system defined function key for cursor entry will cause the same PATCH results.

PATCH ID 10 processing by DOMAMTPG consists of copying the manual EDC output table into a work table and then displaying the work table as described in PATCH ID one processing.

PATCH ID 5 processing by DOMAMTPG consists of copying the work table into the Manual EDC output table eventing this change, and then displaying the manual EDC output table as described in PATCH ID 1 processing.

PATCH ID 6 processing by DOMAMTPG consists of copying the actual power readings from the data base into the work table for each generator and then displaying the work table as described in PATCH ID 1 processing.

PATCH ID 7 processing by DOMAMTPG consists of copying the actual power output into the desired power output for each generator in the work table that is not included in EDC calculations, and then displaying the work table as described in PATCH ID 1 processing.

PATCH ID 8 processing by DOMAMTPG consists of setting participation factors to zero (if the generator is not included in EDC calculations), normalizing the participation factors (for generators included in EDC), and then displaying the work table as described in PATCH ID 1 processing.

PATCH ID 9 processing by DOMAMTPG consists of calculating total system power disPATCHed and total system power, storing their values in the work table, and then displaying the work table as described in PATCH ID one processing.

DOMAMTPG is PATCHed with a PATCH ID of fourteen when a display defined program function key for refresh is depressed. PATCH ID 14 processing outputs the current page's dynamic data to the requesting unit.

EDC CHANGE ENTRY DISPLAY (DOMACEPG): DOMACEPG is PATCHed when the cursor entry function key is depressed (after positioning the cursor on the associated cursor sensitive point defined on the Manual EDC Output Table display). Display processing will cause the background (static) data to be displayed and PATCH DOMACEPG with a PATCH ID of 1.

PATCH ID 1 processing outputs the first page of dynamic data to the requesting unit. Output consists of manual EDC output table values and work table values for each generator. Generator data includes generator number (from 1 to NG), name, status, actual power output, desired power output, participation factor, and an indication of whether the generator is included in EDC calculations. Output also includes system lambda, date and time of disPATCH, and the total number of pages which is equal to the total number of generators. Also, a cursor sensitive point is provided to return to the Manual EDC Output Table display, which will display the entire work table.

Paging to the next page (generator) can be accomplished in two ways. First, depressing a system defined function key to page forward (PATCH ID of 3) or backward (PATCH ID of 4) will cause a PATCH to DOMACEPG. DOMACEPG will output generator data from the manual EDC output table and the work table for the next (or previous) page. Second, positioning the cursor to "F" (forward) or "B" (backward) and depressing the system defined function key for cursor entry will cause similar results.

DOMACEPG is PATCHed with a PATCH ID of 7 when the function key for accepting data is depressed. PATCH ID 7 processing by DOMACEPG consists of processing the data entry change list, storing valid input in the data base work table, refreshing the display screen, or displaying data for a particular generator requested by plant name and generator number. Processing the data entry change list consists of determining which values have been changed and verifying that the values pass limit checking done by Data Entry Services. If not, no update to the data base is made for the invalid parameters. If the data entry request is for data from a particular generator, a check is made to verify that the generator number is associated with the plant requested. If not, an error message is written to the requesting unit informing the user that the requested generator is not associated with the plant requested.

DOMACEPG is PATCHed with a PATCH ID of 2 when this display is replaced on the screen by another display. This PATCH is required to remove paging information from internal storage for this unit.

ECONOMIC DISPATCH CONTROL

Data Base Structure

If a system is to include the EDC programs it must contain all of the arrays listed in section Automatic Generator Control Data Base Structure,, as being necessary for the AGC data base structure. The following additional arrays must also be provided:

1. Arrays established at system generation:

   AA0TEDCC, AAEDCODD, AA00000A, AA00000D, AA00000H, AA00000M, AA00IBMN, AA0GAMMA, AA0INDEX, AA000EXC, AA000RHO, AA000DDP, AA00PTHI, AA00PTLO, AA00000T, AA000BMN, AA00PACT, AA00PDES, AA00PMAX, AA00PMIN, AA0CURVE, AA0ICEDC, AATTGCSM, ALEDCODD

2. Arrays established through DPPXUTIL:

   AA00BMNA, AA00BMNB, AA00BMNC, AA00BMND, AACURVEA, AACURVEB.

The customer supplies the arrays listed under item 2., initialized to their proper values as specified in the System Definition Section of the System/370 Program Reference Manual. The system generation procedure may be used to invoke the offline utility program DPPXUTIL to store these arrays in the data base.

Details of the formats and functions of these arrays are contained in the Data Areas section of this manual.

The additional message definitions necessary for running EDC are contained in member EDCMSGDB of the S370EMS.MSGFILE program listings.

Many of these messages are used to provide the necessary text as input to the alarm macro DOMCALRM. They are listed below. For a more detailed explanation of the messages, reference the message by number in Appendix C of the System/370 Program Reference Manual.

```
DPP701I HH MM SS.T DD/MMM/YY GEN. CCCCCCCC NOT OBEYING AGC
DPP702I HH MM SS.T DD/MMM/YY EDC. COST CURVES DO NOT COVER ECON LIMITS
DPP703I HH MM SS.T DD/MMM/YY AUTO EDC DID NOT CONVERGE
DPP704I HH MM SS.T DD/MMM/YY EDC INITIALIZATION ATTEMPTED BEFORE AGC
DPP705I HH MM SS.T DD/MMM/YY COMP EDC DID NOT CONVERGE
DPP706I HH MM SS.T DD/MMM/YY MULTIPLE EDC INITIALIZATIONS ATTEMPTED
DPP707I HH MM SS.T DD/MMM/YY EDC CYCLIC PROCESSING NOT INITIALIZED
```

<u>EDC Initialization</u> (<u>DOMAEDCI</u>)

For any system containing the EDC application programs, DOMAEDCI is entered in realtime immediately after the AGC initialization program (DOMALFCI) has completed its processing.

DOMAEDCI checks the contents of initialization control item AILEINIT. If it is zero it issues system message DPP704I stating that EDC initialization was attempted before AGC initialization and returns control to the system. If it is not zero or one, it issues system message DPP706I stating that multiple EDC initializations have been attempted and returns control to the system. If AILEINIT is set to one, DOMAEDCI performs EDC initialization and sets the value in AILEINIT to two.

DOMAEDCI computes the number of rows in the BMN matrix AA000BMN (see Figure 10) as the sum of NG (the number of generators), NT (the number of tie lines), and NNCL (the number of non-conforming loads), and stores the result in AI00NROW for use by the cyclic EDC processor.

Control item AIEDCSSP is set non-zero to suspend EDC. The power system operator must make a manual request to activate EDC processing. He may request suspension of EDC processing at any time.

The complete EDC output table, AA0TEDCC, is then initialized. The first NG items in the table are set to zero to indicate that no generators were included in the last EDC. Items NG+1 through 2NG and items 2NG+1 through 3NG are set to the actual megawatt readings of the generators. The system Lambda (incremental cost of delivered power) is arbitrarily set to $3.00 per kilowatt hour. All other items in the table are initialized to zero.

The desired interval between occurrences of automatic dispatch is picked up from AIAEDINT. The user may specify this interval at the time of system generation. The default value is three minutes. The PTIME macro is used to cause the EDC cyclic processor, DOMAEDCB, to be queued at the customer specified interval with an ID of 1 to indicate an "automatic" type entry. The desired interval between complete dispatches is picked up from AICEDINT. The user may specify this interval at the time of system generation. The default value is twenty minutes. The PTIME macro is used to queue DOMAEDCB at this customer specified interval with an ID of 2 to indicate a "complete" type entry.

The program then returns control to the system.

	Gen. # 1		Gen #NG	Tie Line # 1		Tie Line #NT	Non-conforming Load # 1	Non-conforming Load #NNCL
Gen. # 1	$B_{1,1}$	---	$B_{1,NG}$	$B_{1,NG+1}$	---	$B_{1,NG+NT}$	$B_{1,NG+NT+1}$	$B_{1,NG+NT+NNCL}$
⋮	⋮							
Gen. #NG	$B_{NG,1}$		---	---	---	---	---	---
Tie Line # 1	$B_{NG+1,1}$		---	---	---	---	---	---
⋮	⋮							
Tie Line #NT	$B_{NG+NT,1}$		---	---	---	---	---	---
Non-conforming Load # 1	$B_{NG+NT+1,1}$		---	---	---	---	---	---
⋮	⋮							
Non-conforming Load #NNCL	$B_{NG+NT+NNCL,1}$			---	---	---	---	$B_{NG+NT+NNCL,\ NG+NT+NNCL}$

Logical B_{MN} Matrix

	Gen Plant # 1	. . . . . .	Gen. Plant #NP
Gen. Plant # 1	$B_{1,1}$	. . . . . .	$B_{1,NP}$
⋮	⋮		⋮
Gne. Plant #NP	$B_{NP,1}$	. . . . . .	$B_{NP,NP}$
Tie Line # 1	$B_{NP+1,1}$	. . . . . .	$B_{NP+1,NP}$
⋮	⋮		⋮
Tie Line #NT	$B_{NP+NT,1}$	. . . . . .	
Non Conforming Load #1	$B_{NP+NT+1,1}$	. . . . . .	
Non Conforming Load #NNCL	$B_{NP+NT+NNCL\ 1}$	. . . . . .	$B_{NP+NT,NP}$

Stored B_{MN} Matrix

Figure 10. BMN matrix

EDC Cyclic Processor (DOMAEDCB)

DOMAEDCB checks the contents of initialization control item AILEINIT. If it is two normal processing takes place. If it is not two, system message DPP707I is issued stating that EDC cyclic processing has been attempted prior to initializing for it and control is returned to the system.

DOMAEDCB begins normal processing by storing the type entry input parameter in ENTYPE.

If control item AIEDCSSP indicates that EDC is suspended, DOMAEDCB exits without further processing and returns control to the system.

The cyclic processor then determines whether or not the current entry to the program is the first one by checking an indicator word, ENT. If so, the special processing described in the remainder of this paragraph is performed. The address of the System/370 time management table is determined and stored for future use in ATIME. The addresses of various arrays that are needed on this and future entries to DOMAEDCB are determined and stored. The values of NG (the number of generators in the system), NPE (the number of elements in the power vector), NROW (the number of rows in the BMN matrix), and the addresses of the BMN matrix and the cost curve array are determined and stored. EDC output table displacements are calculated at this time for use in cyclic calculations. The indicator word (ENT) is set to show that the first entry to DOMAEDCB has been made.

The current time and date are picked up from the time management table and stored.

The address list that was placed in array AAACTADR by the AGC initialization processor is used in conjunction with the GETITEM macro to pick up the actual power readings for the NG generators, NT tie lines, and NNCL non-conforming loads and store them in array AA00PACT. The signs of the tie line readings are changed. (In AGC flow out of the user's area is considered to be positive but in EDC flow in is considered to be positive.)

Data from the BMN source matrix pointed to by AIBMNPTR is copied into array AA000BMN. Data from the Cost Curve source matrix pointed to by AIOCCPTR is copied into array AA0CURVE. Data from the generator control status matrix AA0TGCSM (see Figure 8) is copied into AATTGCSM. The cost curve data in AA0CURVE (see Figure 11) is updated to reflect the latest fuel costs for each generator as shown in AATTGCSM.

Data from AATTGCSM is copied into AA00PMIN, and AA00PMAX. The actuals from AA00PACT are copied into AA00PDES for use as the first estimate of the economic dispatch solution.

If the entry type was automatic, then the logic described in the remainder of this paragraph is performed. The value of the system Lambda, the incremental cost of delivered power, is initialized from the last EDC output value in the automatic type output table AA0TEDCA. Total system generation is computed by summing the generations in the AA00PACT array for NG generators. For any generator whose generator status code in array AATTGCSM is 4, the appropriate word in the AA0ICEDC array is set to 1 to indicate that the generator is to be dispatched. The total generation to be dispatched is computed by summing the actual power readings from the generators to be dispatched. For any generator whose generator status code in array AATTGCSM is not 4, the appropriate word in AA0ICEDC is set to zero to indicate that the generator is not to be dispatched. Finally, a check is performed to see whether or not the individual generators that are under automatic control are following their AGC commands. To do this, the EDC preprocessor first copies the integrated commands provided by AGC in array AATLFCIC into a temporary storage array. Array AATLFCIC is then reset to zero to allow AGC to begin its next accumulation cycle. The check is performed on all generators that were under automatic control on both the previous and the current automatic EDC entry. If the ratio of the actual change in generation to the commanded change in generation for any generator is less than a given criterion, an alarm is issued using message DPP701I to alert the power system operator that that generator is not obeying its AGC commands.

If the entry type is complete, the logic described in the remainder of this paragraph is performed. The value of the system Lambda, the incremental cost of delivered power, is initialized from the last EDC output value in the complete type output table AAOTEDCC. Total system generation is computed by summing the generations in the AA00PACT array for all NG generators. For any generator whose generator status code in array AATTGCSM is 3 or 4, the appropriate word in array AA0ICEDC is set to 1 to indicate that the generator is to be dispatched. The total generation to be dispatched is computed by summing the actual power readings from the generators to be dispatched. For any generator whose generator status code in array AATTGCSM is neither 3 nor 4, the appropriate word in array AA0ICEDC is set to zero to indicate that the generator is not to be dispatched.

The error code in ERRCODE is initialized to zero. Parameters are initialized such that up to 20 iterations will be allowed in an attempt to get a converged value for Lambda, the system incremental cost of delivered power. The value is considered to be converged when two successive values are within a prescribed tolerance of each other. The iteration proceeds as follows:

∂_i , the reciprocal of the penalty factor for generator i, is computed for each generator based on the desired power vector and the BMN coefficient matrix. It is stored in array AA0GAMMA. D, the main diagonal of the portion of the Bmn matrix associated with generators, is extracted and stored in array AA00000D.

Values are next computed for and stored in the following arrays:

- H - incremental cost of power at generators (from cost curves) - AA00000H
- A - rate of change of H (from cost curves) - AA00000A
- M - cost difference ($M_i = \lambda\gamma_i - H_i$) - AA00000M.

For generators not under EDC, these values are set to zero. For a generator under EDC whose power reading is outside its economic limit: the desired power reading in array AA00PDES is set to the appropriate limit, and the corresponding values in arrays AA00000H and AA00000A are computed before the value of the cost difference M_i is computed. If the desired power reading is at its upper economic limit and M_i is greater than 0, or if the desired power reading is at its lower economic limit and M_i is less than 0, or if the generator is not under EDC control, the appropriate AA000EXC array element is set to 0 to exclude this generator from further calculations. Otherwise, the AA000EXC array element is set to 1 to include the generator.

The maximum allowed desired power value to be computed on this iteration is calculated as the lesser of the far end of the next higher cost curve line segment and the upper economic limit and stored in array AA00PTHI. The minimum allowed desired power value to be computed on this iteration is calculated as the greater of the far end of the next lower cost curve line segment and the lower economic limit and stored in array AA00PTLO. If an error occurs due to an inconsistency between the economic limits in use and the cost curves in use, the error code is set to 1 and the program exits through the EDC post processor DOMAEDCA.

N, the demand difference, is next computed by the formula

$$N = 2\, d_{EDC} - \Sigma P_{DES_i}$$

where d_{EDC} = demand at assumed system load center (computed in DCMAEDCB)

P_{DES_i} = desired power reading

and the summation is over all generators that are to be included in this dispatch.

The exclusion of the final generator from further calculations is overriden if appropriate. If all generators have been marked to be excluded and $/\Sigma P_{DES}/$ is too large then the generator with the maximum cost of delivered power is included. If all generators have been marked to be excluded and $/\Sigma P_{DES}/$ is too small then the generator with the minimum cost of delivered power is included.

$\Delta\lambda$, the desired change in system λ is computed next using only those generators whose readings the AA000EXC array indicates may be changed. The matrix formula used is:

$$\Delta\lambda_i = \frac{N - 2\underline{\bar{\gamma}}\ (A + 2D\lambda\ )^{-1}\ \underline{\bar{M}}}{2\bar{\gamma}\ (A + 2D\lambda\ )^{-1}\bar{\gamma}}$$

This and other formulas used in this section are derived in the System/370 Energy Management System Program Reference Manual.

$\Delta\underline{\bar{P}}$, the incremental change to the desired power settings for each generator subject to change, is then computed and added to the corresponding element of the AA0OPDES array.

$$\Delta\underline{\bar{P}} = (A + 2\lambda\ \ D)^{-1}\ (\underline{\bar{M}} + \Delta\lambda\bar{\gamma}\ )$$

If the new setting in AA0OPDES violates the maximum allowed power in AA0OPTHI or the minimum allowed power in AA0OPTLO, it is set to the appropriate value.

The system Lambda is then updated by the formula

$$\lambda = \lambda + \Delta\lambda$$

This completes a pass through the iteration.

If the error code has been set non-zero, the program exits through the post-processor DOMAEDCA. If twenty iterations have been attempted and $/\Delta\lambda/$ is still greater than the allowable tolerance, or if for any generator $/\Delta f_i/$ is greater than the allowable tolerance, the error code is set to 2 and the processor exits through DOMAEDCA. If less than twenty iterations have been attempted and $/\Delta d\ /$ is greater than the allowable tolerance, or if for any generator $/\Delta f_i/$ is greater than the allowable tolerance, another iteration is attempted. If $/\Delta\lambda/$ and each $/\Delta f_i/$ are less than or equal to the prescribed tolerances, participation factors are computed for all generators under economic disPATCH control using the formula

$$P_i = \frac{\gamma_i / A_i}{\Sigma\gamma_i / A_i}$$

These are placed in array AA000RHO. The appropriate elements of this array are set to zero for any generator not under economic disPATCH control.

Upon completion of the participation factor computation, the program uses the participation factors to adjust the mismatch between the sum of the desired power settings and the sum of the actual power settings for all generators under EDC.

The program next determines whether this entry request was for an automatic or complete economic dispatch.

If the error code is zero, the output is stored in either the automatic type output table, AA0TEDCA, or the complete type output table, AA0TEDCC. The quantities stored are the time and date of the disPATCH, the total system generation, the total power disPATCHed, the system Lambda, an indication of whether or not each generator was disPATCHed, the actual power reading for each generator, the economic desired power reading for each generator, and the participation factor for each generator.

When a successful automatic type EDC has been performed, AIEDCINP is set to 1 to indicate this fact to the AGC programs.

If the error code were set to 1 by the cyclic processor, then alarm message DPP702I stating that the cost curves (see Figure 12) do not cover the economic limits and implying that the automatic (or complete) dispatch was not performed is issued for the power system operator's information.

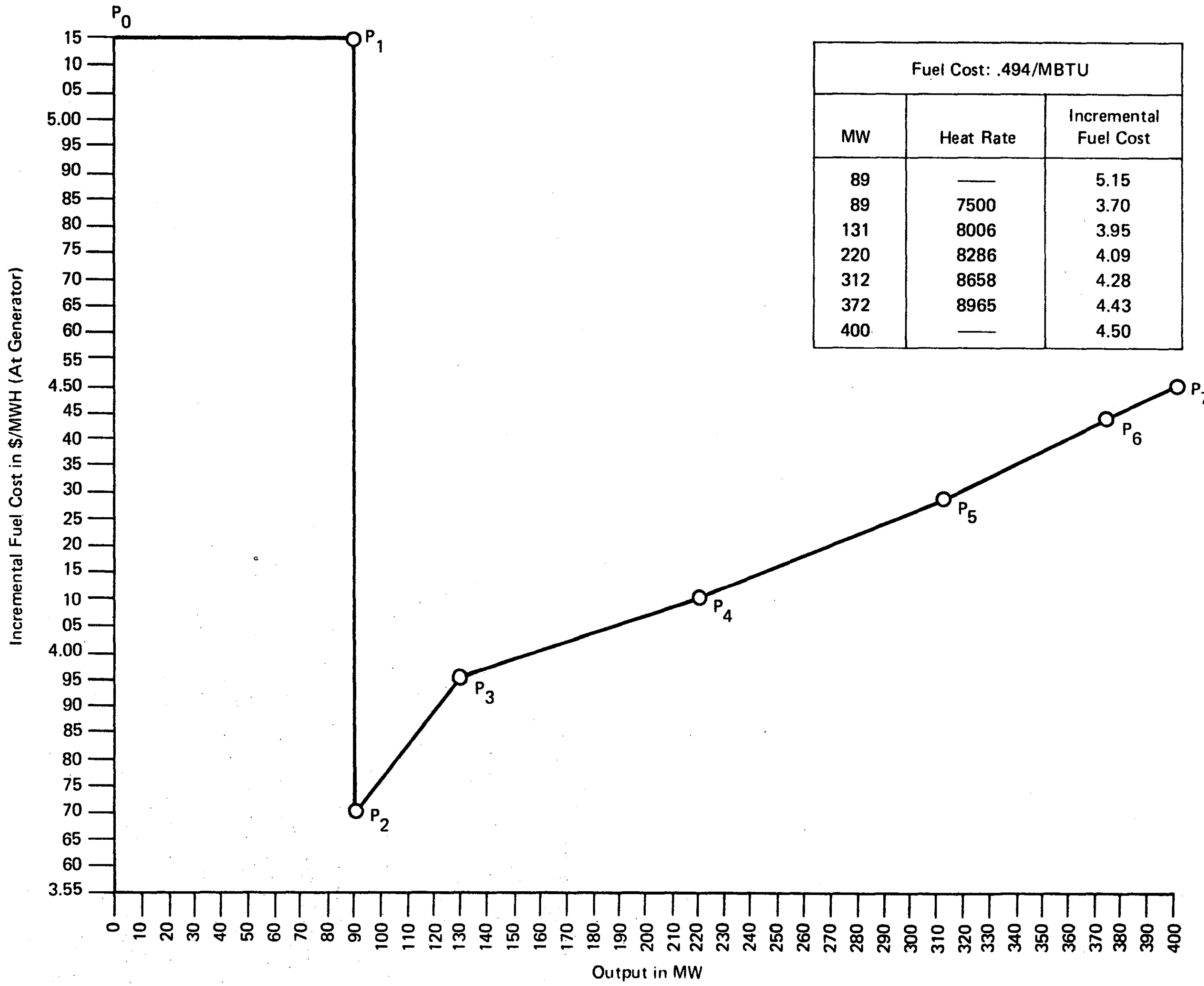
If the error code were set to 2 by the cyclic processor, then alarm message DPP703I or DPP705I stating that the automatic (or complete) EDC did not converge is issued.

Thus, if there is an error in EDC, the output table is not updated or disturbed. AGC continues to use the old values for its computations.

Upon completion of its functions, DOMAEDCB completes processing.

Byte	Function	Type	Dimension
0	Internal generator number	F	-
4	Generator output at P2	E	MW
8	Generator output at P3	E	MW
12	Generator output at P4	E	MW
16	Generator output at P5	E	MW
20	Generator output at P6	E	MW
24	Generator output at P7	E	MW
28	Incremental cost at P2	E	$/MWH
32	Incremental cost at P3	E	$/MWH
36	Incremental cost at P4	E	$/MWH
40	Incremental cost at P5	E	$/MWH
44	Incremental Cost at P6	E	$/MWH
48	Incremental cost at P7	E	$/MWH
52	Slope of line segment 1 (Through P2 and P3	E	($/MWH)/MW
56	Slope of line segment 2 (Through P3 and P4	E	($/MWH)/MW
60	Slope of line segment 3 (Through P4 and P5	E	($/MWH)/MW
64	Slope of line segment 4 (Through P5 and P6	E	($/MWH)/MW
68	Slope of line segment 5 (Through P6 and P7	E	($/MWH)/MW
72	Y - intercept of line segment 1	E	$/MWH
76	Y - intercept of line segment 2	E	$/MWH
80	Y - intercept of line segment 3	E	$/MWH
84	Y - intercept of line segment 4	E	$/MWH
88	Y - intercept of line segment 5	E	$/MWH
92	Incremental cost from P0 to P1	E	$/MWH
96	Fuel cost	E	$/MBTU

Figure 11. Column from cost curve matrix, AA0CURVE



Fuel Cost: .494/MBTU		
MW	Heat Rate	Incremental Fuel Cost
89	——	5.15
89	7500	3.70
131	8006	3.95
220	8286	4.09
312	8658	4.28
372	8965	4.43
400	——	4.50

Figure 12. Incremental cost curve

EDC Operator Interface Processing

EDC operator interface processing provides the capability to monitor and control the EDC application processing with the aid of several displays developed to view and modify data base parameters.

The EDC displays are selected for viewing by positioning the cursor on the associated screen position to the left of the text and depressing the system defined function key for cursor entry.

AUTOMATIC EDC OUTPUT TABLE (DOMAATPG); DOMAATPG is PATCHed when the cursor entry function key is depressed. Display processing will cause the background (static) data to be displayed and PATCH DOMAATPG with a PATCH ID of 1.

PATCH ID 1 processing outputs the first page of dynamic data to the requesting unit. Output consists of an indication of whether the generator is included in EDC calculations, actual power output, desired power output, and desired participation factor for each generator at the first four plants in the system. Other data output (on each page of the display) consists of system lambda, date and time of dispatch, total system power generation, total power dispatched by EDC, and total number of pages required to display the automatic EDC output table.

Paging to the next four plants and their associated generator data can be accomplished in two ways. First, depressing a system-defined function key to page forward (PATCH ID of 3) or background (PATCH ID of 4) will cause a PATCH to DOMAATPG. DOMAATPG will output the next (or previous) page of data. Second, positioning the cursor to "F" (forward) or "B" (backward) and depressing the system defined function key for cursor entry will cause similar results.

DOMAATPG is PATCHed with a PATCH ID of 14 when a display defined program function key for refresh is depressed. PATCH ID 14 processing outputs the current pages dynamic data to the requesting unit.

COMPLETE EDC OUTPUT TABLE (DOMACTPG): DOMACTPG is PATCHed when the cursor entry function key is depressed. Display processing will cause the background (static) data to the displayed and PATCH DOMACTPG with a PATCH ID of 1.

PATCH ID 1 processing outputs the first page of dynamic data to the requesting unit. Output consists of an indication of whether or not the generator is included in EDC calculations, actual power output, desired power output, and desired participation factor for each generator at the first four plants in the system. Other data output (on each page of the display) consists of system lambda, date and time of dispatch, total system power generation, total power dispatched by EDC, and total number of pages required to display the complete EDC output table.

Paging to the next four plants and their associated generator data can be accomplished in two ways. First, pressing a system defined function key to page forward (PATCH ID of 3) or backward (PATCH ID of 4) will cause a PATCH to DOMACTPG. DOMACTPG will output the next (or previous) page of data. Second, positioning the cursor to "F" (forward) or "B" (backward) and depressing the system defined function key for cursor entry will cause similar results.

DOMACTPG is PATCHed with a PATCH ID of 14 when a display defined program function key for refresh is depressed. PATCH ID 14 processing outputs the current pages dynamic data to the requesting unit.

EDC INTERVALS/COSTS/LOSSES (DOMAICUP): DOMAICUP is PATCHed when the cursor entry function key is depressed. Display processing will cause the background (static) data to be displayed and PATCH DOMAICUP with a PATCH ID of 1.

PATCH ID 1 processing outputs the dynamic data to the requesting unit. Output consists of automatic EDC execution time interval, complete EDC execution time interval, cost curve pointer, and BMN matrix pointer.

DOMAICUP is PATCHed with a PATCH ID of 7 when the display defined function key for accepting data is depressed. PATCH ID 7 processing by DOMAICUP consists of processing the data entry change list, storing valid input in the data base, eventing changes, refreshing the display screen, and PATCHing the EDC main processor for time interval changes. Processing the data entry change list consists of determining which values have been changed and verifying that the values passed limit checking done by Data Entry Services If not, no update to the data base is made.

Before making time interval changes the time intervals are converted to an internal format. Current PTIME PATCHes are modified to the new interval(s) and an immediate PATCH is issued in the case of an automatic time interval change.

If the entered value is a cost curve pointer change the following processing is performed. A check is made to verify if all high and low economic power limits in the generator control status matrix lie on the requested cost curve. If not, an error message is issued stating the item name of the first offending generator. (If he desires to do so the power system operator would at this point call up the generator control status matrix display and change the power limits on the offending generator to values which are on the cost curve he wishes to use.) If all power limits lie on the requested curve, the corresponding incremental cost limits are computed and stored in the generator control status matrix and the cost curve pointer is changed to reflect the requested cost curve. All parameters modified or attempted to be modified are evented.

Special Real Time Operating System and Display Management System

The S/370 Energy Management System is dependent upon two Programming RPQ'S for its operation. The two Programming RPQ'S required are the Special Real Time Operating System which enhances OS/VS1 services to support realtime applications, and the Display Management System which provides an operator interface for the 5985 color display stations as well as 3270 Displays. In the text that follows the functions of these PRPQ's as used by the S/370 Energy Management System and additionally available capabilities.

S/370 Energy Management System uses of the Special Real Time Operating System:

- Data base definition, initialization, and management
- Asynchronous tasking (independent)
- Time dependent task creation/queuing
- Realtime message handler
- Data base logging

S/370 Energy Management System uses of the Display Management System:

- Display definition of backgrounds, foregrounds, miats and character font.
- Activation of a display on a display unit or report a display to a printer unit.
- Reading and writing of data from an active display.
- Request the execution of an application program and generate a parameter list through the console keyboard to be passed to the program.
- Request a hardcopy of the contents of an active display to an IBM 3284 or 3286 printer.
- Data entry

Additional capabilities available to a user of the S/370 Energy Management System:

- The creation or modification of a display number in a realtime mode of operation (COMPOSE)

- Program activation via a key interrupt

  Example: A user defines a unique display that is to serve as an input reference point for some new functions he may want to perform. This can be accomplished by DMIAT/ Patch statements in his display miat member or (manual input action table) which allow his program to receive control with register one pointing to a three word input parm list, where the input parms are as follows:

```
Reg 1---> 0      A(DPPXCUT)
          4      A(RESOURCE TABLE)
          8      A(PATCH PARMS)
```

  From the above paramater list the user program has access to many of the tables used by the S/370 Energy Management System.

CHAPTER 4. PROGRAM DESIGN LANGUAGE

Program Design Language (PDL) is used to describe the System/370 Energy Management System program logic. The sections entitled "Purpose", "Syntax Definition", "Semantics Definition" and "Using PDL" describe the general scope of a Program Design Language. The section entitled "Program Design Language Module Listings" lists each PDL module in alphameric order. The section entitled "Directory" is used to cross reference each PDL module to its respective section write-up and method of operation diagrams.

Program Design Language (PDL) provides for natural expression of procedural definitions (programs) at a level of completeness and detail appropriate to the designer's current knowledge of requirements. As understanding grows over time and design parameters and trade offs become apparent, PDL text can evolve, in top down fashion, toward more sophisticated representations which themselves are readily translated into the implementation (programming) language of choice.

At any level (corresponding to time sequence of development) of the evolving system tree structure, this process is completed prior to commitment to the expense of machine implementation. In this connection, PDL together with the corresponding functional specification serves as adequate "as designed" documentation.

PDL thus produced should be considered for inclusion under the control of librarians. When kept up-to-date during development, it then becomes, in conjunction with implementation language code text, a major portion of final documentation.

SYNTAX DEFINITION

RESERVED WORDS

The following keywords are reserved for expressing PDL syntactical structure and may not be used for other purposes:

```
SEGMENT          DO
ENDSEGMENT       WHILE
MAIN             UNTIL
SUBROUTINE       FROM
INCLUDED         TO
DATA             BY
                 ENDDO
INCLUDE
CALL             CASENTRY
                 CASE
GET              ENDCASE
PUT

IF
THEN             TRUE
ELSE             FALSE
ENDIF            UNDEFINED
                 BOOLEAN
                 NUMERIC
                 CHARACTER
                 FILE
```

SEGMENT NAMES

These may be any number of characters and words in length. Multiple-word segment names are written with one blank between words.

Example: ACCOUNTS RECEIVABLE COMPUTATION
Example: TEXT COMPOSITION
Example: LEAST SQUARES
Example: EDITOR

SEGMENT DELIMITATION

PDL code segments are delimited as one-in/one-out units according to the following format. Vertical arrows represent PDL text, and denote the standard indentation convention of two columns.

segment name segment type <u>SEGMENT</u>

ENDSEGMENT segment name

Where,

- Segment name is the descriptive name written as described in the section entitled "Segment Names".
- Segment type is a keyword denoting one of four categories to which the segment belongs. The first three keywords below define segments containing executable code and no data definitions; the last is employed for data segments:

Keyword	Description
MAIN	A segment invoked by the operating system as the control program of the application system.
SUBROUTINE	A segment invoked during execution. This transfer of control generally requires operating system services, and the segment may or may not be in core when it is invoked.
INCLUDED	An executable segment which would be inserted into the text of another segment during assembly or compilation, prior to execution, as specified by the INCLUDE segmentname statement.
DATA	A segment composed entirely of data element definitions, and made available to the corresponding executable segment segment through the INCLUDE segmentname statement.

Example: TEXT COMPOSITION <u>SUBROUTINE</u> <u>SEGMENT</u>

<u>ENDSEGMENT</u> TEXT COMPOSITION

Example: TEXT COMPOSITION DATA1 <u>DATA</u> <u>SEGMENT</u>

<u>ENDSEGMENT</u> TEXT COMPOSITION DATA1

DATA ELEMENT NAMES

These may be any number of characters and words in length. Multiple-work data element names are written with a break (underline) character between words.

Example: DELTA_TIME_1
Example: RHO_SUB_A
Example: LOWER_LIMIT
Example: STEPSIZE

DATA DEFINITION SEGMENTS

Where data definitions are required within an executable segment (that is, segmenttype is MAIN, SUBROUTINE, or INCLUDED-executable), two parallel segments should be created, one for executable text, one for associated data. All data elements must be defined as follows:

```
segmentname DATA SEGMENT

    element name    element type
    element name    element type

ENDSEGMENT segmentname
```

where elementname is as defined in the section entitled "Data Element Names", with optional numeric array dimensions enclosed in parentheses. Elementnames should be alphabetized. Structured data may be represented by indenting subordinate data two columns. Element type may be one of four keywords:

Keyword	Description
NUMERIC	Specifies fixed or floating point binary or decimal data.
CHARACTER	Specifies data which is a continuous string of characters.
BOOLEAN	Specifies an on/off data element to which the value keywords TRUE, FALSE, or UNDEFINED can be assigned.
FILE	Specifies that the associated element is a file name.

Example:

```
TEXT COMPOSITION DATA1 DATA SEGMENT

ALPHA               FILE

BETA(10,10)         CHARACTER

GAMMA(60)           NUMERIC

DELTA               BOOLEAN

ENDSEGMENT TEXT COMPOSITION DATA1
```

LITERAL DATA

PDL text may contain data values introduced directly in the right-hand side of assignment statements, as follows:

Category	Description	Examples
Numeric	Any base or scale may be used.	10.17 Ø 2.935E+02
Character	Any string of characters may be specified, enclosed in double quotes.	"ABC" "370/155" "7_X"
Boolean	PDL provides standard keywords for use here.	TRUE FALSE UNDEFINED

SEGMENT CONTROL STRUCTURE

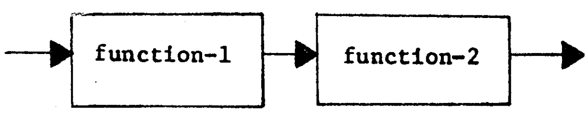
Control logic within a segment may be expressed only in terms of the structure theorem program figures,

- Sequence of operations
- IFTHENELSE
- DO-group

plus the CASE figure, which can of course be represented as nested IFTHENELSE constructions, but which is of direct utility in its own right, and a reasonable addition to the fundamental triad. Refer to the section entitled "Semantics Definition" for a discussion of the semantics of function, predicate, value, and value-list as used below.

Sequence of Operations Figure

Flowchart representation:

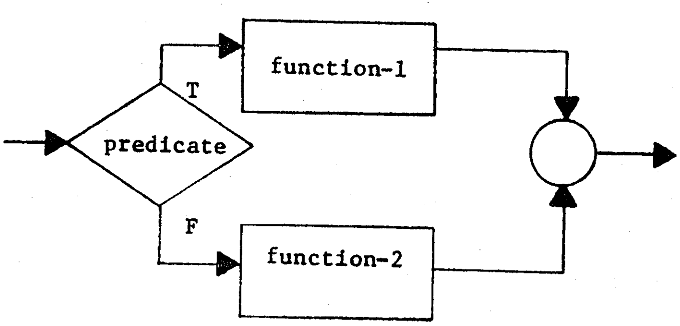


PDL text representation:

```
function-1
function-2
```

IFTHENELSE Figure

Flowchart representation:



function-1

PREDICATE

function-2

PDL text representation:

```
IF predicate THEN

    function-1
ELSE
    function-2

ENDIF
```

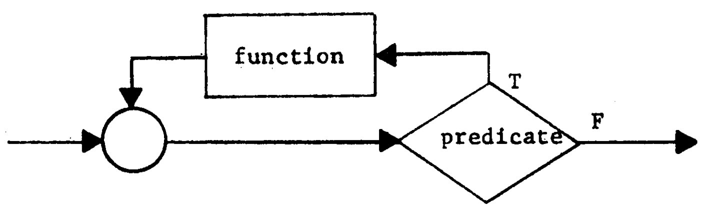
Here the ELSE clause is optional; its omission gives the IFTHEN figure.

DO-group

Three forms of the DO-group are specified; they may be used individually or in combination.

1. DO WHILE form

Flowchart representation:



PDL text representation:

```
DO WHILE predicate

  function

ENDDO
```

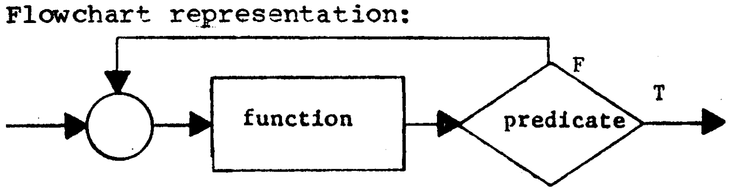
```
DO WHILE ALPHA=BETA

  function

ENDDO
```

Example:

ENDDO

2. DO UNTIL form

Flowchart representation:



PDL text representation:

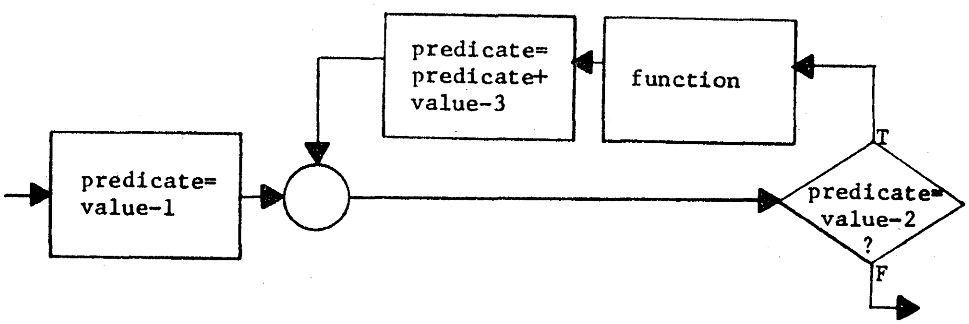
```
DO UNTIL predicate
   function
ENDDO
```

Example:

```
DO UNTIL END_OF_FILE
   function
ENDDO
```

3. DO with iteration form

Flowchart representation:



PDL Text representation:

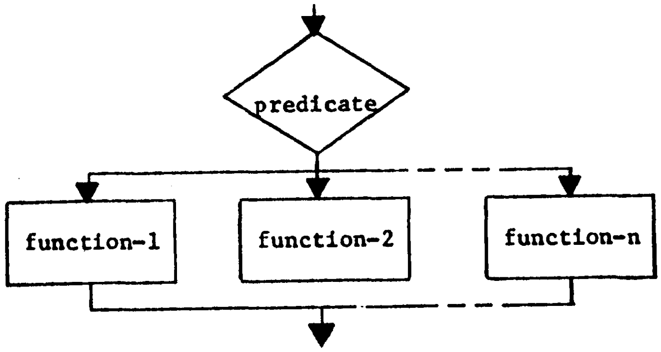
```
DO predicate FROM value-1 TO value-2 BY value-3
   function
ENDDO
```

Example:

```
DO HOUR FROM 1 TO 24 BY STEP
   function
ENDDO
```

CASE-group

Flowchart representation:



PDL text representation:

```
CASENTRY predicate

    CASE value-list-1

        function-1

    CASE value-list-2

        function-2
            •
            •
            •
    CASE value-list-n

        function-n

ENDCASE
```

Where value-list is a series of predicate values for which the given CASE is to be executed. Predicate values which do not appear in a value list prevent the CASE figure from being executed; execution resumes with the statement following ENDCASE.

Example:

```
CASENTRY HOUR

    CASE 1,2,3,4

        function-1

    CASE 8,9,12,14,19

        function-2

    CASE 5,24,23

        function-3

ENDCASE
```

INPUT/OUTPUT

Two keywords are employed to specify transfer of data to and from external files. The PDL text representations are:

GET filename function

PUT filename function

Where filename is any elementname defined in a DATA segment as a file, and function is discussed in the section entitled "Semantics Definition". The following examples omit use of function.

Example: GET INPUT_FILE

PUT OUTPUT_FILE

TOP DOWN EVOLUTION

Tree structures of PDL segments are naturally elaborated from level to level through segment inclusion (INCLUDE statement) and specification of dynamic invocation of segments (CALL statement).

Included Segments

PDL text representation:

INCLUDE segmentname

which denotes presence of an instruction to the compiler or assembler to physically replace the INCLUDE statement with the named segment.

Called Segments

PDL text representation:

CALL segmentname

which denotes presence of a statement which causes invocation of the named subroutine when the statement is executed.

SEMANTICS DEFINITION

SEMANTIC UNITS

The previous definition of PDL syntax utilized a number of semantic units to denote operations performed by code segments:

Semantic Unit	Source
Predicate	IFTHENELSE DO-group CASE
Function	Sequence of Operations IFTHENELSE DO-group CASE GET PUT
Segmentname	SEGMENT/ENDSEGMENT
Elementname	DATA SEGMENT
Value	DO-group with iteration
Value List	CASE

A major portion of the information content or meaning of PDL segments is expressed in terms of these semantic units. The choice left to the program designer is based on the goal of maximum understandability for transfer of technical information between diverse groups of managers, programmers, and users.

EXPRESSING MEANING IN SEMANTIC AND SYNTACTIC UNITS

The fundamental expression of segment meaning in this sense proceeds as follows:

Semantic units may be expressed as:

1. Natural language enclosed in single quotes
2. Statements of the implementation language of choice
3. Combinations of 1 and 2

This form of expression is broadened to encompass the top down procedural elaboration of segments by allowing both syntax and semantics of PDL to be written as defined above. That is, a designer need not code an IFTHENELSE or DO if he is not sure of the construction of that portion of the segment control structure, but need only express the operation in terms of a natural language statement. Later, the statement may be elaborated as an IFTHENELSE, DO, or INCLUDE, or may be totally revised as new information on requirements becomes available.

USING PDL

PDL is a natural representation vehicle for the creative problem-solving process of writing programs. Most importantly, it provides evolving representations, from general first-approximation procedures, to completely detailed solutions ready for translation into the implementation of language of choice. The following example illustrates such evolution.

FUNCTIONAL SPECIFICATIONS

Exactly 100 numbers ranging in value from -1 to 1000 are to be processed. Values from 0 to 499 are to be summed and printed, as are values from 500 to 1000. If a negative value is encountered, summation must be stopped and the partial sums printed.

PDL Version 1

EXECUTABLE SEGMENT

```
SUM MAIN SEGMENT
   INCLUDE SUM DATA
   'INITIALIZATION'
   'OPEN FILES'
   GET INPUT_FILE 100 NUMBERS
   'DETERMINE SUMS, TERMINATE IF NEGATIVE NUMBER FOUND'
   PUT OUTPUT_FILE 'SUMS'
   'CLOSE FILES'

ENDSEGMENT SUM
```

DATA SEGMENT

```
SUM DATA DATA SEGMENT
   INPUT_FILE        FILE
   NUMBERS(100)      NUMERIC
   OUTPUT_FILE       FILE

ENDSEGMENT SUM DATA
```

PDL Version 2

EXECUTABLE SEGMENT

```
SUM MAIN SEGMENT
   INCLUDE SUM DATA
   'INITIALIZATION'
   'OPEN INPUT_FILE, OUTPUT_FILE'
   GET INPUT_FILE, OUTPUT_FILE'
   DO 'FOR EACH VALUE IN NUMBERS' WHILE NOT_NEGATIVE
       IF 'CURRENT VALUE <0' THEN
           NOT_NEGATIVE = FALSE
       ELSE
           IF 'CURRENT VALUE IN LOW RANGE' THEN
               'CONTINUE LOW RANGE SUM'
       ELSE
           'CONTINUE HIGH RANGE SUM'
       ENDIF
   ENDIF
ENDDO
PUT OUTPUT_FILE 'SUMS'
'CLOSE INPUT_FILE, OUTPUT_FILE'

ENDSEGMENT SUM
```

```
DATA SEGMENT

SUM DATA DATA SEGMENT
   INPUT_FILE           FILE
   NOT_NEGATIVE         BOOLEAN
   NUMBERS (100)        NUMERIC
   OUTPUT_FILE          FILE

ENDSEGMENT SUM DATA

PDL Version 3

EXECUTABLE SEGMENT
SUM MAIN SEGMENT
   INCLUDE SUM DATA
   NOT_NEGATIVE = TRUE
   LOW_RANGE_SUM = 0
   HIGH-RANGE_SUM = 0
   'OPEN INPUT_FILE, OUTPUT_FILE'
   GET INPUT_FILE NUMBERS
   DO I FROM 1 TO 100 BY 1 WHILE NOT_NEGATIVE
       IF NUMBERS (I) < 0 THEN
           NOT_NEGATIVE = FALSE
       ELSE
           IF NUMBERS (I) < = 499 THEN
               LOW_RANGE_SUM = LOW_RANGE_SUM + NUMBERS (I)
       ELSE
           IF NUMBER (I) < = 1000 THEN
               HIGH_RANGE_SUM = HIGH_RANGE_SUM + NUMBERS (I)
       ELSE
           PUT OUTPUT_FILE "INPUT VALUE BEYOND RANGE"
           NOT_NEGATIVE = FALSE
       ENDIF
   ENDIF
 ENDIF
ENDDO
PUT OUTPUT_FILE LOW_RANGE_SUM, HIGH_RANGE_SUM
'CLOSE INPUT_FILE, OUTPUT_FILE'

DATA SEGMENT

SUM DATA DATA SEGMENT
   HIGH_RANGE_SUM           NUMERIC
   INPUT_FILE               FILE
   LOW_RANGE_SUM            NUMERIC
   NOT_NEGATIVE             BOOLEAN
   NUMBERS (100)            NUMERIC
   OUTPUT_FILE              FILE

ENDSEGMENT SUM DATA
```

PROGRAM DESIGN LANGUAGE MODULE LISTINGS

This section lists, in alphameric order, the program design language (PDL) for each module in the System/370 Energy Management System.

```
MEMBER NAME  DOMAPDLE
 * DOMAPDLE CONTAINS THE PROGRAM DESIGN LANGUAGE ASSOCIATED WITH THE
 * ECONOMIC DISPATCH CONTROL (EDC) APPLICATION.
 **
 **
 *         DOMAEDCI PERFORMS THE INITIALIZATION REQUIRED BY THE CYCLIC
 *               EDC PROCESSOR DOMAEDCB.
 *
 *         BEGIN DOMAEDCI;
 *         IF LFC INITIALIZATION HAS BEEN DONE SINCE THE LAST RESTART AND
 *               EDC INITIALIZATION HAS NOT BEEN DONE THEN;
 *           PICK UP NG(NUMBER OF GENERATORS),NT(NUMBER OF TIE LINES),NP
 *                 (NUMBER OF PLANTS),AND NNCL (NUMBER OF NON CONFORMING
 *                 LOADS),AND THE TIME INTERVALS BETWEEN AUTOMATIC AND
 *                 COMPLETE EDC'S FROM THE DATA BASE;
 *           COMPUTE NROW=NP+NT+NNCL AND STORE IN AIOONROW;
 *           SUSPEND EDC PROCESSING BY SETTING AIEDCSSP TO ONE;
 *           INITIALIZE THE EDC COMPLETE OUTPUT TABLE AAOTEDCC;
 *           USE PTIME MACRO TO PATCH DOMAEDCB AT THE SPECIFIED INTERVALS
 *                 FOR AUTOMATIC EDC ENTRIES AND FOR COMPLETE ENTRIES;
 *         ELSE;
 *           ISSUE APPROPRIATE SYSTEM MESSAGE;
 *         ENDIF;
 *         EXIT DOMAEDCI;
 **
 **
 *        DOMAEDCB PERFORMS THE CYCLIC EDC COMPUTATIONS
 *
 *         BEGIN DOMAEDCB;
 *         IF EDC INITIALIZATION HAS BEEN DONE THEN;
 *           STORE ENTRY TYPE (COMPLETE OR AUTOMATIC);
 *           IF EDC IS ACTIVATED THEN;
 *             IF FIRST ENTRY THEN;
 *               OBTAIN ADDRESSES OF ARRAYS USED BY DOMAEDCB;
 *               OBTAIN VALUES OF NG,NPE,AND NROW;
 *               CALCULATE OUTPUT TABLE DISPLACEMENTS;
 *               INDICATE THAT FIRST ENTRY HAS BEEN MADE;
 *             ENDIF;
 *             STORE TIME AND DATE;
 *             USE GETITEM & LIST AAACTADR TO PUT ACT.DATA IN AAOOPACT;
 *             CHANGE SIGNS OF TIE LINE READINGS;
 *             STORE BMN MATRIX POINTED TO BY AIBMNPTR IN AAOOOBMN;
 *             STORE COST CURVES POINTED TO BY AIOCCPTR IN AAOCURVE;
 *             COPY GENERATOR CONTROL STATUS TABLE AAOTGCSM INTO AATTGCSM
 *             UPDATE COST CURVES TO REFLECT COSTS SHOWN IN AATTGCSM;
 *             COPY LOW ECONOMIC POWER LIMITS FROM AATTGCSM INTO AAOOPMIN
 *             COPY HI  ECONOMIC POWER LIMITS FROM AATTGCSM INTO AAOOPMAX
 *             COPY ACTUALS FROM AAOPACT INTO AAOOPDES;
 *             SET DEMAND AT SYSTEM LOAD CENTER(DALC) TO ZERO;
 *             IF ENTRY TYPE AUTOMATIC THEN;
 *               COPY DOMAEDB3 (AUTOMATIC TYPE ENTRY INITIALIZATION);
 *               COPY DOMAEDB4 (CHECK GENERATORS COMPLIANCE WITH LFC);
 *             ELSE;
 *               COPY DOMAEDB5 (COMPLETE TYPE ENTRY INITIALIZATION);
 *             ENDIF;
 *             COPY DOMAEDCM (MAIN EDC PROCESSOR);
 *           ENDIF;
 *         ELSE;
 *           ISSUE SYSTEM MESSAGE:''EDC NOT YET INITIALIZED'';
```

```
MEMBER NAME  DOMAPOLE
 *         ENDIF;
 *         EXIT DOMAEDCB:
 **
 **
 *     DOMAEDB3 PERFORMS AUTOMATIC TYPE ENTRY INITIALIZATION
 *
 *         BEGIN DOMAEDB3;
 *         INITIALIZE LAMBDA FROM AUTOMATIC OUTPUT TABLE;
 *         COMPUTE TOTAL SYSTEM GENERATION;
 *         COMPUTE DEMAND = SUM OF GENERATION OF ALL GENERATORS ON
 *               AUTOMATIC CONTROL;
 *         INDICATE IN AAOICEDC THAT ALL GENERATORS UNDER AUTOMATIC
 *               CONTROL AND NO OTHERS ARE TO BE DISPATCHED;
 *         EXIT DOMAEDB3:
 *
 **
 **
 *     DOMAEDB4 CHECKS GENERATORS COMPLIANCE WITH LFC
 *
 *         BEGIN DOMAEDB4;
 *         COPY INTEGRATED COMMANDS FROM AATLFCIC INTO AA00000T;
 *         RESET AATLFCIC TO ZEROS;
 *         DO GENNBR=1 TO NG;
 *           IF GEN.WAS ON AUTO CTRL ON LAST AUTO EDC AND STILL IS THEN;
 *             IF CHANGES COMMANDED BY LFC ARE GREATER THAN CRIT2 THEN;
 *               IF RATIO OF ACTUAL CHANGE TO COMMANDED CHANGE LESS THAN
 *                     CRIT1 THEN;
 *                 ISSUE ALARM STATING THAT THIS GEN.NOT OBEYING LFC;
 *               ENDIF;
 *             ENDIF;
 *           ENDIF;
 *         ENDDO;
 *         EXIT DOMAEDB4;
 *
 **
 **
 *     DOMAEDB5 PERFORMS COMPLETE TYPE ENTRY INITIALIZATION
 *
 *         BEGIN DOMAEDB5;
 *         INITIALIZE LAMBDA FROM COMPLETE OUTPUT TABLE;
 *         COMPUTE TOTAL SYSTEM GENERATION;
 *         COMPUTE DEMAND = SUM OF GENERATION OF ALL GENERATORS ON
 *               AUTOMATIC CONTROL OR ECONOMICALLY VARIABLE;
 *         INDICATE IN AAOICEDC THAT ALL GENERATORS ON AUTOMATIC CONTROL
 *               OR ECONOMICALLY VARIABLE AND NO OTHERS ARE TO BE
 *               DISPATCHED;
 *         EXIT DOMAEDB3:
 *
 **
 **
 *     DOMAEDCM PERFORMS THE MAIN EDC PROCESSING
 **
 **
 *         BEGIN DOMAEDCM;
 *         ERROR_CODE=0;
 *         DELTA_PWR_WITHIN_TOLERANCE = NO;
 *         DELTA_LAMBDA=DELTA_LAMBDA_TOLERANCE + 1.0;
 *         DO ITER = 1 TO 20 WHILE (ERROR_CODE = 0 & ((ABS(DELTA_LAMBDA)
```

```
MEMBER NAME  DOMAPDLE
 *               > DELTA_LAMBDA_TOLERANCE) | (DELTA_PWR_WITHIN_TOLERANCE
 *                     = NO)));
 *             COPY DOMAEDM1 (INVERSE OF PENALTY FACTORS COMPUTATION);
 *             COPY DOMAEDM2 (INCR.COST,SLOPE OF C.C.,COST DIFF CMPS);
 *             IF ERROR_CODE=0 THEN
 *               DO;
 *                 COPY DOMAEDM3 (N COMPUTATION);
 *                 COPY DOMAEM3A (GENERATOR EXCLUSION ADJUSTMENT);
 *                 COPY DOMAEDM4 (DELTA LAMBDA COMPUTATION);
 *                 COPY DOMAEDM5 (DESIRED POWER READINGS COMPUTATIONS);
 *                 LAMBDA=LAMBDA+DELTA_LAMBDA;
 *               END;
 *             ELSE;
 *           END;
 *           IF ERROR_CODE =0 & ABS(DELTA_LAMBDA) ¬> DELTA_LAMBDA_TOLERANCE
 *                 & DELTA_PWR_WITHIN_TOLERANCE = YES THEN
 *             COPY DOMAEDM6 (PARTICIPATION FACTORS COMPUTATION);
 *             COPY DOMAEDM8 (MISMATCH ADJUSTMENT COMPUTATION);
 *           ELSE
 *             IF ERROR_CODE =0 THEN
 *               ERROR_CODE=2;
 *             ENDIF;
 *           ENDIF;
 *           COPY DOMAEDCA(EDC POST PROCESSOR);
 *           EXIT DOMAEDCM;
 **
 **
 *           DOMAEDM1 COMPUTES THE INVERSE OF THE PENALTY FACTOR FOR EACH
 *                 GENERATOR AND FORMS AN ARRAY CONSISTING OF THE MAIN
 *                 DIAGONAL OF THE BMN MATRIX. THESE TWO VALUES ARE THE
 *                 SAME FOR ALL GENERATORS AT A GIVEN GENERATING PLANT.
 **
 **
 *           BEGIN DOMAEDM1;
 *           DO GENNBR=1 TO NG;
 *             IF PENALTY_FACTOR_INDEX(GENNBR) ¬= 0 THEN
 *               DO;
 *                 K = PENALTY_FACTOR_INDEX(GENNBR);
 *                 INV_PEN_FACT(GENNBR) = INV_PEN_FACT(K);
 *                 BMN_DIAG(GENNBR) = BMN_DIAG(K);
 *               END;
 *             ELSE
 *               DO;
 *                 ICOL = INDEX(GENNBR);
 *                 INV_PEN_FACT(GENNBR) = 0;
 *                 DO J=1 TO NPE;
 *                   IROW = INDEX(J);
 *                   INV_PEN_FACT(GENNBR) = INV_PEN_FACT(GENNBR) +
 *                        ACTIVE_BMN_MATRIX(IROW,ICOL) * DESIRED_POWER_
 *                             READING(J);
 *                 END;
 *                 INV PEN FACT(GENNBR)=1-.02*INV PEN FACT(GENNBR)
 *                   BMN_DIAG(GENNBR) = ACTIVE_BMN_MATRIX(IROW,IROW);
 *               END;
 *           END;
 *           EXIT DOMAEDM1;
 **
 **
```

```
MEMBER NAME  DOMAPDLE
 *        DOMAEDM2 COMPUTES THE INCREMENTAL COST OF POWER,RATE OF
 *              CHANGE IN THE INCREMENTAL COST OF POWER, COST DIFFERENCE
 *              AND A MAXIMUM AND MINIMUM ALLOWABLE POWER SETTING ON
 *              THIS ITERATION FOR EACH GENERATOR UNDER EDC. THESE COM-
 *              PUTED VALUES ARE THEN STORED IN THEIR RESPECTIVE ARRAYS.
 *              FOR GENERATORS NOT UNDER EDC, THESE VALUES ARE SET TO
 *              ZERO.
 *              FOR GENERATORS UNDER EDC WHOSE POWER READINGS ARE NOT
 *              WITHIN THE ECONOMIC LIMITS, THE DESIRED POWER READING
 *              IN ARRAY AA00PDES IS SET TO THE APPROPRIATE LIMIT BEFORE
 *              THE COMPUTATION IS MADE.
 *
 **
 **
 *        BEGIN DOMAEDM2:
 *        DO GENNBR=1 TO NG:
 *          IF GEN(GENNBR) IS TO BE DISPATCHED THEN
 *            IF DESIRED POWER(GENNBR) LESS THAN MIN ECON LIMIT THEN
 *              DESIRED POWER(GENNBR) = MIN ECON LIMIT(GENNBR)
 *            IF DESIRED POWER(GENNBR) MORE THAN MAX ECON LIMIT THEN
 *              DESIRED POWER(GENNBR) = MAX ECON LIMIT(GENNBR)
 *            COMPUTE HAME = NO
 *            DO J=2 BY 1 WHILE (COMPUTE HAME = NO AND ERROR CODE = 0)
 *              IF DESIRED POWER(GENNBR) LE COST CURVE(J,GENNBR),THEN
 *               IF J NOT = 2 THEN
 *                  COMPUTE HAME = YES
 *                ELSE
 *                  IF DES.POWER(GENNBR) NE COST CURVE(J,GENNBR),THEN
 *                    ERROR_CODE = 1;
 *                  ELSE
 *                    COMPUTE HAME = YES
 *                    GENNBR = GENNBR + 1
 *              ELSE
 *                IF J = 7 THEN
 *                  ERROR CODE= 1
 *            END
 *                  IF COMPUTE_HAME = YES THEN
 *                    DO:
 *                      INCR_COST(GENNBR) = ACTIVE_COST_CURVE_MATRIX
 *                           (J + 11, GENNBR) * DESIRED_POWER_READING
 *                                (GENNBR) +ACTIVE_COST_CURVE_MATRIX
 *                                     (J + 16, GENNBR);
 *                      RATE_OF_CHNG_INCR_COST (GENNBR) =  ACTIVE_COST_
 *                           CURVE_MATRIX(J + 11, GENNBR);
 *                      COST_DIFF(GENNBR) = LAMBDA * INV_PEN_FACT
 *                           (GENNBR) - INCR_COST(GENNBR);
 *                      GEN_EXCL_COMP_ARRAY(GENNBR) = 1;
 *                      JT = MIN(7,J+1);
 *                      MAX_ALLOWED_PWR(GENNBR) = ACTIVE_COST_CURVE_
 *                            MATRIX(JT,GENNBR);
 *                      JT = MAX(2,J-2);
 *                      MIN_ALLOWED_PWR(GENNBR) = ACTIVE_COST_CURVE_
 *                            MATRIX(JT,GENNBR);
 *                    END;
 *                IF DESIRED POWER(GENNBR) = MIN ECON LIMIT(GENNBR) AND
 *                IF COST DIFF(GENNBR) NEGATIVE THEN
 *                  GEN EXCL COMP ARRAY(GENNBR) = 0
 *                ELSE
```

```
MEMBER NAME  DOMAPDLE
 *                  IF DES.POWER(BENNBR) = MAX ECON LIMIT(GENNBR) AND
 *                  IF COST DIFF(GENNBR) POSITIVE THEN
 *                    GEN EXCL COMP ARRAY(GENNBR) = 0
 *           ELSE
 *             DO;
 *               INCR_COST(GENNBR) = 0;
 *               RATE_OF_CHNG_INCR_ COST(GENNBR) = 0;
 *               COST_DIFF(GENNBR) = 0;
 *               GEN_EXCL_COMP_ARRAY(GENNBR) = 0;
 *             END;
 *         END;
 *         EXIT DOMAEDM2;
 **
 **
 * DOMAEDM3 COMPUTES N,THE DEMAND DIFFENCE. THIS IS THE DIFFERENCE
 * BETWEEN THE TOTAL DELIVERED POWER BASED ON THE ACTUAL GENERATOR
 * READINGS AND THE TOTAL DELIVERED POWER BASED ON THE CURRENT ESTIMATE
 * OF THE DESIRED GENERATOR OUTPUTS.
 **
 **
 *         BEGIN DOMAEDM3;
 *         CALCULATE SUM= SUMMATION OF DES.PWR.
 *                       FOR ALL GENERATORS TO BE DISPATCHED;
 *         N = (DEMAND - SUM) * 2;
 *         EXIT DOMAEDM3;
 **
 **
 * DOMAEM3A OVERRIDES EXCLUSION OF FINAL GENERATOR IF APPROPRIATE
 **
 **
 *         BEGIN DOMAEM3A;
 *         IF ALL GENERATORS EXCLUDED THEN
 *           IF N,THE DEMAND DIFFERENCE, SHOWS SUM PDES TOO LARGE THEN
 *             INCLUDE THE GENERATOR WITH MAX.COST OF DELIVERED POWER
 *                   FROM THE SET OF GENERATORS BEING DISPATCHED WHOSE
 *                   COST DIFFERENCES ARE POSITIVE;
 *           ELSE
 *             INCLUDE THE GENERATOR WITH MIN.COST OF DELIVERED POWER
 *                   FROM THE SET OF GENERATORS BEING DISPATCHED WHOSE
 *                   COST DIFFERENCES ARE NEGATIVE;
 *           ENDIF;
 *         ENDIF;
 *         EXIT DOMAEM3A;
 **
 **
 *         DOMAEDM4 COMPUTES THE DESIRED CHANGE IN THE SYSTEM LAMBDA.
 **
 **
 *         BEGIN DOMAEDM4;
 *         S1 = 0;
 *         S2 = 0;
 *         DO GENNBR = 1 TO NG;
 *           IF GEN_EXCL_COMP_ARRAY(GENNBR) = 0 THEN
 *             TEMP_ARRAY(GENNBR) = 0;
 *           ELSE
 *             DO;
 *               TEMP_ARRAY(GENNBR) =RATE_OF_CHNG_INCR_COST(GENNBR) +
 *                    .02 * BMN_DIAG(GENNBR) * LAMBDA;
```

```
MEMBER NAME  DOMAPDLE
 *               S1 = S1 + COST_DIFF(GENNBR) / TEMP_ARRAY(GENNBR)
 *               S2 = S2 + INV_PEN_FACT(GENNBR) / TEMP_ARRAY(GENNBR)
 *             END;
 *         END;
 *         DELTA_LAMBDA = (N - 2*S1) / (2 * S2);
 *         EXIT DOMAEDM4;
 **
 **
 *         DOMAEDM5 COMPUTES THE CHANGE IN THE DESIRED POWER SETTING
 *               AND ADDS IT TO THE OLD SETTING.
 **
 **
 *         BEGIN PDES_CMP;
 *         BEGIN DOMAEDM5;
 *         DO GENNBR = 1 TO NG;
 *           IF GEN_EXCL_COMP_ARRAY(GENNBR) = 1 THEN
 *             DO;
 *               DELTA_DES_PWR(GENNBR) = (COST_DIFF(GENNBR) + DELTA_
 *                     LAMBDA * INV_PEN_FACT(GENNBR)) / TEMP_ARRAY
 *                           (GENNBR);
 *               DESIRED_POWER_READING(GENNBR) = DESIRED_POWER_READING
 *                    (GENNBR) + DELTA_DES_PWR(GENNBR);
 *               IF DESIRED_POWER_READING(GENNBR) < MIN_ALLOWED_PWR
 *                     (GENNBR) THEN
 *                 DO;
 *                   DELTA_DES_PWR(GENNBR) = MIN_ALLOWED_PWR(GENNBR) -
 *                         (DESIRED_POWER_READING(GENNBR) - DELTA_DES_
 *                               PWR(GENNBR));
 *                   DESIRED_POWER_READING(GENNBR) = MIN_ALLOWED_PWR
 *                         (GENNBR);
|*               END;
 *               IF DESIRED_POWER_READING(GENNBR) > MAX_ALLOWED_PWR
 *                    (GENNBR) THEN
 *                 DO;
 *                   DELTA_DES_PWR(GENNBR) = MAX_ALLOWED_PWR(GENNBR) -
 *                         (DESIRED_POWER_READING(GENNBR) - DELTA_DES_
 *                               PWR(GENNBR));
 *                   DESIRED_POWER_READING(GENNBR) = MAX_ALLOWED_PWR
 *                         (GENNBR);
|*               END;
 *               IF ABS(DELTA_DES_PWR(GENNBR)) > .01 THEN
 *                 DELTA_PWR_WITHIN_TOLERANCE = NO;
|*               END;
 *         END;
 *         EXIT DOMAEDM5;
 **
 **
 *         DOMAEDM6 COMPUTES THE PARTICIPATION FACTORS FOR ALL
 *               GENERATORS UNDER EDC CONTROL. THESE ARE THEN PLACED
 *               IN ARRAY RHO.
 *               FOR GENERATORS NOT UNDER EDC CONTROL, THE PARTICI-
 *               PATION FACTORS ARE SET TO ZERO.
 **
 **
 *         BEGIN DOMAEDM6;
 *         S1=0;
 *         DO GENNBR= 1 TO NG;
 *           IF GEN(GENNBR) IS TO BE DISPATCHED THEN
```

```
MEMBER NAME  DOMAPDLE
*              S1 = S1 + INV_PEN_FACT(GENNBR) / RATE_OF_CHNG_INCR_COST
*                   (GENNBR);
*          END;
*          DO GENNBR = 1 TO NG;
*            IF GEN(GENNBR) IS NOT TO BE DISPATCHED THEN
*              PART_FACT(GENNBR) = 0;
*            ELSE
*              PART_FACT(GENNBR) = (INV_PEN_FACT(GENNBR) / RATE_OF_CHNG_
*                   INCR_COST(GENNBR)) / S1;
*          END;
*          EXIT DOMAEDM6;
**
**
**
*   DOMAEDM8 APPORTIONS THE MISMATCH BETWEEN THE SUM OF THE
*   DESIRED POWER READINGS AND THE SUM OF THE ACTUALS IN ACCORDANCE WITH
*   THE PARTICIPATION FACTORS OF THOSE GENERATORS WHOSE DESIRED POWER
*   SETTINGS HAVE NOT BEEN SET TO EITHER OF THEIR ECONOMIC LIMITS
**
**
*          BEGIN DOMAEDM8;
*          MISMATCH=1;
*          DO I=1 TO 2;
*            DO J=1 TO 5 WHILE MISMATCH GT .01;
*              ZERO OUT ALL TEMPORARY PARTICIPATION FACTORS;
*              SET TEMP.PART.FACTORS OF NON EXCLUDED GENERATORS =ACTUAL
*                    PARTICIPATION FACTORS COMPUTED IN DOMAEDM6;
*              MISMATCH = DEMAND-SUM OF DESIRED POWER SETTINGS FOR ALL
*                    GENERATORS UNDER EDC;
*              NORMALIZE TEMP.PART.FACTORS;
*              DO GENNBR=1 TO NG FOR ALL GENERATORS UNDER EDC;
*                DES.POWER(GENNBR)=DES.POWER(GENNBR)+TEMP.PART.FACTOR *
*                      MISMATCH;
*                IF DES.POWER(GENNBR) OUTSIDE ECONOMIC LIMITS THEN
*                  SET IT TO LIMIT;
*                  EXCLUDE THIS GEN.FROM FURTHER COMPUTATIONS;
*                ENDIF;
*              ENDDO;
*            ENDDO;
*            IF MISMATCH GT .01 THEN
*              INCLUDE ALL GENERATORS BEING DISPATCHED IN MISMATCH CALCS;
*            ENDIF;
*          ENDDO;
*          EXIT DOMAEDM8;
**
*          DOMAEDCA IS THE EDC POST PROCESSOR.
**
**
*          BEGIN DOMAEDCA;
*          IF AUTOMATIC TYPE ENTRY THEN
*            IF ERROR_CODE = 0 THEN
*              DO;
*                CALL DOMASTRO(AAOTEDCA);
*                SET AIEDCINP=1 TO INDICATE TO LFC THAT A SUCCESSFUL
*                      AUTOMATIC EDC HAS BEEN PERFORMED;
*              END;
*            ELSE
*              IF ERROR_CODE =1 THEN
```

```
MEMBER NAME  DOMAPDLE
 *                    AUTOMATIC EDC HAS BEEN PERFORMED;
 *              END;
 *            ELSE
 *              IF ERROR_CODE =1 THEN
 *                WRITE 'EDC COST CURVES DO NOT COVER ECONOMIC LIMITS.
 *                      AUTOMATIC EDC NOT PERFORMED.';
 *              ELSE
 *                WRITE 'AUTOMATIC EDC DID NOT CONVERGE.';
 *          ELSE
 *            IF ERROR_CODE = 0  THEN
 *              DO;
 *                CALL DOMASTRO(AAOTEDCC);
 *              END;
 *            ELSE
 *              IF  ERROR_CODE=1 THEN
 *                WRITE 'EDC COST CURVES DO NOT COVER ECONOMIC LIMITS.
 *                      COMPLETE EDC NOT PERFORMED.';
 *              ELSE
 *                WRITE 'COMPLETE EDC DID NOT CONVERGE.';
 *          EXIT DOMAEDCA;
 **
 **
 *          DOMASTRO:  PROCEDURE(OUTPUT_TABLE);
 *          STORE DATE AND TIME OF DISPATCH IN OUTPUT_TABLE;
 *          STORE TOTAL SYSTEM GENERATION IN OUTPUT_TABLE;
 *          STORE TOTAL POWER DISPATCHED IN OUTPUT_TABLE;
 *          STORE SYSTEM LAMBDA IN OUTPUT_TABLE;
 *          STORE AAOICEDC SETTINGS(UNITS DISPATCHED) IN OUTPUT_TABLE;
 *          STORE ACTUAL POWER READINGS IN OUTPUT_TABLE;
 *          STORE DESIRED POWER READINGS IN OUTPUT_TABLE;
 *          STORE  PARTICIPATION FACTORS IN OUTPUT_TABLE;
 *          EXIT DOMASTRO;
 **
 **
 *          OPERATOR INTERFACE PROCESSING IS DONE BY INDEPENDENT AND
 *            DEPENDENT TASKS THAT ARE PATCHED BY DISPLAY MANAGEMENT
 *            WHEN A FUNTION KEY IS DEPRESSED.
 **
 **
 **
 *   DOMAATPG CONTAINS THE LOGIC NECESSARY TO DISPLAY THE AUTOMATIC EDC
 *          OUTPUT TABLE DATA AND PAGE THRU THAT DATA FOR ANY NUMBER
 *          OF REQUESTING UNITS
 **
 *          SAVE THE ADDRESS OF THE CVT,THE ADDRESS OF THE DISPLAY
 *            PARAMETER LIST,AND THE PATCH ID
 *          IF THIS IS 1ST EXECUTION THEN
 *            INDICATE 1ST EXEC. HAS OCCURED
 *            GET THE ADDRESS OF THE PLANT NAME ARRAY
 *            GET THE NUMBER OF PLANTS(NP)
 *            GET TEXT PARAMETERS
 *            GET THE ADDRESSES OF THE FOLLOWING PARAMETERS
 *              FIRST GENERATORS INCLUSION INDICATOR
 *              FIRST GENERATORS ACTUAL POWER
 *              FIRST GENERATORS DESIRED POWER
 *              FIRST GENERATORS DESIRED PARTICIPATION FACTOR
 *          ENDIF
 *          CONVERT TIME OF DISPATCH FOR OUTPUT
```

```
MEMBER NAME  DOMAPDLE
 *        DETERMINE NUMBER OF PAGES REQUIRED TO OUTPUT TABLE-4 PLTS/PG
 *        IF PATCH ID IS 1 THEN PROCESS 1ST PAGE REQUEST
 *          STORE PAGE NUMBER ONE IN CEATAB ENTRY FOR EDC (CEAEDCM)
 *        ENDIF
 *        IF PATCH ID IS 3 OR 4 OR 14 THEN PROCESS PAGING REQUEST
 *          OR REFRESH
 *          IF PATCH ID IS 14 THEN STORE CURRENT PAGE NUMBER
 *          ELSE
 *            IF PATCH ID IS 3 THEN PAGE FORWARD
 *              INCREMENT CURRENT PAGE NO.(WRAP-AROUND IF ON LAST PAGE)
 *            ELSE PAGE BACKWARD(PATCH ID 4)
 *              DECREMENT CURRENT PAGE NO.(WRAP-AROUND IF ON FIRST PAGE)
 *            ENDIF
 *          ENDIF
 *        ENDIF
 *        ISSUE DINFO'S FOR PAGE NO.,TOTAL PAGES, HOURS, MINUTES,
 *          AND SECONDS FOR TIME OF DISPATCH
 *        DETERMINE ADDRESSES OF DATA FOR PAGE REQUESTED
 *        FOR EACH SET OF 4 PLANTS DO (1 LINE OF OUTPUT)
 *          ISSUE DINFO FOR ONE LINE OF PLANT NAMES
 *          FOR EACH PLANT DO
 *            ISSUE DINFO-INCLUSION INDICATOR FOR EACH GENERATOR
 *            ISSUE DINFO-GENERATOR NUMBER FOR EACH GENERATOR
 *            ISSUE DINFO-ACTUAL POWER OF EACH GENERATOR
 *            ISSUE DINFO-DESIRED POWER OF EACH GENERATOR
 *            ISSUE DINFO-DESIRED PARTICIPATION FACTOR OF EACH GEN.
 *          ENDDO
 *        ENDDO
 *        ISSUE DISPUP'S TO OUTPUT DYNAMIC DATA TO REQUESTING UNIT
 *  EXIT DOMAATPG
 **
 *  DOMACTPG CONTAINS THE LOGIC NECESSARY TO DISPLAY THE COMPLETE EDC
 *        OUTPUT TABLE DATA AND PAGE THRU THAT DATA FOR ANY NUMBER
 *        OF REQUESTING UNITS
 **
 *        SAVE THE ADDRESS OF THE CVT,THE ADDRESS OF THE DISPLAY
 *          PARAMETER LIST, AND THE PATCH ID
 *        IF THIS IS 1ST EXECUTION THEN
 *          INDICATE 1ST EXEC. HAS OCCURED
 *          GET THE ADDRESS OF THE PLANT NAME ARRAY
 *          GET THE NUMBER OF PLANTS(NP)
 *          GET TEXT PARAMETERS
 *          GET THE ADDRESSES OF THE FOLLOWING PARAMETERS
 *            FIRST GENERATORS INCLUSION INDICATOR
 *            FIRST GENERATORS ACTUAL POWER
 *            FIRST GENERATORS DESIRED POWER
 *            FIRST GENERATORS DESIRED PARTICIPATION FACTOR
 *        ENDIF
 *        CONVERT TIME OF DISPATCH FOR OUTPUT
 *        DETERMINE NUMBER OF PAGES REQUIRED TO OUTPUT TABLE-4 PLTS/PG
 *        IF PATCH ID IS 1 THEN PROCESS 1ST PAGE REQUEST
 *          STORE PAGE NUMBER ONE IN CEATAB ENTRY FOR EDC(CEAEDCM)
 *        ENDIF
 *        IF PATCH ID IS 3 OR 4 OR 14 THEN PROCESS PAGING REQUEST
 *          OR REFRESH
 *          IF PATCH ID IS 14 THEN STORE CURRENT PAGE NUMBER
 *          ELSE
 *            IF PATCH ID IS 3 THEN PAGE FORWARD
```

```
MEMBER NAME  DOMAPDLE
 *              INCREMENT CURRENT PAGE NO.(WRAP-AROUND IF ON LAST PAGE)
 *            ELSE
 *              DECREMENT CURRENT PAGE NO.(WRAP-AROUND IF ON FIRST PAGE)
 *            ENDIF
 *          ENDIF
 *        ENDIF
 *        ISSUE DINFO'S FOR PAGE NO.,TOTAL PAGES,HOURS,MINUTES,
 *           AND SECONDS FOR TIME OF DISPATCH
 *        DETERMINE ADDRESSES OF DATA FOR REQUESTED
 *        FOR EACH SET OF 4 PLANTS DO (1 LINE OF OUTPUT)
 *          ISSUE DINFO FOR ONE LINE OF PLANT NAMES
 *          FOR EACH PLANT DO
 *            ISSUE DINFO-INCLUSION INDICATOR-EACH GENERATOR
 *            ISSUE DINFO-GENERATOR NUMBER-EACH GENERATOR
 *            ISSUE DINFO-ACTUAL POWER-EACH GENERATOR
 *            ISSUE DINFO-DESIRED POWER-EACH GENERATOR
 *            ISSUE DINFO-DESIRED PARTICIPATION FACTOR-EACH GENERATOR
 *          ENDDO
 *        ENDDO
 *        ISSUE DISPUP'S TO OUTPUT DYNAMIC DATA TO REQUESTING UNIT
 *  EXIT DOMACTPG
 **
 *  DOMAICUP CONTAINS THE LOGIC NECESSARY TO DISPLAY THE COMPLETE AND
 *        AUTOMATIC EDC INTERVALS,THE COST CURVE POINTER,AND THE BMN
 *        MATRIX POINTER AND THE LOGIC NECESSARY TO PROCESS DATA ENTRY
 *        CHANGES REQUESTED FOR THIS DISPLAY
 **
 *        SAVE THE ADDRESS OF THE CVT,THE ADDRESS OF THE DISPLAY PARMS,
 *             THE PATCH ID, AND THE ADDRESS OF
 *             THE DATA ENTRY CHANGE LIST OR THE AIDIN PARMS.
 *        IF THIS IS FIRST EXECUTION THEN
 *          INDICATE FIRST EXECUTION HAS OCCURRED
 *          GET ADDRESS OF EDC
 *          INTERVALS, ACCESS/FUNCTION AREA ID, AND
 *          TEXT PARAMETERS
 *        ENDIF
 *        IF PATCH ID IS 7 THEN PROCESS DATA ENTRY PATCH
 *          DO LOOP THRU DATA ENTRY CHANGE LIST
 *            IF DATA ENTRY VALUES PASSED LIMIT CHECKING THEN
 *              IF ENTERED VALUE IS AUTO EDC INTERVAL CHANGE
 *                CONVERT TO INTERNAL FORMAT AND STORE IN DATA BASE
 *                ISSUE PTIME MOD,FOR AUTO EDC,TO NEW INTERVAL
 *                ISSUE IMMEDIATE PATCH TO EDC FOR AUTO TYPE EDC EXEC.
 *                EVENT CHANGE TO AUTO EDC INTERVAL
 *              ELSE
 *                IF ENTERED VALUE IS COMPLETE EDC INTERVAL CHANGE
 *                  CONVERT TO INTERNAL FORMAT AND STORE IN DATA BASE
 *                  ISSUE PTIME MOD FOR,COMPLETE EDC,TO NEW INTERVAL
 *                  EVENT CHANGE TO COMPLETE EDC INTERVAL
 *                ELSE
 *                  IF ENTERED VALUE IS COST CURVE POINTER CHANGE
 *                    CHECK TO SEE IF ALL HIGH AND LOW ECONOMIC POWER
 *                     LIMITS IN THE GENERATOR CONTROL STATUS
 *                     MATRIX LIE ON THE REQUESTED COST CURVE.
 *                     IF NOT ISSUE AN ERROR MESSAGE AND DO NOT
 *                     CHANGE THE COST CURVE POINTER. IF ALL POWER
 *                     LIMITS LIE ON THE REQUESTED CURVE,COMPUTE
 *                     THE CORRESPONDING INCREMENTAL COST LIMITS,
```

MEMBER NAME DOMAPDLE

```
 *                    STORE THEM IN THE GENERATOR CONTROL STATUS
 *                    MATRIX,AND CHANGE THE COST CURVE POINTER TO
 *                    REFLECT THE REQUESTED COST CURVE
 *                    EVENT COST CURVE CHANGE
 *                 ELSE
 *                   IF ENTERED VALUE IS BMN MATRIX POINTER CHANGE
 *                     CHANGE BMN MATRIX POINTER TO REQUESTED VALUE
 *                     EVENT BMN MATRIX POINTER CHANGE
 *                   ELSE
 *                     DO NOTHING
 *                   ENDIF
 *                 ENDIF
 *               ENDIF
 *             ENDIF
 *           ELSE
 *             EVENT BAD DATA ENTRY ATTEMPT
 *           ENDIF
 *         ENDDO
 *         LOG AAEDCODD ARRAY CONTAINING CHANGED VALUES.
 *       ENDIF
 *       CONVERT EDC INTERVALS TO OUTPUT FORMAT
 *       ISSUE DINFO'S FOR AUTO AND COMPLETE INTERVALS
 *       ISSUE DISPUP'S TO OUTPUT DYNAMIC DATA TO REQUESTING UNIT
 *   EXIT DOMAICUP
 **
 **
 *   END OF EDC DISPLAY SUPPORT PROGRAMS   *********
 **
```

```
MEMBER NAME  DOMAPOLL
 * DOMAPOLL CONTAINS THE PROGRAM DESIGN LANGUAGE ASSOCIATED WITH THE AGC
 * (LFC) APPLICATION OF THE SYSTEM 370 ENERGY MANAGEMENT SYSTEM.
 *
 **
 **
 **
 *          BEGIN DOMALFCI;
 *          IF LFC INITIALIZATION HAS NOT OCCURRED SINCE LAST RESTART THEN
 *            INITIALIZE INTEGRATED VALUES OF LFC COMMANDS SINCE LAST
 *                  AUTOMATIC EDC TO ZERO FOR EACH GENERATOR IN AATLFCIC;
 *            INITIALIZE LFC OUTPUT TABLE,AALFCOUT,TO ZEROES;
 *            PICK UP NG(NUMBER OF GENERATORS),NT(NUMBER OF TIE LINES),AND
 *                  NNCL(NUMBER OF NON CONFORMING LOADS) FROM DATA BASE;
 *            COMPUTE NUMBER OF ELEMENTS IN POWER VECTOR,NPE,(NPE=NG+NT+
 *                  NNCL) AND STORE IN AIOOONPE;
 *            COMPUTE THE ADDRESSES OF THE ANALOG DATA VALUES OF EACH GEN-
 *                  ERATOR,TIE LINE,AND NON CONFORMING LOAD AND STORE THEM
 *                  IN ARRAY AAACTADR;
 *            INITIALIZE THE EDC AUTOMATIC OUTPUT TABLE AAOTEDCA;
 *            SET AIEDCINP TO ZERO TO SHOW EDC NOT YET PERFORMED;
 *            SET AILFCSSP NONZERO TO SUSPEND LFC;
 *            PICK UP DESIRED LFC INTERVAL FROM AILFCINT;
 *            USE PTIME MACRO TO PATCH CYCLIC LFC PROCESSOR(DOMALFCB) AT
 *                  THE DESIRED INTERVALS;
 *            SHOW THAT LFC INITIALIZATION HAS BEEN DONE;
 *          ELSE
 *            ISSUE SYSTEM MESSAGE ''MULTIPLE LFC INITIALIZATIONS TRIED'';
 *          ENDIF
 *          EXIT DOMALFCI:
 **
 **
 *          BEGIN DOMALFCB;
 *          IF LFC INITIALIZATION HAS OCCURRED SINCE LAST RESTART THEN
 *            IF FIRST ENTRY THEN
 *              COPY DOMALFB1; (SPECIAL PROCESSING FOR FIRST ENTRY)
 *            ENDIF;
 *            OBTAIN CURRENT VALUES OF LFC PARAMETERS TO BE USED ON THIS
 *                  PASS THRU LFC;
 *            FLOAT THE LFC TIME INTERVAL AND CONVERT IT TO SECONDS;
 *            UPDATE CONTINUOUS MONITOR FAILOVER HALFWORD;
 *            STORE CURRENT TIME AND DATE;
 *            COPY DOMALFB2:   PERFORM SCHEDULED TIE FLOW COMPUTATIONS
 *            IF LFC NOT SUSPENDED THEN
 *              IF AILFCGO SHOWS SCAN COMPLETE THEN
 *                UPDATE AIPLFCGO TO SHOW PREVIOUS SCAN COMPLETE;
 *                UPDATE AILFCGO  TO SHOW NEXT SCAN NOT YET COMPLETE;
 *                IF EDC INPUT IS AVAILABLE THEN
 *                  IF PRIMARY FREQUENCY READING AVAILABLE THEN
 *                    DO LFCB:   DO THE LFC (DOMALFB4)
 *                  ELSE
 *                    IF BACKUP FREQUENCY OK OR ON FLAT TIE LINE CTRL.THEN
 *                      DO LFCB;
 *                    ELSE
 *                      ISSUE ALARM:'UNABLE TO DO LFC:NO SYS.FREQ.DATA';
 *                    ENDIF;
 *                  ENDIF;
 *                ELSE
 *                  ISSUE ALARM:'UNABLE TO DO LFC:NO EDC OUTPUT';
```

```
MEMBER NAME  DOMAPDLL
 *              ENDIF;
 *            ELSE
 *              IF AIPLFCGO SHOWS PREVIOUS SCAN COMPLETE THEN
 *                UPDATE AIPLFCGO TO SHOW PREVIOUS SCAN NOT COMPLETE;
 *              ELSE
 *                ISSUE ALARM:'UNABLE TO DO LFC:INCOMPLETE DATA SCAN';
 *              ENDIF;
 *            ENDIF;
 *          ELSE
 *            SHOW RUNNING MODE OF SUSPENDED IN LFC OUTPUT TABLE;
 *          ENDIF;
 *        ELSE
 *          ISSUE SYSTEM MESSAGE:'LFC TRIED BEFORE BEING INITIALIZED';
 *        ENDIF;
 *        EXIT DOMALFCB;
 **
 **
 *        DOMALFB1 CONTAINS SPECIAL PROCESSING TO BE DONE ON THE FIRST
 *        ENTRY TO THE LFC CYCLIC PROCESSOR DOMALFCB
 **
 **
 *        BEGIN DOMALFB1;
 *        INDICATE FIRST ENTRY MADE;
 *        OBTAIN ADDRESSES OF ARRAYS USED BY DOMALFCB;
 *        OBTAIN NG(NUMBER OF GENERATORS),NT(NUMBER OF TIE LINES),NCIS
 *              (NUMBER OF INTERCHANGE SCHEDULES),AND THE ITEM NAMES OF
 *              THE PRIMARY AND BACK UP FREQUENCY SENSORS;
 *        OBTAIN THE ADDRESSES OF THE PRIMARY AND BACKUP FREQ.SENSORS;
 *        OBTAIN ADDRESSES OF ITEMS USED BY DOMALFCB;
 *        CALCULATE EDC OUTPUT TABLE DISPLACEMENTS;
 *        EXIT DOMALFB1;
 **
 **
 *        DOMALFB2 PERFORMS THE NET SCHEDULED TIE FLOW COMPUTATIONS
 *
 *        BEGIN DOMALFB2;
 *        SET NET SCHEDULED TIE FLOW (NSTF) TO ZERO;
 *        IF NCIS GREATER THAN ZERO THEN
 *          COPY AADOICSM INTO AADTICSM;
 *          FOR EACH INTERCHANGE SCHEDULE DO;
 *            IF CURRENT SCHEDULED VALUE NOT = DESIRED SCH.VALUE 1 THEN
 *              IF CURRENT DATE BETWEEN START DATE 1 AND STOP DATE1 THEN
 *                CONVERT START DATE 1 TO DAYS SINCE 1900;
 *                CONVERT STOP DATE 1 TO DAYS SINCE 1900;
 *                CONVERT CURRENT DATE TO DAYS SINCE 1900;
 *                COMPUTE CURRENT TIME IN .01 SECONDS SINCE MIDNIGHT
 *                        PRECEEDING START DATE;
 *                COMPUTE STOP TIME 1 IN .01 SECS. SINCE MIDNIGHT
 *                        PRECEEDING START DATE;
 *                IF CURRENT TIME IS AFTER START TIME1 AND BEFORE STOP
 *                        TIME1 THEN
 *                  UPDATE CURRENT SCHEDULED VALUE;(CSV=CSV+(LFC INTER-
 *                          VAL / (STOP TIME - CURRENT TIME))*(DSV-CSV))
 *                ENDIF;
 *              ENDIF;
 *            ENDIF;
 *            IF CURRENT SCHEDULED VALUE NOT = DESIRED SCH.VALUE 2 THEN
 *              IF CURRENT DATE BETWEEN START DATE 2 AND STOP DATE2 THEN
```

```
MEMBER NAME  DOMAPOLL
 *                  CONVERT START DATE 2 TO DAYS SINCE 1900;
 *                  CONVERT STOP DATE 2 TO DAYS SINCE 1900;
 *                  CONVERT CURRENT DATE TO DAYS SINCE 1900;
 *                  COMPUTE CURRENT TIME IN .01 SECONDS SINCE MIDNIGHT
 *                        PRECEEDING START DATE;
 *                  COMPUTE STOP TIME 2 IN .01 SECONDS SINCE MIDNIGHT
 *                        PRECEEDING START DATE;
 *                  IF CURRENT TIME IS AFTER START TIME 2 AND BEFORE STOP
 *                        TIME 2 THEN
 *                    UPDATE CURRENT SCHEDULED VALUE;
 *                  ENDIF;
 *                ENDIF;
 *              ENDIF;
 *              NSTF=NSTF+ CURRENT SCHEDULED VALUE;
 *            ENDDO;
 *            COPY AAOTICSM INTO AAODICSM;
 *          ENDIF;
 *          EXIT DOMALFB2;
 **
 **
 *          DOMALFB4 IS THE TOP LEVEL CONTROL OF THE LFC ALGORITHM
 *          BEGIN DOMALFB4;
 *          BGNSEG LFCB;
 *          STORE ACTUAL SYSTEM FREQUENCY;
 *          COPY GENERATOR CONTROL STATUS TABLE AAOTGCSM INTO AATTTGCS;
 *          COPY HI.EC.POWER LIMITS FROM AATTTGCS TO AAPMAXEC;
 *          COPY LO.EC.POWER LIMITS FROM AATTTGCS TO AAPMINEC;
 *          COPY HI.EM.POWER LIMITS FROM AATTTGCS TO AAPMAXEM;
 *          COPY LO.EM.POWER LIMITS FROM AATTTGCS TO AAPMINEM;
 *          COPY EDC PARTICIPATION FACTORS FROM AAOTEDCA INTO AAOORHOO;
 *          COPY EDC DESIRED POWER SETTINGS FROM AAOTEDCA INTO AAOPDEDC;
 *          FOR EACH GENERATOR DO;
 *            IF DATA FOR THE GENERATOR IS GOOD THEN
 *              STORE IT IN AALFCACT;
 *            ELSE
 *              IF THE GENERATOR IS BASE LOADED OR ON AUTOMATIC CNTRL THEN
 *                ISSUE ALARM:'GEN(NAME) REGARDED AS OUT OF SERVICE FOR
 *                      THIS PASS DUE TO MISSING DATA';
 *                CHANGE STATUS CODE IN GCSM TO OUT OF SERVICE;
 *              ENDIF;
 *            ENDIF;
 *          ENDDO;
 *          RAMPD=0;              SET RAMP DELTA = 0
 *          FOR EACH GENERATOR DO;
 *            IF STATUS CODE INDICATES THE GENERATOR IS OUT OF SERVICE,OFF
 *                  CONTROL,OR ECONOMICALLY VARIABLE THEN
 *              SET ITS UNIT CORRECTION ZERO IN AALFCUNC;
 *              SHOW IT TO BE OFF CONTROL IN AAOLFCIC;
 *            ELSE
 *              IF STATUS CODE INDICATES IT IS ON AUTOMATIC CONTROL THEN
 *                SET ITS UNIT CORRECTION = 0 IN AALFCUNC;
 *                SHOW IT TO BE ON CONTROL IN AAOLFCIC;
 *              ELSE
 *                DELTA=DESIRED BASE LOAD POINT - ACTUAL POWER READING;
 *                ALLOWABLE CORRECTION=LFC INTERVAL * DESIRED RAMP RATE;
 *                IF |DELTA| > ALLOWABLE CORRECTION THEN
 *                  ADJUST |DELTA| TO = |ALLOWABLE CORRECTION|;
 *                ENDIF;
```

```
MEMBER NAME  DOMAPDLL
 *              SET UNIT CORRECTION FOR THIS GEN.IN AALFCUNC TO DELTA;
 *              SHOW IT TO BE OFF CONTROL IN AAOLFCIC;
 *              RAMPD=RAMPD + DELTA;
 *            ENDIF;
 *          ENDIF;
 *        ENDDO;
 *        SET ERRCODE =0;
 *        SET NATF =0;        NATF IS NET ACTUAL TIE FLOW
 *        IF LFC MODE ADJUSTMENT IS NOT ZERO AND # OF TIE LINES IS NOT
 *              ZERO THEN
 *          DO THE FOLLOWING FOR EACH TIE LINE WHILE ERRCODE = 0;
 *              CHECK TO SEE IF ITS DATA IS USABLE;
 *              IF THE DATA IS NOT USABLE AND IS NOT OUT OF SERVICE-SELF
 *                IF FREQUENCY BIAS IS NOT ZERO THEN
 *                  SET LFC MODE ADJUSTMENT TO ZERO;
 *                  ISSUE ALARM:'NO USABLE DATA FOR TIE LINE(NAME).LFC
 *                        ON FLAT FREQUENCY CONTROL';
 *                ELSE
 *                  SET ERRCODE=1;
 *                ENDIF;
 *              ELSE
 *                NATF = NATF + READING FROM THIS TIE LINE;
 *              ENDIF;
 *            ENDIF;
 *          ENDDO;
 *        ENDIF;
 *        IF ERRCODE=0 THEN
 *          COPY DOMALFM1;    GENERATION CAPABILITY TESTS
 *          COPY DOMALFM2;    AREA REQUIREMENT COMPUTATION
 *          IF |SMOOTHED AREA REQUIREMENT| LE Y THEN
 *            COPY DOMALFM3;  DELTA GEN.SINCE LAST EDC COMPUTATIONS
 *            COPY DOMALFM4;  ADJUSTMENT OF DESIRED POWER READING
 *            IF |SAR| GT X THEN
 *              COPY DOMALFM5;     NORMAL MODE COMPUTATION
 *            ELSE
 *              COPY DOMALFM6;     DEAD BAND MODE COMPUTATION
 *            ENDIF;
 *          ELSE
 *            COPY DOMALFM7;       EMERGENCY ASSIST MODE COMPUTATION
 *          ENDIF
 *          COPY DOMALFCA;         POST PROCESSING
 *        ELSE
 *          ISSUE ALARM:'UNABLE TO DO LFC:LACK OF TIE LINE DATA.';
 *        ENDIF;
 *        ENDSEG LFCB;
 *        EXIT DOMALFB4;
 **
 **
 *    DOMALFM1 CHECKS THE GENERATION CONTROL CAPABILITY OF THE SYSTEM
 *
 *        BEGIN DOMALFM1;
 *        SET CAPABILITY TO LOWER GENERATION TO ZERO;
 *        SET CAPABILITY TO RAISE GENERATION TO ZERO;
 *        FOR EACH GENERATOR DO;
 *          IF AAOLFCIC SHOWS THIS GENERATOR UNDER LFC CONTROL THEN
 *            CAP.TO RAISE=CAP.TO RAISE +(HI.EMERG.LIMIT-ACT.POWER.RDG);
 *            CAP.TO LOWER=CAP.TO LOWER +(LO.EMERG.LIMIT-ACT.POWER.RDG);
 *          ENDIF;
```

```
MEMBER NAME  DOMAPDLL
 *         ENDDO;
 *         IF CAP.TO.RAISE < RAISE CRITERION THEN
 *           ISSUE ALARM:'LFC CAP.TO RAISE GEN.< PRESCRIBED STANDARD';
 *         ENDIF;
 *         IF CAP.TO.LOWER > LOWER CRITERION THEN
 *           ISSUE ALARM:'LFC CAP.TO LOWER GEN.< PRESCRIBED STANDARD';
 *         ENDIF;
 *         EXIT DOMALFM1;
 **
 **
 *         DOMALFM2 COMPUTES THE INSTANTANEOUS AREA REQUIREMENT (IAR) AND
 *           THE SMOOTHED AREA REQUIREMENT (SAR)
 *
 *         BEGIN DOMALFM2;
 *         MAKE A COPY OF THE OLD LFC OUTPUT TABLE IN AALFCOTT;
 *         IAR=T(NATF-NSTF)-10K(FREQS+FREQO-FREQA);
 *         SAR=(1-N)SAR + N(IAR);
 *         ABSSAR=|SAR|;
 *         EXIT DOMALFM2;
 **
 **
 *         DOMALFM3 CALCULATES DELTA,THE DIFFERENCE BETWEEN THE TOTAL
 *         CURRENT GENERATION OF ALL GENERATORS UNDER LFC AND THE TOTAL
 *         GENERATION THAT THEY HAD AT THE TIME OF THE LAST AUTOMATIC EDC
 *         IT ALSO INITIALIZES PARTICIPATION FACTORS AND EXCLUSION
 *         INDICATORS.
 *
 *         BEGIN DOMALFM3;
 *         SET EDC_DEMAND = 0;          GENERATION AT LAST EDC TIME
 *         SET LFC_DEMAND = 0:          GENERATION AT THIS LFC TIME
 *         SET SUMPF=0;                 SUM OF PARTICIPATION FACTORS
 *         DO THE FOLLOWING FOR EACH GENERATOR;
 *           IF THIS GENERATOR UNDER LFC CONTROL THEN
 *             EDC_DEMAND = EDC_DEMAND + ITS DESIRED ECONOMIC SETTING;
 *             LFC_DEMAND = LFC_DEMAND + ITS ACTUAL POWER READING;
 *             INDICATE THIS GENERATOR TO BE INCLUDED IN AALFCEXC;
 *             SET ITS PARTICPATION FACTOR IN AALFCRHO TO SETTING AT LAST
 *                   AUTOMATIC EDC;
 *             SUMPF = SUMPF + PARTICIPATION FACTOR OF THIS GENERATOR;
 *           ENDIF;
 *         ENDDO;
 *         NORMALIZE PARTICIPATION FACTORS;(USE DOMABLK4)
 *         DELTA=LFC_DEMAND-EDC_DEMAND;
 *         EXIT DOMALFM3;
 **
 **
 *        (DOMALFM4 ADJUSTS THE ECONOMIC DESIRED POWER SETTINGS OF
 *         AAOPDEDC TO ACCOUNT FOR DELTA AND STORES THE ADJUSTED ECON.
 *         DESIRED SETTINGS IN AAOPDLFC.)
 *
 *       BEGIN DOMALFM4;
 *       SUMPF=1;
 *       WHILE SUMPF > 0 DO;
 *         FOR EVERY  GENERATOR DO;
 *           IF  AAOICLFC  SHOWS THIS GENERATOR ON CONTROL THEN
 *             IF  AALFCEXC  SHOWS THIS GENERATOR NOT EXCLUDED THEN
 *               ADJUSTED ECON.  SETTING = DELTA*PARTICIPATION FACTOR
 *                     + ECON.  SETTING AT THE TIME OF THE LAST EDC;
```

```
MEMBER NAME  DOMAPDLL
 *             ENDIF:
 *           ENDIF;
 *         ENDDO;
 *
 *        (IF THE GENERATION HAS GONE DOWN SINCE THE LAST EDC THEN IN
 *         MAKING THE ADJUSTMENT WE MAY HAVE VIOLATED THE LOW ECONOMIC
 *         LIMITS.IF WE HAVE VIOLATED THEM FOR A GIVEN GENERATOR WE SET
 *         THE DESIRED POWER SETTING TO THE LOW ECONOMIC LIMIT,ADJUST THE
 *         VALUE OF DELTA,EXCLUDE THE GENERATOR FROM FURTHER CALCULATIONS
 *         SET ITS PARTICIPATION FACTOR TO ZERO,AND ADJUST THE PARTICIPA-
 *         TION FACTORS OF ALL GENERATORS.IF THERE ARE NO GENERATORS LEFT
 *         ON WHICH TO MAKE THE CHANGES WE INDICATE THAT WE CAN'T CORRECT
 *         FOR ECONOMICS AND SET THE SOLELY ECONOMIC CORRECTION DELTA TO
 *         ZERO FOR ALL GENERATORS.AFTER MAKING THE NECESSARY ADJUSTMENTS
 *         FOR THE VIOLATION OF LIMITS BY A SINGLE GENERATOR WE REPEAT
 *         THE ENTIRE DOMALFM4 CALCULATION FROM THE BEGINNING.)
 *
 *         OUTLIMIT=0;
 *         IF DELTA < 0 THEN
 *           FOR EACH GENERATOR DO UNTIL OUTLIMIT ¬= 0;
 *             IF AAOICLFC SHOWS THIS GENERATOR ON CONTROL THEN
 *               IF ADJUSTED ECON.SETTING < LO.ECON.LIMIT THEN
 *                 ADJ.ECON.SETTING = LO.ECON.LIMIT;
 *                 DELTA=DELTA-(LFC DESIRED ECONOMIC SETTING - EDC DES-
 *                        IRED ECONOMIC SETTING);
 *                 EXCLUDE GENERATOR FROM FURTHER CALCULATIONS;(AALFCEXC)
 *                 SET ITS PARTICIPATION FACTOR TO ZERO;(AALFCRHO)
 *                 SUMPF=SUM OF PARTICIPATION FACTORS OF ALL GENERATORS
 *                        SHOWN UNDER LFC CONTROL IN AAOICLFC;
 *                 OUTLIMIT=1;
 *                 IF SUMPF ¬> ZERO THEN
 *                   FOR EACH GENERATOR DO;
 *                     IF AAOICLFC SHOWS GEN. IS UNDER CONTROL THEN
 *                       SET ITS ECONOMIC CORRECTION IN AAODELTP TO ZERO;
 *                     ENDIF;
 *                   ENDDO;
 *                   ISSUE ALARM:'LFC COULD NOT CORRECT FOR ECONOMICS';
 *                 ELSE
 *                   FOR EACH GENERATOR DO;
 *                     IF AAOICLFC INDICATES GENERATOR ON CONTROL THEN
 *                       NORMALIZE ITS PARTICIPATION FACTOR BY DIVIDING
 *                              IT BY SUMPF;
 *                     ENDIF;
 *                   ENDDO;
 *                 ENDIF;
 *               ENDIF;
 *             ENDIF;
 *           ENDDO;
 *           IF OUTLIMIT=0 THEN
 *             FOR EACH GENERATOR DO;
 *               IF AAOICLFC SHOWS THIS GEN ON CONTROL THEN
 *                 ECONOMIC CORRECTION(AAODELTP)=LFC DESIRED POWER SET-
 *                        TING - ACTUAL POWER READING);
 *               ENDIF;
 *             ENDDO;
 *             SUMPF=0;
 *           ENDIF;
 *         ELSE
```

```
MEMBER NAME  DOMAPOLL
 *          FOR EACH GENERATOR DO UNTIL OUTLIMIT ¬= 0;
 *            IF AAOICLFC SHOWS THIS GENERATOR ON CONTROL THEN
 *              IF ADJUSTED ECON.SETTING > HI.ECON.LIMIT THEN
 *                ADJ.ECON.SETTING = HI.ECON.LIMIT;
 *                DELTA=DELTA-(LFC DESIRED ECON.SETTING - EDC DESIRED
 *                      ECONOMIC SETTING);
 *                EXCLUDE GENERATOR FROM FURTHER CALCULATIONS;(AALFCEXC)
 *                SET ITS PARTICIPATION FACTOR TO ZERO;(AALFCRHO)
 *                SUMPF=SUM OF PARTICIPATION FACTORS OF ALL GENERATORS
 *                      SHOWN UNDER LFC CONTROL IN AAOICLFC;
 *                OUTLIMIT=1;
 *                IF SUMPF ¬> ZERO THEN
 *                  FOR EACH GENERATOR DO;
 *                    IF AAOICLFC SHOWS GEN. UNDER CONTROL THEN
 *                      SET ITS ECONOMIC CORRECTION IN AAODELTP TO ZERO;
 *                    ENDIF;
 *                  ENDDO;
 *                  ISSUE ALARM:'LFC COULD NOT CORRECT FOR ECONOMICS';
 *                ELSE
 *                  FOR EACH GENERATOR DO;
 *                    IF AAOICLFC SHOWS GENERATOR ON CONTROL THEN
 *                      NORMALIZE ITS PARTICIPATION FACTOR BY DIVIDING
 *                            IT BY SUMPF;
 *                    ENDIF;
 *                  ENDDO;
 *                ENDIF;
 *              ENDIF;
 *            ENDIF;
 *          ENDDO;
 *          IF OUTLIMIT=0 THEN
 *            FOR EACH GENERATOR DO;
 *              IF AAOICLFC SHOWS THIS GEN.ON CONTROL THEN
 *                ECONOMIC CORRECTION(AAODELTP)= LFC DESIRED POWER SET-
 *                      TING - ACTUAL POWER READING);
 *              ENDIF;
 *            ENDDO;
 *            SUMPF=0;
 *          ENDIF;
 *        ENDIF;
 *      ENDDO;
 *      EXIT DOMALFM4;
**
**
 *        DOMALFM5 PERFORMS THE NORMAL MODE COMPUTATIONS
 *
 *        BEGIN DOMALFM5;
 *        SET LFC DYNAMIC MODE TO 2(NORMAL) IN AALFCOTT;
 *        SUMPF=0;
 *        FOR EACH GENERATOR DO;
 *          IF AAOICLFC SHOWS THIS GENERATOR ON CONTROL THEN
 *            SET AALFCEXC TO INCLUDE GENERATOR IN COMPUTATION;
 *            SET LFC PARTICIPATION FACTORS IN AALFCRHO TO THOSE COMPUT-
 *                  ED DURING THE LAST AUTOMATIC EDC;
 *            SUMPF=SUMPF + PARTICIPATION FACTOR FOR THIS GENERATOR;
 *          ENDIF;
 *        ENDDO;
 *        FOR EACH GENERATOR DO;
 *          IF AAOICLFC SHOWS GENERATOR ON CONTROL THEN
```

```
MEMBER NAME  DOMAPDLL
*            NORMALIZE ITS PARTICIPATION FACTOR BY DIVIDING BY SUMPF;
*          ENDIF;
*        ENDDO;
*        USAR=SAR+RAMPD;  (USABLE AREA REQ.= SMOOTHED AREA REQ.+ RAMP
*              DELTA USED TO CONTROL BASE LOADED GENERATORS)
*        SUMPF=1;
*        TUSAR=USAR
*        DO WHILE SUMPF > 0;
*          FOR EACH GENERATOR DO;
*            IF AAOICLFC SAYS ON CTRL. AND AALFCEXC SAYS INCLUDE THEN
*              COMPUTE UNIT CORRECTION AND STORE IN AALFCUNC; (UNIT
*                    CORRECTION=GAINB(AAODELTP)+GAINC*USAR(AALFCRHO))
*              ADJ.ECON.SETTING IN AAOPDLFC = UNIT CORRECTION + ACTUAL;
*            ENDIF;
*          ENDDO;
*          OUTLIMIT=0;
*          FOR  EACH  GENERATOR DO UNTIL OUTLIMIT IS NON ZERO;
*            IF  AAOICLFC  INDICATES ON CONTROL THEN
*              IF PERMISSIVE CONTROL=YES THEN
*                IF TUSAR < 0 THEN
*                  IF ADJUSTED ECON. SETTING < ACTUAL VALUE OR
*                  IF HIGH ECON. LIMIT < ACTUAL VALUE THEN
*                    ADJUSTED ECON. SETTING = ACTUAL VALUE
*                    STORE ADJUSTED ECON. SETTING IN DATABASE
*                    SET IHI=1
*                    OUTLIMIT=1
*                  ELSE
*                    IF ADJUSTED ECON. SETTING > HIGH ECON. LIMIT THEN
*                      STORE HIGH ECONOMIC LIMIT IN DATABASE
*                      SET IHI=1
*                      OUTLIMIT=1
*                    ENDIF
*                  ENDIF
*                ELSE                    TUSAR > 0
*                  IF ADJUSTED ECON. SETTING > ACTUAL VALUE OR
*                  IF LOW ECON. LIMIT > ACTUAL VALUE
*                    ADJUSTED ECON. SETTING = ACTUAL VALUE
*                    STORE ADJUSTED ECON. SETTING IN DATABASE
*                    SET IHI=0
*                    OUTLIMIT=1
*                  ELSE
*                    IF ADJUSTED ECON. SETTING < LOW ECON. LIMIT THEN
*                      STORE ADJUSTED ECON. SETTING IN DATABASE
*                      SET IHI=0
*                      OUTLIMIT=1
*                    ENDIF
*                  ENDIF
*                ENDIF
*              ELSE                    PERMISSIVE CONTROL=NO
*                IF ADJ. ECON. SETTING OUTSIDE OF LIMITS THEN
*                  OUTLIMIT=1
*                  IF ADJ. ECON. SETTING OUTSIDE OF HIGH LIMIT THEN
*                    SET ADJ. ECON. SETTING TO HI LIMIT;
*                    SET IHI = 1;
*                  ELSE
*                    SET ADJ. ECON  SETTING TO LO LIMIT;
*                    SET IHI = 0;
*                    ENDIF;
```

```
MEMBER NAME  DOMAPDLL
*                ENDIF;
*              ENDIF
*              IF OUTLIMIT > 0 THEN
*                IF GAINC NOT 0 THEN
*                  USAR = USAR+((ADJ.ECON.-ACTUAL)-GAINB(AAODELTP))/
*                   GAINC)
*                ENDIF
*                EXCLUDE   GENERATOR FROM CALCULATIONS; (AALFCEXC =1)
*                SET  ITS  PARTICIPATION FACTOR TO ZERO IN AALFCRHO;
*                SUMPF = SUM OF PART.FACTORS FOR ALL GEN. THAT AAOICLFC
*                        SHOWS ON CONTROL;
*                IF  SUMPF  > 0 THEN
*                  FOR  EACH  GENERATOR DO;
*                    IF  AAOICLFC  SHOWS GENERATOR ON CONTROL THEN
*                      NORMALIZE   ITS PARTICIPATION FACTOR BY DIVIDING
*                              IT BY SUMPF;
*                    ENDIF;
*                  ENDDO;
*                ELSE
*                  ERRCODE=1;
*                  IF IHI = 1 THEN
*                    ISSUE ALARM:'LFC UNABLE TO CORRECT SAR WITH
*                            GENERATION AT HIGH ECONOMIC LIMITS';
*                  ELSE
*                    ISSUE ALARM:'LFC UNABLE TO CORRECT SAR WITH
*                            GENERATION AT LOW ECONOMIC LIMITS';
*                  ENDIF;
*                ENDIF;
*              ENDIF;
*            ENDIF;
*          ENDDO;
*        ENDDO;
*        SLOP=0;
*        IF PERMISSIVE CONTROL = YES AND (ERRCODE=1 OR GAINC=0) THEN
*          FOR EACH GENERATOR DO
*            IF AAOICLFC SHOWS GENERATOR ON CONTROL THEN
*            UNIT CORRECTION = 0
*            ENDIF
*          ENDDO
*        ELSE
*        FOR EACH GENERATOR DO;
*          IF AAOICLFC SHOWS THIS GENERATOR UNDER CONTROL THEN
*            UNIT CORRECTION = (LFC DESIRED SETTING - ACTUAL) + SLOP;
*            SLOP=0;
*            MAX.ALLOWED CORRECTION =(MAX.SHORT TERM RAMP RATE)(LFC
*                  INTERVAL);
*            IF |UNIT CORRECTION| > MAX.ALLOWED CORRECTION THEN
*              IF UNIT CORRECTION IS NEGATIVE THEN
*                CHANGE SIGN OF MAX ALLOWED CORRECTION;
*                SET UNIT CORRECTION = MAX.ALLOWED CORRECTION;
*              ENDIF;
*              STORE UNIT CORRECTION IN AALFCUNC;
*            ELSE
*            STORE UNIT CORRECTION IN AALFCUNC;
*            MIN.ALLOWED CORRECTION =(MIN.USABLE RATE)(LFC INTERVAL);
*            IF |UNIT CORRECTION| < MIN.ALLOWED CORRECTION THEN
*              SLOP = UNIT CORRECTION FOR THIS GENERATOR;
*              SET UNIT CORRECTION=0;
```

```
MEMBER NAME  DOMAPDLL
*               STORE UNIT CORRECTION IN AALFCUNC;
*             ENDIF;
*           ENDIF;
*         ENDDO;
*         ENDIF
*         EXIT DOMALFM5;
**
**
*         DOMALFM6 PERFORMS THE DEAD BAND MODE COMPUTATIONS
*         SET LFC DYNAMIC MODE TO 1(DEAD BAND) IN AALFCOTT;
*         FOR EACH GENERATOR DO;
*           IF AAOICLFC SHOWS THIS GENERATOR ON CONTROL THEN
*             UNIT CORRECTION=DEAD BAND GAIN (AAODELTP)
*             MAX.ALLOWED CORRECTION=(MAX.SHORT TERM RAMP RATE)(LFC
*                    INTERVAL);
*             IF |UNIT CORRECTION| > MAX.ALLOWED CORRECTION THEN
*               IF UNIT CORRECTION IS NEGATIVE THEN
*                 CHANGE SIGN OF MAX.ALLOWED CORRECTION;
*               ENDIF;
*               STORE MAX.ALLOWED CORRECTION IN AALFCUNC;
*             ELSE
*               STORE UNIT CORRECTION IN AALFCUNC;
*             ENDIF;
*           ENDIF;
*         ENDDO;
*         EXIT DOMALFM6
**
**
*         DOMALFM7 PERFORMS EMERGENCY ASSIST MODE COMPUTATIONS
*
*         BEGIN DOMALFM7;
*         SET LFC DYNAMIC MODE TO 3(EMERGENCY ASSIST) IN AALFCOTT;
*         IF SAR IS NEGATIVE THEN
*           FOR EACH GENERATOR DO;
*             IF AAOICLFC SHOWS GENERATOR ON CONTROL OR
*              UNIT BASE LOADED TO BE USED IN EMER. ASSIST MODE THEN
*               UNIT CORRECTION =GAINA(HI EMERG.LIMIT-ACTUAL);
*               IF |UNIT CORRECTION| > MAX.ALLOWED CORRECTION THEN
*                 ADJUST UNIT CORRECTION ACCORDINGLY;
*               ENDIF;
*             ENDIF;
*           ENDDO;
*         ELSE
*           FOR EACH GENERATOR DO;
*             IF AAOICLFC SHOWS GENERATOR ON CONTROL THEN
*               UNIT CORRECTION= GAINA(LO EMERG.LIMIT-ACTUAL);
*               IF |UNIT CORRECTION| > MAX.ALLOWED CORRECTION THEN
*                 ADJUST UNIT CORRECTION ACCORDINGLY;
*               ENDIF;
*             ENDIF;
*           ENDDO;
*         ENDIF;
*         EXIT DOMALFM7;
**
**
*         DOMALFCA PERFORMS POSTPROCESSING FOLLOWING THE LFC ALGORITHM
*
*         BEGIN DOMALFCA;
```

```
MEMBER NAME  DOMAPDLL
*         IF DYNAMIC MODE WAS EMERGENCY ASSIST ON THIS LFC ENTRY AND THE
*               PREVIOUS ONE THEN
*           IF THE SIGN OF SAR DIFFERS ON THIS ENTRY AND PREVIOUS 1 THEN
*             SUPPRESS LFC (IN AILFCSSP);
*             SET RUNNING MODE = 0 IN AALFCOTT;
*             ISSUE ALARM:'LFC HAS SUSPENDED ITSELF';
*           ENDIF;
*         ENDIF;
*         IF LFC IS NOT SUSPENDED THEN
*           IF LFC MODE ADJUSTMENT=1 AND FREQ.BIAS ¬= 0 THEN
*             SET RUNNING MODE TO TIE LINE BIAS IN AALFCOTT;
*           ELSE
*             IF LFC MODE ADJ.=0 AND FREQ.BIAS ¬=0 THEN
*               SET RUNNING MODE TO FLAT FREQ.CONTROL IN AALFCOTT;
*             ELSE
*               IF LFC MODE ADJ.=1 AND FREQ.BIAS =0 THEN
*                 SET RUNNING MODE TO FLAT TIE LINE CONTROL IN AALFCOTT;
*               ELSE
*                 ISSUE ALARM:'UNDEFINED RUNNING MODE IN LFC';
*               ENDIF;
*             ENDIF;
*           ENDIF;
*           FOREACH GENERATOR DO;
*             STORE UNIT CORRECTION IN AALFCOTT;
*             ADD UNIT CORRECTION TO VALUE IN AATLFCIC TO ACCUMULATE
*                   COMMANDS BETWEEN EDC ENTRIES;
*           ENDDO;
*           COPY DATA FROM AALFCOTT INTO AALFCOUT;
*           CALL DOMALFCO; (CUSTOMIZED INTERFACE PROCESSOR THAT CONVERTS
*                 UNIT CORRECTIONS FROM MW TO PULSE LENGTHS.)
*         ELSE
*           SET RUNNING MODE TO SUSPENDED
*         ENDIF;
*         EXIT DOMALFCA;
**
**
*         DOMALFCO IS THE OUTPUT INTERFACE PROCESSOR.
*
*         BEGIN DOMALFCO;
*         IF THIS IS THE FIRST ENTRY TO DOMALFCO THEN
*           DETERMINE THE ADDRESSES OF ARRAYS USED BY DOMALFCO;
*           MAKE PROVISIONS TO LOCK AALFCPDO WHEN NECESSARY;
*           STORE THE ITEM NAMES OF GENERATOR STATUS DATA IN AALFCPDO;
*           SHOW THAT THE FIRST ENTRY HAS BEEN MADE;
*         ENDIF;
*         FOR EACH GENERATOR DO;
*           NUMBER OF PULSE INTERVALS = A(UNIT CORRECTION) + B;
*           FIX NUMBER OF PULSES AND STORE IN TEMPORARY ARRAY;
*         ENDDO;
*         LOCK AALFCPDO;
*         STORE OUTPUTS IN AALFCPDO;
*         UNLOCK AALFCPDO;
*         CALL DOMCGENO; (SUPERVISORY CONTROL PROCESSOR THAT INITIATES
*               TRANSMISSION OF PULSES)
*         EXIT DOMALFCO;
**
**
*         OPERATOR INTERFACE PROCESSING IS DONE BY INDEPENDENT AND
```

```
MEMBER NAME   DOMAPDLL
*              DEPENDENT TASKS THAT ARE PATCHED BY DISPLAY MANAGEMENT
*              WHEN A FUNTION KEY IS DEPRESSED.
**
**
*  DOMALMUP ISSUES THE DISPLAY MANAGEMENT MACRO NECESSARY TO DISPLAY
*        THE LFC MENU OR THE LFC 'ONLY' MENU OR THE EDC MENU.
**
*        SAVE THE ADDRESS OF THE CVT,THE ADDRESS OF THE DISPLAY
*             PARAMETER LIST,AND THE PATCH ID.
*        IF THE PATCH ID IS 1 THEN
*          LOAD THE ADDRESS OF THE AIDIN PARM LIST
*          ISSUE DISPUP MACRO(DLIST OPTION) TO DISPLAY LFC OR EDC MENU
*             ON THE REQUESTING UNIT.
*        ENDIF
*  EXIT DOMALMUP
**
*  DOMALTPG CONTAINS LOGIC NECESSARY TO DISPLAY THE LFC OUTPUT TABLE
*        DATA AND PAGE THRU THAT DATA FOR ANY NUMBER OF REQUESTNG UNITS
**
*        SAVE THE ADDRESS OF THE CVT,THE ADDRESS OF THE DISPLAY
*             PARAMETER LIST, AND THE PATCH ID.
*        IF THIS IS 1ST EXECUTION THEN
*          INDICATE 1ST EXEC. HAS OCCURED
*          GET TEXT PARAMETER
*          GET NUMBER OF PLANTS
*          GET ADDRESS OF PLANT NAME ARRAY
*          GET ADDRESS OF LFC OUTPUT TABLE
*        ENDIF
*        CALCULATE THE NO. OF PAGES(NP)-MAX OF 8 PLANTS / PAGE
*        IF PATCH ID IS 1 THEN PROCESS 1ST PAGE REQUEST
*          STORE PAGE NUMBER ONE IN CEATAB ENTRY FOR LFC(CEALFCM)
*        ENDIF
*        IF PATCH ID IS 3 OR 4 OR 14 THEN PROCESS PAGING REQUEST
*           OR REFRESH
*          IF PATCH ID IS 14 THEN STORE CURRENT PAGE NUMBER
*          ELSE
*            IF PATCH ID IS 3 THEN PAGE FORWARD
*              INCREMENT CURRENT PAGE NO.(WRAP-AROUND IF ON LAST PAGE)
*            ELSE PAGE BACKWARD(PATCH ID 4)
*              DECREMENT CURRENT PAGE NO.(WRAP-AROUND IF ON FIRST PAGE)
*            ENDIF
*          ENDIF
*        ENDIF
*        ISSUE DINFO MACRO FOR PAGE NO. AND TOTAL NO.OF PAGES
*        DETERMINE ADDRESSES OF DATA FOR REQUESTED PAGE
*        FOR EACH SET OF 8 PLANTS DO
*          ISSUE DINFO FOR ONE LINE OF PLANT NAMES
*          ISSUE DINFO FOR ONE LINE OF 'GEN' CHARACTERS
*          FOR EACH PLANT DO
*            ISSUE DINFO FOR UNIT CORRECTION  OF EACH GENERATOR
*            ISSUE DINFO FOR GENERATOR NUMBER OF EACH GERERATOR
*          ENDDO
*        ENDDO
*        ISSUE DISPUP MACRO TO OUTPUT DYNAMIC DATA TO REQUESTING UNIT
*  EXIT DOMALTPG
**
*  DOMAMTUP CONTAINS LOGIC NECESSARY TO DISPLAY MODE CHANGE AND TIME
*        CORRECTION DATA AND PROCESS DATA ENTRY CHANGES REQUESTED BY
```

```
MEMBER NAME  DOMAPDLL
*         THE DISPATCHER FOR THIS DISPLAY.
**
*         SAVE THE ADDRESS OF THE CVT,THE ADDRESS OF THE DISPLAY PARMS,
*              THE PATCH ID,AND THE ADDRESS OF THE
*              DATA ENTRY CHANGE LIST OR THE AIDIN PARMS.
*         IF THIS IS 1ST EXECUTION THEN
*           INDICATE 1ST EXEC. HAS OCCURED
*           GET ADDRESSES OF THE FOLLOWING PARAMETERS
*             RUNNING MODE
*             FREQUENCY BIAS(K)
*             LFC MODE ADJ.(T)
*             CNTL WD TO SUSP LFC
*             1ST GEN. STATUS
*             DATE TO START TIME CORRECTION
*             DATA TO STOP TIME CORRECTION
*             TIME TO START TIME CORRECTION
*             TIME TO STOP TIME CORRECTION
*             TIME SPEED UP/SLOW DOWN IND.
*             FREQUENCY OFFSET
*             NO. GENERATORS
*             ACCESS/FUNCTION AREA
*             TEXT PARAMETERS
*           GET ADDRESS OF SRTOS TIME ARRAY
*         ENDIF
*         IF PATCH ID IS 7 THEN  PROCESS DATA ENTRY PATCH
*           ZERO OR BLANK OUT ANY WORK AREAS
*           LOAD APPROPRIATE ADDRESSES FOR PROCESSING DATA ENTRY CHANGES
*           DO LOOP THRU DATA ENTRY CHANGE LIST
*             MOVE LIMIT CHECKED VALUES INTC WORK AREA
*             SET BAD INPUT INDICATOR IF ANY VALUE INPUT FAILED LIMIT CK
*           ENDDO
*           IF ALL INPUT VALUES PASSED LIMIT CHECKING THEN
*             IF A MODE CHANGE IS REQUESTED THEN
*               IF MODE REQUESTED IS SUSPENDED MODE THEN
*                 SET APPROPRIATE PARAMETERS IN DATA BASE TO SUSPEND LFC
*                 EVENT MODE CHANGED TO SUSPENDED
*               ELSE
*               IF AT LEAST ONE GENERATOR IS ON AUTOMATIC CONTROL THEN
*                 SET APPROPRIATE PARMS IN DATA BASE FOR REQUESTED MODE
*                 EVENT REQUESTED MODE CHANGE
*               ELSE
*                 ISSUE ERROR MESSAGE AND FORCE SUSPENDED MODE
*               ENDIF
*               ENDIF
*             ENDIF
*             IF A TIME CORRECTION IS REQUESTED THEN
*               CONVERT TIMES TO FIXED POINT 1/100 SEC
*               IF DATES ARE TODAY OR TOMORROW AND
*               IF TIMES ARE LESS THAN 24 HOURS AND
*               IF START TIME IS GREATER THAN CURRENT TIME AND
*               IF STOP  TIME IS GREATER THAN START TIME THEN
*                 STORE TIME CORRECTION VALUES IN DATA BASE
*                 ISSUE PTIME TO DOMATCOR AT START TIME-PASS FREQ OFFSET
*                 ISSUE PTIME TO DOMATCOR AT STOP TIME-PASS ZERO
*                 EVENT TIME CORRECTION PARAMETERS
*               ELSE
*                 ISSUE ERROR MESSAGE
*                 EVENT BAD TIME CORRECTION DATA ENTERED
```

```
MEMBER NAME  DOMAPDLL
*                ENDIF
*              ENDIF
*            ELSE
*              EVENT BAD DATA ENTRY PARAMETERS
*            ENDIF
*            LOG AALFCODD ARRAY
*          ENDIF
*          IF PATCH ID IS 1 THEN INITIAL REQUEST
*            PICK UP PATCH PARMS DIFFERENT FROM PATCH ID 7
*          ENDIF
*          IF PATCH ID IS 5 THEN PROCESS DELETION OF TIME CORRECTION
*            IF ACCESS/FUNCTION AREA OF REQUESTOR IS VALID THEN
*              ISSUE PTIME DELETE FOR DOMATCOR
*              ZERO TIME CORRECTION ARRAY IN DATA BASE
*              EVENT TIME CORRECTION DELETION
*            ELSE
*              ISSUE ERROR MESSAGE
*              EVENT ACCESS/FUNCTION VIOLATION
*            ENDIF
*          ENDIF
*          CONVERT DATES AND TIMES FOR OUTPUT
*          ISSUE DINFO FOR DATES AND TIMES
*          ISSUE DISPUP TO OUTPUT DYNAMIC DATA TO REQUESTING UNIT
*   EXIT DOMAMTUP
**
**
*   DOMATCOR STORES FREQUENCY OFFSET PARAMETER INTO THE DATA BASE
**
*          SAVE THE ADDRESS OF THE CVT
*          LOAD THE FREQUENCY OFFSET PARAMETER FROM THE PARM LIST
*          IF THIS IS 1ST EXECUTION THEN
*            INDICATE 1ST EXEC HAS OCCURED
*            GET ADDRESS OF FREQUENCY OFFSET
*          ENDIF
*          STORE FREQUENCY OFFSET IN DATA BASE
*   EXIT DOMATCOR
**
*   DOMATLPG CONTAINS THE LOGIC NECESSARY TO DISPLAY TIE LINE DATA, PAGE
*          THRU THAT DATA FOR ANY NUMBER OF REQUESTING UNITS,AND PROCESS
*          DATA ENTRY CHANGES FOR THIS DISPLAY.
**
*          SAVE THE ADDRESS OF THE CVT,THE ADDRESS OF THE DISPLAY PARMS
*               OR DOMATLDP PARMS,THE PATCH ID,AND THE ADDRESS OF THE
*               DATA ENTRY CHANGE LIST OR THE AIDIN PARMS.
*          IF THIS IS FIRST EXECUTION THEN
*            INDICATE 1ST EXEC. HAS OCCURED
*            GET THE NUMBER OF INTERCHANGE COMPANIES
*            GET THE ADDRESSES OF THE FOLLOWING PARAMETERS:
*              RUNNING MODE
*              COMPANY NAME
*              COMPANY CURRENT SCHEDULED VALUE
*              COMPANY DESIRED SCHEDULED VALUE - SCHEDULE 1 & 2
*              DATE TO START COMPANY SCHEDULED CHANGES - SCHEDULE 1 & 2
*              TIME TO START COMPANY SCHEDULED CHANGES - SCHEDULE 1 & 2
*              DATE TO STOP  COMPANY SCHEDULED CHANGES - SCHEDULE 1 & 2
*              TIME TO STOP  COMPANY SCHEDULED CHANGES - SCHEDULE 1 & 2
*              MAXIMUM RATE OF CHANGE - SCHEDULE 1 & 2
*              TEXT PARAMETERS
```

```
MEMBER NAME  DOMAPDLL
*           GET ADDRESS OF SRTOS TIME ARRAY
*         ENDIF
*         IF PATCH ID IS 100 THEN PROCESS INITIAL REQUEST-1ST PAGE
*           GETMAIN 16 BYTE AREA FOR PAGING INFO FOR REQUESTING UNIT
*           BUILD UNIT CONTROL BLOCK FOR PAGING(UCBP) AND CHAIN IT
*             ONTO THE RESOURCE TABLE FOR REQUESTING UNIT
*           FOR COMPANY NAMES AND ASSOCIATED PCKE POINTS IN SYSTEM DO
*             ISSUE DINFO FOR ONE LINE OF COMPANY NAMES
*             ISSUE DINFO FOR ONE LINE OF COMPANY POKE POINTS
*           ENDDO
*           ISSUE DISPUP TO OUTPUT COMPANY NAMES AND ASSOCIATED
*             POKE POINTS TO THE REQUESTING UNIT
*         ELSE
*           IF PATCH ID IS LESS THAN OR EQUAL TO 51 THEN PAGING REQUEST
*             DO SEARCH FOR REQUESTING UNITS UCBP
*             IF PATCH ID IS LESS THAN OR EQUAL TO 24 THEN
*               USE PATCH ID AS PAGE REQUESTED BY THIS UNIT
*             ELSE
*               IF PATCH ID IS 51 THEN PAGE FORWARD
*                 INCREMENT CURRENT PAGE NO.
*               ELSE PAGE BACKWARD(PATCH ID OF 50)
*                 DECREMENT CURRENT PAGE NO.
*               ENDIF
*             ENDIF
*           ELSE
*             IF PATCH ID IS 98 OR 99 THEN DELETE SCHEDULE 1 OR 2
*               DELETE REQUESTED SCHEDULE 1 OR 2 BY
*               ZEROING PARAMETERS IN DATA BASE
*               FOR CURRENT PAGE (COMPANY).
*               EVENT SCHEDULE DELETION.
*             ELSE PROCESS DATA ENTRY PATCH(PATCH ID OF 52)
*               ZERO OR BLANK OUT ANY WORK AREAS
*               DO SEARCH FOR REQUESTING UNITS UCBP
*               LOAD NEEDED ADDRESSES
*               DO LOOP THRU DATA ENTRY CHANGE LIST
*                 MOVE LIMIT CHECKED VALUES INTO WORK AREA
*               ENDDO
*               IF CURRENT SCHEDULED VALUE HAS CHANGED AND
*               IF RUNNING MODE IS SUSPENDED THEN
*                 ACCEPT CHANGE TO CSV
*                 EVENT CSV CHANGE
*               ELSE
*                 ISSUE ERROR MESSAGE
*               ENDIF
*               CONVERT TIMES TO FIXED POINT 1/100 SEC
*               IF DATES ARE TODAY OR TOMORROW AND
*               IF TIMES ARE LESS THAN 24 HOURS AND
*               IF START TIME IS GREATER THAN CURRENT TIME AND
*               IF STOP TIME IS GREATER THAN START TIME AND
*               IF (TIME TO STOP - TIME TO START)*MAX RATE OF CHANGE IS
*                 GREATER THAN OR EQUAL TO|DES.SCHED.VAL-CSV|THEN
*                 ACCEPT INTERCHANGE SCHEDULE CHANGES
*                 STORE INTERCHANGE SCHEDULE CHANGES IN DATA BASE
*                 EVENT INTERCHANGE SCHEDULES 1 AND 2
*               ENDIF
*             ENDIF
*           ENDIF
*         ENDIF
```

```
MEMBER NAME  DOMAPDLL
*         ISSUE DAUPDAT MACRO TO PROVIDE DISPLAY PROCESSING WITH THE
*           ADDRESSES OF THE COMPANY DATA TC BE OUTPUT-REQUESTED PAGE
*         CONVERT DATES AND TIMES FOR OUTPUT
*         ISSUE DINFO FOR DATES,TIMES,PAGE,AND TIE LINE STATUS
*         IF PATCHID NE 52,98,99, OR 100 THEN
*           ISSUE DISPUP TO OUTPUT DYNAMIC DATA TO REQUESTING UNIT
*         ENDIF
*   EXIT DOMATLPG
**
**
*   DOMAGSPG CONTAINS THE LOGIC NECESSARY TO DISPLAY THE GENERATOR
*         CONTROL STATUS MATRIX DATA,PAGE THRU THAT DATA FOR ANY NUMBER
*         OF REQUESTING UNITS,AND PROCESS DATA ENTRY CHANGES.
**
*         SAVE THE ADDRESS OF THE CVT,THE ADDRESS OF THE DISPLAY PARMS,
*              THE PATCH ID,AND THE ADDRESS OF THE
*              DATA ENTRY CHANGE LIST OR THE AIDIN PARMS.
*         IF THIS IS FIRST EXECUTION THEN
*           INDICATE FIRST EXECUTION HAS OCCURED
*           GET ADDRESS OF PLANT NAME ARRAY AND
*             GENERATOR CONTROL STATUS MATRIX ARRAY
*           GET THE NUMBER OF GENERATORS(NG)
*           GET TEXT PARAMETERS
*         ENDIF
*         IF PATCH ID IS 2 THEN PROCESS EXIT PATCH
*           DO SEARCH FOR REQUESTING UNITS UNIT CONTROL BLOCK FOR PAGING
*           UNCHAIN UCBP FOR THIS UNIT
*           FREEMAIN 16 BYTE AREA USED FOR UCBP
*         ELSE
*           IF PATCH ID IS 1 THEN PROCESS INITIAL REQUEST-1ST PAGE
*             DO SEARCH FOR REQ. UNITS UCBP
*             IF NOT FOUND, THEN
*               GETMAIN 16 BYTE AREA FOR PAGING INFO FOR REQUESTING UNIT
*               BUILD UNIT CONTROL BLOCK FOR PAGING(UCBP) AND CHAIN IT
*                 ONTO THE RESOURCE TABLE FOR REQUESTING UNIT
*             ELSE
*               USE EXISTING UCBP
*               STORE PAGE NUMBER 1 AS CURRENT PAGE
*             ENDIF
*           ELSE
*             IF PATCH ID IS 3 OR 4 THEN PROCESS PAGING REQUEST
*               DO SEARCH FOR REQUESTING UNITS UCBP
*               IF PATCH ID IS 3 THEN PAGE FORWARD
*                 INCREMENT CURRENT PAGE NO.(WRAP AROUND IF ON LAST PG)
*               ELSE PAGE BACKWARD (PATCH ID 4)
*                 DECREMENT CURRENT PAGE NO.(WRAP AROUND IF ON FIRST PG)
*               ENDIF
*             ELSE
*               IF PATCH ID IS 7 THEN PROCESS DATA ENTRY PATCH
*                 DO SEARCH FOR REQUESTING UNITS UCBP
*                 LOAD NEEDED ADDRESSES
*                 DO LOOP THRU DATA ENTRY CHANGE LIST
*                   IF VALUE ENTERED THEN
*                     SAVE EVENT TEXT
*                   ENDIF
*                   SEARCH THROUGH ADDRESS TABLE
*                   IF VALUE ENTERED THEN
*                     LIMIT CHECK THE VALUE
```

```
MEMBER NAME  DOMAPDLL
*                     IF VALUE PASSES THE LIMIT CHECK THEN
*                       DO NECESSARY CONVERSIONS
*                       STORE NEW VALUE IN DATA BASE
*                       EVENT DATA ENTRY CHANGES
*                     ELSE
*                     ISSUE ERROR MESSAGE
*                     ENDIF
*                   ENDIF
*                   IF EVENT GENERATED
*                    WRITE THE EVENT TO THE SCRATCH PAD ZONE
*                   ENDIF
*                 ENDDO
*                 LOG GENERATOR CONTROL STATUS MATRIX
*               ENDIF
*             ENDIF
*           ENDIF
*           ISSUE DINFO FOR PAGE NO. REQUESTED
*           ISSUE DAUPDAT MACRO TO PROVIDE DISPLAY PROCESSING WITH THE
*             ADDRESSES OF THE GCSM DATA TO BE OUTPUT(REQUESTED PAGE)
*           DO NECESSARY CONVERSIONS OF DATA FOR OUTPUT
*           ISSUE DINFO FOR CONVERTED GCSM DATA
*           ISSUE DISPUP TO OUTPUT DYNAMIC DATA TO REQUESTING UNIT
*         ENDIF
*   EXIT DOMAGSPG
**
**
*   DOMAESUS CONTAINS LOGIC NECESSARY TO SUSPEND OR ACTIVATE EDC IF THE
*         POKE POINT IS ACTIVATED ON THE GENERATOR CONTROL STATUS MATRIX
*         DISPLAY FROM A VALID ACCESS / FUNCTION AREA
**
*         SAVE THE ADDRESS OF THE CVT,THE ADDRESS OF THE DISPLAY
*              PARAMETER LIST, AND THE PATCH ID
*         IF THIS IS 1ST EXECUTION THEN
*           INDICATE 1ST EXEC. HAS OCCURED
*           GET ADDRESS OF EDC SUSPEND/ACTIVATE INDICATOR,OF 1ST GEN.
*             STATUS INDICATOR,AND OF ACCESS / FUNCTION AREA IDS
*           GET NUMBER OF GENERATORS
*           GET TEXT PARAMETER ADDRESSES
*         ENDIF
*         IF ACCESS / FUNCTION AREA OF REQUESTOR IS VALID THEN
*           IF PATCH ID IS 1 THEN SUSPEND EDC PROCESSING
*             SET SUSPEND/ACTIVATE INDICATOR IN DATA BASE TO SUSPEND EDC
*             EVENT EDC SUSPENDED
*           ELSE
*             IF PATCH ID IS 2 THEN PROCESS ACTIVATE PATCH
*               IF AT LEAST ONE GENERATOR IS ON AUTO CONTROL THEN
*                 SET SUSPEND/ACTIVATE IND.IN DATA BASE TO ACTIVATE EDC
*                 PATCH EDC FOR AUTOMATIC TYPE ENTRY
*                 PATCH EDC FOR COMPLETE TYPE ENTRY
*                 EVENT EDC ACTIVATED
*               ELSE
*                 ISSUE ERROR MESSAGE TO REQUESTING UNIT
*               ENDIF
*             ENDIF
*           ENDIF
*         ELSE
*           ISSUE ERROR MESSAGE TO REQUESTING UNIT
*           EVENT ACCESS/FUNCTION AREA VIOLATION
```

```
MEMBER NAME  DOMAPDLL
*         ENDIF
*   EXIT DOMAESUS
**
*   DOMAGPUP CONTAINS LOGIC NECESSARY TO DISPLAY GENERAL PARAMETER DATA,
*         AND PROCESS DATA ENTRY CHANGES REQUESTED FOR THIS DISPLAY
**
*         SAVE THE ADDRESS OF THE CVT,THE ADDRESS OF THE DISPLAY PARMS
*              OR DOMAGPDP PARMS,THE PATCH ID,AND THE ADDRESS OF THE
*              DATA ENTRY CHANGE LIST OR THE AIDIN PARMS.
*         IF THIS IS FIRST EXECUTION THEN
*           INDICATE FIRST EXECUTION HAS OCCURRED
*           GET TEXT PARAMETER ADDRESSES
*         ENDIF
*         IF PATCH ID IS 7 THEN PROCESS DATA ENTRY REQUEST
*           DO LOOP THRU DATA ENTRY CHANGE LIST
*             IF DATA ENTRY VALUE PASSED LIMIT CHECKING THEN
*               STORE NEW VALUE IN DATA BASE
*               EVENT DATA ENTRY CHANGES
*             ENDIF
*             IF ENTERED VALUE IS A CHANGE TO LFC INTERVAL THEN
*               CONVERT TO FIXED POINT 1/100 SEC AND STORE IN DATA BASE
*               ISSUE PTIME MOD TO LFC FOR NEW INTERVAL
*               EVENT LFC INTERVAL CHANGE
*             ENDIF
*           ENDDO
*           LOG AALFCODD ARRAY
*         ENDIF
*         CONVERT LFC INTERVAL FOR OUTPUT
*         ISSUE DINFO FOR LFC INTERVAL
*         ISSUE DISPUP TO OUTPUT DYNAMIC DATA TO REQUESTING UNIT
*   EXIT DOMAGPUP
**
**
*   DOMAMTPG CONTAINS LOGIC NECESSARY TO DISPLAY THE MANUAL EDC OUTPUT
*         TABLE DATA OR WORK TABLE DATA,PAGE THRU THAT DATA FOR ANY  NO.
*         OF REQUESTING UNITS,AND PROCESS TABLE UPDATE TYPE PATCHES.
**
*         SAVE THE ADDRESS OF THE CVT,THE ADDRESS OF THE DISPLAY PARM
*              LIST,AND THE PATCH ID.
*         IF THIS IS 1ST EXECUTION THEN
*           INDICATE 1ST EXEC.HAS OCCURED
*           GET ADDRESSES OF PLANT NAME ARRAY,MANUAL OUTPUT TABLE,AND
*             WORK TABLE ARRAY'S
*           GET NO.OF GENERATORS(NG) AND NO.OF PLANTS(NP)
*           GET ADDRESSES OF 1ST GEN.INC.IND.,1ST ACTUAL POWER,1ST
*             DESIRED POWER,AND 1ST PARTICIPATION FACTOR
*           GET TEXT DATA
*         ENDIF
*         IF PATCH ID IS GREATER THAN 4 AND LESS THAN 11 THEN
*           GET ADDRESS OF ERROR MESSAGE TEXT
*           IF A MATCH ON PRIMARY ACCESS AREA/FUNCTION CODE
*            NOT FOUND OR
*             IF A MATCH ON SECONDARY ACCESS AREA/FUNCTION CODE
*              NOT FOUND OR
*               IF A MATCH ON TERTIARY ACCESS AREA/FUNCTION CODE
*                NOT FOUND THEN
*                 FORMAT THE EVENT
*                 ISSUE THE EVENT
```

```
MEMBER NAME  DOMAPDLL
*                WRITE THE MESSAGE TO THE SCRATCH PAD ZONE
*              ENDIF
*            ENDIF
*          ENDIF
*        ENDIF
*        IF PATCH ID IS 1
*          INDICATE THAT WORK TABLE IS DISPLAYED
*        ENDIF
*        IF PATCH ID IS 10 THEN
*          COPY MANUAL EDC OUTPUT TABLE DATA INTO WORK TABLE
*        ENDIF
*        IF PATCH ID IS 5 THEN
*          COPY WORK TABLE DATA INTO MANUAL EDC OUTPUT TABLE
*          EVENT WORK TABLE IS STORED IN MANUAL TABLE
*        ENDIF
*        IF PATCH ID IS 6 THEN
*          COPY ACTUAL POWER READINGS FROM DATA BASE INTO WORK TABLE
*        ENDIF
*        IF PATCH ID IS 7 THEN
*          COPY ACTUAL INTO DESIRED WITHIN WORK TABLE FOR EACH GEN.
*           NOT INCLUDED IN EDC
*        ENDIF
*        IF PATCH ID IS 8 THEN
*          SET PART.FACTORS TO ZERO(ALL NON-EDC INCLUDED GENERATORS)
*          NORMALIZE PART_FACTORS INCLUDED IN EDC
*        ENDIF
*        IF PATCH ID IS 9 THEN
*          CALCULATE TOTAL SYSTEM GENERATION AND TOTAL SYSTEM POWER
*           DISPATCHED AND STORE THEM IN THE WORK TABLE
*        ENDIF
*        CONVERT TIME OF DISPATCH FOR OUTPUT
*        DETERMINE NUMBER OF DISPLAY PAGES
*        IF PATCH ID IS 1 THEN PROCESS INITIAL REQUEST-1ST PAGE
*          STORE PAGE NUMBER ONE IN CEATAB ENTRY FOR LFC(CEALFCM)
*        ENDIF
*        IF PATCH ID IS 3 OR 4 OR 14 THEN PROCESS PAGING REQUEST
*         OR REFRESH
*          IF PATCH ID IS 14 THEN STORE CURRENT PAGE NUMBER
*          ELSE
*            IF PATCH ID IS 3 THEN PAGE FORWARD
*              INCREMENT CURRENT PAGE NO.(WRAP-AROUND IF ON LAST PAGE)
*            ELSE PAGE BACKWARD(PATCH ID 4)
*              DECREMENT CURRENT PAGE NO.(WRAP-AROUND IF ON FIRST PAGE)
*            ENDIF
*          ENDIF
*        ENDIF
*        ISSUE DINFO'S FOR PAGE NO.,TOTAL PAGES,HOURS,MINUTES,AND
*           SECONDS FOR TIME OF DISPATCH
*        DETERMINE ADDRESSES OF DATA FOR PAGE REQUESTED
*        FOR EACH SET OF 4 PLANTS DO (1 LINE OF OUTPUT)
*          ISSUE DINFO FOR ONE LINE OF PLANTS NAMES
*          FOR EACH PLANT DO
*            ISSUE DINFO-INCLUSION INDICATOR FOR EACH GENERATOR
*            ISSUE DINFO-GENERATOR NUMBER FOR EACH GENERATOR
*            ISSUE DINFO-ACTUAL POWER OF EACH GENERATOR
*            ISSUE DINFO-DESIRED POWER OF EACH GENERATOR
*            ISSUE DINFO-DESIRED PARTICIPATION FACTOR OF EACH GEN.
*          ENDDO
```

```
MEMBER NAME  DOMAPDLL
*        ENDDO
*        ISSUE DISPUP'S TO OUTPUT DYNAMIC DATA TO REQUESTING UNIT
*  EXIT DOMAMTPG
**
*  DOMACEPG CONTAINS LOGIC NECESSARY TO DISPLAY GENERATOR DATA FROM THE
*        MANUAL EDC OUTPUT TABLE AND THE WORK TABLE,PAGE THRU THAT DATA
*        FOR ANY NUMBER OF REQUESTING UNITS,AND PROCESS DATA ENTRY
*        CHANGES TO THE WORK TABLE PORTION OF THE DISPLAY
**
*        SAVE THE ADDRESS OF THE CVT,THE ADDRESS OF THE DISPLAY PARMS,
*             THE PATCH ID,AND THE ADDRESS OF THE
*             DATA ENTRY CHANGE LIST OR THE AIDIN PARMS
*        IF THIS IS FIRST EXECUTION THEN
*          INDICATE FIRST EXEC HAS OCCURED
*          GET ADDRESSES OF PLANT NAME ARRAY AND GEN.CONTROL STATUS MX
*          GET NUMBER OF GENERATORS(NG)
*          GET ADDRESSES OF THE FOLLOWING PARAMETERS FOR BOTH TABLES
*            INCLUSION INDICATOR FOR 1ST GENERATOR
*            TIME OF DISPATCH
*            DATE OF DISPATCH
*          GET ADDRESS OF SRTOS TIME ARRAY
*        ENDIF
*        IF PATCH ID IS 2 THEN PROCESS EXIT PATCH
*          DO SEARCH FOR REQUESTING UNITS UNIT CONTROL BLOCK FOR PAGING
*          UNCHAIN UCBP FOR THIS UNIT
*          FREEMAIN 16 BYTE AREA USED FOR UCBP
*        ELSE
*          IF PATCH ID IS 1 THEN PROCESS INITIAL REQUEST-1ST PAGE
*            DO SEARCH FOR REQ. UNITS UCBP
*            IF NOT FOUND, THEN
*              GETMAIN 16 BYTE AREA FOR PAGING INFO FOR REQUESTING UNIT
*              BUILD UNIT CONTROL BLOCK FOR PAGING(UCBP) AND CHAIN IT
*                ONTO THE RESOURCE TABLE FOR THE REQUESTING UNIT
*            ELSE
*              USE EXISTING UCBP
*              STORE PAGE NUMBER 1 AS CURRENT PAGE
*            ENDIF
*          ELSE
*            IF PATCH ID IS 3 OR 4 THEN PROCESS PAGING REQUEST
*              DO SEARCH FOR REQUESTING UNITS UCBP
*              IF PATCH ID IS 3 THEN PAGE FORWARD
*                INCREMENT CURRENT PAGE NO.(WRAP AROUND IF ON LAST PG)
*              ELSE PAGE BACKWARD(PATCH ID 4)
*                DECREMENT CURRENT PAGE NO.(WRAP AROUND IF ON 1ST PAGE)
*              ENDIF
*            ELSE PROCESS DATA ENTRY REQUEST(PATCH ID 7)
*              DO SEARCH FOR REQUESTING UNITS UCBP
*              ZERO OR BLANK OUT ANY WORK AREAS
*              LOAD ANY ADDRESSES USED IN CHANGE LIST LOOP
*              DO LOOP THRU DATA ENTRY CHANGE LIST
*                IF ENTERED VALUE PASSED LIMIT CHECKING THEN
*                  MOVE VALUE  INTO WORK AREA
*                  DO ANY NECESSARY CONVERSIONS
*                  IF ENTRY IS A PAGE REQUEST(A PARTICULAR GEN.S DATA)
*                    SAVE PAGE REQUESTED OR IF INVALID OUTPUT ERROR MSG
*                  ENDIF
*                ENDIF
*              ENDDO
```

```
MEMBER NAME  DOMAPDLL
*               IF DATE ENTERED IS TODAY OR TOMORROW THEN
*                 ACCEPT DATE
*               ENDIF
*               IF TIME ENTERED IS LESS THAN 24 HOURS THEN
*                 ACCEPT TIME
*               ENDIF
*               STORE ACCEPTED INPUT VALUES IN DATA BASE
*             ENDIF
*           ENDIF
*           ISSUE DAUPDAT MACRO TO PROVIDE DISPLAY PROCESSING THE ADDR'S
*             NECESSARY TO OUTPUT THE REQUESTED PAGE OF DATA
*           CONVERT GEN. STATUS FOR OUTPUT
*           ISSUE DINFO FOR GEN. STATUS
*           ISSUE DINFO FOR PAGE NUMBER REQUESTED
*           CONVERT DATES AND TIMES FOR OUTPUT
*           ISSUE DINFO FOR DATES AND TIMES
*           ISSUE DINFO FOR INCLUSION INDICATORS
*           ISSUE DISPUP MACRO'S TO OUTPUT DYNAMIC DATA TO REQ.ING  UNIT
*         ENDIF
*   EXIT DOMACEPG
**
**
*   END OF LFC DISPLAY SUPPORT PROGRAMS  **********
**
```

```
MEMBER NAME  DOMCAGCK
 *        DOMCAGCK MAIN SEGMENT
 *          IF PATCH ID IS 1 THEN
 *            SET FLAG TO DEACTIVATE AGC OUTPUT
 *            ISSUE A MESSAGE TO SYSTEM MESSAGE ZONE OF INPUT UNIT ID
 *            ISSUE EVENT INDICATING AGC OUTPUT DEACTIVATED
 *          ELSE
 *            IF PATCH ID IS 2 THEN
 *              SET FLAG TO ACTIVATE AGC OUTPUT
 *              ISSUE A MESSAGE TO SYSTEM MESSAGE ZONE OF INPUT UNIT ID
 *              ISSUE EVENT INDICATING AGC OUTPUT ACTIVATED
 *            ENDIF
 *          ENDIF
 *        ENDSEGMENT DOMCAGCK
```

```
MEMBER NAME  DOMCALD1
 * DOMCALD1 MAIN SEGMENT
 *    SET END OF LIST INDICATOR IN STAE PARAMETER LIST
 *    SAVE ADDRESS OF ACCESS AREA TABLE FROM EMSCVT
 *    SAVE PATCH ID
 *    IF CEALARM FIELD IN CEATAB IS ZERO THEN
 *      GET AN AREA FOR ACE USING GETWA MACRO
 *      ERROR EXIT IF RETURN CODE IS NOT ZERO TO ERR1
 *      STORE ADDRESS IN CEATAB
 *      ZERO ACE
 *    ENDIF
 *    STORE ACE INFORMATION IN STAE PARAMETER LIST
 *    GET ADDRESS OF LOCK, SYSGEN OPTIONS ARRAY, ALARM CONDITION TABLE,
 *     AND TYPES TABLE FROM EMSCVT
 *    IF PATCH ID IS 8 THEN
 *      SET CASE ID TO 1
 *    ENDIF
 *    CASE ENTRY - PATCH ID
 *
 *      CASE 1 AND 8 - BUILD DETAIL ALARM DISPLAY FOR SELECTED ACCESS
 *                     AREA AND FUNCTION
 *        IF PATCH ID IS 1 THEN
 *          ZERO ACE
 *          BLANK ACCES AND FUNCTION AREA NAMES IN SAVE AREA
 *        ELSE
 *          IF ID IS EQUAL TO 8 THEN
 *            CLEAN UP SCREEN USING DINFO MACRO
 *            ISSUE DISPUP MACRO TO WRITE BLANKS TO SCREEN
 *            ERROR EXIT TO ERR5 IF MACRO FAILS
 *            SAVE NEW ACCESS AREA AND/OR FUNCTION NAME FROM PARM LIST
 *          ENDIF
 *        ENDIF
 *        SEARCH FOR ACCESS AREA NAME IN PARAMETERS PASSED -UNTIL-DO LOOP
 *        EXITIF FOUND
 *          OBTAIN ACCESS AREA ID USING DOBTAIN MACRO
 *          ERREXIT TO ERR3 IF ACCESS AREA ID NOT FOUND
 *          STORE ID IN ACE (ALARM CONTROL ELEMENT)
 *        ORELSE
 *          INCREMENT POINTER TO PARAMETER LIST
 *        ENDLOOP
 *          USE DEFAULT ACCESS AREA ID FROM DISPLAY CONTROL ELEMENT (DCE)
 *          OBTAIN ACCESS AREA NAME USING DOBTAIN MACRO
 *        END SEARCH
 *        SEARCH FOR FUNCTION NAME IN PARAMETERS PASSED - UNTIL-DO LOOP
 *        EXITIF FOUND
 *          OBTAIN FUNCTION ID USING DOBTAIN MACRO
 *          ERREXIT TO ERRA IF FUNCTION ID NOT FOUND
 *          STORE FUNCTION ID IN ACE
 *        ORELSE
 *          INCREMENT POINTER TO PARAMETER LIST
 *        ENDLOOP
 *          USE DEFAULT FUNCTION ID FROM DCE
 *          OBTAIN FUNCTION NAME USING DOBTAIN MACRO
 *        END SEARCH
 *        POINT TO ACCESS AREA ARRAY
 *        SEARCH ARRAY FOR MATCH ON ACCESS AREA ID - UNTIL-DO LOOP
 *        EXITIF ID MATCHES
 *        ORELSE
 *          INCREMENT POINTER TO TABLE
```

```
MEMBER NAME  DOMCALD1
 *        ENDLOOP
 *          ERREXIT TO ERR3 SINCE MATCH NOT FOUND
 *        END SEARCH
 *        IF PATCH ID IS EQUAL TO 1 THEN
 *          TURN BACKLIGHTS ON OR OFF USING DLITES MACRO
 *        ENDIF
 *        UPDATE ACCESS AREA NAME AND FUNCTION NAME USING DINFO MACRO
 *        ERREXIT TO ERR5 IF MACRO FAILS
 *        IF PATCH ID IS 8 THEN
 *          ZERO ALARM COUNT FIELD
 *        ENDIF
 *        SEARCH FOR FUNCTION ID IN ACCESS AREA TABLE - UNTIL-DO LOOP
 *        EXIT IF FOUND
 *          SAVE ADDRESS OF FUNCTION INFORMATION
 *        ORELSE
 *          INCREMENT POINTER TO FUNCTION INFORMATION IN ACCESS AREA TAB.
 *        ENDLOOP
 *          ERREXIT TO ERR9 SINCE FUNCTION ID NOT FOUND
 *        END SEARCH
 *        IF ANY ALARMS EXIST FOR SELETED ACCESS AREA AND FUNCTION THEN
 *          IF MORE THAN ONE PAGE OF ALARMS THEN
 *            SET NUMBER TO RETRIEVE TO PAGE COUNT NUMBER
 *          ENDIF
 *          CALL DOMCALD3 - MODULE TO RETRIEVE ALARMS
 *          DETERMINE PAGE NUMBER AND TOTAL NUMBER OF PAGES
 *          GET AREA FOR DISPLAY BUFFER USING GETMAIN MACRO
 *          SAVE ADDRESS OF BUFFER
 *          STORE BUFFER INFORMATION IN STAE PARAMETER LIST
 *          CALL DOMCALD2 - MODULE TO UPDATE DISPLAY
 *          RESET STAE LIST TO REMOVE BUFFER INFORMATION
 *          ERREXIT TO ERR5 IF DISPLAY WAS NOT UPDATED
 *          DO DOMZONE - TO CLEAN UP SCRATCH PAD ZONE
 *        ELSE
 *          DO DOMMESG - TO OBTAIN MESSAGE INDICATING NO ALARMS EXIST
 *          DISPLAY PAGE NUMBER AND TOTAL NUMBER OF PAGES USING DINFO
 *          DO DOMZONE - TO DISPLAY MESSAGE OBTAINED
 *        ENDIF
 *
 *     CASE 2 AND 3 - ACKNOWLEDGEMENT OR DELETION OF ALARMS
 *        IF ACCESS AREA BEING VIEWED DOES NOT MATCH ACCESS AREA IN DCE
 *          ERREXIT TO ERR6
 *        ENDIF
 *        IF FUNCTION BEING VIEWED DOES NOT MATCH FUNCTION IN DCE THEN
 *          ERREXIT TO ERR6
 *        ENDIF
 *        DETERMINE WHICH ALARM BEING VIEWED IS TO BE ACTED UPON
 *        POINT TO FUNCTION INFORMATION
 *        CALL DOMCALD4 - MODULE TO ACKNOWLEDGE OR DELETE ALARMS
 *        IF RETURN CODE IS NOT ZERO THEN
 *          SET CONTROL FOR MESSAGE NUMBER TO BE RETRIEVED
 *        ENDIF
 **** NOTE: THIS CASE DROPS THROUGH TO THE NEXT CASE TO REFRESH SCREEN**
 *
 *     CASE 4,5 AND 6 - PAGE BACKWARD, FORWARD OR REFRESH SCREEN
 *        CALCULATE TOTAL NUMBER OF PAGES
 *        IF THERE ARE ANY ALARMS TO DISPLAY THEN
 *          DETERMINE WHICH PAGE TO DISPLAY BASED ON CASE ID DOING
 *           WRAPAROUND WHEN NECESSARY
```

```
MEMBER NAME  DOMCALD1
 *          DETERMINE NUMBER OF ALARMS TO BYPASS IF ANY AND NUMBER OF
 *           ALARMS TO DISPLAY
 *          POINT TO FUNCTION INFORMATION
 *          CALL DOMCALD3
 *        ELSE
 *          IF MESSAGE NUMBER CONTROL NOT SET THEN
 *            SET MESSAGE NUMBER CONTROL
 *          ENDIF
 *        ENDIF
 *        DO DOMMESG
 *        DO DOMZONE
 *        GET BUFFER AREA FOR DISPLAY BUFFER USING GETMAIN MACRO
 *        SAVE BUFFER ADDRESS
 *        STORE BUFFER INFORMATION IN STAE PARAMETER LIST
 *        CALL DOMCALD2
 *        RESET BUFFER FIELD IN STAE LIST
 *        ERREXIT TO ERR5 IF RETURN CODE IS NOT ZERO
 *        IF MESSAGE NUMBER CONTROL IS NOT SET THEN
 *          DO DOMZONE - TO CLEAN UP SCRATCH PAD ZONE
 *        ENDIF
 *
 *      CASE 7 - DISPLAY CHANGE
 *        FREE SAVED AREAS USING FREEWA AND FREEMAIN MACROS
 *        MOVE END OF LIST INDICATOR TO STAE PARAMETER LIST
 *
 *    ENDCASE
 *    EXIT WITH RETURN CODE SET TO ZERO
 *
 *    ERRENTER ERR1  - NO GETWA CORE AVAILABLE
 *      SET UP FOR MESSAGE # 333
 *      DO DOMMESG
 *      DO DOMZONE
 *    ERRENTER ERR3  - ACCESS AREA INVALID
 *      SET UP FOR MESSAGE # 379
 *      DO DOMMESG
 *      DO DOMZONE
 *    ERRENTER ERR5  - BAD RETURN CODE FROM DINFO
 *      SET UP FOR MESSAGE # 381
 *      DO DOMMESG
 *      DO DOMZONE
 *    ERRENTER ERR6  - ACCESS AREA OR FUNCTION DO NOT MATCH
 *      SET UP FOR MESSAGE # 361
 *      DO DOMMESG
 *      DO DOMZONE
 *    ERRENTER ERR9  - FUNCTION INVALID FOR ACCESS AREA
 *      SET UP FOR MESSAGE # 389
 *      DO DOMMESG
 *      DO DOMZONE
 *    ERRENTER ERRA  - FUNCTION INVALID
 *      SET UP FOR MESSAGE # 363
 *      DO DOMMESG
 *      DO DOMZONE
 *    ERRETURN
 *    EXIT WITH RETURN CODE SET TO ZERO
 * ENDSEGMENT DOMCALD1
 *
 * DOMMESG SUBROUTINE SEGMENT
 *    GET MESSAGE USING MESSAGE MACRO
```

```
MEMBER NAME  DOMCALD1
 * ENDSEGMENT DOMMESG
 *
 * DOMZONE SUBROUTINE SEGMENT
 *    WRITE MESSAGE TO SCRATCH PAD ZONE USING DWZONE MACRO
 * ENDSEGMENT DOMZONE
```

```
MEMBER NAME  DOMCALD2
 * DOMCALD2 MAIN SEGMENT
 *   CLEAN UP DISPLAY BUFFER AREA BY BLANKING IT
 *   IF THERE ARE ANY ALARMS THEN
 *     SAVE FIRST SEQUENCE NUMBER
 *     UNTIL ALL ALARMS BUILT DO
 *       BUILD ALARM LINE IN BUFFER USING TYPE AND CONDITION TABLES
 *       INCREMENT POINTER TO NEXT ALARM RECORD
 *       INCREMENT POINTER TO NEXT POSITION IN BUFFER
 *     ENDDO
 *   ENDIF
 *   UPDATE PAGE NO. AND TOTAL NUMBER OF PAGES USING DINFO MACRO
 *   ERREXIT TO ERR1 IF RETURN CODE IS NOT ZERO
 *   IF THERE ARE ANY ALARMS TO DISPLAY THEN
 *     UNTIL ALL ALARMS DISPLAYED DO
 *       UPDATE DISPLAY WITH POKE POINT USING DINFO MACRO
 *       ERREXIT TO ERR1 IF RETURN CODE IS NOT ZERO
 *       IF ALARM IS ACKNOWLEDGED THEN
 *      . UPDATE DISPLAY USING DINFO MACRO (ATTRIBUTE OF GREEN)
 *       ELSE
 *         UPDATE DISPLAY USING DINFO MACRO (ATTRIBUTE OF RED)
 *       ENDIF
 *       ERREXIT TO ERR1 IF RETURN CODE IS NOT ZERO
 *       INCREMENT POINTERS AND DECREMENT COUNTS
 *     ENDDO
 *   ENDIF
 *   IF ENTIRE DISPLAY HAS NOT BEEN UPDATED THEN
 *     UNTIL IT HAS THEN
 *       UPDATE DISPLAY WITH BLANKS USING DINFO MACRO
 *     ENDDO
 *   ENDIF
 *   WRITE DATA TO SCREEN USING DISPUP MACRO
 *   ERROR EXIT TO ERR1 IF MACRO FAILS
 *   FREE BUFFER AREA USING FREEMAIN MACRO
 *  EXIT WITH RETURN CODE SET TO ZERO
 *
 *   ERRENTER ERR1
 *   FREE BUFFER AREA USING FREEMAIN MACRO
 *     EXIT WITH RETURN CODE OF 1
 * ENDSEGMENT DOMCALD2
```

```
MEMBER NAME  DOMCALD3
 * DOMCALD3 MAIN SEGMENT
 *    GET SAVE AREA FOR ALARMS USING GETMAIN MACRO
 *    STORE ALARM SAVE AREA INFORMATION IN STAE PARAMETER LIST
 *    LOCK DETAIL ALARM RECORD LIST USING LOCK MACRO
 *    IF THERE ARE ANY ALARMS TO BYPASS THEN
 *      UNTIL THEY ARE BYPASSED DO
 *        POINT TO NEXT ALARM IN CHAIN
 *      ENDDO
 *    ENDIF
 *    UNTIL ALL ALARMS MOVED DO
 *      MOVE ALARM TO SAVE AREA
 *      POINT TO NEXT ALARM
 *      POINT TO NEXT POSITION IN SAVE AREA
 *    ENDDO
 *    UNLOCK DETAIL ALARM RECORD USING LOCK MACRO
 *    EXIT WITH RETURN CODE SET TO ZERO
 * ENDSEGMENT DOMCALD3
```

```
MEMBER NAME  DOMCALD4
 * DOMCALD4 MAIN SEGMENT
 *    ERROR EXIT TO ERR2 IF THERE ARE NO ALARMS
 *    LOCK DETAIL ALARM RECORD LIST USING LOCK MACRO
 *    SEARCH FOR ALARM RECORD TO BE ACTED UPON
 *    EXITIF FOUND
 *      IF ALARM RECORD HAS NOT CHANGED THEN
 *        IF THE PATCH ID IS EQUAL TO 2 THEN
 *          IF THE ACKNOWLEDGED INDICATOR IS ALREADY ON THEN
 *            BRANCH TO NOPROCES
 *          ENDIF
 *          SET ON ACKNOWLEDGED INDICATOR IN ALARM RECORD
 *          SET ON ACKNOWLEDGED INDICATOR IN DATA BASE ITEM IF ANALOG OR
 *           STATUS POINT
 *          IF WALLBOARD INDICATOR ON THEN
 *            DO WALLBD
 *          ENDIF
 *          DO LOG IF STATUS POINT
 *        ELSE
 *          RECHAIN ALARM RECORDS
 *          ADJUST COUNT IN RCB
 *          IF ALL ALARMS DELETED FOR THE TERMINAL THEN
 *            DELETE GENERAL ALARM USING DALARM MACRO
 *          ENDIF
 *          ADJUST COUNT IN ACCESS AREA TABLE
 *          IF ALARM IS STATUS ALARM THEN
 *            IF WALLBOARD INDICATOR ON THEN
 *              DO WALLBD
 *            ENDIF
 *            DO LOG
 *            TURN OFF ALARM INDICATORS IN STATUS ITEM
 *            ISSUE ENTITY BASED ON TYPE OF DEVICE AND STATUS SETTING
 *             USING DISPENT MACRO
 *          ELSE
 *            IF ALARM IS PULSE COUNTER ALARM THEN
 *              TURN OFF ALARM INDICATOR IN PC ITEM
 *              ISSUE ENTITY CHANGE USING DISPENT MACRO
 *            ELSE
 *              IF ALARM IS ANALOG ALARM THEN
 *                TURN OFF ALARM INDICATORS IN ANALOG ITEM
 *                IF WALLBOARD INDICATOR IS ON THEN
 *                  DO WALLBD
 *                ENDIF
 *                ISSUE ENTITY CHANGE USING DISPENT MACRO
 *              ENDIF
 *            ENDIF
 *          ENDIF
 *          IF ALARM IS MESSAGE TYPE THEN
 *            TURN OFF ALARM INDICATOR IN MESSAGE ITEM
 *          ENDIF
 *          BUILD EVENT MESSAGE AND ISSUE IT USING SCEVENT MACRO
 *          BUILD TYPER MESSAGE
 *          FREE ALARM RECORD AREA USING FREEWA MACRO
 *        ENDIF
 *      ELSE
 *      UNLOCK ALARM RECORD CHAIN USING LOCK MACRO
 *      ERREXIT TO ERR1
 *      ENDIF
 *    ORELSE
```

```
MEMBER NAME  DOMCALD4
 *      POINT TO NEXT ALARM IN CHAIN
 *    ENDLOOP
 *      UNLOCK ALARM RECORD CHAIN USING LOCK MACRO
 *      ERREXIT TO ERR2  - ALARM NOT FOUND
 *    END SEARCH
 *    UNLOCK ALARM RECORD CHAIN USING LOCK MACRO
 *    PUT TYPER MESSAGE TO ALARM TYPER(S) USING MESSAGE MACRO
 * NOPROCES -LABEL
 *    EXIT WITH RETURN CODE SET TO ZERO
 *
 *    ERRENTER ERR1
 *      SET RETURN CODE TO 1 -ALARM CHANGED CONDITION
 *    ERRENTER ERR2
 *      SET RETURN CODE TO 2 -ALARM NOT FOUND
 *    ERRETURN
 *    EXIT WITH RETURN CODE SET AS ABOVE
 * ENDSEGMENT DOMCALD4
 *
 *
 * WALLBD SUBROUTINE SEGMENT
 *    GET AREA FOR PARAMETER LIST USING GETWA MACRO
 *    IF RETURN CODE IS ZERO THEN
 *      SET UP PARAMETER LIST
 *      PATCH WALLBOARD PROCESSOR (DOMCWBPR)
 *      IF PATCH FAILS THEN
 *        FREE AREA USING FREEWA MACRO
 *      ENDIF
 *    ENDIF
 * ENDSEGMENT WALLBD
 *
 * LOG SUBROUTINE SEGMENT
 *    GET AREA FOR PARAMETER LIST USING GETWA MACRO
 *    IF RETURN CODE IS ZERO THEN
 *      SET UP PARAMETER LIST
 *      GET TIME USING PTIME MACRO
 *      PATCH CHANGE OF STATUS LOG PROCESSOR (DOMCSLOG)
 *    ENDIF
 * ENDSEGMENT LOG
```

```
MEMBER NAME  DOMCALM2
 * DOMCALM2 MAIN SEGMENT
 *    ZERO STAE PARAMETER LIST
 *    ISSUE STAE MACRO
 *    GET AREA TO BUILD PARAMETER LIST FOR ALARMS PROCESSOR USING GETWA
 *    IF RETURN CODE IS NOT ZERO THEN
 *      SET MACRO RETURN CODE TO 12
 *    ELSE
 *    STORE PARAMETER LIST INFORMATION IN STAE LIST
 *      IF CONFLICTING PARAMETERS THEN
 *        SET MACRO RETURN CODE TO 16 AND FREEWA PARAMETER LIST AREA
 *      ELSE
 *        IF MISSING PARAMETERS THEN
 *          SET MACRO RETURN CODE TO 20 AND FREEWA PARAMETER LIST AREA
 *        ELSE
 *          SET ON MACRO INDICATOR IN PARAMETER LIST
 *          IF TEXT BIT IS ON THEN
 *            SET ON TEXT INDICATOR IN PARAMETER LIST
 *          ELSE
 *             MOVE S/7 AND EMT IDS TO PARAMETER LIST
 *          ENDIF
 *          IF DELETE BIT IS ON THEN
 *            SET ON DELETE INDICATOR IN PARAMETER LIST
 *          ENDIF
 *          IF CONDITION CODE BIT IS ON THEN
 *            MOVE CONDITION CODE TO PARAMETER LIST
 *          ENDIF
 *          IF TEXT BIT IS OFF THEN
 *            IF TYPE BIT IS OFF THEN
 *              IF TYPE FIELD DOES NOT CONTAIN STATUS, ANALOG OR
 *               COUNTER THEN
 *                FREE PARAMETER LIST AREA USING FREEWA MACRO
 *                DELETE STAE MACRO
 *                SET MACRO RETURN CODE TO 4
 *                EXIT
 *              ELSE
 *                SET ON TYPE INDICATORS IN PARAMETER LIST
 *              ENDIF
 *            ELSE
 *              SET ON TYPE INDICATORS ACCORDING TO MACRO LIST SETTINGS
 *            ENDIF
 *          ENDIF
 *          IF POINT BIT IS ON THEN
 *            MOVE POINT NAME TO PARAMETER LIST
 *          ELSE
 *            IF TEXT BIT IS ON THEN
 *              MOVE TEXT TO PARAMETER LIST
 *            ENDIF
 *          ENDIF
 *          IF APPOINT BIT IS ON THEN
 *            MOVE APPOINT NAME TO PARAMETER LIST
 *          ENDIF
 *          IF ACB BIT IS ON THEN
 *            MOVE ACB NAME TO PARAMETER LIST
 *          ENDIF
 *          PTIME FOR CURRENT TIME AND DATE
 *          MOVE TIME AND DATE TO PARAMETER LIST
 *          PATCH ALARM PROCESSOR (DOMCALR1)
 *          IF RETURN CODE IS NOT ZERO THEN
```

```
MEMBER NAME  DOMCALM2
 *            SET MACRO RETURN CODE TO 8 AND FREEWA PARAMETER LIST
 *          ELSE
 *            WAIT ON ECB
 *              IF DOMCALR1 DID NOT PROCESS THEN
 *                SET RETURN CODE TO 8
 *              ELSE
 *                SET MACRO RETURN CODE TO 0
 *              ENDIF
 *          ENDIF
 *        ENDIF
 *      ENDIF
 *    ENDIF
 *    DELETE STAE MACRO
 *    EXIT WITH RETURN CODE SET AS ABOVE
 * ENDSEGMENT DOMCALM2
```

```
MEMBER NAME  DOMCALR1
 * DOMCALR1 MAIN SEGMENT
 *    MOVE END OF LIST INDICATOR TO STAE LIST
 *    SET OFF FLAGS
 *    POINT TO PARAMETER LIST
 *    STORE PARAMETER LIST INFORMATION IN STAE LIST
 *    CONVERT DATE AND TIME TO EBCDIC FORMAT
 *    CALCULATE TOTAL NUMBER OF ALARMS TO PROCESS
 *    GET AREA FOR ENTITY LIST BUFFER BASED ON NUMBER OF ALARMS USING
 *     GETWA MACRO
 *    ERROR EXIT TO ERR1 IF RETURN CODE IS NOT ZERO
 *    UNTIL ALL SET BUFFERS OR MACRO PARAMETER LIST PROCESSED DO
 *      UNTIL ALL ALARMS IN BUFFER PROCESSED DO
 *        SET OFF NO PROCESS FLAG
 *        IF SET HEADER THEN
 *          IF MACRO INITIATED THEN
 *            IF MESSAGE TYPE THEN
 *              POINT TO ACB NAME
 *            ELSE
 *              CALCULATE RCB NAME USING S/7 AND EMT IDS PASSED
 *            ENDIF
 *            GET ADDRESS OF RCB OR ACB USING GETITEM MACRO
 *            ERROR EXIT IF RETURN CODE IS NOT ZERO TO ERR5
 *          ELSE
 *            GET ADDRESS OF RCB FROM SET
 *          ENDIF
 *          SAVE S/7 ID AND EMT ID FROM RCB FOR GENERAL ALARM ID
 *          GET CBLOC ADDRESS, CONDITION TABLE ADDRESS, SYSGEN OPTIONS
 *           TABLE ADDRESS AND ACCESS AREA TABLE ADDRESS FROM EMSCVT
 *          STRTSRCH UNTIL ACCESS AREA ENTRY FOUND DO
 *          EXIT IF MATCH FOUND ON ACCESS AREA ID
 *          ORELSE
 *            POINT TO NEXT ACCESS AREA ENTRY
 *          ENDLOOP
 *            ERROR EXIT TO ERR3 IF NO MATCH FOUND
 *          ENDSRCH
 *          IF MESSAGE TYPE ALARM THEN
 *            DO DOMMESG
 *          ENDIF
 *        ENDIF
 *        IF NOT A MESSAGE TYPE ALARM THEN
 *          IF A MACRO GENERATED ALARM THEN
 *            POINT TO ITEM NAME
 *            GET ADDRESS OF ITEM USING GETITEM MACRO
 *            ERROR EXIT TO ERR6 IF ITEM NOT FOUND
 *          ENDIF
 *        ENDIF
 *        IF STATUS POINT ALARM THEN
 *          CALL DOMCALR3
 *          ERROR EXIT TO ERR1 IF RETURN CODE IS NOT ZERO
 *        ELSE
 *          IF ANALOG POINT ALARM THEN
 *            CALL DOMCALR4
 *            ERROR EXIT TO ERR1 IF RETURN CODE IS NOT ZERO
 *          ELSE
 *            CALL DOMCALR5  - PULSE COUNTER ALARM
 *            ERROR EXIT TO ERR1 IF RETURN CODE IS NOT ZERO
 *          ENDIF
 *        ENDIF
```

```
MEMBER NAME  DOMCALR1
*       IF NOT MESSAGE TYPE ALARM THEN
*         IF NAME IS BLANK THEN
*           SET ON NO PROCESS FLAG
*         ENDIF
*       ENDIF
*       CALL DOMCALR6
*       ERROR EXIT TO ERR1 IF RETURN CODE IS NOT ZERO
*       POINT TO NEXT ENTRY IN BUFFER
*     ENDDO
*     POINT TO NEXT BUFFER
*   ENDDO
*   UNTIL ALL INPUT BUFFERS ARE FREE, DO
*     OBTAIN POINTER TO NEXT BUFFER
*     FREEWA THE CURRENT BUFFER
*   ENDDO
*   CLEAR STAE LIST
*   IF ANY ALARMS TO BE EVENTED THEN
*     PATCH DOMCEVT5
*     IF PATCH FAILS THEN
*       FREE EVENTS BUFFER CHAIN
*     ENDIF
*   ENDIF
*   IF ANY ENTITIES IN BUFFER LIST THEN
*     ISSUE DISPENT MACRO
*   ENDIF
*   DO WALLBD
*   DO LOG
*   MOVE END OF LIST INDICATOR TO STAE LIST
*   EXIT WITH RETURN CODE SET TO ZERO
*
*   ERROR ENTER ERR1
*     SET UP FOR MESSAGE # 459  - NO GETWA CORE
*   ERROR ENTER ERR3
*     SET UP FOR MESSAGE # 488  - NO MATCH ON ACCESS AREA CODE
*   ERROR ENTER ERR4
*     SET UP FOR MESSAGE # 461  - UNABLE TO GET APPOINT ITEM
*   ERROR ENTER ERR5
*     SET UP FOR MESSAGE # 462  - UNABLE TO GET RCB OR ACB
*   ERROR ENTER ERR6
*     SET UP FOR MESSAGE # 463  - UNABLE TO GET ITEM PASSED BY MACRO
*   ERROR RETURN
*     ISSUE MESSAGE USING MESSAGE MACRO
*     IF ANY ALARMS TO BE EVENTED THEN
*       PATCH DOMCEVT5
*       IF PATCH FAILS THEN
*         FREE EVENTS BUFFER CHAIN
*       ENDIF
*     ENDIF
*     MOVE END OF LIST INDICATOR TO STAE LIST
*     WHILE BUFFERS REMAIN DO
*       FREE BUFFER USING FREEWA MACRO
*       POINT TO NEXT BUFFER
*     ENDDO
*     IF ENTITY LIST BUFFER HAS ANY ENTRIES IN IT THEN
*       ISSUE DISPENT MACRO
*     ENDIF
*     DO WALLBD
*     DO LOG
```

```
MEMBER NAME  DOMCALR1
 *    EXIT WITH RETURN CODE SET TO FOUR
 * ENDSEGMENT DOMCALR1
 *
 * DOMMESG SUBROUTINE SEGMENT
 *    BUILD MESSAGE TYPE ALARM RECORD
 *    GET APPOINT ADDRESS USING GETITEM MACRO
 *    ERROR EXIT TO ERR4 IF RETURN CODE IS NOT ZERO
 *    IF DELETE INDICATOR IS ON IN MACRO LIST THEN
 *      SET ON DELETE FLAG
 *    SET ON UPDATE FLAG
 *      SET OFF ALARM INDICATOR IN APPOINT ITEM
 *    ELSE
 *      SET ON ADD ALARM FLAG
 *      SET ON ALARM INDICATOR IN APPOINT ITEM
 *    ENDIF
 * ENDSEGMENT DOMMESG
 *
 * WALLBD SUBROUTINE SEGMENT
 *    IF ADDRESS OF WALLBOARD BUFFER IS NOT ZERO THEN
 *      PATCH WALLBOARD PROCESSOR (DOMCWBPR)
 *      IF PATCH FAILS THEN
 *        WHILE BUFFERS REMAIN DO
 *          FREE BUFFER USING FREEWA MACRO
 *          POINT TO NEXT BUFFER IN CHAIN
 *        ENDDO
 *      ENDIF
 *    ENDIF
 * ENDSEGMENT WALLBD
 *
 * LOG SUBROUTINE SEGMENT
 *    IF LOG BUFFER ADDRESS IS NOT ZERO THEN
 *      PATCH CHANGE OF STATUS LOG PROCESSOR (DOMCSLOG)
 *    ENDIF
 * ENDSEGMENT LOG
```

```
MEMBER NAME  DOMCALR3
 * DOMCALR3 MAIN SEGMENT
 *    MOVE INFORMATION TO RECORD AREA FROM ITEM AND RCB
 *    IF DEVICE TYPE IS TCT TYPE 3 OR
 *    IF DEVICE TYPE IS MOTOR OPERATED SWITCH THEN
 *      IF GROUP ADDRESS IS ZERO THEN
 *        START SEARCH ON RCB STATUS DATA FOR GROUP
 *        EXIT IF GROUP FOUND
 *        OR ELSE
 *          POINT TO NEXT GROUP
 *        END LOOP
 *          ERROR EIXT TO ONE
 *        END SEARCH
 *      ENDIF
 *      DETERMINE IF ITEM IS EVEN OR ODD ONE OF PAIR
 *      STORE ADDRESS OF EVEN ITEM IN RECORD AREA
 *    ENDIF
 *    DETERMINE WHETHER ENTITY IS TO BE ADDED OR DELETED AND WHICH
 *     ATTRIBUTE CODE TO USE
 *    ESTABLISH CONDITION CODE BASED ON TYPE OF DEVICE AND STATUS
 *     SETTING OR CODE PASSED
 *    MOVE REMAINING INFORMATION TO RECORD AREA
 *    IF PREVIOUS ALARM EXISTS THEN
 *      SET ON UPDATE FLAG
 *      SET OFF ACKNOWLEDGED FLAG IN ITEM
 *    ENDIF
 *    IF DELETE ALRM INDICATOR ON THEN
 *      SET ON DELETE FLAG
 *      SET OFF ALARM AND ACKNOWLEDGED FLAGS IN ITEM
 *    ELSE
 *      IF POINT IS ALARMABLE AND
 *      IF POINT IS IN SERVICE THEN
 *        SET ON ADD ALARM FLAG
 *        SET ON ALARM OUTSTANDING FLAG IN ITEM
 *      ENDIF
 *    ENDIF
 *    SET UP ENTITY NAME
 *    ADD ENTITY NAME AND INDICATOR TO ENTITY LIST BUFFER
 *    IF WALLBOARD FLAG IS ON THEN
 *      IF WALLBOARD ADDRESS IS ZERO THEN
 *        GET BUFFER AREA USING GETWA MACRO
 *        ERROR EXIT TO TWO IF RETURN CODE IS NOT ZERO
 *        SAVE ADDRESS OF BUFFER
 *      ELSE
 *        POINT TO CURRENT BUFFER
 *        IF BUFFER IS FULL THEN
 *          GET AREA FOR NEXT BUFFER USING GETWA MACRO
 *          ERROR EXIT TO TWO IF RETURN CODE IS NOT ZERO
 *          CHAIN NEW BUFFER AND SAVE ADDRESS
 *        ENDIF
 *      ENDIF
 *      ADD INFORMATION TO WALLBOARD BUFFER
 *      ADJUST COUNTS AND POINTERS
 *    ENDIF
 *    IF STATUS POINT IS WIRED THEN
 *      IF POINT IS NOT ALARMABLE OR
 *      IF POINT IS OUT OF SERVICE THEN
 *        SET ON EVENT ONLY FLAG
 *      ENDIF
```

```
MEMBER NAME  DOMCALR3
 *       IF LOG BUFFER ADDRESS IS ZERO THEN
 *         GET AREA FOR LOG BUFFER USING GETWA MACRO
 *         ERROR EXIT TO TWO IF RETURN CODE IS NOT ZERO
 *         SAVE ADDRESS OF BUFFER
 *       ENDIF
 *       MOVE CHANGE INFORMATION TO CHANGE OF STATUS LOG BUFFER
 *       IF LOG BUFFER IS FULL THEN
 *         PATCH CHANGE OF STATUS LOG PROCESSOR (DOMCSLOG)
 *         ZERO LOG BUFFER ADDRESS
 *       ENDIF
 *     ENDIF
 *     EXIT WITH RETURN CODE SET TO ZERO
 *
 *     ERROR ENTER ONE  - ITEM IS NOT PART OF STATUS FOR RCB
 *       SET NO PROCESS FLAG ON
 *       ZERO RETURN CODE
 *
 *     ERROR ENTER TWO - NO GETWA
 *       SET RETURN CODE TO FOUR
 *
 *     ERROR RETURN
 *     EXIT WITH RETURN CODE SET AS ABOVE
 *
 *   ENDSEGMENT DOMCALR3
```

```
MEMBER NAME  DOMCALR4
 * DOMCALR4 MAIN SEGMENT
 *   BUILD ANALOG ALARM RECORD
 *   IF ALARM ALREADY EXISTS FOR POINT THEN
 *     SET ON UPDATE FLAG
 *     SET OFF ACKNOWLEDGED FLAG IN ITEM
 *   ENDIF
 *   IF DELETE INDICATOR IS ON THEN    (IN SET)
 *     CLEAR ERROR CONDITION FLAGS IN ANALOG ITEM IF IN SERVICE
 *     TURN ON DELETE FLAG
 *     TURN OFF ALARM INDICATOR IN ANALOG ITEM
 *     TURN OFF ACKNOWLEDGED FLAG IN ITEM
 *     ADD ENTITY TO ENTITY LIST BUFFER WITH DELETE CODE
 *   ELSE
 *     SET UP CONDITION CODE
 *     IF POINT IS OUT OF SERVICE OR
 *     IF POINT IS NOT ALARMABLE THEN
 *       SET ON EVENT ONLY FLAG
 *       SET ON ADD FLAG
 *     ELSE
 *       SET ON ADD ALARM FLAG
 *       SET ON ALARM INDICATOR IN ANALOG ITEM
 *       ADD ENTITY TO ENTITY LIST BUFFER WITH APPROPRIATE ATTRIBUTE CODE
 *     ENDIF
 *   ENDIF
 *   IF WALLBOARD FLAG IS ON THEN
 *     IF WALLBOARD BUFFER ADDRESS IS ZERO THEN
 *       GET AREA FOR BUFFER USING GETWA MACRO
 *       ERROR EXIT TO TWO IF RETURN CODE IS NOT ZERO
 *       SAVE ADDRESS OF AREA
 *     ELSE
 *       POINT TO CURRENT BUFFER
 *       IF BUFFER IS FULL THEN
 *         GET AREA FOR NEXT BUFFER USING GETWA MACRO
 *         ERROR EXIT TO TWO IF RETURN CODE IS NOT ZERO
 *         CHAIN NEW BUFFER AND SAVE ADDRESS
 *       ENDIF
 *     ENDIF
 *     ADD INFORMATION TO WALLBOARD BUFFER
 *     ADJUST COUNTS AND POINTERS TO BUFFER
 *   ENDIF
 *   EXIT WITH RETURN CODE SET TO ZERO
 *
 *   ERROR ENTER TWO - NO GETWA AVAILABLE
 *     SET RETURN CODE TO FOUR
 *   ERROR RETURN
 *   EXIT WITH RETURN CODE SET AS ABOVE
 * ENDSEGMENT DOMCALR4
```

```
MEMBER NAME  DOMCALR5
 * DOMCALR5 MAIN SEGMENT
 *    BUILD ALARM RECORD FOR PULSE COUNTER DATA
 *    IF ALARM INDICATOR IS ON IN COUNTER ITEM THEN
 *      SET ON UPDATE FLAG
 *    ENDIF
 *    IF DELETE FLAG IS ON THEN   (IN SET)
 *      ADJUST ERROR FLAGS IF NOT OUT OF SERVICE
 *      TURN OFF ALARM INDICATOR IN COUNTER ITEM
 *      ADD ENTITY TO ENTITY LIST BUFFER WITH DELETE CODE
 *      TURN ON DELETE FLAG
 *    ELSE
 *      SET UP ALARM CONDITION CODE
 *      IF POINT IS OUT OF SERVICE OR
 *      IF POINT IS NOT ALARMABLE THEN
 *        SET ON ADD FLAG
 *        SET ON EVENT ONLY FLAG
 *      ELSE
 *        SET ON ADD ALARM FLAG
 *        SET ON ALARM INDICATOR IN COUNTER ITEM
 *        ADD ENTITY TO ENTITY LIST BUFFER WITH APPROPRIATE ATTRIBUTE
 *      ENDIF
 *    ENDIF
 *    EXIT WITH RETURN CODE SET TO ZERO
 * ENDSEGMENT DOMCALR5
```

```
MEMBER NAME  DOMCALR6
* DOMCALR6 MAIN SEGMENT
*   ESTABLISH ADDRESSABILITY
*   IF EVENT ONLY FLAG IS ON THEN
*     IF DO NOT PROCESS FLAG IS OFF THEN
*       DO DOMCMSG
*     ENDIF
*   ELSE
*     IF DO NOT PROCESS FLAG IS OFF THEN
*       IF POINT IS IN SERVICE THEN
*         IF NOT A DELETE REQUEST THEN
*           BUILD AUDIBLE ALARM MESSAGE
*           SEND MESSAGE TO SYSTEM/7 USING S7WRITE MACRO
*           IF NO ALARMS OUTSTANDING OR
*           IF GENERAL ALARM IS ACKNOWLEDGED THEN
*             ISSUE GENERAL ALARM USING DALARM MACRO
*           ENDIF
*         ENDIF
*         IF DELETE REQUEST AND
*         IF NO PREVIOUS ALARM EXISTS THEN
*           SET ON DO NOT PROCESS FLAG
*         ENDIF
*         SEARCH CAAATBL FOR FUNCTION WITHIN ACCESS
*         EXITIF  MATCH FOUND, THEN
*           IF ALARM COUNT IS NOT ZERO THEN (IN FUNCTION INFO)
*             IF ALARM COUNT IS NOT ZERO AND
*             IF PREVIOUS ALARM RECORD EXISTS THEN
*               DO DOMCOMP
*             ENDIF
*           ENDIF
*           IF NO PROCESS FLAG IS OFF THEN
*             IF ADD ALARM FLAG IS ON THEN
*               ADD 1 TO SEQUENCE NUMBER
*               STORE SEQUENCE NUMBER IN ALARM RECORD
*               GET AREA FOR ALARM RECORD USING GETWA MACRO
*               ERROR EXIT TO ERR1 IF MACRO FAILS
*               MOVE RECORD TO AREA GOTTEN
*               LOCK ON ALARM RECORD USING LOCK MACRO
*               ADD ALARM TO CHAIN
*               UPDATE ALARM COUNT IN ACCESS AREA TABLE
*               UPDATE CHAIN POINTERS IN ACCESS AREA TABLE
*               UNLOCK ALARM CHAIN USING LOCK MACRO
*               ADJUST ALARM COUNT IN RCB ITEM
*             ENDIF
*           ENDIF
*         ORELSE
*           POINT TO NEXT FUNCTION ITEM
*         ENDLOOP
*           FUNCTION MATCH NOT FOUND SO DEFAULT TO ZERO
*         ENDSRCH
*         IF ALARM PROCESSED THEN
*           DO DOMCMSG
*         ENDIF
*       ENDIF
*     ENDIF
*   ENDIF
*   EXIT WITH RETURN CODE OF ZERO
*
*   ERROR ENTER ERR1
```

```
MEMBER NAME  DOMCALR6
*      SET RETURN CODE TO FOUR
*      EXIT
*
* ENDSEGMENT DOMCALR6
*
* DOMCOMP SUBROUTINE SEGMENT
*    LOCK ON ALARM CHAIN USING LOCK MACRO
*    IF NOT MESSAGE TYPE ALARM THEN
*      START SEARCH UNTIL ALL ALARMS IN CHAIN COMPARED
*      EXIT IF MATCH FOUND ON POINT NAME THEN
*        IF DELETE THEN
*          DO DOMCDEL
*        ELSE
*          IF CONDITION IS THE SAME THEN
*            IF ANALOG POINT THEN
*              SET ON DO NOT PROCESS FLAG
*            ELSE
*              UPDATE PART OF ALARM RECORD
*            ENDIF
*          ELSE
*            UPDATE ALL OF ALARM RECORD
*          ENDIF
*          SET OFF ADD ALARM FLAG
*          SET ON ALARM UPDATED FLAG
*        ENDIF
*      ORELSE
*        POINT TO NEXT ALARM IN CHAIN
*      END LOOP - NO MATCH FOUND
*        IF DELETE THEN
*          SET ON DO NOT PROCESS FLAG
*          SET OFF DELETE FLAG
*        ENDIF
*      END SEARCH
*    ELSE - MESSAGE TYPE ALARM
*      START SEARCH UNTIL ALL RECORDS COMPARED
*      EXIT IF MATCH ON ACB NAME AND ALARM TEXT FOUND THEN
*        IF DELETE THEN
*          DO DOMCDEL
*        ENDIF
*      ORELSE
*        POINT TO NEXT ALARM IN CHAIN
*      END LOOP
*        SET OFF DELETE INDICATOR
*        SET ON DO NOT PROCESS FLAG
*      END SEARCH
*    ENDIF
*    UNLOCK ALARM CHAIN USING LOCK MACRO
* END SEGMENT DOMCOMP
*
* DOMCDEL SUBROUTINE SEGMENT
*    DELETE ALARM RECORD FROM CHAIN
*    ADJUST CHAIN POINTERS IN ACCESS AREA ITEM
*    ADJUST ALARM COUNT IN RCB ITEM
*    IF ALARM COUNT IN RCB ITEM IS ZERO THEN
*      DELETE GENERAL ALARM USING DALARM MACRO
*      SET ON ALL ALARMS DELETED FLAG
*    ENDIF
*    ADJUST ALARM COUNT IN ACCESS AREA ITEM
```

```
MEMBER NAME  DOMCALR6
*    SET OFF ADD ALARM FLAG
*    FREE ALARM RECORD AREA USING FREEWA MACRO
* ENDSEGMENT DOMCDEL
*
* DOMCMSG SUBROUTINE SEGMENT
*    DETERMINE TYPE OF ACTION PERFORMED
*    IF EVENTS BUFFER NOT YET GOTTEN THEN
*      GET AREA USING GETWA MACRO
*      ERROR EXIT TO ERR1 IF MACRO FAILS
*      ADD STAE LIST ENTRY FOR BUFFER
*      SET BUFFER COUNT FIELD TO ONE
*      SAVE ADDRESS OF START OF FIRST BUFFER
*      SAVE ADDRESS OF NEXT (FIRST) ENTRY IN BUFFER
*    ELSE
*      IF  BUFFER NOT YET FULL, THEN
*        INCREMENT BUFFER COUNT FIELD BY ONE
*      ELSE
*        GET A NEW BUFFER (GETWA)
*        IF BUFFER NOT OBTAINED, THEN
*          PATCH DOMCEVT5 PASSING ADDRESS OF FIRST BUFFER
*          IF PATCH FAILS, THEN
*            FREE BUFFER CHAIN
*          ENDIF
*          ZERO BUFFER ADDRESS SAVE AREA
*          DELETE STAE LIST ENTRY FOR BUFFER CHAIN
*        ELSE
*          SET BUFFER COUNT FIELD TO ONE
*          SAVE ADDRESS OF FIRST ENTRY ELEMENT
*        ENDIF
*      ENDIF
*    ENDIF
*    BUILD EVENT RECORD IN BUFFER
*    IF EVENT ONLY FLAG IS ON THEN
*      IF ALARM IS ANALOG OR PULSE COUNTER THEN
*        GET MESSAGE # 304 USING MESSAGE MACRO
*      ELSE
*        GET MESSAGE # 334 USING MESSAGE MACRO
*      ENDIF
*      IF MACRO DID NOT FAIL THEN
*        MOVE MESSAGE TEXT TO EVENTS BUFFER
*      ENDIF
*    ENDIF
*    MOVE ACCESS AREA AND FUNCTION CODES TO BUFFER
*    ADJUST BUFFER POINTER
*    IF EVENT ONLY FLAG IS OFF THEN
*      BUILD ALARM TYPER MESSAGE
*      USE DOBTAIN MACRO TO GET ACCESS AREA AND FUNCTION NAMES
*      USE TYPES TABLE AND ALARM CONDITION TABLE TO CONVERT CODES TO
*       PHRASES
*      ISSUE MESSAGE TO TYPERS USING MESSAGE MACRO (MESSAGE NO. 300)
*    ENDIF
* ENDSEGMENT DOMCMSG
```

```
MEMBER NAME  DOMCAND1
 *        DOMCAND1 MAIN SEGMENT
 *          CASENTRY - PATCH ID
 *            CASE 2 - (BUILD SCAN CONTROL DISPLAY)
 *              GETMAIN FOR SCAN CONTROL ELEMENT (SCE)
 *              BUILD SCE
 *              PUT ADDRESS OF SCE IN DISPLAY CONTROL ELEMENT (DCE)
 *              GETMAIN FOR DISPLAY BUFFER AREA
 *              BUILD DISPLAY FROM SCAN ID TABLE
 *              SAVE ADDRESS OF BUFFER AND FIRST AND LAST SCAN ID'S IN SC
 *              WRITE DISPLAY
 *
 *            CASE 4 - (ALTERATION TO SCAN)
 *              UNTIL ALL INPUT FROM DISPLAY READ DO
 *                IF FIELDS CHANGED THEN
 *                  ISSUE VARYSCAN MACRO FOR CHANGES
 *                  WAIT FOR RETURN CODE
 *                  IF RETURN CODE IS NOT ZERO THEN
 *                    UPDATE DISPLAY WITH APPROPRIATE MESSAGE
 *                  ELSE
 *                    UPDATE DISPLAY WITH REQUEST ACCEPTED MESSAGE
 *                  ENDIF
 *                ENDIF
 *              ENDDO
 *
 *            CASE 6 - (DISPLAY CHANGE)
 *                FREEMAIN SCE AND DISPLAY BUFFER AREA
 *                SET SCE ADDRESS IN DCE TO ZEROES
 *
 *            CASE 8  - (PAGE FORWARD)
 *                BUILD DISPLAY IN BUFFER FROM SCAN ID TABLE STARTING AT
 *                     LAST ID IN SCE AND CHANGE SCE
 *              UPDATE DISPLAY
 *
 *            CASE 10 - (PAGE BACKWARD)
 *                BUILD DISPLAY IN BUFFER FROM SCAN ID TABLE STARTING AT
 *                     FIRST ID IN SCE READING BACKWARDS  + CHANGE SCE
 *              UPDATE DISPLAY
 *
 *            CASE 12 - (REPLY FROM SYSTEM/7)
 *              PROCESS REPLY
 *              UPDATE SCAN ARRAY IF SCM COMMAND WAS SUCCESSFUL
 *              EVENT RESULT OF SCM COMMAND
 *              LOG SCAN ARRAY IF SCM COMMAND WAS SUCCESSFUL
 *
 *            CASE 14 - (REFRESH)
 *              BUILD DISPLAY IN BUFFER FROM SCAN ARRAY STARTING AT THE
 *                 ADDRESS IN THE SCE CONTROL ELEMENT
 *              WRITE THE UPDATED DISPLAY TO THE SCREEN
 *
 *          ENDCASE
 *        ENDSEGMENT DOMCAND1
 *
```

```
MEMBER NAME  DOMCBLD1
 * DOMCBLD1 MAIN SEGMENT
 *    GET THE ADDRESS OF THE XCVT USING GETCVT MACRO
 *    SET OFF THE REFRESH FLAG
 *    TURN KEY BACKLIGHTS ON OR OFF USING DLITES MACRO
 *    BLANK BUFFER AREA
 *    FOR THE NUMBER OF LOGICAL IDS DO -(UNTIL-DO LOOP)
 *      MOVE TAB AND POKE POINT CHARACTERS TO BUFFER
 *      MOVE DATA ENTRY CHARACTERS TO BUFFER
 *    ENDDO
 *    CALCULATE THE NUMBER OF UNIT GROUPS AND SIZE OF THE BOX
 *    FILL LINE BUFFERS WITH HORIZONTAL LINE CHARACTERS AND APPROPRIATE
 *      CORNER CHARACTER FOR SIZE OF BOX
 *    FILL APPROPRIATE BUFFER FOR VERTICAL LINE
 *    FOR NUMBER OF S/7 UNITS DO -(UNTIL-DO LOOP)
 *      MOVE UNIT ID TO UNIT STATUS BUFFER
 *      ADJUST POINTERS TO UNIT ID AND TO BUFFER
 *    ENDDO
 *    GET MESSAGE # 362 - CONTAINS STATUS OF UNIT PHRASES
 *    ZERO CCELIDS FIELD IN CCE
 *    SAVE BUFFER FIELD ADDRESSES
 *    UNTIL ALL LOGICAL IDS PROCESSED DO
 *      SAVE LOGICAL ID IN CCELIDS FIELD
 *      ADJUST POINTER TO CCELIDS FIELD
 *      CONVERT LOGICAL ID TO DECIMAL
 *      POINT TO LOGICAL ID BUFFER
 *      UNPACK LOGICAL ID INTO BUFFER
 *      SAVE LOGICAL ID IN SAVE ID FIELD
 *      ADJUST POINTER TO BUFFER
 *      POINT TO PRIMARY ID BUFFER AND TO ATTRIBUTE LIST
 *      IF THE ACTIVE FLAG IS ON THEN
 *        POINT TO UNIT ID
 *        MOVE ID TO BUFFER
 *        IF THE SCANNING FLAG IS OFF THEN
 *          SET ON FLASHING BIT IN ATTRIBUTE BYTE
 *        ENDIF
 *      ENDIF
 *      ADJUST POINTERS TO BUFFER AND ATTRIBUTE LIST
 *      POINT TO BACKUP UNIT BUFFER
 *      IF BACKUP FLAG IS ON THEN
 *        POINT TO UNIT ID
 *        MOVE IT TO BUFFER
 *      ENDIF
 *      ADJUST POINTER TO BUFFER
 *      POINT TO AVAILABLE UNITS BUFFER, PFLAGS, UNIT STATUS BUFFER,
 *       AND UNIT IDS
 *      FOR THE NUMBER OF UNITS DO -(UNTIL-DO LOOP)
 *        IF THE ABLE FLAG IS ON THEN
 *          MOVE THE UNIT ID TO AVAILABLE UNITS BUFFER
 *          ADJUST THE AVAILABLE UNITS BUFFER POINTER
 *          IF THE READY FLAG IS ON THEN
 *            IF THE IPLDME FLAG IS ON THEN
 *              MOVE THE IPLED MSG AND LOGICAL ID TO UNIT STATUS BUFFER
 *            ELSE
 *              IF THE IPLNGME FLAG IS ON THEN
 *                MOVE IPLING MSG AND LOGICAL ID TO UNIT STATUS BUFFER
 *              ELSE
 *                IF THE IPLDYOU FLAG IS OFF AND
 *                IF THE IPLNGYOU FLAG IS OFF THEN
```

```
MEMBER NAME  DOMCBLD1
 *                   MOVE NOT IPLED MSG TO UNIT STATUS BUFFER
 *                 ENDIF
 *               ENDIF
 *             ENDIF
 *           ENDIF
 *         ENDIF
 *         ADJUST POINTERS TO PFLAGS, UNIT ID AND UNIT-STATUS BUFFER
 *       ENDDO
 *       ADJUST POINTER TO AVAILABLE UNITS BUFFER
 *     ENDDO
 *     POINT TO UNIT STATUS BUFFER
 *     FOR NUMBER OF UNITS IN BUFFER DO -(WHILE-DO LOOP)
 *       IF THERE IS NO MSG IN FIELD THEN
 *         MOVE NOT READY MSG TO FIELD
 *       ENDIF
 *       ADJUST POINTER TO BUFFER
 *     ENDDO
 *     POINT TO DISPLAY UNIT ID, DYNAMIC ID LIST, FIRST BUFFER, ATTRIBUTE
 *      LIST, NUMBER OF ITEMS LIST, BUFFER LENGTH LIST
 *     FOR THE NUMBER OF DYNAMIC FIELDS DO-(UNTIL-DO LOOP)
 *       SET UP DATA FOR SCREEN USING DINFO MACRO
 *       ERROR EXIT TO ERR1 IF RETURN CODE IS NOT ZERO
 *       ADJUST ALL POINTERS
 *     ENDDO
 *     WRITE DATA TO SCREEN USING DISPUP MACRO
 *     ERROR EXIT TO ERR1 IF RETURN CODE IS NOT ZERO
 *     EXIT WITH RETURN CODE SET TO ZERO
 *
 *     ERROR ENTER ERR1
 *       SET RETURN CODE TO 1
 *     EXIT WITH RETURN CODE SET TO ONE
 *
 * ENDSEGMENT DOMCBLD1
```

```
MEMBER NAME  DOMCBLD2
 * DOMCBLD2 MAIN SEGMENT
 *    GET ADDRESS OF XCVT USING GETCVT MACRO
 *    GET ADDRESS OF FIRST RCB FOR PAGE FROM RCB LIST
 *    STORE IT IN CCE
 *    BLANK ENTIRE BUFFER AREA
 *    SET CONTROLS FOR FIELDS IN BUFFER
 *    GET STATUS OF TERMINAL PHRASES USING MESSAGE MACRO (MSG. # 335)
 *    SAVE NUMBER OF RCBS TO DISPLAY
 *    UNTIL ALL RCBS PROCESSED DO
 *      MOVE POKE POINT TO BUFFER
 *      ADJUST BUFFER POINTER
 *      MOVE TERMINAL NAME TO BUFFER
 *      ADJUST BUFFER POINTER
 *      IF RCB IS NOT OUT OF SERVICE THEN
 *        IF TERMINAL IS MANUAL THEN
 *          MOVE STATUS TO BUFFER
 *        ELSE - TERMINAL IS IN SERVICE
 *          MOVE STATUS TO BUFFER
 *        ENDIF
 *      ELSE - TERMINAL IS OUT OF SERVICE
 *        IF TERMINAL IS OUT OF SERVICE SELF THEN
 *          MOVE STATUS TO BUFFER
 *        ELSE - IT IS OUT OF SERVICE OTHER
 *          MOVE STATUS TO BUFFER
 *        ENDIF
 *      ENDIF
 *      ADJUST BUFFER POINTER
 *      MOVE ID OF S/7 TO WHICH TERMINAL IS ATTACHED TO BUFFER
 *      ADJUST BUFFER POINTER
 *    ENDDO
 *    IF PCC II DISPLAY IS NOT UP THEN
 *      BRING UP DISPLAY USING DISPREQ MACRO
 *      ERROR EXIT TO ERR1 IF RETURN CODE IS NOT ZERO
 *      TURN BACKLIGHTS ON USING DLITES MACRO
 *    ELSE
 *      SET OFF SAME DISPLAY INDICATOR
 *    ENDIF
 *    IF THE S/7 IS ACTIVE THEN
 *      ADJUST ATTRIBUTE TO INDICATE WHETHER SCANNING OR NOT
 *    ELSE
 *      BLANK THE S/7 ID SAVE AREA
 *    ENDIF
 *    POINT TO PAGE NUMBER + TOTAL NUMBER OF PAGES
 *    DO DOMDINF
 *    POINT TO S/7 ID
 *    DO DOMDINF
 *    POINT TO UNIT ID
 *    DO DOMDINF
 *    POINT TO FIRST COLUMN OF POKE POINTS
 *    DO DOMDINF
 *    POINT TO FIRST COLUMN OF NAMES
 *    DO DOMDINF
 *    POINT TO FIRST COLUMN OF STATUS OF TERMINALS
 *    DO DOMDINF
 *    POINT TO FIRST COLUMN OF S/7 IDS
 *    DO DOMDINF
 *    POINT TO SECOND COLUMN OF POKE POINTS
 *    DO DOMDINF
```

```
MEMBER NAME  DOMCBLD2
 *    POINT TO SECOND COLUMN OF NAMES
 *    DO DOMDINF
 *    POINT TO SECOND COLUMN OF STATUS OF TERMINALS
 *    DO DOMDINF
 *    POINT TO SECOND COLUMN OF S/7 IDS
 *    DO DOMDINF
 *    WRITE DYNAMIC AREAS TO SCREEN USING DISPUP MACRO
 *    ERROR EXIT IF RETURN CODE IS NOT ZERO TO ERR2
 *    EXIT RETURN CODE=0
 *
 *    ERROR ENTER ERR1 - DISPREQ MACRO FAILED
 *      SET RETURN CODE TO 1
 *
 *    ERROR ENTER ERR2 - DINFO/DISPUP MACRO FAILED
 *      SET RETURN CODE TO 2
 *
 *    ERROR ENTER ERR3 - S/7 ID NOT FOUND IN S7CT
 *      SET RETURN CODE TO 3
 *    ERROR RETURN
 *    EXIT WITH RETURN CODE SET AS ABOVE
 * ENDSEGMENT DOMCBLD2
 *
 *
 * DOMDINF SUBROUTINE SEGMENT
 *    SET UP DATA FOR DISPLAY USING DINFO MACRO
 *    ERROR EXIT TO ERR2 IF RETURN CODE IS NOT ZERO
 * ENDSEGMENT DOMDINF
```

```
MEMBER NAME  DOMCBLD3
 * DOMCBLD3 MAIN SEGMENT
 *    GET ADDRESS OF XCVT USING GETCVT MACRO
 *    GET MESSAGES 335, 336 AND 386 USING MESSAGE MACRO
 *    CALCULATE NUMBER OF POINTS TO DISPLAY AND SAVE NUMBER
 *    BLANK THE BUFFER AREA
 *    IF PCC III IS NOT CURRENTLY UP THEN
 *      WRITE THE BACKGROUND USING DISPREQ MACRO
 *      ERROR EXIT IF THE RETURN CODE IS NOT ZERO TO ERR1
 *      TURN ON APPROPRIATE BACKLIGHTS USING DLITES MACRO
 *    ENDIF
 *    TURN OFF SAME DISPLAY FLAG (CCESDY)
 *    SAVE BUFFER POINTERS
 *    IF ANALOG DATA THEN
 *      DO DBANALOG
 *    ELSE
 *      IF PULSE COUNTER DATA THEN
 *        DO DBPC
 *      ELSE -STATUS DATA
 *        DO DBSTAT
 *      ENDIF
 *    ENDIF
 *    POINT TO PAGE NUMBER AND TOTAL NUMBER OF PAGES
 *    DO DINF
 *    POINT TO S/7 ID
 *    DO DINF
 *    POINT TO TYPE OF DATA
 *    DO DINF
 *    POINT TO TERMINAL NAME
 *    DO DINF
 *    POINT TO POKE POINTS FOR FIRST COLUMN
 *    DO DINF
 *    POINT TO POINT NAMES FOR FIRST COLUMN
 *    DO DINF
 *    POINT TO TYPES BUFFER FOR FIRST COLUMN
 *    DO DINF
 *    POINT TO STATUS OF POINTS FOR FIRST COLUMN
 *    DO DINF
 *    POINT TO POKE POINTS FOR SECOND COLUMN
 *    DO DINF
 *    POINT TO POINT NAMES FOR SECOND COLUMN
 *    DO DINF
 *    POINT TO TYPES BUFFER FOR SECOND COLUMN
 *    DO DINF
 *    POINT TO STATUS OF POINTS FOR SECOND COLUMN
 *    DO DINF
 *    ISSUE DISPUP MACRO TO WRITE DISPLAY TO SCREEN
 *    ERROR EXIT TO ERR1 IF RETURN CODE IS NOT ZERO
 *    EXIT RETURN CODE OF ZERO
 *
 *    ERROR ENTER ERR1
 *    EXIT RETURN CODE OF ONE
 *
 *
 * DBANALOG SUBROUTINE SEGMENT
 *    POINT TO ANALOG NAMES LIST
 *    ADD DISPLACEMENT INTO ANALOG NAMES LIST
 *    IF MORE THAN ONE COLUMN THEN
 *      SET COUNT TO 2
```

```
MEMBER NAME  DOMCBLD3
 *    ELSE
 *      SET COUNT TO 1
 *    ENDIF
 *    UNTIL COUNT IS ZERO DO
 *      UNTIL ALL DATA FOR COLUMN IS PROCESSED DO
 *        MOVE POKE POINT TO BUFFER
 *        ADJUST BUFFER POINTER
 *        MOVE ANALOG POINT NAME TO BUFFER
 *        ADJUST BUFFER POINTER
 *        CALCULATE DISPLACEMENT INTO TYPES ARRAY USING CODE IN ITEM
 *        MOVE TYPE TO BUFFER
 *        ADJUST POINTER TO BUFFER
 *        DETERMINE STATUS OF POINT
 *        MOVE IT TO BUFFER
 *        ADJUST POINTER TO BUFFER
 *      ENDDO
 *      IF COUNT IS 2 THEN
 *        DO DCOL2
 *      ENDIF
 *    ENDDO
 *  ENDSEGMENT DBANALOG
 *
 *
 *  DBPC SUBROUTINE SEGMENT
 *    IF MORE THAN ONE COLUMN THEN
 *      SET COUNT TO 2
 *    ELSE
 *      SET COUNT TO 1
 *    ENDIF
 *    UNTIL COUNT IS ZERO DO
 *      UNTIL ALL DATA FOR COLUMN IS PROCESSED DO
 *        MOVE POKE POINT TO BUFFER
 *        ADJUST BUFFER POINTER
 *        MOVE PULSE COUNTER NAME TO BUFFER
 *        ADJUST BUFFER POINTER
 *        CALCULATE DISPLACEMENT INTO TYPES ARRAY USING CODE IN ITEM
 *        MOVE TYPE TO BUFFER
 *        ADJUST POINTER TO BUFFER
 *        DETERMINE STATUS OF POINT
 *        MOVE IT TO BUFFER
 *        ADJUST BUFFER POINTER
 *      ENDDO
 *      IF COUNT IS 2 THEN
 *        DO DCOL2
 *      ENDIF
 *    ENDDO
 *  ENDSEGMENT DBPC
 *
 *
 *  DBSTAT SUBROUTINE SEGMENT
 *    IF MORE THAN ONE COLUMN OF DATA THEN
 *      SET COUNT TO 2
 *    ELSE
 *      SET COUNT TO 1
 *    ENDIF
 *    UNTIL COUNT IS ZERO DO
 *      UNTIL ALL DATA FOR COLUMN PROCESSED DO
 *        UNTIL ALL ITEMS IN STATUS GROUP PROCESSED DO
```

```
MEMBER NAME  DOMCBLD3
 *          MOVE POKE POINT TO BUFFER
 *          ADJUST BUFFER POINTER
 *          MOVE STATUS ITEM NAME TO BUFFER
 *          ADJUST BUFFER POINTER
 *          CALCULATE DISPLACEMENT INTO TYPES ARRAY USING CODE IN ITEM
 *          MOVE TYPE TO BUFFER
 *          ADJUST POINTER TO BUFFER
 *          DETERMINE STATUS OF POINT
 *          MOVE IT TJ BUFFER
 *          ADJUST BUFFER POINTER
 *        ENDDO
 *        ADJUST POINTER TO STATUS GROUP TO BYPASS HEADER
 *      ENDDO
 *      IF COUNT IS 2 THEN
 *        DO DCOL2
 *      ENDIF
 *    ENDDO
 *  ENDSEGMENT DBSTAT
 *
 *
 *  DCOL2 SUBROUTINE SEGMENT
 *    ADJUST BUFFER POINTERS FOR SECOND COLUMN
 *  ENDSEGMENT DCOL2
 *
 *
 *  DINF SUBROUTINE SEGMENT
 *    SET UP DATA TO BE WRITTEN TO SCREEN USING DINFO MACRO
 *    ERROR EXIT IF RETURN CODE IS NOT ZERO TO ERR1
 *  ENDSEGMENT DINF
```

```
MEMBER NAME  DOMCDC01
 * DOMCDC01 MAIN SEGMENT
 *   INITIALIZE STAE LIST
 *   POINT TO PARAMETER LIST
 *   SAVE PATCH ID
 *   IF PATCH ID IS GREATER THAN 18 THEN
 *     SET CASE ID TO 6
 *   ENDIF
 *   IF PATCH ID IS EQUAL TO NINE THEN
 *     SET CASE ID TO PDC REPLY (8)
 *   ENDIF
 *   GET ADDRESSES OF DCT INDEX TABLE, SYSGEN OPTIONS TABLE AND PDC
 *    OPTIONS TABLE FROM EMSCVT AND SAVE THEM
 *
 *   CASE ENTRY - PATCH ID
 *
 *     CASE 4 AND 8 - INITIALIZATION FOR MACRO AND DISPLAY
 *       CALL DOMCDC06
 *       ERROR EXIT IF RETURN CODE IS NOT ZERO TO ERR1
 *       IF PATCH ID IS 2 THEN
 *         DO DOMDINF
 *       ELSE
 *         IF EXECUTED INDICATOR IS ON THEN
 *           DO DOMEVNT
 *           IF ACTION IS OPEN/TRIP OR CLOSE THEN
 *             DO DOMWORD
 *           ENDIF
 *           POST ECB WITH RETURN CODE OF 0 AND MSG # 314
 *           DO DOMCLN
 *         ENDIF
 *       ENDIF
 *
 *     CASE 12 AND 20 - ARM OR EXECUTE FROM DISPATCHER
 *       DO DOMDGET
 *       IF DCT ADDRESS IS ZERO THEN
 *         SET ERROR CODE TO 18
 *         ERROR EXIT TO ERR1
 *       ENDIF
 *       IF PATCH ID IS 10 AND
 *       IF ARMED INDICATOR IS ON THEN
 *         SET UP FOR MSG # 313
 *         DO DOMMESG
 *         DO DOMDINF
 *       ELSE
 *         CALL DOMCDC07
 *         IF RETURN CODE IS 50 (DEVICE READY) THEN
 *           SET UP FOR MESSAGE # 312
 *           DO DOMMESG
 *           DO DOMDINF
 *           DO DOMEVNT
 *         ELSE
 *           ERROR EXIT TO ERR1 IF RETURN CODE IS NOT ZERO
 *         ENDIF
 *         IF EXECUTED INDICATOR IS ON THEN
 *           IF ACTION IS OPEN/TRIP OR CLOSE THEN
 *             DO DOMWORD
 *           ENDIF
 *           DO DOMDINF
 *           DO DOMEVNT
```

```
MEMBER NAME  DOMCDCO1
 *          DO DOMCLN
 *        ENDIF
 *      ENDIF
 *
 *     CASE 16 - PDC REPLY FROM S/7
 *       CALL DOMCDC08
 *       ERROR EXIT TO ERR1 IF RETURN CODE IS GREATER THAN 8
 *       IF RETURN CODE IS LESS THAN 8 THEN
 *         IF RETURN CODE IS 4 THEN
 *           DO DOMCLN
 *         ENDIF
 *         IF DISPLAY INITIATED THEN
 *           DO DOMDINF
 *         ENDIF
 *         DO DOMEVNT
 *       ENDIF
 *
 *     CASE 24,28,32,AND 36 - COS RECEIVED, TIME-OUTS, CANCEL
 *       IF PATCH ID IS 14 THEN
 *         DO DOMDGET
 *         IF DCT ADDRESS IS ZERO THEN
 *           BRANCH TO END
 *         ENDIF
 *       ENDIF
 *       CALL DOMCDC09
 *       ERROR EXIT TO ERR1 IF RETURN CODE GREATER THAN 8
 *       IF RETURN CODE IS NOT 8 THEN
 *         IF DISPLAY INITIATED THEN
 *           DO DOMDINF
 *         ENDIF
 *         DO DOMEVNT
 *         DO DOMCLN
 *       ENDIF
 *END-LABEL
 *
 *   END CASE
 *   EXIT WITH RETURN CODE SET TO ZERO
 *
 *   ERROR ENTER ERR1
 *
 *     CASE ENTRY - ERROR CODE
 *
 *       CASE 2 - ACCESS AREA/FUNCTION INVALID
 *         SET UP FOR MSG # 308
 *         DO DOMMESG
 *         DO DOMDINF
 *         GET MESSAGE # 309 USING MESSAGE MACRO
 *         ISSUE EVENT USING SCEVENT MACRO
 *
 *       CASE 5 - DINFO MACRO FAILED
 *         IF THE SAME DISPLAY IS UP THEN
 *           ISSUE MESSAGE # 322 USING MESSAGE MACRO
 *           IF DCT ADDRESS IS NOT ZERO THEN
 *             DO DOMCLN
 *           ENDIF
 *         ENDIF
 *
 *       CASE 6 - INVALID COMMAND
```

```
MEMBER NAME  DOMCDC01
 *          IF MACRO INITIATED THEN
 *            POST ECB WITH RETURN CODE OF 24 AND MSG # 303
 *          ELSE
 *            SET UP FOR MESSAGE # 303
 *            DO DOMMESG
 *            DO DOMDINF
 *          ENDIF
 *          DO DOMCLN
 *
 *        CASE 7 - DEVICE ALREADY ARMED
 *          SET UP FOR MSG # 469
 *          DO DOMMESG
 *          DO DOMDINF
 *
 *        CASE 9 - DEVICE TAGGED - OPTION NOT UNTAG
 *          SET UP FOR MSG # 318
 *          DO DOMMESG
 *          IF DISPLAY INITIATED THEN
 *            DO DOMDINF
 *          ELSE
 *            POST ECB WITH RETURN CODE OF 28 AND MSG # 318
 *          ENDIF
 *          DO DOMCLN
 *
 *        CASE 10 -  GETITEM FAILED FOR PDC ITEM IN CAPDC ARRAY
 *          SET UP FOR MSG # 323
 *          DO DOMMESG
 *          IF DISPLAY INITIATED THEN
 *            DO DOMDINF
 *          ELSE
 *            POST ECB WITH RETURN CODE OF 24 AND MSG # 323
 *          ENDIF
 *          DO DOMCLN
 *
 *        CASE 11 - PDC MSG-NO CONTROL IN PROGRESS
 *          SET UP FOR MSG # 325
 *          ISSUE MESSAGE USING MESSAGE MACRO TO GENERAL EVENTS AND
 *           GENERAL ALARM TYPERS
 *          GET MESSAGE USING MESSAGE MACRO
 *          ISSUE EVENT USING SCEVENT MACRO
 *
 *        CASE 12 - COS-NO CONTROL IN PROGRESS
 *          ISSUE ALARM USING DOMCALRM MACRO
 *
 *        CASE 13 - DEVICE ALREADY IN REQUESTED STATE
 *          SET UP FOR MSG # 316
 *          DO DOMMESG
 *          IF DISPLAY INIATED THEN
 *            DO DOMDINF
 *          ELSE
 *            POST ECB WITH RETURN CODE OF 32 AND MSG # 316
 *          ENDIF
 *          DO DOMCLN
 *
 *        CASE 14 - GETITEM FOR RCB FAILED
 *          IF PATCH ID IS 2 THEN
 *            DO DOMDINF
 *          ENDIF
```

```
MEMBER NAME  DOMCDC01
 *
 *       CASE 16 - UNABLE TO COMMUNICATE WITH S/7
 *         GET MESSAGE # 330 USING MESSAGE MACRO
 *         IF DISPLAY INITIATED THEN
 *           DO DOMDINF
 *         ELSE
 *           POST ECB WITH RETURN CODE OF 40 AND MSG # 330
 *         ENDIF
 *         ISSUE MESSAGE # 330 USING MESSAGE MACRO TO SYSTEM PROGRAMMER
 *          ROUTING CODE
 *         DO DOMCLN
 *
 *       CASE 17 - DEVICE IS OUT OF SERVICE
 *         SET UP FOR MSG # 387
 *         DO DOMMESG
 *         IF MACRO INITIATED THEN
 *           POST ECB WITH RETURN CODE OF 48 AND MSG # 387
 *         ELSE
 *           DO DOMDINF
 *         ENDIF
 *         IF DCT ADDRESS IS NOT ZERO THEN
 *           DO DOMCLN
 *         ENDIF
 *
 *       CASE 18 - NO CONTROL ACTION IN PROGRESS
 *         SET UP FOR MSG # 424
 *         DO DOMMESG
 *         DO DOMDINF
 *
 *       CASE 19 - DEVICE IS NOT CONTROLLABLE
 *         SET UP FOR MSG # 319
 *         DO DOMMESG
 *         IF MACRO INITIATED THEN
 *           POST ECB WITH RETURN CODE OF 52 AND MSG # 319
 *         ELSE
 *           DO DOMDINF
 *         ENDIF
 *         IF DCT ADDRESS IS NOT ZERO THEN
 *           DO DOMCLN
 *         ENDIF
 *
 *       CASE 20 - TCT TYPE3 IS AUTO - NO CONTROL
 *         SET UP FOR MSG # 489
 *         DO DOMMESG
 *         IF DISPLAY INITIATED THEN
 *           DO DOMDINF
 *         ELSE
 *           POST USER ECB WITH RETURN CODE OF 28 AND MSG # 489
 *         ENDIF
 *         DO DOMCLN
 *
 *       CASE 21 - NO GETWA FOR STATUS LOG LIST
 *         SET UP FOR MSG # 467
 *         DO DOMMESG
 *         IF DISPLAY INITIATED THEN
 *           DO DOMDINF
 *         ELSE
 *           POST ECB WITH RETURN CODE OF 76 AND MSG # 467
```

```
MEMBER NAME  DOMCDC01
 *         ENDIF
 *         DO DOMCLN
 *
 *     END CASE
 *     EXIT WITH RETURN CODE SET TO ZERO
 * ENDSEGMENT DOMCDC01
 *
 * DOMDGET SUBROUTINE SEGMENT
 *   SET UP POINTERS TO DCT INDEX TABLE, DCE, DCT, RCB AND ITEM
 * ENDSEGMENT DOMDGET
 *
 * DOMCLN SUBROUTINE SEGMENT
 *   IF DISPLAY INITIATED THEN
 *     ZERO CEATAB FIELD
 *   ENDIF
 *   ZERO DCT INDEX TABLE ENTRY
 *   FREE DCT USING FREEWA MACRO
 *   RESET STAE LIST
 * ENDSEGMENT DOMCLN
 *
 * DOMDINF SUBROUTINE SEGMENT
 *   IF DISPLAY IS UP THEN
 *     WRITE TO SCREEN USING DINFO MACRO
 *     IF RETURN CODE IS NOT ZERO THEN
 *       SET ERROR CODE TO 5
 *       ERROR EXIT TO ERR1
 *     ENDIF
 *   ENDIF
 * ENDSEGMENT DOMDINF
 *
 * DOMEVNT SUBROUTINE SEGMENT
 *   ISSUE EVENT USING SCEVENT MACRO
 * ENDSEGMENT DOMEVNT
 *
 * DOMMESG SUBROUTINE SEGMENT
 *   GET MESSAGE USING MESSAGE MACRO
 * ENDSEGMENT DOMMESG
 *
 * DOMWORD SUBROUTINE SEGMENT
 *   STRTSRCH FOR NUMBER OF GROUPS IN STATUS FOR RCB
 *   EXIT IF GROUP FOUND
 *MOS2ND-LABEL
 *     CHANGE BIT IN GROUP WORD FOR APPROPRIATE ITEM
 *     IF MOS TYPE OF DEVICE THEN
 *       POINT TO NEXT ITEM
 *       BRANCH TO MOS2ND
 *     ENDIF
 *   ORELSE
 *     POINT TO NEXT STATUS GROUP
 *   ENDLOOP
 *   ENDSRCH
 * ENDSEGMENT DOMWORD
```

```
MEMBER NAME  DOMCDC02
 * DOMCDC02 MAIN SEGMENT
 *    PATCH DOMCDC01 WITH PARAMETER LIST AND PATCH ID OF 16
 * ENDSEGMENT DOMCDC02
```

```
MEMBER NAME  DOMCDC03
 * DOMCDC03 MAIN SEGMENT
 *    PATCH DOMCDC01 WITH PARAMETER LIST AND PATCH ID OF 18
 * ENDSEGMENT DOMCDC03
```

```
MEMBER NAME  DOMCDC04
* DOMCDC04 MAIN SEGMENT
*    IF A SINGLE DEVICE IS TO BE CONTROLLED THEN
*      DO DOMNAME
*      BUILD PARAMETER LIST FOR DOMCDC01
*      UNTIL RETURN CODE FROM DOMCDC01 IS NOT EQUAL TO X'38' AND
*      UNTIL IT IS NOT EQUAL TO X'3C' DO
*        DO DOMPTCH
*      ENDDO
*      IF RETURN CODE IS EQUAL TO X'FF' THEN
*        SET RETURN CODE TO X'48'
*      ELSE
*        SET RETURN CODE TO RETURN CODE FROM DOMCDC01
*        IF THE AREA INDICATOR IS ON THEN
*          DO DOMMESG
*          MOVE ITEM NAME TO AREA
*        ENDIF
*      ENDIF
*    ELSE  -LIST OF DEVICES TO BE CONTROLLED
*      UNTIL ALL DEVICES IN LIST PROCESSED DO
*        BUILD PARAMETER LIST FOR DOMCDC01
*        DO DOMNAME
*        UNTIL RETURN CODE IS NOT EQUAL TO X'38' AND
*        UNTIL RETURN CODE IS NOT EQUAL TO X'3C' DO
*          DO DOMPTCH
*        ENDDO
*        IF RETURN CODE IS X'FF' THEN
*          SET RETURN CODE TO X'48'
*        ELSE
*          IF RETURN CODE IS NOT ZERO THEN
*            ERROR EXIT IF THE LIST IS DEPENDENT TO ONE
*            SET RETURN CODE TO X'40'
*          ENDIF
*        ENDIF
*        POINT TO NEXT DEVICE IN LIST
*      ENDDO
*      IF THE AREA INDICATOR IS ON THEN
*        IF THE DEVICE FAILED TO EXEC,THEN
*          SET MSG#326 - DEVICE FAILED
*        ELSE
*          SET MSG#314 - CONTROL COMPLETE
*        ENDIF
*        DO DOMMESG
*      ENDIF
*    ENDIF
*    POST THE ECB WITH APPROPRIATE RETURN CODE
*    EXIT WITH RETURN CODE SET TO ZERO
*
*    ERROR ENTER ONE
*      SET RETURN CODE TO X'44'
*      IF THE AREA INDICATOR IS ON THEN
*        DO DOMMESG
*      ENDIF
*      POST THE ECB WITH RETURN CODE
*      EXIT WITH RETURN CODE SET TO ZERO
* ENDSEGMENT DOMCDC04
*
*
* DOMNAME SUBROUTINE SEGMENT
```

```
MEMBER NAME  DOMCDC04
*   BUILD RCB ITEM NAME FROM S/7 ID AND EMT ID IN PARAMETER LIST
* ENDSEGMENT DOMNAME
*
*
* DOMPTCH SUBROUTINE SEGMENT
*   PATCH DOMCDC01 WITH A PATCH ID OF 4
*   IF RETURN CODE IS NOT ZERO THEN
*     SET RETURN CODE TO X'FF'
*   ELSE
*     WAIT ON PATCH ECB
*     IF PATCH FAILED THEN
*       SET RETURN CODE TO X'FF'
*     ELSE
*       WAIT ON MACRO ECB
*     ENDIF
*   ENDIF
* ENDSEGMENT DOMPTCH
*
*
* DOMMESG SUBROUTINE SEGMENT
*   BLANK MESSAGE AREA
*   GET MESSAGE USING MESSAGE MACRO
*   MOVE MESSAGE TO USER AREA
* ENDSEGMENT DOMMESG
```

```
MEMBER NAME  DOMCDC05
* DOMCDC05 MAIN SEGMENT
*    IF NOT EXEC TIME OUT,AND
*    IF NOT A VERIFY TIME OUT,THEN
*      IF DISPLAY INITIATED THEN
*        DELETE ARM TIME-OUT USING PTIME MACRO
*      ENDIF
*    ENDIF
*    START TO BUILD PDC MESSAGE
*    GET PDC ITEM FROM CAPDC ARRAY USING GETITEM MACRO
*    ERROR EXIT TO TEN IF MACRO RETURN CODE IS NOT ZERO
*    BUILD REST OF PDC MESSAGE FROM PDC ITEM DEPENDING ON DEVICE TYPE
*     AND ACTION OR ON TIME-OUT
*    CONVERT MESSAGE TEXT TO ASCII USING ASCICONV MACRO
*    SEND MESSAGE TO SYSTEM/7 USING S7WRITE MACRO
*    ERROR EXIT TO SIXT IF RETURN CODE IS NOT ZERO
*    START EXECUTE TIME-OUT USING PTIME MACRO AND INTERVAL IN CASYS
*    SET ON EXECUTING FLAGS IN ITEM AND DCT
*    IF MOTOR OPERATED SWITCH THEN
*      SET ON EXECUTING FLAG IN SECOND ITEM
*    ENDIF
*    SET FLAG INDICATORS IN STAE LIST
*    EXIT WITH RETURN CODE SET TO ZERO
*
*    ERROR ENTER SIXT - UNABLE TO COMMUNICATE WITH SYSTEM/7
*      SAVE RETURN CODE FROM MACRO
*      SET RETURN CODE TO SIXTEEN
*
*    ERROR ENTER TEN - GETITEM FOR PDC ARRAY FAILED
*      SET RETURN CODE TO TEN
*
*    ERROR RETURN
*    EXIT WITH RETURN CODE SET AS ABOVE
*
* ENDSEGMENT DOMCDC05
```

```
MEMBER NAME  DOMCDC06
 * DOMCDC06 MAIN SEGMENT
 *   CASE ENTRY
 *
 *     CASE 4 - DISPLAY INITIATED
 *       SAVE ITEM AND RCB NAMES FROM PARAMETER LIST
 *       DO DOMINDEX
 *       ERROR EXIT TO TWO IF ACCESS AREA DOES NOT MATCH
 *       DO DOMFIND
 *       ERROR EXIT TO TWO IF FUNCTION DOES NOT MATCH
 *       DO DOMDCTB
 *       SET UP FOR MSG # 388
 *       DO DOMMESG
 *       DO DOMEVNT
 *       SET UP FOR MSG # 472
 *       DO DOMMESG
 *       DO DOMOPGET
 *       MOVE APPROPRIATE OPTIONS TO MESSAGE BASED ON STATUS OF POINT
 *       START SELECTION TIME-OUT USING PTIME MACRO
 *
 *     CASE 8  - MACRO INITIATED
 *       SAVE ITEM AND RCB NAMES FROM PARAMETER LIST
 *       DO DOMINDEX
 *       DO DOMFIND
 *       DO DOMDCTB
 *       SET UP FOR MSG # 388
 *       DO DOMMESG
 *       DO DOMEVNT
 *       DO DOMOPGET
 *       COMPARE ACTION TO TYPE OF DEVICE
 *       ERROR EXIT TO SIX IF ACTION INVALID
 *       MOVE ACTION PHRASE TO DCT
 *       SET ACTION CODE IN DCT
 *       CALL DOMCDC11
 *       ERROR EXIT TO SEVT IF RETURN CODE IS 17
 *       ERROR EXIT TO NINT IF RETURN CODE IS 19
 *       ERROR EXIT TO TWEN IF RETURN CODE IS NOT ZERO
 *       IF DEVICE DID NOT EXECUTE THEN
 *         SET UP FOR MESSAGE # 312
 *         DO DOMMESG
 *         DO DOMEVNT
 *         CALL DOMCDC05
 *         ERROR EXIT TO TWEN IF RETURN CODE IS NOT ZERO
 *
 *   END CASE
 *   EXIT WITH RETURN CODE OF ZERO
 *
 *   ERROR ENTER ONE - DEVICE ALREADY BEING CONTROLLED
 *     SET UP FOR MSG # 302
 *     DO DOMMESG
 *     IF MACRO INITIATED THEN
 *       POST ECB WITH RETURN CODE OF 56
 *     ENDIF
 *     ZERO RETURN CODE
 *
 *   ERROR ENTER TWO - INVALID ACCESS OR FUNCTION AREA
 *     SET ERROR CODE TO 2
 *
 *   ERROR ENTER THRE - DEVICE NOT DEFINED IN DATA BASE
```

```
MEMBER NAME  DOMCDC06
 *      SET UP FOR MSG # 301
 *      DO DOMMESG
 *      IF MACRO INITIATED THEN
 *        POST ECB WITH RETURN CODE OF 4 AND MESSAGE # 301
 *      ENDIF
 *      ZERO RETURN CODE
 *
 *    ERROR ENTER FOUR  -DCT INDEX TABLE FULL
 *      SET UP FOR MSG # 315
 *      DO DOMMESG
 *      IF MACRO INITIATED THEN
 *        POST ECB WITH RETURN CODE OF 60
 *      ENDIF
 *      ZERO RETURN CODE
 *
 *    ERROR ENTER FIVE -2ND REQUEST INVALID
 *      SET UP FOR MSG # 466
 *      DO DOMMESG
 *      ZERO RETURN CODE
 *
 *    ERROR ENTER SIX   -INVALID COMMAND
 *      SET RETURN CODE TO 6
 *
 *    ERROR ENTER EIGH  -NO GETWA AVAILABLE
 *      SET UP FOR MSG # 333
 *      DO DOMMESG
 *      IF MACRO INIATED THEN
 *        POST ECB WITH RETURN CODE OF 80 AND MSG # 333
 *      ENDIF
 *      ZERO RETURN CODE
 *
 *    ERROR ENTER FORT   -GETITEM FOR RCB FAILED
 *      SET UP FOR MSG # 317
 *      DO DOMMESG
 *      IF MACRO INIATED THEN
 *        POST ECB WITH RETURN CODE OF 84 AND MSG # 317
 *      ENDIF
 *      SET RETURN CODE TO 14
 *
 *    ERROR ENTER SEVT - DEVICE IS OUT OF SERVICE
 *      SET RETURN CODE TO 17
 *
 *    ERROR ENTER NINT - DEVICE IS NOT CONTROLLABLE
 *      SET RETURN CODE TO 19
 *
 *    ERROR ENTER TWEN   -BAD RETURN CODE FROM DOMCDC05 OR DOMCDC11
 *
 *    ERROR RETURN
 *    EXIT WITH RETURN CODE SET AS ABOVE
 *  ENDSEGMENT DOMCDC06
 *
 *  DOMFIND SUBROUTINE SEGMENT
 *    GET ADDR OF DEVICE ITEM USING GETITEM MACRO
 *    ERROR EXIT TO THRE IF RETURN CODE IS NOT ZERO
 *    ERROR EXIT TO SEVT IF DEVICE IS OUT OF SERVICE
 *    ERROR EXIT TO NINT IF DEVICE IS NOT CONTROLLABLE
 *    IF DEVICE IS MOS OR
 *    IF DEVICE IS TCT TYPE-3 THEN
```

```
MEMBER NAME  DOMCDC06
 *      POINT TO SECOND ITEM
 *      ERROR EXIT IF DEVICE IS OUT OF SERVICE TO SEVT
 *    ENDIF
 *    ERROR EXIT TO NINT IF DEVICE IS A GENERATOR
 * ENDSEGMENT DOMFIND
 *
 * DOMDCTB SUBROUTINE SEGMENT
 *    STRTSRCH ON DCT INDEX TABLE -(UNTIL-DO LOOP)
 *    EXITIF EMPTY SLOT IN TABLE FOUND
 *      IF DISPLAY INITIATED THEN
 *        ERROR EXIT IF CEATAB ENTRY IS NOT ZERO FIVE
 *      ENDIF
 *      GET AREA FOR DCT USING GETWA MACRO
 *      ERROR EXIT TO EIGH IF RETURN CODE IS NOT ZERO
 *      BUILD DCT FROM ITEM AND RCB
 *      SET STAE INFORMATION IN STAE LIST
 *      STORE ADDR OF DCT IN CEATAB AND DCE
 *    ORELSE
 *      POINT TO NEXT SLOT IN TABLE
 *    ENDLOOP
 *      ERROR EXIT TO FOUR
 *    ENDSRCH
 * ENDSEGMENT DOMDCTB
 *
 * DOMINDEX SUBROUTINE SEGMENT
 *    UNTIL DCT INDEX TABLE SEARCHED DO
 *      IF MATCH FOUND ON ITEM NAME THEN
 *        ERROR EXIT ONE
 *      ELSE
 *        POINT TO NEXT ENTRY IN TABLE
 *      ENDIF
 *    ENDDO
 *    GET ADDR OF RCB USING GETITEM MACRO
 *    ERROR EXIT TO FORT IF RETURN CODE IS NOT ZERO
 * ENDSEGMENT DOMINDEX
 *
 * DOMOPGET SUBROUTINE SEGMENT
 *    CALCULATE DISPLACEMENT INTO OPTIONS TABLE BASED ON DEVICE TYPE
 * ENDSEGMENT DOMOPGET
 *
 * DOMEVNT SUBROUTINE SEGMENT
 *    ISSUE EVENT USING SCEVENT MACRO
 * ENDSEGMENT DOMEVNT
 *
 * DOMMESG SUBROUTINE SEGMENT
 *    GET MESSAGE USING MESSAGE MACRO
 * ENDSEGMENT DOMMESG
```

```
MEMBER NAME  DOMCDC07
 * DOMCDC07 MAIN SEGMENT
 *   CASE ENTRY  - CASE ID
 *
 *     CASE 12 - OPTIONS SELECTED
 *       ERROR EXIT IF DEVICE ARMED TO SEVE
 *       DELETE SELECTION TIME-OUT USING PTIME MACRO
 *       IF PATCH ID IS 23 THEN
 *         SAVE ACTION WORD FOR MESSAGES
 *         SET ACTION CODE TO TAG
 *       ELSE
 *         IF PATCH ID IS 24 THEN
 *           SAVE ACTION WORD
 *           SET ACTION CODE TO UNTAG
 *         ELSE
 *           IF PATCH ID IS 19 THEN
 *             SET ACTION CODE TO AUTOMATIC
 *             IF DEVICE IS TCT TYPE 2 OR 3 THEN
 *               SAVE ACTION WORD
 *             ELSE
 *               ERROR EXIT TO SIX
 *             ENDIF
 *           ELSE
 *             IF PATCH ID IS 20 THEN
 *               SAVE ACTION WORD
 *               IF DEVICE IS TCT TYPE 1 OR 3 THEN
 *                 SET ACTION CODE TO RAISE
 *               ELSE
 *                 SET ACTION CODE TO OPEN
 *               ENDIF
 *             ELSE
 *               IF PATCH ID IS 21 THEN
 *                 SAVE ACTION WORD FOR MESSAGES
 *                   IF DEVICE IS TCT TYPE 1 OR 3 THEN
 *                     SET ACTION CODE TO LOWER
 *                   ELSE
 *                     SET ACTION CODE TO CLOSE
 *                   ENDIF
 *               ELSE
 *                 IF PATCH ID IS 22 THEN
 *                   SAVE ACTION WORD
 *                   SET ACTION CODE TO MANUAL
 *                   IF DEVICE IS NOT TCT TYPE 2 OR 3 THEN
 *                     ERROR EXIT TO SIX
 *                   ENDIF
 *                 ENDIF
 *               ENDIF
 *             ENDIF
 *           ENDIF
 *         ENDIF
 *       ENDIF
 *       CALL DOMCDC11
 *       ERROR EXIT TO TWEN IF RETURN CODE IS NOT ZERO
 *       SET RETURN CODE TO 50
 *       ERROR EXIT TO TWEN
 *
 *     CASE 20  -EXECUTE REQUEST
 *       CALL DOMCDC05
 *       ERROR EXIT TO TWEN IF RETURN CODE IS NOT ZERO
```

```
MEMBER NAME  DOMCDC07
 *
 *    ENDCASE
 *    EXIT WITH RETURN CODE SET AS ABOVE
 *
 *    ERROR ENTER SIX - INVALID COMMAND
 *      SET RETURN CODE TO SIX
 *
 *    ERROR ENTER SEVE - DEVICE ALREADY ARMED
 *      SET RETURN CODE TO 7
 *
 *    ERROR ENTER TWEN
 *    ERROR RETURN
 *    EXIT WITH RETURN CODE SET AS ABOVE
 *  ENDSEGMENT DOMCDC07
 *
```

```
MEMBER NAME  DOMCDC08
* DOMCDC08 MAIN SEGMENT
*   MOVE INPUT DATA TO SAVE AREAS DEPENDING ON TYPE OF REPLY
*   IF PDC REPLY IN TRANSACTION CODE 86 OR
*   IF PDC REPLY IS TRANSACTION CODE 16 THEN
*     CONVERT TO EBCDIC USING ASCICONV MACRO
*     RELEASE INPUT BUFFER USING RLSEBUFF MACRO
*   ENDIF
*   IF 370 IS NOT CONTROLLING CPU THEN
*     GET ADDRESS OF ITEM IN DATA BASE USING GETITEM MACRO
*     IF RETURN CODE IS ZERO THEN
*       DO UN#TAG
*     ENDIF
*     ZERO RETURN CODE
*   ELSE - 370 IS THE CONTROLLING CPU
*     START SEARCH FOR ENTRY IN DCT INDEX TABLE
*     EXIT IF MATCH FOUND ON ITEM NAME THEN
*       DELETE EXECUTE TIME-OUT USING PTIME MACRC
*       IF RETURN CODE IS ZERO THEN
*         IF TAG OR UNTAG REPLY THEN
*           DO UN#TAG
*           SET ON EXECUTED INDICATOR
*           SET OFF EXECUTING INDICATORS IN ITEM AND DCT
*           SET UP FOR MESSAGE # 314
*           DO DOMMESG
*           CALL DOMCDC11 - TO  LOG CHANGE OF STATUS
*           IF MACRO INITIATED THEN
*             POST ECB WITH RETURN CODE OF ZERO AND MSG # 314
*           ENDIF
*           SET RETURN CODE TO FOUR
*         ELSE
*           IF PDC REPLY IS NOT FROM THE RDA THEN
*             SET UP FOR MESSAGE # 314
*             DO DOMMESG
*             IF ALARM IS OUTSTANDING THEN
*               DO DOMDALM - TO DELETE ALARM
*             ENDIF CONDITION CODE TO 9
*             SET OFF EXECUTING FLAGS
*             SET ON EXECUTED FLAG
*            IF NOT A RAISE OR LOWER COMMAND THEN
*             FLIP STATUS BIT IN DATA BASE ITEM
*             DO DOMWORD
*            ENDIF
*             IF ACTION IS OPEN/TRIP OR CLOSE THEN
*               IF STATUS BIT IS ON THEN
*                 DELETE ENTITY USING DISPENT MACRO
*               ELSE
*                 ISSUE ENTITY USING DISPENT MACRO
*               ENDIF
*               CALL DOMCDC11 - TO LOG CHANGE OF STATUS
*             ENDIF
*             IF MACRO INITIATED,THEN
*               POST MACRO ECB WITH ACTION COMPLETE MSG# - MSG#314
*             ENDIF
*             IF WALLBOARD FLAG IS ON THEN
*               GET AREA FOR PARAMETERS USING GETWA MACRO
*               IF RETURN CODE FROM MACRO IS ZERO THEN
*                 SET UP PARAMETERS
*                 PATCH WALLBOARD PROCESSOR (DOMCWBPR)
```

```
MEMBER NAME  DOMCDC08
*                    IF RETURN CODE IS NOT ZERO THEN
*                      FREE AREA USING FREEWA MACRO
*                    ENDIF
*                  ENDIF
*                ENDIF
*                SET RETURN CODE TO FOUR
*              ELSE - REPLY IS FROM THE RDA
*                IF EXECUTION TIME-OUT THEN
*                  SET UP FOR MESSAGE # 306
*                  DO DOMMESG
*                  DO DOMDALM - TO ISSUE ALARM
*                  IF MACRO INITIATED THEN
*                    POST ECB WITH RETURN CODE OF 12 AND MSG # 306
*                  ENDIF
*                  SET OFF EXECUTING FLAGS
*                  IF MOTOR OPERATED SWITCH THEN
*                    SET OF EXECUTING FLAG IN SECOND ITEM
*                  ENDIF
*                  SET RETURN CODE TO FOUR
*                ENDIF
*              ENDIF
*            ENDIF
*          ELSE - RETURN CODE IS NOT ZERO
*            SET UP FOR MESSAGE # 307
*            DO DOMMESG
*            DO DOMDALM
*            IF MACRO INITIATED THEN
*              POST ECB WITH RETURN CODE OF 16 AND MSG # 307
*            ENDIF
*            SET OFF EXECUTING FLAGS
*            IF MOTOR OPERATED SWITCH THEN
*              SET OFF EXECUTING FLAG IN SECOND ITEM
*            ENDIF
*            SET RETURN CODE TO FOUR
*          ENDIF
*        OR ELSE
*          POINT TO NEXT ENTRY IN DCT INDEX TABLE
*        END LOOP
*          ERROR EXIT TO ELEV
*        END SEARCH
*      ENDIF
*      EXIT WITH RETURN CODE SET AS ABOVE
*
*      ERROR ENTER ELEV  -PDC MESSAGE-NO CONTROL IN PROGRESS
*        SET RETURN CODE TO 11
*      ERROR ENTER TWEN  -BAD RETURN CODE FROM DOMCDC11
*      ERROR RETURN
*      EXIT WITH RETURN CODE SET AS ABOVE
*  ENDSEGMENT DOMCDC08
*
*  DOMMESG SUBROUTINE SEGMENT
*    GET MESSAGE USING MESSAGE MACRO
*  ENDSEGMENT DOMMESG
*
*  DOMDALM SUBROUTINE SEGMENT
*    ISSUE OR DELETE ALARM USING DOMCALRM MACRO
*  ENDSEGMENT DOMDALM
*
```

```
MEMBER NAME  DOMCDC08
* DOMWORD SUBROUTINE SEGMENT
*    STRTSRCH UNTIL ALL STATUS GROUPS CHECKED DO
*    EXITIF STATUS GROUP TO WHICH ITEM BELONGS IS FOUND
*      FIND MATCH ON ADDR OF ITEM IN GROUP ADJUSTING MASK
*MOS2ND-LABEL
*      CHANGE BIT IN GROUP WORD ACCORDING TO MASK
*      IF MOS THEN
*        BRANCH TO MOS2ND TO DO 2ND ITEM IN PAIR
*      ENDIF
*    ORELSE
*      POINT TO NEXT STATUS GROUP
*    ENDLOOP
*    ENDSRCH
* ENDSEGMENT DOMWORD
*
* UN#TAG SUBROUITNE SEGMENT
*    IF TAG REPLY THEN
*      SET ON TAG FLAG
*      ISSUE ENTITY USING DISPENT MACRO
*    ELSE
*      IF UNTAG REPLY THEN
*        SET OFF TAG FLAG
*        DELETE ENTITY USING DISPENT MACRO
*      ENDIF
*    ENDIF
* ENDSEGMENT UN#TAG
```

```
MEMBER NAME  DOMCDC09
 * DOMCDC09 MAIN SEGMENT
 *   CASE ENTRY -CASE ID
 *     CASE 24  -CHANGE OF STATUS RECEIVED
 *        POINT TO PREVIOUS ITEM
 *        STRTSRCH FOR NUMBER OF DCT INDEX ENTRIES
 *        EXITIF MATCH FOUND ON ITEM RECEIVED OR
 *        EXITIF MATCH FOUND ON PREVIOUS ITEM
 *          IF MOS THEN
 *            SET OFF EXECUTING FLAG IN ITEM
 *            IF COS RECEIVED INDICATES COMMAND COMPLETE THEN
 *              SET RETURN CODE TO 8
 *              BRANCH TO MOSOUT
 *            ELSE
 *              POINT TO FIRST ITEM IN PAIR
 *            ENDIF
 *          ENDIF
 *          IF MOS THEN
 *            IF EXECUTING FLAG IS ON IN 1ST OR 2ND ITEM THEN
 *              SET IT OFF
 *            ENDIF
 *          ENDIF
 *          DELETE TIME-OUT USING PTIME MACRO
 *          SET UP FOR MSG # 314
 *          DO DOMMESG
 *          IF ACTION IS OPEN/TRIP, CLOSE, MANUAL OR AUTOMATIC THEN
 *            SET UP ENTITY NAME
 *            ISSUE OR DELETE ENTITY USING DISPENT MACRO BASED ON STATUS
 *             FLAG IN ITEM
 *          ENDIF
 *          IF ALARM IS OUTSTANDING THEN
 *            DO DOMDALM -TO DELETE ALARM
 *          ENDIF
 *          IF MACRO INITIATED THEN
 *            POST ECB WITH RETURN CODE OF ZERO AND MSG # 314
 *          ENDIF
 *          SET OFF EXECUTING FLAGS IN DCT AND ITEM
 *          ZERO RETURN CODE
 *        ORELSE
 *          POINT TO NEXT ENTRY IN DCT INDEX TABLE
 *        ENDLOOP
 *          ERROR EXIT TO TWEL
 *        ENDSRCH
 *        IF WALLBOARD FLAG IS ON THEN
 *          GET AREA FOR PARAMETERS USING GETWA MACRO
 *          IF RETURN CODE IS ZERO THEN
 *            BUILD PARAMETER LIST
 *            PATCH WALLBOARD PROCESSOR (DOMCWBPR)
 *            IF PATCH MACRO FAILS THEN
 *              FREE AREA USING FREEWA MACRO
 *            ENDIF
 *          ENDIF
 *        ENDIF
 *MOSOUT -LABEL
 *        SET ON EXECUTED FLAG
 *        CALL DOMCDC11 - TO LOG CHANGE OF STATUS
 *        SET OFF EXECUTED FLAG
 *
 *     CASE 28 -CANCEL REQUEST OR DISPLAY CHANGE
```

```
MEMBER NAME  DOMCDC09
 *       SET UP FOR MSG # 321
 *       DO DOMMESG
 *       SET OFF PDC FLAGS IN ITEM
 *       IF MOS THEN
 *         SET OFF EXECUTING FLAG IN SECOND ITEM
 *       ENDIF
 *       IF ARM OR SELECT TIME-OUT OUTSTANDING THEN
 *         DELETE TIME-OUT USING PTIME MACRO
 *       ELSE
 *         IF EXECUTE TIME-OUT OUTSTANDING THEN
 *           DELETE TIME-OUT USING PTIME MACRO
 *         ENDIF
 *       ENDIF
 *       ZERO RETURN CODE
 *
 *     CASE 32 - EXECUTE TIME-OUT
 *       IF MATCH ON ID FOUND
 *         IF CHANGE OF STATUS FLAG OFF THEN
 *           SET ON EXECUTION TIME-OUT FLAG
 *         ELSE
 *           IF RAISE, LOWER, TAG OR UNTAG THEN
 *             SET ON EXECUTION TIME-OUT FLAG
 *           ELSE
 *             SET ON VERIFY TIME-OUT FLAG
 *           ENDIF
 *         ENDIF
 *         CALL DOMCDC05
 *       ENDIF
 *       SET RETURN CODE TO EIGHT
 *
 *     CASE 36 - ARM TIME-OUT
 *       IF MATCH ON ID FOUND THEN
 *         IF DISPLAY INITIATED AND
 *         IF DEVICE IS NOT ARMED THEN
 *           SET UP FOR MESSAGE # 311
 *         ELSE
 *           SET UP FOR MESSAGE # 305
 *         ENDIF
 *         DO DOMMESG
 *         ZERO RETURN CODE
 *       ELSE
 *         SET RETURN CODE TO EIGHT
 *       ENDIF
 *
 *   ENDCASE
 *   EXIT WITH RETURN CODE SET AS ABOVE
 *
 *   ERROR ENTER TWEL
 *     SET ERROR CODE TO 12
 *   ERROR ENTER TWEN
 *   ERROR RETURN
 *   EXIT WITH RETURN CODE SET AS ABOVE
 * ENDSEGMENT DOMCDC09
 *
 * DOMDALM SUBROUTINE SEGMENT
 *   ISSUE OR DELETE ALARM USING DOMCALRM MACRO
 * ENDSEGMENT DOMDALM
 *
```

```
MEMBER NAME  DOMCDC09
 * DOMMESG SUBROUTINE SEGMENT
 *    GET MESSAGE USING MESSAGE MACRO
 * ENDSEGMENT DOMMESG
 *
```

```
MEMBER NAME  DOMCDC10
 * DOMCDC10 MAIN SEGMENT
 *   GET AREA FOR PARAMETER LIST USING GETWA MACRO
 *   ERROR EXIT TO ERR1 IF RETURN CODE IS NOT ZERO
 *   MOVE MACRO PARAMETERS TO AREA GOTTEN
 *   ZERO ECBS
 *   PATCH DOMCDC04 WITH PARAMETER LIST
 *   WAIT ON ECB
 *   IF PATCH IS SUCCESSFUL
 *     SET RETURN CODE TO ZERO
 *   ELSE
 *     SET RETURN CODE TO FOUR
 *   ENDIF
 *   EXIT WITH RETURN CODE SET AS AOVE
 *
 *   ERROR ENTER ERR1
 *     SET RETURN CODE TO EIGHT
 *   EXIT WITH RETURN CODE
 * ENDSEGMENT DOMCDC10
```

```
MEMBER NAME  DOMCDC11
 * DOMCDC11 MAIN SEGMENT
 *    ESTABLISH ADDRESSABILITY
 *    IF DEVICE HAS ALREADY EXECUTED THEN
 *      DO DOMLOG
 *      IF MOS TYPE DEVICE THEN
 *        POINT TO SECOND ITEM IN PAIR
 *        DO DOMLOG
 *      ENDIF
 *    ELSE
 *      IF TAG OPTION SELECTED THEN
 *        ERROR EXIT TO SEVE IF DEVICE IS ALREADY TAGGED
 *      ELSE
 *        IF UNTAG OPTION SELECTED THEN
 *          ERROR EXIT TO SEVE IF DEVICE IS NOT TAGGED
 *        ELSE
 *          ERROR EXIT TO NINE IF DEVICE IS TAGGED
 *          IF OPEN ACTION SELECTED AND
 *          IF DEVICE IS A SWITCH, BREAKER, OR MOS THEN
 *            ERROR EXIT TO SEVE IF DEVICE IS ALAREAY OPEN
 *          ELSE
 *            IF CLOSE ACTION SELECTED AND
 *            IF DEVICE IS A SWITCH, BREAKER, OR MOS THEN
 *              ERROR EXIT TO SEVE IF DEVICE IS ALREADY CLOSED
 *            ELSE
 *              IF RAISE ACTION SELECTED AND
 *              IF DEVICE IS A TCT 1 OR A TCT3 THEN
 *                ERROR EXIT TO THRE IF TCT 3 DEVICE IN AUTOMATIC STATE
 *              ELSE
 *                IF LOWER ACTION SELECTED AND
 *                IF DEVICE IS TCT 1 OR TCT 3 THEN
 *                  ERROR EXIT TO THRE IF TCT 3 IN AUTOMATIC STATE
 *                ELSE
 *                  IF MANUAL ACTION SELECTED AND
 *                  IF DEVICE IS TCT 2 OR TCT 3 THEN
 *                    ERROR EXIT TO SEVE IF DEVICE ALREADY MANUAL
 *                  ELSE
 *                    IF AUTOMATIC ACTION SELECTED AND
 *                    IF DEVICE IS TCT 2 OR TCT 3 THEN
 *                      ERROR EXIT TO SEVE IF DEVICE ALREADY AUTOMATIC
 *                    ELSE
 *                      ERROR EXIT TO SIX - INVALID COMMAND
 *                    ENDIF
 *                  ENDIF
 *                ENDIF
 *              ENDIF
 *            ENDIF
 *          ENDIF
 *        ENDIF
 *      ENDIF
 *      ERROR EXIT TO SEVT IF DEVICE IS OUT OF SERVICE
 *      ERROR EXIT TO NINT IF DEVICE IS NOT CONTROLLABLE
 *      IF DEVICE IS A MANUAL DEVICE THEN
 *        IF ACTION IS TAG OR UNTAG THEN
 *          FLIP TAG FLAG IN ITEM
 *          IF DEVICE IS MOS OR TCT 3 THEN
 *            POINT TO SECOND ITEM
 *            FLIP TAG FLAG IN SECOND ITEM
 *          ENDIF
```

```
MEMBER NAME  DOMCDC11
 *          BUILD TAG NAME FOR ENTITY
 *          ISSUE DISPENT MACRO TO ADD OR DELETE ENTITY
 *        ELSE
 *          IF MOS DEVICE THEN
 *            POINT TO SECOND ITEM IN PAIR
 *            IF OPEN ACTION SELECTED
 *              SET ON STATUS BIT IN FIRST ITEM
 *              SET OFF STATUS BIT IN SECOND ITEM
 *            ELSE
 *              IF CLOSE ACTION SELECTED
 *                SET OFF STATUS BIT IN FIRST ITEM
 *                SET ON STATUS BIT IN SECOND ITEM
 *              ENDIF
 *            ENDIF
 *            SET UP NAME FOR ENTITY
 *            ISSUE OR DELETE ENTITY USING DISPENT MACRO
 *            BRANCH TO MOSEND
 *          ENDIF
 *          IF ACTION IS OPEN, TRIP, OR CLOSE THEN
 *            FLIP STATUS BIT IN ITEM
 *            BUILD ENTITY NAME
 *            ISSUE OR DELETE ENTITY USING DISPENT MACRO
 *          ENDIF
 *MOSSEND   SET ON EXECUTED FLAG IN DCT
 *          IF ALARM OUTSTANDING THEN
 *            DO DOMDALM
 *          ENDIF
 *          DO DOMLOG
 *          IF MOS DEVICE THEN
 *            POINT TO SECOND ITEM
 *            DO DOMLOG
 *          ENDIF
 *          IF WALLBOARD FLAG IS ON THEN
 *            GET AREA FOR PARAMETERS USING GETWA MACRO
 *            IF MACRO WORKED THEN
 *              SET UP PARAMETERS IN AREA GOTTEN
 *              PATCH WALLBOARD PROCESSOR - DOMCWBPR
 *              IF PATCH FAILS THEN
 *                FREE PARAMETER AREA USING FREEWA MACRO
 *              ENDIF
 *            ENDIF
 *          ENDIF
 *        ENDIF
 *      ELSE
 *        START PTIME FOR ARM TIME-OUT USING PTIME MACRO
 *      ENDIF
 *      SET ON ARMED INDICATOR IN DCT
 *    ENDIF
 *    EXIT WITH RETURN CODE OF ZERO
 *
 *    ERROR ENTER ONE   - NO GETWA FOR STATUS LOG
 *      SET RETURN CODE TO 21
 *    ERROR ENTER THRE  - TCT 3 IS AUTOMATIC-ACTION NOT MANUAL
 *      SET RETURN CODE TO 20
 *    ERROR ENTER SIX   - INVALID COMMAND
 *      SET RETURN CODE TO 6
 *    ERROR ENTER SEVE  - DEVICE ALREADY IN REQUESTED STATE
 *      SET RETURN CODE TO 13
```

```
MEMBER NAME  DOMCDC11
 *    ERROR ENTER NINE   - DEVICE TAGGED-ACTION NOT UNTAG
 *      SET RETURN CODE TO 9
 *    ERROR ENTER TEN   - INVALID COMMAND FOR TCT
 *      SET RETURN CODE TO 10
 *    ERROR ENTER SEVT   - DEVICE OUT OF SERVICE
 *      SET RETURN CODE TO 17
 *    ERROR ENTER NINT   - DEVICE NOT CONTROLLABLE
 *      SET RETURN CODE TO 19
 *    ERROR RETURN
 *    EXIT WITH RETURN CODE SET
 *  ENDSEGMENT DOMCDC11
 *
 *  DOMDALM SUBROUTINE SEGMENT
 *    USE DOMCALRM MACRO TO DELETE ALARM
 *  ENDSEG DOMDALM
 *
 *  DOMLOG SUBROUTINE SEGMENT
 *    GET AREA FOR PARAMETERS USING GETWA MACRO
 *    BUILD PARAMETER LIST
 *    USE PTIME MACRO TO GET DATE AND TIME FOR LIST
 *    PATCH DOMCSLOG WITH PARAMETER LIST
 *  ENDSEGMENT DOMLOG
```

```
MEMBER NAME  DOMCDIG1
 * DOMCDIG1 MAIN SEGMENT
 *
 *         READ THE OPTIONS AND THE LINE TO BE PRINTED
 *         IF THERE IS ANY ERROR OR INCONSISTANCY THEN
 *           DISPLAY MESSAGE INDICATING THE INCONSISTANCY
 *           EXIT
 *         ENDIF
 *         IF ROUTING CODE WAS SPECIFIED THEN
 *           WRITE TO THIS ROUTING CODE
 *         ELSE
 *           IF GENERAL TYPER WAS SPECIFIED THEN
 *             WRITE TO THE GENERAL EVENTS AND ALARMS TYPER
 *           ELSE
 *             IF ACCESS AREA NOT SPECIFIED
 *               PICK-UP THIS DISPLAY ACCESS AREA
 *             ENDIF
 *             IF FUNCTION CODE NOT SPECIFIED
 *               PICK-UP THIS DISPLAY FUNCTION CODE
 *             ENDIF
 *             IF EVENT TYPER WAS SPECIFIED THEN
 *               WRITE TO THIS ACCESS AREA/FUNCTION CODE EVENT TYPER
 *             ELSE
 *               WRITE TO THIS ACCESS AREA/FUNCTION CODE ALARM TYPER
 *             ENDIF
 *           ENDIF
 *         ENDIF
 *         WRITE BACK TO THE SCREEN THE OPTIONS USED
 * ENDPROG
```

```
MEMBER NAME  DOMCEVD1
*     DOMCEVD1  --  PROCESS EVENTS RETRIEVAL REQUEST
*       SAVE OLD ECE ADDRESS
*       GET AREA FOR NEW ECE
*       PUT ECE POINTER IN DCE
*       GET AREA FOR NEW PAGE SAVE AREA
*       PUT NEW PAGE SAVE AREA ADDRESS IN NEW ECE
*       IF  EVENTS DISPLAY NOT CURRENTLY ON CRT, THEN
*         ASSUME DEFAULT PARAMETERS = ALL BLANKS
*       ELSE
*         READ OPERATOR'S CHOICES FROM THE MENU DISPLAY
*       ENDIF
*       IF ALL INPUT PARAMETERS VALID, THEN
*         FILL ECE FIELDS WITH INPUT PARAMETERS
*         FORMAT DISPLAY HEADINGS INTO NEW PAGE SAVE AREA FOR DISPLAY
*         IF EVENTS LOG DISPLAY NOT CURRENTLY ON CRT, THEN
*           ISSUE A DISPLAY REQUEST FOR EVENTS LOG DISPLAY
*         ENDIF
*         PATCH EVENTS LOG DISPLAY PROCESSING PROGRAM
*         FREE OLD ECE AND OLD PAGE SAVE AREA
*       ELSE
*         FREE NEW ECE
*         FREE NEW PAGE SAVE AREA
*         DISPLAY ERROR MESSAGE
*       ENDIF
*       IF EVENTS DISPLAY CURRENTLY IN CYCLIC MODE, THEN
*         DELETE PTIME OF DISPLAY PROGRAM
*         PURGE WORK QUEUE OF ANY QUEUED PTIME ELEMENTS
*       ENDIF
*       EXIT PROGRAM
*     EXIT DOMCEVD1
```

```
MEMBER NAME  DOMCEVD2
*        DOMCEVD2 MAIN SEGMENT
*          IF THIS IS NOT INITIAL, REFRESH OR PTIME CASE AND
*          IF PTIME IS ACTIVE THEN
*            CANCEL PTIME
*            PURGE WORK QUEUE OF EXISTING PTIME ELEMENTS
*          ENDIF
*          IF THIS IS A CHANGE CASE THEN
*            FREE ECE AREA
*            FREE PAGE SAVE AREA
*          ELSE
*            ENQUE THE EVENT FILE
*            CASENTRY PATCHID
*              CASE 1 - INITIAL OR PTIME
*                INCLUDE INITIAL SEGMENT
*              CASE 2 - FORWARD
*                SET FORWARD INDICATOR IN ECE
*                CALL CHANGE PAGE ROUTINE
*              CASE 3-  BACKWARD
*                SET BACKWARD INDICATOR IN ECE
*                CALL CHANGE PAGE ROUTINE
*              CASE 4 - REFRESH
*                NO ADDITIONAL CODE
*            ENDCASE
*            INCLUDE DISPLAY SEGMENT
*            DEQUEUE ON EVENT FILE
*          ENDIF
*          EXIT DOMCEVD2
*        ENDSEGMENT DOMCEVD2
*
*        INITIAL SEGMENT
*          FILL EVENTS DISPLAY HEADING IF NOT PTIME CASE
*          IF TIME WAS SPECIFIED THEN
*            READ THE FILE USING DATE AND TIME AS KEY
*            IF RECORD NOT FOUND THEN
*              PERFORM BINARY SEARCH TO FIND CLOSEST RECORD
*            ENDIF
*            PUT RECORD SEQUENCE NUMBER IN ECE
*            SET FORWARD INDICATOR IN ECE
*            CALL CHANGE PAGE ROUTINE
*          ELSE
*            IF PTIME IS NOT ACTIVE THEN
*              ISSUE PTIME AT SYSGENED RATE
*            ENDIF
*            PUT NEWEST RECORD SEQUENCE NUMBER IN ECE
*            SET BACKWARD INDICATOR IN ECE
*            CALL CHANGE PAGE ROUTINE
*          ENDIF
*        ENDSEGMENT INITIAL
*
*        DISPLAY SEGMENT
*          IF THERE ARE EVENTS TO BE DISPLAYED THEN
*            IF BACKWARD DIRECTION WAS SPECIFIED THEN
*              INVERT RECORD NUMBER LIST
*            ENDIF
*            READ, PUT TO THE SCREEN AND SAVE RELEVANT INFORMATION IN
*            THE PAGE SAVE AREA OF THE RECORDS WHOSE NUMBERS ARE IN THE
*            RECORD NUMBER LIST
*          ELSE
```

```
MEMBER NAME  DOMCEVD2
*             DISPLAY DPP351 TO SCRATCH PAD
*           ENDIF
*         ENDSEGMENT DISPLAY
*
*         CHANGE PAGE SUBROUTINE SEGMENT
*           IF FORWARD DIRECTION SPECIFIED THEN
*             PICK-UP OLD PAGE NEWEST EVENT SEQUENCE NUMBER FROM ECE
*           ELSE
*             PICK-UP OLD PAGE OLDEST EVENT SEQUENCE NUMBER FROM ECE
*           ENDIF
*           UNTIL FILL PAGE OR
*           UNTIL CURRENT POSITION REACHED,DO
*             IF FORWARD INDICATOR IS ON THEN
*               ADD 1 TO RECORD SEQUENCE NUMBER
*             ELSE
*               SUBTRACT 1 FROM RECORD SEQUENCE NUMBER
*             ENDIF
*             CALL READCOMPARE ROUTINE
*             ENDIF
*           ENDDO
*           UPDATE OLDEST AND NEWEST RECORD SEQUENCE NUMBER ON PAGE IN
*                   ECE
*         ENDSEGMENT CHANGE PAGE
*
*         READCOMPARE SEGMENT
*           IF NEEDED INDEX RECORD IS NOT IN STORAGE THEN
*             READ IT
*           ENDIF
*           IF ACCESS AREA, FUNCTION CODE AND TYPE MATCH REQUEST THEN
*             PUT THE RECORD NUMBER IN THE RECORD NUMBER LIST
*           ENDIF
*         ENDSEGMENT READCOMPARE
```

```
MEMBER NAME  DOMCEVD3
*        DOMCEVD3 MAIN SEGMENT
*          FIND ECE FROM POINTER IN DCE
*          LINK TO DOMCEVD2 FOR EVENTS RETRIEVAL
*          UPDATE DISPLAY WITH CURRENT EVENTS
*        ENDSEGMENT DOMCEVD3
```

```
MEMBER NAME  DOMCEVD4
*        IF NOW PTIMING, THEN
*          DELETE THE PTIME
*          PURGE THE WORK QUEUE
*        ENDIF
*        ERREXIT M352 IF EVENTS RETRIEVED VIA FUNCTION 9
*        IF EVENT NOT FROM FUNCTION 0, AND
*        IF EVENT NOT FROM PRIME FA, AND
*        IF EVENT NOT FROM SECONDARY FA, AND
*        IF EVENT NOT FROM TERTIARY FA, THEN
*          ERREXIT M352, INVALID AA/FA
*        ENDIF
*        IF EVENT NOT FROM ACCESS 0, AND
*        IF EVENT NOT FROM PRIME AA, AND
*        IF EVENT NOT FROM SECONDARY AA, AND
*        IF EVENT NOT FROM TERTIARY AA, THEN
*          ERREXIT M352, INVALID AA/FA
*        ENDIF
*        READ COMMENT FROM SCREEN
*        ERREXIT M384 IF READ ERROR
*        UNTIL ALL BYTES OF INPUT PROCESSED, DO
*          IF UNDERSCORE, THEN
*            SUBSTITUTE A BLANK
*          ENDIF
*        ENDDO
*        VERIFY EVENT NUMBER IS VALID AND ON PAGE DISPLAYED
*        STRTSRCH PROCESS ALL EVENTS/PAGE
*        EXITIF IF EVENT FOUND, THEN
*          SAVE DYNAMIC ID
*        ORELSE
*          NEXT EVENT NUMBER
*        ENDLOOP
*          ERREXIT M353, EVENT # NOT FOUND
*        ENDSRCH
*        OBTAIN ACCESS AND FUNCTION NAMES
*        LOCK EVENT FILE
*        READ EVENT RECORD FROM DISK
*        DO SEGMENT WAITONIO
*        IF NO I/O ERROR, THEN
*          VERIFY THAT THE RECORD HAS NOT BEEN OVERLAYED
*          ERREXIT M354 IF KEYS DON'T MATCH
*          ERREXIT M354 IF ACCESS AREAS DON'T MATCH
*          ERREXIT M354 IF FUNCTION AREAS DON'T MATCH
*          MOVE COMMENT TO EVENT RECORD
*          WRITE EVENT RECORD BACK TO FILE
*          DO SEGMENT WAITONIO
*        ENDIF
*        UNLOCK EVENTS FILE
*        IF NO I/O ERROR, THEN
*          OD SEGMENT PRINTIT
*        ELSE ERROR
*          WRITE ERROR MESSAGE TO TYPERS
*        ENDIF
*        IF NO I/O ERRORS, THEN
*          WRITE COMMENT BACK TO SCREEN
*        ELSE
*          INDICATE UNABLE TO COMPLY, RETRY
*        ENDIF
*        WRITE RESULTS TO SCRATCH PAD ZONE
```

```
MEMBER NAME  DOMCEVD4
*        IF UPDATE SUCCESSFUL, THEN
*          REINITIALIZE COMMENT LINE FIELD
*        ENDIF
*        EXIT PROGRAM
*
*        BGNSEG WAITONIO    WAIT AND VERIFY I/O
*          IF I/O ERROR, THEN
*            INDICATE I/O ERROR
*          ENDIF
*        ENDSEG WAITONIO
*
*        BGNSEG   PRINTIT    ROUTE COMMENT TO TYPERS
*          STRTSRCH ALL CAAATBL ACCESS AREAS
*          EXITIF IF ACCESS AREA MATCH, THEN
*            STRTSRCH FOR FUNCTION
*            EXITIF IF MATCH ON FUNCTION, THEN
*              SAVE ROUTING CODE
*            ORELSE
*              NEXT FUNCTION ITEM
*            ENDLOOP
*              DEFAULT TO ZERO ROUTING CODE
*            ENDSRCH
*          ORELSE
*            CHECK NEXT ACCESS ITEM
*          ENDLOOP
*            SAVE DEFAULT ROUTING CODE
*          ENDSRCH
*          IF I/O ERROR, THEN
*            ROUTE MESSAGE 367 TO TYPER(S)
*          ELSE
*            ROUTE MESSAGE 357, TEXT OF EVENT, TO TYPER(S)
*            ROUTE MESSAGE 358, COMMENT OF EVENT, TO TYPER(S)
*          ENDIF
*        ENDSEG PRINTIT
*
*        ERREXIT ROUTINE CODE - INDICATE ONE OF THE FOLLOWING MESSAGES
*        M384 - UNABLE TO ANSWER REQUEST
*        M369 - BAD EVENT #, BLENK/NONNUM
*        M352 - INVALID ACCESS/FUNCTION
*        M353 - EVENT NOT ON DISPLAY
*        M354 - EVENT NOT IN FILE
*        UNLOCK EVENTS FILE
*        DISPLAY ERROR MESSAGE AND EXIT PROGRAM
```

```
MEMBER NAME  DOMCEVNT
*    DOMCEVNT    -    SCEVENT MACRO PROCESSOR
*       IF LENGTH OF TEXT SUPPLIED > ZERO, THEN
*          IF LENGTH OF TEXT SUPPLIED > MAXIMUM, THEN
*             DEFAULT TO MAXIMUM LENGTH
*             SET RETURN CODE EQ MAX LENGTH USED
*          ENDIF
*          OBTAIN BUFFER FOR EVENTS LOGGER PARAMETER LIST
*          OBTAIN SRTOS TIME, TIME OF THE EVENT
*          MOVE TEXT TO BUFFER
*          IF  TYPE CODE IS VALID, THEN
*             MOVE TYPE CODE TO BUFFER
*             IF ACCESS IS ZERO, THEN
*                USE BLANK ACCESS NAME
*             ELSE
*                IF ACCESS NAME SPECIFIED, THEN
*                   OBTAIN ACCESS ID
*                ELSE
*                   OBTAIN ACCESS NAME
*                ENDIF
*             ENDIF
*             IF  FUNCTION IS ZERO, THEN
*                USE BLANK FUNCTION NAME
*             ELSE
*                IF  FUNCTION ID SPECIFIED, THEN
*                   OBTAIN FUNCTION NAME
*                ELSE
*                   OBTAIN FUNCTION ID
*                ENDIF
*             ENDIF
*             PATCH DOMCEVT1 PASSING THE COMPLETED PARAMETER LIST
*             SET RETURN CODE EQ 0
*          ELSE
*             SET RETURN CODE EQ INVALID TYPE
*          ENDIF
*       ELSE
*          SET RETURN CODE EQ INVALID TEXT LENGTH
*       ENDIF
*       RETURN TO CALLER
```

```
MEMBER NAME  DOMCEVT1
*        DOMCEVT1 MAIN SEGMENT
*          LOCK THE EVENT FILE FOR UPDATE
*          REPLACE EMS STANDARD TYPE CODE BY USER DEFINED TYPE CODE
*          MOVE EVENT DATA TO OUTPUT AREA
*          OBTAIN DATE AND TIME FOR EVENT KEY
*          IF THE REQUIRED INDEX RECORD IS NOT IN STORAGE THEN
*            WRITE THE IN STORAGE ONE TO THE FILE
*            READ THE REQUIRED INDEX RECORD
*          ENDIF
*          UPDATE CORRESPONDING ACCESS AREA, FUNCTION CODE AND TYPE
*          FIELDS IN THE INDEX RECORD
*          WRITE THE EVENT RECORD
*          DO TYPERS ROUTINE
*          FREE PASSED PARAMETER AREA
*          DEQUEUE ON EVENT FILE
*          EXIT DOMCEVT1
*        END   DOMCEVT1
*
*        TYPERS SEGMENT - WRITE THE EVENT TO TYPERS
*          FORMAT THE EVENT PROPERLY TO BE WRITTEN
*          FIND THE ROUTING CODE OF THE PRINTER FOR THAT ACCESS AREA
*          WRITE EVENT TO GENERAL AND ACCESS AREA TYPER(S)
*        ENDSEGMENT TYPERS
```

```
MEMBER NAME  DOMCEVT5
* DOMCEVT5 MAIN SEGMENT
*       OBTAIN CURRENT SRTOS DATE AND TIME FOR EVENTS KEY
*       OBTAIN USER SPCEIFIED ALARM TYPE CODE
*       LOCK THE EVENTS FILE FOR UPDATE
*       DO UNTIL ALL BUFFERS ARE PROCESSED
*         EXTRACT TIME OF EVENT FOR INCLUSION IN EVENT RECORD
*         DO NUMBER OF ALARMS PASSED IN THE AREA
*           INSERT TIME OF EVENT INTO EVENT RECORD
*           SAVE ECTABLE FIELDS
*           MOVE TEXT TO OUTPUT AREAS
*           MOVE AAID, FCID AND TYPE TO THE INDEX AND EVENT RECORD
*           IF THE INDEX RECORD IS FULL
*             WRITE IT TO THE FILE
*             READ THE NEXT ONE
*           ENDIF
*           WRITE THE EVENT RECORD TO THE FILE
*           UPDATE ECTABLE FIELDS
*           IF AA/FC DIFFERS FROM PAST LOOP
*             MOVE NEW AA/FC TO THE TYPER AREA
*             FIND NEW ROUTING CODE FOR THIS AA/FC
*           ENDIF
*           IF AN I/O ERROR OCCURRED WHEN WRITING TO THE FILE
*             RESTORE ECTABLE FIELDS
*             WRITE ERROR MESSAGE TO THE GENERAL AND AA/FC TYPERS
*           ENDIF
*           WRITE EVENT TO THE GENERAL AND AA/FC TYPERS
*           ADVANCE POINTER TO NEXT ALARM IN THE PASSED AREA
*         ENDDO
*         FREE THE PASSED AREA
*       ENDDO
* END PROGRAM DOMCEVT5
```

```
MEMBER NAME  DOMCFGDA
 * DOMCFGDA COPY SEGMENT  - PCC I DISPLAY
 *   CALCULATE SIZE OF S7COMM TABLE
 *   GET AREA FOR S7COMM USING GETWA MACRO
 *   ERREXIT TO ERR1 IF RETURN CODE IS NOT ZERO
 *   GET S7COMM ARRAY USING GETARRAY MACRO
 *   ERREXIT TO ERR2 IF RETURN CODE IS NOT ZERO
 *   IF NOT ORIGINAL ENTRY INTO PROGRAM THEN
 *     IF DISPLAY PCCI IS NOT UP THEN
 *       ADJUST INDICATORS
 *       SET DO NOT RELEASE FLAG ON
 *     ELSE
 *       SET SAME PAGE (REFRESH) FLAG
 *     ENDIF
 *   ELSE
 *     SET INDICATORS
 *   ENDIF
 *   CALL DOMCBLD1  - PCCI DISPLAY BUILDER
 *   ERREXIT TO ERR3 IF RETURN CODE IS NOT ZERO
 * ENDSEGMENT DOMCFGDA
```

```
MEMBER NAME  DOMCFGDB
 * DOMCFGDB COPY SEGMENT  - PCC II DISPLAY
 *    IF DISPLAY PCC I IS NOT UP THEN
 *      ADJUST INDICATORS
 *    ELSE
 *      ADJUST INDICATORS
 *      CONVERT S/7 ID FROM PARAMETER LIST AND STORE IN CCE
 *    ENDIF
 *    SET DO NOT RELEASE FLAG ON
 *    GET ADDRESS TO S/7 CONTROL TABLE FROM EMSCVT
 *    SEARCH FOR MATCH ON S/7 ID (UNTIL-DO LOOP)
 *    EXITIF MATCH FOUND
 *    ORELSE
 *      POINT TO NEXT ENTRY IN S/7 CONTROL TABLE
 *    ENDLOOP
 *    END SEARCH
 *    GET ADDRESS OF RCB LIST FROM S/7 CONTROL TABLE
 *    ERROR EXIT TO ERR9 IS NO RCBS DEFINED FOR SYSTEM/7
 *    DETERMINE NUMBER OF PAGES AND STORE IN CCE
 *    CALL DOMCBLD2  - PCC II BUILDER
 *    ERREXIT IF RETURN CODE IS NOT ZERO TO ERR6
 * ENDSEGMENT DOMCFGDB
```

```
MEMBER NAME  DOMCFGDC
 * DOMCFGDC COPY SEGMENT  - PCC III DISPLAY
 *    ADJUST INDICATORS IN CCE
 *    SET ON DO NOT RELEASE FLAG
 *    GET ADDRESS OF S/7 CONTROL TABLE FROM CCE
 *    GET ADDRESS OF RCB LIST FROM S7CT
 *    SEARCH RCBS IN LIST FOR MATCH TO RCB NAME IN PARAMETER LIST
 *    EXITIF MATCH FOUND
 *      STORE ADDRESS OF RCB IN CCE
 *    ORELSE
 *      POINT TO NEXT RCB IN RCB LIST
 *    ENDLOOP
 *    END SEARCH
 *    CALCULATE NUMBER OF PAGES:
 *    IF ANY ANALOG POINTS FOR TERMINAL THEN
 *      CALCULATE NUMBER OF ANALOG PAGES
 *    ENDIF
 *    IF ANY PULSE COUNTER POINTS FOR TERMINAL THEN
 *      CALCULATE NUMBER OF COUNTER PAGES
 *    ENDIF
 *    IF ANY STATUS GROUPS FOR TERMINAL THEN
 *      MULTIPLY NUMBER OF STATUS GROUPS BY 16 (NUMBER OF ITEMS IN GROUP)
 *    ENDIF
 *    ERREXIT IF THERE ARE NO POINTS FOR TERMINAL TO ERR4
 *    STORE NUMBER OF PAGES IN CCE
 *    IF THERE IS ANALOG DATA THEN
 *      GET ADDRESS OF START OF ANALOG DATA FROM RCB
 *      MOVE TYPE OF DATA INDICATOR TO CCE
 *    ELSE
 *      IF THERE IS COUNTER DATA THEN
 *        GET ADDRESS OF START OF COUNTER DATA FROM RCB
 *        MOVE TYPE OF DATA INDICATOR TO CCE
 *      ELSE
 *        GET ADDRESS OF START OF STATUS DATA FROM RCB
 *        BYPASS GROUP HEADER
 *        MOVE TYPE OF DATA INDICATOR TO CCE
 *      ENDIF
 *    ENDIF
 *    STORE NUMBER OF POINTS IN CCE
 *    ZERO NUMBER OF POINTS TO BYPASS FIELD IN CCE
 *    CALL DOMCBLD3  - PCC III DISPLAY BUILDER
 *    ERREXIT TO ERR7 IF RETURN CODE IS NOT ZERO
 * ENDSEGMENT DOMCFGDC
```

```
MEMBER NAME  DOMCFGDD
 * DOMCFGDD COPY SEGMENT - FORWARD + BACKWARD PAGING + REFRESH - PCC II
 *    GET ADDRESS OF S/7 CONTROL TABLE FROM CCE
 *    IF REFRESH INDICATOR NOT ON THEN
 *      IF PAGE BACKWARD INDICATOR IS ON THEN
 *        SUBTRACT 1 FROM PAGE NUMBER OR WRAPAROUND TO LAST PAGE
 *        STORE PAGE NUMBER IN CCE
 *      ELSE
 *        ADD 1 TO PAGE NUMBER OR WRAPAROUND TO FIRST PAGE
 *        STORE PAGE NUMBER IN CCE
 *      ENDIF
 *    ENDIF
 *    IF PAGE NUMBER IS NOT EQUAL TO 1 THEN
 *      CALCULATE DISPLACEMENT INTO RCB LIST
 *    ELSE
 *      ZERO DISPLACEMENT INDICATOR
 *    ENDIF
 *    GET ADDRESS OF RCB LIST FROM S7CT
 *    ADD DISPLACEMENT TO IT
 *    SET ON SAME DISPLAY INDICATOR
 *    CALL DOMCBLD2
 *    ERREXIT IF RETURN CODE IS NOT ZERO TO ERR6
 * ENDSEGMENT DOMCFGDD
```

```
MEMBER NAME  DOMCFGDE
 * DOMCFGDE COPY SEGMENT -FORWARD,BACKWARD PAGING + REFRESH -PCCIII
 *   IF NOT REFRESH THEN
 *     ESTABLISH PAGE NUMBER DOING WRAPAROUND WHERE NECESSARY
 *   ENDIF
 *   FIND START OF POINTS FOR THE TERMINAL (ANALOG, COUNTER OR STATUS)
 *
 *   UNTIL THE PROPER PAGE IS FOUND DO
 *     IF NOT THE FIRST PAGE OF THE DISPLAY THEN
 *       FIND NUMBER OF PAGES IN TYPE OF POINT
 *       IF PAGE NUMBER IS GREATER THAN TOTAL NUMBER OF PAGES FOR
 *        TYPE THEN
 *         SUBTRACT TOTAL NUMBER OF PAGES FOR TYPE FROM PAGE NUMBER
 *         IF TYPE IS ANALOG THEN
 *           IF THERE IS COUNTER DATA THEN
 *             POINT TO COUNTER DATA
 *           ELSE
 *             POINT TO STATUS DATA
 *           ENDIF
 *         ELSE -IT IS COUNTER DATA
 *           POINT TO STATUS DATA
 *         ENDIF
 *       ELSE
 *         IF PAGE NUMBER IS LESS THAN TOTAL NUMBER OF PAGES FOR TYPE
 *          THEN  (PAGE NUMBER IS IN THIS TYPE OF DATA)
 *           FIND NUMBER OF POINTS TO BYPASS
 *           IF ANALOG DATA THEN
 *             BYPASS NUMBER OF POINTS TO REACH PAGE NUMBER
 *             SET POINTER TO NEXT ANALOG POINT TO DISPLAY
 *           ELSE
 *             IF COUNTER DATA THEN
 *               BYPASS NUMBER OF POINTS TO REACH PAGE NUMBER
 *               SET POINTER TO NEXT COUNTER POINT TO DISPLAY
 *             ELSE - IT IS STATUS DATA
 *               FIND NUMBER OF STATUS GROUPS TO BYPASS
 *               IF ANY PARTIAL GROUPS TO BYPASS THEN
 *                 BYPASS REMAINING POINTS TO REACH PAGE NUMBER
 *                 SET POINTER TO NEXT STATUS POINT TO DISPLAY
 *               ELSE
 *                 BYPASS STATUS GROUP HEADER
 *               ENDIF
 *             ENDIF
 *           ENDIF
 *         ELSE   -PAGE NUMBER IS EQUAL TO TOTAL NUMBER OF PAGES IN TYPE
 *           IF ANALOG DATA THEN
 *             IF ANY COUNTER DATA THEN
 *               POINT TO START OF COUNTER DATA
 *             ELSE
 *               POINT TO START OF STATUS DATA
 *             ENDIF
 *           ELSE  -IT IS COUNTER DATA
 *             POINT TO START OF STATUS DATA
 *           ENDIF
 *         ENDIF
 *       ENDIF
 *     ELSE -LESS THAN ONE PAGE OF POINTS
 *       POINT TO START OF FIRST TYPE OF DATA
 *     ENDIF
 *   ENDDO
```

```
MEMBER NAME  DOMCFGDE
 *    SET ON SAME DISPLAY INDICATOR (CCESDY)
 *    CALL DOMCBLD3
 *    ERREXIT IF THE RETURN CODE IS NOT ZERO TO ERR7
 * ENDSEGMENT DOMCFGDE
```

```
MEMBER NAME  DOMCFGDF
 * DOMCFGDF COPY SEGMENT -DISPLAY CHANGE CLEAN-UP
 *   IF THE DO NOT RELEASE INDICATOR IS OFF THEN
 *     ZERO THE CCE ADDRESS IN THE CEATAB
 *     FREE THE CCE AREA USING THE FREEWA MACRO
 *   ELSE
 *     SET OFF THE DO NOT RELEASE INDICATOR
 *   ENDIF
 * ENDSEGMENT DOMCFGDF
```

```
MEMBER NAME  DOMCFGDG
 * DOMCFGDG COPY SEGMENT
 *    IF PCC III DISPLAY IS UP THEN
 *      IF PATCH ID IS FOR PAGE FORWARD OR
 *      IF PATCH ID IS FOR PAGE BACKWARD THEN
 *        POINT TO PARTIAL SCREEN READ INFORMATION
 *        IF FIRST BYTE IS NOT UNDERLINE CHARACTER THEN
 *          ERROR EXIT TO ERR5 IF BLANKS
 *          ERROR EXIT TO ERR5 IF NOT NUMERIC
 *          CONVERT TO BINARY
 *          ERROR EXIT TO ERR5 IF PAGE REQUESTED GREATER THAN TOTAL
 *           NUMBER OF PAGES OR IF ZERO
 *          STORE PAGE NUMBER IN CCE
 *          SET PATCH ID FOR REFRESH
 *        ENDIF
 *      ENDIF
 *    ENDIF
 * ENDSEGMENT DOMCFGDG
```

```
MEMBER NAME  DOMCFGDH
 * DOMCFGDH COPY SEGMENT -ERROR EXIT ROUTINES
 *    ERRENTER ERR1 -NO GETWA CORE AVAILABLE
 *      SET UP FOR MESSAGE # 333
 *      SET VARIABLE COUNT TO ZERO
 *      DO MESG
 *      DO ZONE
 *
 *    ERRENTER ERR2 -GETARRAY FAILED FOR S7COMM TABLE
 *      SET UP FOR MESSAGE # 410
 *      SET VARIABLE COUNT TO ZERO
 *      DO MESG
 *      DO ZONE
 *
 *    ERRENTER ERR3 -UNABLE TO WRITE PCC I DISPLAY
 *      SET UP FOR MESSAGE # 411
 *      SET VARIABLE COUNT TO ZERO
 *      DO MESG
 *      DO ZONE
 *
 *   ERRENTER ERR5 -INVALID PAGING REQUEST
 *     SET UP FOR MESSAGE 376
 *     SET VARIABLE COUNT TO ZERO
 *     DO MESG
 *     DO ZONE
 *
 *    ERRENTER ERR6 -UNABLE TO WRITE PCC II DISPLAY
 *      SET UP FOR MESSAGE 421
 *      SET VARIABLE COUNT TO ZERO
 *      DO MESG
 *      DO ZONE
 *
 *    ERRENTER ERR7 -UNABLE TO WRITE PCC III DISPLAY
 *      SET UP FOR MESSAGE 450
 *      SET VARIABLE COUNT TO ZERO
 *      DO MESG
 *      DO ZONE
 *
 *    ERRENTER ERR4 -NO POINTS DEFINED FOR TERMINAL
 *      SET UP FOR MESSAGE 347
 *      FIND ADDRESS OF RCB NAME FOR VARIABLE
 *      DO MESG
 *      DO ZONE
 *
 *    ERRENTER ERR9 -NO TERMINALS DEFINED FOR S/7
 *      SET UP FOR MESSAGE 449
 *      SET UP S/7 ID FOR VARIABLE
 *      DO MESG
 *      DO ZONE
 *
 *    ERRETURN
 *    EXIT
 * ENDSEGMENT DOMCFGDH
 *
 * MESG SUBROUTINE SEGMENT
 *    BLANK MESSAGE AREA
 *    IF VARIABLE COUNT IS ZERO THEN
 *      GET MESSAGE USING MESSAGE MACRO
 *    ELSE
```

```
MEMBER NAME  DOMCFGDH
 *      GET MESSAGE WITH VARIABLE USING MESSAGE MACRO
 *    ENDIF
 * ENDSEGMENT MESG
 *
 * ZONE SUBROUTINE SEGMENT
 *    SET UP DISPLAY UNIT ID
 *    WRITE MESSAGE TO SCRATCH PAD ZONE USING DWZONE MACRO
 * ENDSEGMENT ZONE
```

```
MEMBER NAME  DOMCFGD1
 * DOMCFGD1 MAIN SEGMENT
 *    SAVE ADDRESS OF STAE PARAMETER LIST FROM RESOURCE TABLE
 *    MOVE END OF LIST INDICATOR TO STAE LIST
 *    SAVE PATCH ID
 *    DO DOMGCCE
 *    COPY DOMCFGDG - DIRECT PAGING PROCESS
 *    IF PATCH ID IS FOR PAGE FORWARD OR BACKWARD OR FOR REFRESH THEN
 *      SET PROPER INDICATORS AND CASE IDS
 *    ENDIF
 *    CASE ENTRY -CASE ID
 *
 *      CASE 4 - BUILD PCC I DISPLAY
 *        COPY DOMCFGDA
 *
 *      CASE 8 - BUILD PCC II DISPLAY
 *        COPY DOMCFGDB
 *
 *      CASE 12 - BUILD PCC III DISPLAY
 *        COPY DOMCFGDC
 *
 *      CASE 16 - UPDATE PCC II DISPLAY
 *        COPY DOMCFGDD
 *
 *      CASE 20 - UPDATE PCC III DISPLAY
 *        COPY DOMCFGDE
 *
 *      CASE 24 - DISPLAY CHANGE
 *        COPY DOMCFGDF
 *    ENDCASE
 *    EXIT
 *    COPY DOMCFGDH - ERROR EXITS
 *    COPY DOMCFGDI - CONSTANTS
 *    COPY DOMCFGDJ - DSECTS
 * ENDSEGMENT DOMCFGD1
 *
 *
 * DOMGCCE SUBROUTINE SEGMENT
 *    IF PATCH ID IS 32 THEN
 *      SET CASE ID TO 24 FOR REFRESH
 *    ENDIF
 *    ESTABLISH POINTERS TO AID, DCE, AND CEATAB
 *      MOVE INFORMATION ABOUT CEATAB ENTRY TO STAE PARAMETER LIST
 *    IF THERE IS NO PREVIOUS CCE THEN
 *      GET AN AREA FOR CCE USING GETWA MACRO
 *      ERREXIT IF RETURN CODE FROM MACRO IS NOT ZERO TO ERR1
 *      INITIALIZE CCE AREA
 *      IF PATCH ID IS NOT EQUAL TO 4 THEN
 *        SET CASE ID TO 4
 *      ENDIF
 *    ELSE
 *      BLANK MESSAGE AREA
 *      IF PATCH ID IS NOT EQUAL TO 32 THEN
 *        DO ZONE - TO BLANK SCRATCH PAD ZONE
 *      ENDIF
 *    ENDIF
 *    MOVE INFORMATION ABOUT CCE TO STAE LIST
 *    ADJUST END OF LIST INDICATOR IN STAE LIST
 * ENDSEGMENT DOMGCCE
```

```
MEMBER NAME  DOMCFGD2
 *   DOMCFGD2 MAIN TASK
 *       DISPLAY  COMMAND REJECT AND
 *       EXIT IF THIS DISPLAY IS NOT ALLOWED TO ISSUE THIS COMMAND
 *       IF PATCHID LESS OR EQUAL TO 4 THEN (VARY S/7)
 *         VARY THE S/7 OFFLINE, BACK-UP OR PRIMARY (PATCHID 2,3 OR 4)
 *         USING THE VARYS7 MACRO
 *       ELSE
 *         IF PATCHID LESS OR EQUAL TO 8 THEN (VARY TERMINAL)
 *           VARY THE TERMINAL OFFLINE OR ONLINE    (PATCHID 7 OR 8)
 *           USING THE VARYCONF MACRO
 *         ELSE
 *           IF PATCHID GREATER OR EQUAL TO 11 (VARY POINT)
 *             VARY THE POINT OFFLINE OR ONLINE (PATCHID 12 OR 11)
 *             USING THE VARYCONF MACRO
 *           ENDIF
 *         ENDIF
 *       ENDIF
 *       ACCORDING TO THE MACRO RETURN CODE ISSUE THE PROPER MESSAGE
 *     EXIT
 *   ENDSEGMENT DOMCFGD2
```

```
MEMBER NAME  DOMCGENO
 *        DOMCGENO MAIN SEGMENT
 *          GET ADDR AND ITEM COUNT OF ARRAY AALFCPDO
 *          LOCK ARRAY AALFCPDO
 *          DO UNTIL ALL ENTRIES ARE PROCESSED
 *            SEARCH ARRAY AALFCPDO UNTIL ALL ENTRIES WITH THE SAME S/7 X
 *                  ID HAVE BEEN GROUPED
 *          ENDDO
 *            IF ANY ONE OF THE PULSE DURATIONS ARE NON-ZERO THEN
 *              GETWA BUFFER FOR 8 BYTES PLUS 10 * NUMBER OF ENTRIES
 *              CONVERT DATA IN BUFFER TO ASCII
 *              S7WRITE DATA TO SPECIFIED S/7
 *            ENDIF
 *          UNLOCK ARRAY AALFCPDO
 *        ENDSEGMENT DOMCGENO
```

```
MEMBER NAME  DOMCHART
 * DOMCHART MAIN SEGMENT
 *    ESTABLISH ADDRESSABILITY
 *    GET ADDRESS OF TYPES TABLE FROM EMSCVT
 *    SAVE PATCH ID
 *    ZERO SAVE AREAS
 *    CASE ENTRY - PATCH ID
 *      CASE  -  BUILD DISPLAY OR REFRESH (1 OR 2)
 *        IF PATCH ID EQUAL TO INITIAL BUILD
 *          TURN BACKLIGHTS ON OR OFF USING DLITES MACRO
 *          ZERO RECORDER COUNT
 *        ENDIF
 *        DO DOMBLD
 *
 *      CASE  -  DATA ENTRY FOR CHANGES (3)
 *        IF MASTER DISPATCHER ACCESS AREA AND FUNCTION DO NOT MATCH
 *          ONE OF PAIRS DEFINED FOR DISPLAY UNIT THEN
 *          ERROR EXIT TO ERR1
 *        ENDIF
 *        CALL DOMCHRT2
 *
 *    ENDCASE
 *    EXIT WITH RETURN CODE SET TO ZERO
 *
 *    ERROR ENTER ERR1
 *      SET UP FOR MESSAGE # 308
 *      DO MESG
 *      POINT TO DISPLAY UNIT ID
 *      DO ZONE
 *    ERROR RETURN
 *    EXIT WITH RETURN CODE SET TO ZERO
 * ENDSEGMENT DOMCHART
 *
 * DOMBLD SUBROUTINE SEGMENT
 *    CALL DOMCHRT1
 * ENDSEGMENT DOMBLD
 *
 * MESG SUBROUTINE SEGMENT
 *    GET MESSAGE USING MESSAGE MACRO
 * ENDSEGMENT MESG
 *
 * ZONE SUBROUTINE SEGMENT
 *    SET UP DISPLAY UNIT ID
 *    DISPLAY MESSAGE IN SCRATCH PAD ZONE USING DWZONE MACRO
 * ENDSEGMENT ZONE
```

```
MEMBER NAME  DOMCHRTA
 * DOMCHRTA MAIN SEGMENT
 *    ESTABLISH ADDRESSABILITY
 *    SAVE PATCH ID
 *    GET ADDRESSES OF STRIPCHART ARRAY AND SYSGEN OPTIONS ARRAY FROM
 *      EMSCVT
 *    SAVE NUMBER OF RECORDERS
 *    IF PATCH ID EQUAL TO COMMAND REQUEST (1) THEN
 *      IF MACRO NOT ISSUED BY DISPLAY PROGRAM REQUEST THEN
 *        ERROR EXIT TO ERR1 IF RECORDER ID LESS THAN ONE
 *        ERROR EXIT TO ERR1 IF RECORDER ID GREATER THAN HIGHEST ID
 *          SYSGENED
 *      ENDIF
 *      USE RECORDER ID TO CALCULATE DISPLACEMENT INTO STRIPCHART ARRAY
 *      IF MACRO NOT ISSUED BY DISPLAY PROGRAM THEN
 *        IF ACTION IS NOT ON OR OFF THEN
 *          ERROR EXIT TO ERR2
 *        ENDIF
 *        IF ACTION IS ON THEN
 *          TURN ON RECORDER ON FLAG
 *        ELSE
 *          TURN ON RECORDER OFF FLAG
 *        ENDIF
 *        IF ON FLAG IS ON THEN
 *          ERROR EXIT TO ERR8 IF RECORDER IS ACTIVE
 *        ELSE
 *          ERROR EXIT TO ERR9 IF RECORDER IS AVAILABLE
 *        ENDIF
 *        IF TIME MARK OPTION IS NOT YES OR NO THEN
 *          ERROR EXIT TO ERR3
 *        ENDIF
 *        IF OPTION IS YES THEN
 *          SET ON TIME MARK FLAG
 *        ENDIF
 *        IF RECORDER IS TO BE TURNED ON THEN
 *          ERROR EXIT IF NAME NOT PASSED IN LIST TO ERR6
 *          SET UP NAME FOR MACRO
 *          GET ADDRESS OF POINT USING GETITEM MACRO
 *          ERROR EXIT TO ERR6 IF MACRO FAILS
 *          IF ANALOG POINT THEN
 *            SET ON ANALOG ITEM FLAG IN ARRAY ITEM
 *          ELSE
 *            IF PULSE COUNTER DATA THEN
 *              SET ON PC ITEM FLAG IN ARRAY ITEM
 *            ELSE
 *              ERROR EXIT TO ERR6 - INVALID POINT TYPE
 *            ENDIF
 *          ENDIF
 *          MOVE POINT NAME TO ARRAY ITEM
 *          SAVE POINT ADDRESS IN ARRAY ITEM
 *          GET ADDRESS OF RCB ARRAY USING GETARRAY MACRO
 *          ERROR EXIT TO ERR7 IF MACRO FAILS
 *          IF ANALOG POINT THEN
 *                  START SEARCH FOR RCB TO WHICH POINT BELONGS
 *            EXIT IF RCB FOUND THEN
 *              FIND DISPLACEMENT INTO NAMES LIST FOR POINT
 *              MOVE TYPE CODE TO ARRAY ITEM
 *            ORELSE
 *              POINT TO NEXT RCB ITEM
```

```
MEMBER NAME  DOMCHRTA
 *            ENDLOOP
 *              ERROR EXIT TO ERR6 IF NOT FOUND
 *            END SEARCH
 *          ELSE
 *            MOVE TYPE CODE (PC) TO ARRAY ITEM
 *          ENDIF
 *        ELSE
 *          ERROR EXIT TO ERR6 IF POINT NAME NOT PASSED
 *          ERROR EXIT TO ERR6 IF NAME DOES NOT MATCH NAME IN ARRAY ITEM
 *        ENDIF
 *        IF SCALE FACTOR A IS PASSED THEN
 *          ERROR EXIT TO ERR4 IF NOT WITHIN VALID RANGE
 *          MOVE SCALE FACTOR TO ARRAY ITEM
 *        ELSE
 *          MOVE DEFAULT VALUE TO ARRAY ITEM (ONE)
 *        ENDIF
 *        IF SCALE FACTOR B IS PASSED THEN
 *          ERROR EXIT TO ERR5 IF NOT WITHIN VALID RANGE
 *          MOVE SCALE FACTOR TO ARRAY ITEM
 *        ELSE
 *          MOVE DEFAULT VALUE TO ARRAY ITEM (ZERO)
 *        ENDIF
 *      ELSE
 *        IF ACTION IS ON THEN
 *          TURN RECORDER ON INDICATOR ON
 *        ELSE
 *          TURN RECORDER OFF INDICATOR ON
 *        ENDIF
 *        IF TIME MARK OPTION IS YES THEN
 *          TURN TIME MARK OPTION FLAG ON
 *        ENDIF
 *      ENDIF
 *      BUILD MESSAGE (TRANSACTION CODE X'08') TO SEND TO SYSTEM/7
 *      PUT END OF MESSAGE INDICATOR IN MESSAGEAREA
 *      CONVERT FROM EBCDIC TO ASCII
 *      DO FINDECB
 *      IF FIRST ECB IS ZERO THEN
 *        SET PATCH ID TO 3
 *      ELSE
 *        IF SECOND ECB IS ZERO THEN
 *          SET PATCH ID TO 4
 *        ELSE
 *          ERROR EXIT TO ERRB
 *        ENDIF
 *      ENDIF
 *      SAVE ECB ADDRESS
 *      SAVE RECORDER ID
 *      SEND MESSAGE TO SYSTEM/7 USING S7WRITE MACRO
 *      ERROR EXIT TO ERRA IF MACRO FAILS
 *      SET ON IN PROGRESS FLAG IN ARRAY ITEM
 *      SET TIME-OUT UP USING PTIME MACRO FOR 30 SECONDS
 *    ELSE
 *      DO FINDECB
 *      IF PATCH ID IS EQUAL TO SYSTEM/7 REPLY (2) THEN
 *        SAVE REPLY
 *        RELEASE INPUT BUFFER USING RLSEBUFF MACRO
 *        MATCH ECB TO RECORDER ID
 *        ERROR EXIT TO ERRC IF NO MATCH FOUND
```

```
MEMBER NAME  DOMCHRTA
 *       DELETE TIME-OUT USING PTIME MACRO
 *       USE RECORDER ID TO FIND DISPLACEMENT INTO STRIPCHART ARRAY
 *       ERROR EXIT TO ERRC IF IN PROGRESS FLAG IS ON
 *       ERROR EXIT TO ERRD IF RETURN CODE FROM S/7 IS NOT ZERO
 *       SET OFF IN PROGRESS FLAG
 *       IF CODE IN MESSAGE INDICATES RECORDER ON THEN
 *         SET ON ACTIVE FLAG IN ARRAY ITEM
 *         IF CYCLIC PROCESSOR NOT YET STARTED THEN
 *           START CYCLIC PROCESSOR (DOMCHRTC) USING PTIME MACRO AT
 *            BASIC SCAN CYCLE RATE
 *         ENDIF
 *         ADD 1 TO NUMBER OF ACTIVE RECORDERS
 *       ELSE
 *         ZERO ARRAY ITEM
 *         SUBTRACT 1 FROM NUMBER OF ACTIVE RECORDERS
 *         IF NO RECORDERS ARE NOW ACTIVE THEN
 *           DELETE CYCLIC PROCESSOR USING PTIME MACRO
 *         ENDIF
 *       ENDIF
 *       LOG CASCHART ARRAY USING PUTLOG MACRO
 *       SET UP FOR MESSAGE 496
 *       DO MESG
 *       DO EVENT
 *       ZERO ECB
 *     ELSE
 *       IF PATCH ID EQUAL TO TIME-OUT (3 OR 4) THEN
 *         ERROR EXIT IF NO MATCH FOUND ON ECB TO ERRE
 *       ENDIF
 *       USE RECORDER ID TO FIND DISPLACEMENT INTO ARRAY
 *       SET UP FOR MESSAGE # 497
 *       DO MESG
 *       DO EVENT
 *       ZERO ECB
 *       IF RECORDER IS NOT ON THEN
 *         ZERO ARRAY ITEM
 *       ELSE
 *         SET OFF COMMAND IN PROGRESS FLAG IN ARRAY ITEM
 *       ENDIF
 *     ENDIF
 *   ENDIF
 *   IF PATCH ID IS FOR TIME-OUT THEN
 *     SET RETURN CODE TO 52
 *   ELSE
 *     ZERO RETURN CODE
 *   ENDIF
 *   IF NOT CHANGE REQUEST THEN
 *     POST SECOND USER ECB
 *   ENDIF
 *   ZERO RETURN CODE
 *   EXIT WITH RETURN CODE SET TO ZERO
 *
 *   ERROR ENTER ERR1  - INVALID ID
 *     SET RETURN CODE TO FOUR
 *
 *   ERROR ENTER ERR2  - INVALID ACTION
 *     SET RETURN CODE TO EIGHT
 *
 *   ERROR ENTER ERR3  - INVALID TIME MARK
```

```
MEMBER NAME  DOMCHRTA
 *      SET RETURN CODE TO TWELVE
 *
 *    ERROR ENTER ERR4  - INVALID SCALE A
 *      SET RETURN CODE TO SIXTEEN
 *      ZERO ARRAY ENTRY
 *
 *    ERROR ENTER ERR5  - INVALID SCALE B
 *      SET RETURN CODE TO TWENTY
 *      ZERO ARRAY ENTRY
 *
 *    ERROR ENTER ERR6  - INVALID POINT NAME
 *      SET RETURN CODE TO TWENTY-FOUR
 *
 *    ERROR ENTER ERR7  - RCB GETARRAY FAILED
 *      SET RETURN CODE TO TWENTY-EIGHT
 *      ZERO ARRAY ITEM
 *
 *    ERROR ENTER ERR8  - RECORDER ALREADY ON
 *      SET RETURN CODE TO THIRTY-TWO
 *
 *    ERROR ENTER ERR9  - RECORDER ALREADY OFF
 *      SET RETURN CODE TO THIRTY-SIX
 *
 *    ERROR ENTER ERRA  - S7WRITE MACRO FAILED
 *      IF ON FLAG IS ON THEN
 *        ZERO ARRAY ITEM
 *      ENDIF
 *      ZERO ECB
 *      SET RETURN CODE TO FORTY
 *
 *    ERROR ENTER ERRB  - ECBS IN USE
 *      IF ON FLAG IS ON THEN
 *        ZERO ARRAY ITEM
 *      ENDIF
 *      SET RETURN CODE TO FORTY-FOUR
 *
 *    ERROR ENTER ERRC  - INVALID SCC REPLY
 *      ZERO RETURN CODE
 *
 *    ERROR ENTER ERRD  - SCC COMMAND FAILED
 *      SET UP FOR MESSAGE # 498
 *      DO MESG
 *      DO EVENT
 *      IF RECORDER IS NOT ACTIVE THEN
 *        ZERO ARRAY ITEM
 *      ENDIF
 *      SET RETURN CODE TO FORTY-EIGHT
 *      POST SECOND USER ECB
 *      ZERO ECB
 *
 *    ERROR ENTER ERRE  - INVALID TIME-OUT
 *      ZERO RETURN CODE
 *
 *    ERROR RETURN
 *      IF RETURN CODE IS NOT ZERO THEN
 *        SET UP FOR MESSAGE # 468
 *        DO MESG
 *        DO EVENT
```

```
MEMBER NAME  DOMCHRTA
 *      ENDIF
 *    EXIT WITH RETURN CODE SET
 * ENDSEGMENT DOMCHRTA
 *
 * FINDECB SUBROUTINE SEGMENT
 *    FIND DISPLACEMENT INTO ARRAY TO ECBS
 * ENDSEGMENT FINDECB
 *
 * MESG SUBROUTINE SEGMENT
 *    ISSUE MESSAGE MACRO TO GET MESSAGE TEXT
 * ENDSEGMENT MESG
 *
 * EVENT SUBROUTINE SEGMENT
 *    ISSUE EVENT USING SCEVENT MACRO
 * ENDSEGMENT EVENT
```

```
MEMBER NAME  DOMCHRTC
 * DOMCHRTC MAIN SEGMENT
 *    ESTABLISH ADDRESSABILITY
 *    GET ADDRESS OF STRIPCHART ARRAY AND SYSGEN OPTIONS ARRAY
 *    GET NUMBER OF RECORDERS FROM ARRAY
 *    ZERO MESSAGE AREA
 *    UNTIL ALL RECORDERS PROCESSED DO
 *      IF ACTIVE FLAG IS ON THEN
 *        IF ANALOG ITEM THEN
 *          SUBTRACT B COEFFICIENT FROM MOST RECENT SCANNED VALUE
 *          DIVIDE RESULT BY A COEFFICIENT
 *        ELSE - PULSE COUNTER
 *          DIVIDE MOST RECENT SCANNED VALUE BY A COEFFICIENT
 *        ENDIF
 *        SET POSITIVE OR NEGATIVE SIGN
 *        CONVERT FLOATING POINT VALUE TO BINARY
 *        STORE VALUE IN TABLE
 *      ENDIF
 *      POINT TO NEXT SLOT IN TABLE
 *      POINT TO NEXT ARRAY ENTRY
 *    ENDDO
 *    MOVE END OF MESSAGE INDICATOR TO TABLE
 *    BUILD MESSAGE HEADER - TRANSACTION CODE X'13'
 *    SEND MESSAGE TO SYSTEM/7 USING S7WRITE MACRO
 *    EXIT WITH RETURN CODE OF ZERO
 * ENDSEGMENT DOMCHRTC
```

```
MEMBER NAME  DOMCHRT1
 * DOMCHRT1 MAIN SEGMENT
 *    ESTABLISH ADDRESSABILITY
 *    GET BUFFER BUILD AREA USING GETWA MACRO
 *    ERROR EXIT TO ERR1 IF MACRO FAILS
 *    SET UP FOR MESSAGE # 494
 *    DO MESG
 *    ERROR EXIT TO ERR2 IF MACRO FAILS
 *    IF PATCH ID IS NOT DATA ENTRY THEN
 *      ZERO ERROR FLAGS
 *    ENDIF
 *    GET ADDRESS OF STRIPCHART ARRAY
 *    ESTABLISH POINTERS TO BUFFER SECTIONS
 *    BLANK BUFFER AREA
 *    ZERO RECORDER ID SECTION IN BUFFER
 *    SAVE NUMBER OF RECORDERS FROM ARRAY
 *    UNTIL ALL RECORDERS PROCESSED DO
 *      STORE RECORDER ID IN BUFFER
 *      IF UNDEFINED INDICATOR IS ON THEN
 *        MOVE UNDEFINED PHRASE TO BUFFER
 *        ADJUST BUFFER POINTER
 *        ADJUST POINTERS TO OTHER BUFFER SECTIONS
 *      ELSE
 *        IF COMMAND IS IN PROGRESS THEN
 *          MOVE IN PROGRESS PHRASE TO BUFFER
 *        ELSE
 *          MOVE DATA ENTRY CHARACTERS TO BUFFER
 *        ENDIF
 *        ADJUST POINTER TO ACTION SECTION
 *        IF ACTIVE INDICATOR IS ON THEN
 *          MOVE ACTIVE PHRASE TO BUFFER SECTION
 *          ADJUST SECTION POINTER
 *          MOVE POINT NAME TO BUFFER SECTION
 *          POINT TO A SCALE FACTOR
 *          DO CONVERT
 *          MOVE CONVERTED SCALE FACTOR TO BUFFER SECTION
 *          POINT TO B SCALE FACTOR
 *          DO CONVERT
 *          MOVE CONVERTED SCALE FACTOR TO BUFFER SECTION
 *          IF TIME MARK INDICATOR ON THEN
 *            MOVE YES PHRASE TO BUFFER SECTION
 *          ELSE
 *            MOVE NO PHRASE TO BUFFER SECTION
 *          ENDIF
 *          GET POINT TYPE FROM TYPES TABLE USING TYPE CODE
 *          ADJUST BUFFER SECTION POINTER
 *        ELSE
 *          MOVE AVAILABLE PHRASE TO BUFFER SECTION
 *          ADJUST BUFFER SECTION POINTER
 *          MOVE DATA ENTRY CHARACTERS TO REMAINING BUFFER SECTIONS
 *          ADJUST BUFFER POINTERS
 *        ENDIF
 *        IF ERROR FLAG ARE NOT ZERO THEN
 *          MOVE ERROR MESSAGE TO BUFFER SECTION
 *          IF INVALID ACTION FLAG ON THEN
 *            POINT TO CURRENT POSITION IN BUFFER AND START OF BUFFER
 *             SECTION AREA AND APPROPRIATE MESSAGE
 *            DO INVALID
 *          ENDIF
```

```
MEMBER NAME  DOMCHRT1
 *          IF INVALID NAME FLAG ON THEN
 *            POINT TO CURRENT POSITION IN BUFFER, START OF BUFFER AND
 *             APPROPRIATE MESSAGE
 *            DO INVALID
 *          ENDIF
 *          IF INVALID SCALE A FACTOR FLAG ON THEN
 *            POINT TO CURRENT POSITION IN BUFFER, START OF BUFFER AND
 *             APPROPRIATE MESSAGE
 *            DO INVALID
 *          ENDIF
 *          IF INVALID SCALE B FACTOR FLAG ON THEN
 *            POINT TO CURRENT POSITION IN BUFFER, START OF BUFFER AND
 *             APPROPRIATE MESSAGE
 *            DO INVALID
 *          ENDIF
 *          IF INVALID TIME MARK OPTION FLAG ON THEN
 *            POINT TO CURRENT POSITION IN BUFFER, START OF BUFFER AND
 *             APPROPRIATE MESSAGE
 *            DO INVALID
 *          ENDIF
 *          IF DOMCSCHT MACRO FAILED OR
 *          IF GETARRAY MACRO FAILED THEN
 *            POINT TO CURRENT POSITION IN BUFFER, START OF BUFFER AND
 *             APPROPRIATE MESSAGE
 *             DO INVALID
 *          ENDIF
 *        ENDIF
 *      ENDIF
 *      POINT TO NEXT ERROR FLAG BYTE
 *      POINT TO NEXT STRIPCHART ARRAY ENTRY
 *      INCREMENT RECORDER ID ENTRY
 *    ENDDO
 *    POINT TO DISPLAY UNIT ID
 *    POINT TO FIRST DYNAMIC ID
 *    SET UP REPEAT NUMBER FOR MACRO
 *    POINT TO ATTRIBUTE
 *    POINT TO BUFFER SECTION - RECORDER IDS
 *    DO DINF
 *    POINT TO NEXT DYNAMIC ID
 *    ALTER REPEAT NUMBER
 *    POINT TO BUFFER SECTION - RECORDER STATUS
 *    DO DINF
 *    POINT TO NEXT DYNAMIC ID
 *    POINT TO ATTRIBUTE
 *    POINT TO BUFFER SECTION - ACTION DATA ENTRY FIELD
 *    DO DINF
 *    POINT TO NEXT DYNAMIC ID
 *    ALTER REPEAT NUMBER
 *    SET UP LOOP CONTROL COUNT (NUMBER OF RECORDERS)
 *    POINT TO BUFFER SECTION - POINT INFORMATION
 *    UNTIL LOOP COUNT IS ZERO DO
 *      IF RECORDER IS ACTIVE THEN
 *        POINT TO GREEN ATTRIBUTE
 *      ELSE
 *        POINT TO YELLOW ATTRIBUTE
 *      ENDIF
 *      DO DINF
 *      POINT TO NEXT DYNAMIC ID
```

```
MEMBER NAME  DOMCHRT1
 *      ADJUST BUFFER POINTER
 *    ENDDO
 *    IF ERROR MESSAGE AREA IS NOT BLANK THEN
 *      SET UP LOOP CONTROL COUNT
 *      POINT TO FIRST ERROR BUFFER AREA
 *      POINT TO SECOND ERROR BUFFER AREA
 *      UNTIL LOOP COUNT IS ZERO DO
 *        IF FIRST AREA IS BLANK THEN
 *          IF SECOND AREA IS BLANK THEN
 *            POINT TO NEXT ERROR BUFFER AREA
 *          ELSE
 *            MOVE INFORMATION FROM SECOND AREA TO FIRST AREA
 *            BLANK SECOND AREA
 *            POINT TO NEXT GROUP FOR FIRST AREA
 *            POINT TO NEXT GROUP FOR SECOND AREA
 *          ENDIF
 *        ELSE
 *          ADJUST BOTH POINTERS TO NEXT ERROR BUFFER AREAS
 *        ENDIF
 *      ENDDO
 *    ELSE
 *      IF PATCH ID IS DATA ENTRY CHANGES THEN
 *        MOVE ALL UPDATES SUCCESSFUL MESSAGE TO BUFFER SECTION
 *      ENDIF
 *    ENDIF
 *    POINT TO BUFFER SECTION
 *    POINT TO ATTRIBUTE - WHITE
 *    DO DINF
 *    POINT TO NEXT DYNAMIC ID
 *    POINT TO NEXT ATTRIBUTE - RED,PROTECTED
 *    POINT TO BUFFER AREA - INVALID ITEM LINE
 *    DO DINF
 *    POINT TO NEXT DYNAMIC ID
 *    POINT TO NEXT INVALID ITEM BUFFER LINE
 *    DO DINF
 *    POINT TO NEXT DYNAMIC ID
 *    POINT TO NEXT INVALID ITEM BUFFER LINE
 *    DO DINF
 *    BLANK MESSAGE AREA
 *    DO ZONE
 *    SET UP FOR DISPUP MACRO
 *    WRITE DYNAMIC INFORMATION TO SCREEN USING DISPUP MACRO
 *    ERROR EXIT TO ERR2 IF MACRO RETURN CODE IS NOT ZERO
 *    WAIT ON ECB
 *    ERROR EXIT IF ECB INDICATES MACRO FAILED
 *    EXIT WITH RETURN CODE SET TO ZERO
 *
 *    ERROR ENTER ERR1  - NO GETWA
 *      SET UP FOR MESSAGE # 333
 *      DO MESG
 *      POINT TO DISPLAY UNIT ID
 *      DO ZONE
 *
 *    ERROR ENTER ERR2 - UNABLE TO UPDATE DISPLAY
 *      SET UP FOR MESSAGE # 495
 *      DO MESG
 *      DO ZONE
 *
```

```
MEMBER NAME  DOMCHRT1
 *   ERROR RETURN
 *   EXIT WITH RETURN CODE SET TO ZERO
 * ENDSEGMENT DOMCHRT1
 *
 * CONVERT SUBROUTINE SEGMENT
 *   CONVERT BINARY SCALE FACTOR TO DECIMAL
 *   UNPACK FACTOR TO MAKE DISPLAYABLE
 * ENDSEGMENT CONVERT
 *
 * INVALID SUBROUTINE SEGMENT
 *   IF CURRENT ADDRESS EQUAL TO START OF BUFFER SECTION ADDRESS THEN
 *     MOVE MESSAGE PHRASE TO AREA
 *     SET UP TRANSLATE TABLE
 *     FIND END OF PHRASE AND ADD BLANK
 *   ENDIF
 *   CONVERT RECORDER ID TO DECIMAL
 *   MOVE ID TO ERROR BUFFER AREA
 *   ADJUST CURRENT POSITION BUFFER
 * ENDSEGMENT INVALID
 *
 * DINF SUBROUTINE SEGMENT
 *   ISSUE DINFO MACRO FOR SECTION
 *   ERROR EXIT TO ERR2 IF MACRO FAILS
 * ENDSEGMENT DINF
 *
 * MESG SUBROUTINE SEGMENT
 *   GET MESSAGE USING MESSAGE MACRO
 * ENDSEGMENT MESG
 *
 * ZONE SUBROUTINE SEGMENT
 *   POINT TO DISPLAY UNIT ID AND MESSAGE
 *   WRITE MESSAGE TO SCRATCH PAD ZONE USING DWZONE MACRO
 * ENDSEGMENT ZONE
```

```
MEMBER NAME  DOMCHRT2
 * DOMCHRT2 MAIN SEGMENT
 *    ESTABLISH ADDRESSABILITY
 *    GET ADDRESS OF STRIPCHART ARRAY FROM EMSCVT
 *    SAVE NUMBER OF RECORDERS
 *    ZERO ERROR FLAG BYTES
 *    POINT TO RECORDER ID READ FROM DISPLAY SCREEN
 *    UNTIL ALL RECORDERS PROCESSED DO
 *      IF RECORDER IS DEFINED THEN
 *        IF ACTION FIELD HAS DATA ENTERED THEN
 *          IF FIELD IS BLANK THEN
 *            SET ON INVALID ACTION ERROR FLAG
 *          ELSE
 *            IF FIELD HAS ON IN IT THEN
 *              IF RECORDER IS ACTIVE THEN
 *                SET ON INVALID ACTION ERROR FLAG
 *              ELSE
 *                SET ON TURN ON FLAG
 *              ENDIF
 *            ELSE
 *              IF FIELD HAS OFF IN IT THEN
 *                IF RECORDER IS AVAILABLE THEN
 *                  SET ON INVALID ACTION ERROR FLAG
 *                ENDIF
 *              ELSE
 *                SET ON INVALID ACTION ERROR FLAG
 *              ENDIF
 *            ENDIF
 *          ENDIF
 *          IF RECORDER IS TO BE TURNED ON THEN
 *            SET UP NAME FOR GETITEM MACRO
 *            GET ADDRESS OF ITEM USING GETITEM MACRO
 *            IF RETURN CODE IS NOT ZERO THEN
 *              SET ON INVALID NAME ERROR FLAG
 *            ELSE
 *              IF POINT IS ANALOG THEN
 *                SET ON ANALOG FLAG IN ARRAY ITEM
 *              ELSE
 *                IF POINT IS COUNTER THEN
 *                  SET ON PC FLAG IN ARRAY ITEM
 *                ELSE
 *                  SET ON INVALID NAME ERROR FLAG
 *                ENDIF
 *              ENDIF
 *              IF NAME IS VALID THEN
 *                SAVE ADDRESS OF POINT IN ARRAY ITEM
 *                SAVE NAME OF POINT IN ARRAY ITEM
 *              ENDIF
 *            ENDIF
 *            SET UP TRANSLATE TABLE
 *            POINT TO A SCALE FACTOR INPUT FIELD
 *            SET ON A FLAG
 *            DO TRANSLTE
 *            READJUST TRANSLATE TABLE
 *            SET OFF A FLAG
 *            POINT TO B SCALE FACTOR FIELD
 *            DO TRANSLTE
 *            IF TIME OPTION FIELD HAS DATA THEN
 *              IF DATA IS YES THEN
```

```
MEMBER NAME  DOMCHRT2
 *                SET ON TIME MARK FLAG IN ARRAY ITEM
 *                SET ON TIME MARK FLAG IN FLAG BYTE
 *              ELSE
 *                IF DATA IS NOT NO THEN
 *                  SET ON INVALID TIME MARK ERROR FLAG
 *                ENDIF
 *              ENDIF
 *            ENDIF
 *          ENDIF
 *          IF NO ERRORS FOUND THEN
 *            SET UP MACRO PARAMETER POINTERS
 *            ISSUE DOMCSCHT MACRO
 *            IF RETURN CODE IS NOT ZERO THEN
 *              SET ON MACRO FAILED ERROR FLAG
 *              IF RECORDER WAS TO BE TURNED ON THEN
 *                ZERO ARRAY ITEM ENTRY
 *              ENDIF
 *            ELSE
 *              IF RCB ADDRESS NOT YET GOTTEN THEN
 *                GET RCB ARRAY ADDRESS USING GETARRAY MACRO
 *                IF RETURN CODE IS ZERO THEN
 *                  CALCULATE LENGTH OF ARRAY
 *                  SAVE ARRAY ADDRESS AND LENGTH
 *                ELSE
 *                  BRANCH TO NORCB
 *                ENDIF
 *              ELSE
 *                PICK UP ADDRESS OF RCB ARRAY AND LENGTH FROM SAVE AREAS
 *              ENDIF
 *              IF ANALOG FLAG ON THEN
 *                START SEARCH FOR RCB TO WHICH POINT BELONGS
 *                EXIT IF RCB FOUND THEN
 *                  PICK UP ADDRESS OF ANALOG NAMES LIST
 *                  CALCULATE DISPLACEMENT INTO NAMES LIST
 *                  MOVE TYPE CODE TO STRIPCHART ARRAY ITEM
 *                ORELSE
 *                  POINT TO NEXT RCB ITEM
 *                ENDLOOP
 *                END SEARCH
 *              ELSE
 *                MOVE TYPE TO ARRAY ITEM (PC)
 *              ENDIF
 *            ENDIF
 *          ENDIF
 *NORCB     TURN OFF TIME MARK FLAG
 *          TURN OFF  RECORDER ON FLAG
 *        ENDIF
 *      ENDIF
 *      POINT TO NEXT PSREAD ENTRY
 *      POINT TO NEXT ERROR FLAG BYTE
 *      ADJUST RECORDER ID NUMBER
 *      POINT TO NEXT ENTRY IN STRIPCHART ARRAY
 *    ENDDO
 *    EXIT WITH RETURN CODE SET TO ZERO
 *  ENDSEGMENT DOMCHRT2
 *
 *  TRANSLTE SUBROUTINE SEGMENT
 *    IF DATA ENTERED IN FIELD THEN
```

```
MEMBER NAME  DOMCHRT2
 *     TRANSLATE TO CHECK FOR INVALID CHARACTERS
 *     IF INVALID CHARACTER FOUND THEN
 *       IF A FLAG IS ON THEN
 *         SET ON INVALID SCALE A ERROR FLAG ON
 *       ELSE
 *         SET INVALID SCALE B FLAG ON
 *       ENDIF
 *     ELSE
 *       IF FIELD IS BLANK THEN
 *         IF A FLAG IS ON THEN
 *           SET ON INVALID A SCALE ERROR FLAG
 *         ELSE
 *           SET ON INVALID B SCALE ERROR FLAG
 *         ENDIF
 *       ELSE
 *         FIND NUMBER OF BLANKS AROUND NUMBER (WHILE-DO LOOP)
 *         PAD NUMBER WITH ZEROS AFTER RIGHT JUSTIFYING IN FIELD
 *         CONVERT EBCDIC NUMBER TO BINARY
 *         IF A FLAG IS ON THEN
 *           IF SCALE VALUE IS GREATER THAN MAXIMUM OR
 *           IF SCALE VALUE IS LESS THAN MINIMUM THEN
 *             SET ON INVALID SCALE A FACTOR ERROR FLAG
 *           ELSE
 *             STORE VALUE IN ARRAY ITEM
 *           ENDIF
 *         ELSE
 *           IF SCALE VALUE IS GREATER THAN MAXIMUM OR
 *           IF SCALE VALUE IS LESS THAN MINIMUM THEN
 *             SET ON INVALID SCALE B ERROR FLAG
 *           ELSE
 *             STORE VALUE IN ARRAY ITEM
 *           ENDIF
 *         ENDIF
 *       ENDIF
 *     ENDIF
 *   ELSE
 *     IF A FLAG IS ON THEN
 *       STORE DEFAULT VALUE IN ARRAY ITEM (ONE)
 *     ELSE
 *       STORE DEFAULT VALUE IN ARRAY ITEM (ZERO)
 *     ENDIF
 *   ENDIF
 * ENDSEGMENT TRANSLTE
```

```
MEMBER NAME  DOMCIF
 *   CSECT  DOMCIF              COMPILER INTERFACE SUBROUTINE
 *
 *      OVERLAY COMPILER STAE WITH INTERFACE STAE AT DOIFSTAE
 *      CANCEL COMPILER SPIE
 *
 *      IF MACRO ID MULTIPLE OF 4, AND
 *      IF MACRO ID NOT NEGATIVE, AND
 *      IF MACRO ID NOT GREATER THAN LARGEST ID, THEN
 *        PERFORM THE SEGMENT TO EXECUTE THE DESIRED SERVICE
 *      ELSE
 *        SET RETURN CODE TO MINUS ONE
 *      ENDIF
 *      STORE THE RETURN CODE IN USER SUPPLIED AREA
 *         RESTORE STAE AND SPIE
 *      EXIT  THE PROGRAM
 *
 *
 *      DOIFSTAE                    STAE ROUTINE
 *         SAVE COMPLETION CODE IN STAE EXIT PARAMETER LIST
 *         LOAD ADDRESS OF STAE RETRY ROUTINE, DOMRCVRY
 *         REQUEST RETRY OPTION
 *         EXIT STAE ROUTINE
 *      ENDSEG  DOIFSTAE
 *      DOMRCVRY                    STAE RECOVERY ROUTINE
 *         IF  WORKAREA PROVIDED, THEN
 *           SAVE STAE WORKAREA ADDRESS
 *           LOAD INTERFACE WORKAREA
 *           LOAD ADDRESS OF PSW AT ENTRY TO ABEND
 *         ELSE  NO WORK AREA PROVIDED
 *           LOAD ADDRESS OF FIRST SAVEAREA THIS TCB
 *           STRTSRCH            SEARCH SAVEAREA CHAIN
 *           EXITIF   INTERFACE PGM ADDRESS IS IN WORD 4 OF SAVEAREA
 *             LOAD ADDRESS OF INTERFACE SAVEAREA
 *             LOAD A ZERO FOR ABEND ADDRESS
 *           ENDLOOP
 *           ENDSRCH
 *         ENDIF
 *         ISSUE DPP227 - COMPLETION CODE, ABEND ADDRESS AND MACRO ID
 *         IF  STAE WORKAREA PROVIDED, THEN
 *           FREEMAIN WORKAREA
 *         ENDIF
 *         SAVE ABEND COMPLETION CODE
 *         FREEMAIN INTERFACE WORKAREA
 *         RESTORE  REGISTERS 2-12
 *         ABEND WITH ORIGINAL COMPLETION CODE
 *      ENDSEG DOMRCVRY
 *
 *
 *      BGNSEG   DOMIF00   DOMCLGET/DOMCLPUT MACRO SERVICE, ID=0
 *         MOVE ALL PARAMETERS TO INTERFACE WORK SPACE
 *         BRANCH AND LINK TO THE MACRO PROCESSOR
 *         IF RETURN CODE IS ZERO, THEN
 *           STORE ADDR RETURNED DATA INTO COMPILER PARAMETER LIST
 *         ENDIF
 *      ENDSEG   DOMIF00
 *
 *      BGNSEG   DOMIF04   DOMCFREE MACRO SERVICE, ID=4
 *         BRANCH AND LINK TO THE MACRO PROCESSOR
```

```
MEMBER NAME  DOMCIF
 *      ENDSEG    DOMIF04
 *
 *      BGNSEG    DOMIF08   DOMCALRM MACRO SERVICE
 *         REPOSITION MACRO TYPE IN PARAMETER LIST
 *         BRANCH AND LINK TO THE MACRO PROCESSOR
 *      ENDSEG    DOMIF08
 *
 *      BGNSEG    DOMIF12          ASCICONV MACRO SERVICE
 *         LOAD  DATA ADDRESS AND DATA LENGTH
 *         IF ASCTYPE IS ZERO, THEN
 *           EXECUTE ASCICONV TYPE=ASCII
 *         ELSE        CONVERT TO EBCDIC
 *           EXECUTE ASCICONV TYPE=EBCDIC
 *         ENDIF
 *      BGNSEG    DOMIF16   SCEVENT MACRO SERVICE ID=16
 *         BRANCH AND LINK TO MACRO PROCESSOR
 *      ENDSEG    DOMIF16
 *
 *      BGNSEG    DOMIF20   SCDEVICE MACRO SERVICE ID=20
 *         REPOSITION FLAGS IN COMPILER PARAMETER LIST
 *         BRANCH AND LINK TO MACRO PROCESSOR
 *      ENDSEG    DOMIF20
 *
 *      BGNSEG    DOMIF24   STRIPCHRT MACRO SERVICE ID=24
 *         CONSTRUCT NEW PARAMETER LIST IN INTERFACE WORK AREA
 *         BRANCH AND LINK TO MACRO PROCESSOR
 *      ENDSEG    DOMIF24
 *
 *      BGNSEG    DOMIF28   S7WRITE MACRO SERVICE ID=28
 *         MOVE PARAMETER TO INTERFACE WORK AREA
 *         BRANCH AND LINK TO MACRO PROCESSOR
 *      ENDSEG    DOMIF28
 *
 *      BGNSEG    DOMIF32   VARYCONF MACRO SERVICE ID=32
 *         MOVE AND RESTRUCTURE PARAMETERS TO INTERFACE WORKAREA
 *         BRANCH AND LINK TO MACRO PROCESSOR
 *      ENDSEG    DOMIF32
 *
 *      BGNSEG    DOMIF36   VARYSCAN MACRO PROCESSOR ID=36
 *         MOVE AND RECONSTRUCT PARAMETERS TO INTERFACE WORKAREA
 *         BRANCH AND LINK TO MACRO PROCESSOR
 *      ENDSEG    DOMIF36
 *
 *      BGNSEG    DOMIF40   RLSEBUFF MACRO PROCESSOR ID=40
 *         LOAD THE BUFFER ADDRESS
 *         EXECUTE THE MACRO
 *      ENDSEG    DOMIF40
 *
 *      BGNSEG    DOMIF44   WAIT MACRO PROCESSOR
 *         LOAD THE ECB ADDRESS
 *         WAIT ON THE ECB
 *         ZERO THE POST BIT
 *      ENDSEG    DOMIF44
 *
 *
 *    END  DOMCIF
```

```
MEMBER NAME  DOMCINIT
 *        DOMCINIT MAIN SEGMENT
 *          CASENTRY PATCH ID
 *            CASE 1 - COLDSTART
 *              DO SEGMENT DOMCCOLD
 *              DO SEGMENT DOMCWARM
 *              CLEAR ALL ALARM AND ERROR FLAGS IN SENSOR BASE DATA      X
 *                    ARRAYS
 *            CASE 2 - WARM START
 *              IF THE DATA BASE IS CONSISTANT THEN
 *                DO SEGMENT DOMCWARM
 *              ELSE
 *                DO SEGMENT DOMCCOLD
 *                DO SEGMENT DOMCWARM
 *              ENDIF
 *          ENDCASE
 *        ENDSEGMENT DOMCINIT
 *
 *        DOMCCOLD SUBROUTINE SEGMENT
 *          GETARRAY TYPE=ADDR FOR STATUS,COUNTER,ANALOG,AND NAMES LIST X
 *                DATA ARRAYS CASTATUS,CACOUNT,CANALOG,AND CANAME
 *            IF GETARRAY NOT SUCCESSFUL THEN
 *              ISSUE ERROR MESSAGE INDICATING GETARRAY FAILURE
 *            ENDIF
 *            DO UNTIL ALL RCB'S ARE UPDATED
 *              RESOLVE ADDRESSES OF RCB'S STATUS,COUNTER,ANALOG,AND     X
 *                    NAMES LIST AND PLACE THEM IN THE RCB
 *            ENDDO
 *            LOAD SYSGENED POINT SUMMATION MODULE DOMTPTSM
 *            DO UNTIL END OF MODULE REACHED
 *              GET THE NUMBER OF POINTS IN EACH GROUP
 *              PUT HEADER INFORMATION (GROUP NO.AND NO.OF POINTS IN     X
 *                    GROUP) IN POINT SUMMATION TABLE
 *              DO UNTIL ALL POINTS HAVE BEEN PROCESSED
 *                GETITEM TYPE=ADDR FOR EACH POINT NAME
 *                IF GETITEM NOT SUCESSFUL THEN
 *                  ISSUE ERROR MESSAGE INDICATING GETITEM FAILURE
 *                ENDIF
 *                PLACE ADDRESS OF EACH POINT IN POINT SUMMATION TABLE
 *              ENDDO
 *            ENDDO
 *        ENDSEGMENT DOMCCOLD
 *
 *        DOMCWARM SUBROUTINE SEGMENT
 *          GET ADDRESS OF DEVICE CONTROL INDEX TABLE FROM EMSCVT
 *          ZERO THE DEVICE CONTROL INDEX TABLE
 *          OPEN THE EVENT LOG FILE DATA SET AND INITIALIZE THE EVENT    X
 *                LOG FILE
 *          IF THIS IS A VALID WARM START THEN
 *            DO SEGMENT DOMCIALM TO SCAN THE DATA BASE AND ISSUE ALARMSX
 *                  AS REQUIRED
 *            DO SEGMENT DOMTPCNT TO UPDATE PULSE COUNTER DATA UP TO     X
 *                  CURRENT TIME
 *          ENDIF
 *        ENDSEGMENT DOMCWARM
 *
 *        DOMCIALM SUBROUTINE SEGMENT
 *          DO UNTIL ALL RCB'S ARE SEARCHED
 *            EXAMINE RCB'S STATUS,PULSE COUNTER,AND ANALOG DATA FOR     X
```

```
MEMBER NAME  DOMCINIT
 *                ERROR CONDITIONS
 *            BUILD SCAN EXCEPTION TABLE ENTRIES FOR ALL REQUIRED ALARMS
 *          ENDDO
 *          IF ANY SCAN EXCEPTION TABLE ENTRIES WERE BUILT THEN
 *            PATCH DOMCALR1 PASSING POINTER TO FIRST SCAN EXCEPTION    X
 *                  TABLE
 *            IF PATCH WAS NOT SUCCESSFUL THEN
 *              FREE SPACE OBTAINED FOR SCAN EXCEPTION TABLE
 *              ISSUE ERROR MESSAGE INDICATING PATCH FAILURE TO DOMCALR1
 *            ENDIF
 *          ENDIF
 *        ENDSEGMENT DOMCIALM
 *
 *        DOMTPCNT SUBROUTINE SEGMENT
 *          GETBLOCK TO BRING IN MOST CURRENT COPY OF CACOUNT
 *          DETERMINE THE AMOUNT OF DOWN TIME IN MINUTES
 *          IF DOWN TIME IS GREATER THAN 60 MINUTES THEN
 *            DO UNTIL ALL PC POINTS ARE PROCESSED
 *              DETERMINE NUMBER OF MINUTES UNTIL NEXT EVEN HOUR
 *              NUMBER OF MINUTES TIMES EXTRAPALATION VALUE
 *              ADD TO ACCUMULATED AMOUNT AND STORE ACCUMULATOR
 *            ENDDO
 *            PUTLOG FOR CACOUNT
 *            DO UNTIL PC DATA TIME IS UP TO LAST EVEN HOUR OF CURRENT  X
 *                  TIME
 *              DO UNTIL ALL PC POINTS ARE PROCESSED
 *                60 TIMES EXTRAPALATION VALUE
 *                ADD TO ACCUMULATED AMOUNT AND STORE IT BACK
 *              ENDDO
 *              PUTLOG FOR CACOUNT
 *            ENDDO
 *            DETERMINE NUMBER OF MINUTES FROM LAST EVEN HOUR UNTIL     X
 *                  LAST CURRENT EVEN MINUTE
 *            DO UNTIL ALL PC POINTS ARE PROCESSED
 *              NUMBER OF MINUTES UNTIL LAST EVEN MINUTE TIMES          X
 *                    EXTRAPALATION VALUE
 *              ADD TO ACCUMULATED AMOUNT AND STORE IT BACK
 *            ENDDO
 *            PUTBLOCK FOR CACOUNT
 *          ELSE
 *            IF DOWN TIME IS OVER AN EVEN HOUR MARK THEN
 *              DO UNTIL ALL PC POINTS ARE PROCESSED
 *                DETERMINE NUMBER OF MINUTES UNTIL NEXT EVEN HOUR
 *                NUMBER OF MINUTES TIMES EXTRAPALATION VALUE
 *                ADD TO ACCUMULATED AMOUNT AND STORE IT BACK
 *              ENDDO
 *              PUTLOG FOR CACOUNT
 *            DETERMINE NUMBER OF MINUTES FROM LAST EVEN HOUR UNTIL     X
 *                  LAST CURRENT EVEN MINUTE
 *            DO UNTIL ALL PC POINTS ARE PROCESSED
 *              NUMBER OF MINUTES UNTIL LAST EVEN MINUTE TIMES          X
 *                    EXTRAPALATION VALUE
 *              ADD TO ACCUMULATED AMOUNT AND STORE IT BACK
 *            ENDDO
 *            PUTBLOCK FOR CACOUNT
 *            ELSE
 *              DO UNTIL ALL PC POINTS ARE PROCESSED
 *                DETERMINE NUMBER OF MINUTES SINCE THIS DATA WAS GOOD
```

```
MEMBER NAME  DOMCINIT
 *                NUMBER OF MINUTES TIMES EXTRAPALATION VALUE
 *                ADD TO ACCUMULATED AMOUNT AND STORE IT BACK
 *              ENDDO
 *              PUTBLOCK FOR CACOUNT
 *            ENDIF
 *          ENDIF
 *        ENDSEGMENT DOMTPCNT
```

```
MEMBER NAME  DOMCLFRE
 *   BEGIN DOMCLFRE
 *      /*  THIS CSECT ISSUES A FREEWA MACRO TO FREE BUFFER AREA WHICH
 *          WAS RETURNED TO A USER VIA THE DOMCLGET MACRO. */
 *   GET ADDRESS OF DPPXCVT
 *   ISSUE FREEWA MACRO PROVIDING ADDRESS PASSED FROM DOMCFREE MACRO AND
 *      DPPXCVT ADDRESS
 *   RETURN
 *   END DOMCLFRE
```

```
MEMBER NAME  DOMCLRMT
 *   BEGIN DOMCLRMT
 *   /* THIS IS THE CONTROL PROGRAM FOR THE DOMCLGET AND DOMCLPUT MACROS.
 *      A MACRO ID OF 0X OR 2X WILL INDICATE INPUTS FROM DOMCLGET MACRO
 *      FOR CURRENT OR LOGGED HISTORY DATA RESPECTIVELY. A MACRO ID OF
 *      4X OR 6X WILL INDICATE INPUTS FROM DOMCLPJT MACRO FOR CURRENT OR
 *      LOGGED HISTORY DATA RESPECTIVELY. */
 *   RETRIEVE AND SAVE DPPXCVT ADDRESS
 *   SET RETURN CONDITION CODE TO ZERO
 *   DO GETMAIN MACRO FOR WORK AREA
 *   EQUATE LENGTHS OF RCB STATUS ANALOG NAME-LIST AND COUNTER DSECTS TO
 *      CONSTANTS
 *   ISSUE GETARRAY MACRO TO PROVIDE LENGTHS OF INCORE RCB,STATUS,COUNTER
 *      ANALOG, AND NAMES-LIST ARRAYS
 *   BUILD ARRAY NAMES TO BE USED USING S/7 ID AND RMT ID
 *   CASENTRY
 *     CASE  ID=0  GET CURRENT DATA
 *       INCLUDE GET CURRENT PROCESSING SEGMENT (DOMCLRGC)
 *     CASE  ID=2  GET LOGGED HISTORY DATA
 *       INCLUCE GET HISTORY PROCESSING SEGMENT (DOMCLRGH)
 *     CASE  ID=4  PUT CURRENT DATA
 *       INCLUDE PUT CURRENT PROCESSING SEGMENT (DOMCLRPC)
 *     CASE  ID=6  PUT LOGGED HISTORY DATA
 *       INCLUDE PUT HISTORY PROCESSING SEGMENT (DOMCLRPH)
 *   ENDCASE
 *   STORE ADDRESS OF RETURN BUFFER IN REG 1
 *   STORE RETURN CONDITION CODE IN REG 15
 *   FREE WORK AREA BJFFER USING FREEMAIN MACRO
 *   RESTORE REGISTERS
 *   END DOMCLRMT
 *
 *
 *
 *   BEGIN DOMCLRGC SUBROUTINE
 *      /* GET CURRENT PROCESSING SEGMENT. */
 *   DO BUFFGET SUBROUTINE USING LENGTH OF RCB ITEM
 *   ISSUE A GETITEM MACRO AND PLACE RETURNED RCB ITEM IN BUFFER
 *   CASENTRY
 *     CASE ID=0 GET CURRENT RCB DATA
 *       GO TO ENDCASE
 *     CASE ID=2 GET CURRENT ANALOG DATA AND NAMES LIST DATA (DOMCLRCA)
 *       CALCULATE LENGTH OF ANALOG ITEMS (#)(LENGTH OF ONE)
 *       CALCULATE LENGTH OF NAMES LIST ITEMS (# OF ANALOG)(LENGTH OF ONE
 *          NAMES LIST ITEM)
 *       DO BUFFGET SUBROUTINE USING SUM OF CALCULATED LENGTHS
 *       MOVE ANALOG AND NAMES LIST ITEMS TO BUFFER
 *     CASE ID=4 GET CURRENT COUNTER DATA  (DOMCLRCC)
 *       CALCULATE LENGTH OF COUNTER ITEMS (#)(LENGTH OF ONE)
 *       DO BUFFGET SUBROUTINE USING LENGTH OF COUNTER ITEMS
 *       MOVE COUNTER ITEMS TO BUFFER
 *     CASE ID=6 GET CURRENT STATUS DATA  (DOMCLRCS)
 *       CALCULATE LENGTH OF STATUS ITEMS (#)(LENGTH OF ONE)
 *       DO BUFFGET SUBROUTINE USING LENGTH OF STATUS ITEMS
 *       MOVE STATUS ITEMS TO BUFFER
 *   ENDCASE
 *   IF LOW-ORDER 4 BITS OF MACRO ID EQUALS ZERO THEN
 *     SAVE BUFFER ADDRESS FOR RCB
 *   ELSE
 *     FREE BUFFER AREA USED FOR RCB WITH FREEWA MACRO
```

```
MEMBER NAME  DOMCLRMT
 *   ENDIF
 *   END DOMCLRGC
 *
 *
 *
 *   BEGIN DOMCLRGH SUBROUTINE
 *      /* GET HISTORY PROCESSING SEGMENT  */
 *   SET ARRAY ADDRESS POINTERS TO ZERO
 *   CASENTRY
 *     CASE ID=0 GET HISTORY RCB DATA
 *       INCLUDE GET HISTORY RCB SEGMENT (DOMCLGHR)
 *     CASE ID=2 GET HISTORY ANALOG/NAMES LIST DATA
 *       INCLUDE GET HISTORY ANALOG SEGMENT (DOMCLGHA)
 *     CASE ID=4 GET HISTORY COUNTER DATA
 *       INCLUDE GET HISTORY COUNTER SEGMENT (DOMCLGHC)
 *     CASE ID=6 GET HISTORY STATUS DATA
 *       INCLUDE GET HISTORY STATUS SEGMENT (DOMCLGHS)
 *   ENDCASE
 *   FREE BUFFER AREA USED FOR ARRAYS WITH FREEMAIN MACRO
 *   END DOMCLRGH
 *
 *
 *
 *   BEGIN DOMCLGHR SUBROUTINE
 *      /* GET HISTORY RCB */
 *   DO GETBUFF SUBROUTINE USING LENGTH OF RCB ARRAY
 *   DO LOGHIST SUBROUTINE FOR RCB ARRAY
 *   PLACE ARRAY HEADER INFORMATION IN FEEDBACK BUFFER
 *   SET RCB ARRAY ADDRESS EQUAL TO BUFFER ADDRESS
 *   BUMP ARRAY ADDRESS PAST HEADER INFORMATION
 *   DO   DOMCLSCH SUBROUTINE
 *   DO BUFFGET SUBROUTINE FOR RCB DSECT LENGTH
 *   SAVE ADDRESS TO BE RETURNED
 *   MOVE LENGTH OF DATA AND MACRO ID TO BUFFER
 *   MOVE RCB DATA TO BUFFER
 *   MOVE LENGTH AND DISPLACEMENT INFO. TO FEEDBACK BUFFER
 *   END DOMCLGHR
 *
 *
 *
 *   BEGIN DOMCLGHA SUBROUTINE
 *      /* GET ANALOG AND NAMES LIST ARRAYS */
 *   LENGTH = LENGTH OF RCB, ANALOG, AND NAMES LIST ARRAYS
 *   DO GETBUFF SUBROUTINE USING CALCULATED LENGTH
 *   DO LOGHIST SUBROUTINE FOR RCB ARRAY
 *   SET RCB ARRAY ADDRESS EQUAL TO BUFFER ADDRESS
 *   DO LOGHIST SUBROUTINE FOR NAMES LIST ARRAY
 *   SET NAMES LIST ARRAY ADDRESS EQUAL TO BUFFER ADDRESS
 *   DO LOGHIST SUBROUTINE FOR ANALOG ARRAY
 *   SET ANALOG ARRAY ADDRESS EQUAL TO BUFFER ADDRESS
 *   MOVE ARRAY HEADER INFORMATION INTO FEEDBACK BUFFER
 *   BUMP ADDRESSES PAST ARRAY HEADERS
 *   DO   DOMCLSCH SUBROUTINE  - SEARCH
 *   SAVE ADDRESS TO BE RETURNED
 *   CALCULATE LENGTH OF ANALOG AND NAMES LIST ITEMS
 *   DO BUFFGET SUBROUTINE USING LENGTHS OF ANALOG AND NAMES ITEMS
 *   MOVE ANALOG AND NAME LIST ITEMS TO BUFFER
 *   MOVE LENGTH AND DISPLACEMENT INFO. TO FEEDBACK BUFFER
```

```
MEMBER NAME  DOMCLRMT
 *   END DOMCLGHA
 *
 *
 *
 *   BEGIN DOMCLGHC SUBROUTINE
 *      /* GET PULSE COUNTER HISTORY ARRAY */
 *   LENGTH = LENGTH OF RCB AND COUNTER ARRAYS
 *   DO GETBUFF SUBROUTINE USING CALCULATED LENGTH
 *   DO LOGHIST SUBROUTINE FOR RCB ARRAY
 *   SET RCB ARRAY ADDRESS EQUAL TO BUFFER ADDRESS
 *   DO LOGHIST SUBROUTINE FOR COUNTER ARRAY
 *   SET COUNTER ADDRESS EQUAL TO BUFFER ADDRESS
 *   MOVE ARRAY HEADER INFORMATION TO FEEDBACK BLOCK
 *   BUMP ARRAY ADDRESSES PAST HEADER
 *   DO DOMCLSCH SUBROUTINE  - SEARCH
 *   CALCULATE LENGTH OF COUNTER ITEMS
 *   DO BUFFGET SUBROUTINE USING LENGTH OF COUNTER ITEMS
 *   SAVE ADDRESS TO BE RETURNED
 *   MOVE LENGTH AND DISPLACEMENT INFO. TO FEEDBACK BUFFER
 *   MOVE COUNTER ITEMS TO BUFFER
 *   END DOMCLGHC
 *
 *
 *
 *   BEGIN DOMCLGHS SUBROUTINE
 *      /* GET STATUS HISTORY ARRAY DATA */
 *   LENGTH = LENGTH OF RCB AND STATUS ARRAYS
 *   DO GETBUFF SUBROUTINE USING CALCULATED LENGTH
 *   DO LOGHIST SUBROUTINE FOR RCB ARRAY
 *   SET RCB ARRAY ADDRESS EQUAL TO BUFFER ADDRESS
 *   DO LOGHIST SUBROUTINE FOR STATUS ARRAY
 *   SET STATUS ARRAY ADDRESS EQUAL TO BUFFER ADDRESS
 *   MOVE HEADER INFORMATION INTO FEEDBACK BUFFER
 *   BUMP ARRAY ADDRESSES PAST HEADER
 *   DO DOMCLSCH SUBROUTINE  - SEARCH
 *   CALCULATE LENGTH OF STATUS ITEMS
 *   DO BUFFGET SUBROUTINE USING LENGTH OF STATUS ITEMS
 *   SAVE ADDRESS TO BE RETURNED
 *   MOVE LENGTH AND DISPLACEMENT INFO. TO FEEDBACK BUFFER
 *   MOVE STATUS ITEMS TO BUFFER
 *   END DOMCLGHS
 *
 *
 *
 *   BEGIN DOMCLRPC SUBROUTINE
 *      /* THIS SECTION PROCESSES PUT CURRENT DATA */
 *   IF OFFLIN STATUS BITS ARE OFF THEN
 *     ISSUE MACRO ERROR EXIT-
 *   ENDIF
 *   IF MACRO ID EQ XXXX0000 THEN
 *   ELSE
 *     IF MACRO ID = XXXX0010 THEN
 *       CALCULATE LENGTH OF ANALOG ITEM
 *       STORE ANALOG ITEM IN DATA BASE
 *     ELSE
 *       IF MACRO ID EQ XXXX0100 THEN
 *         CALCULATE LENGTH OF COUNTER ITEMS
 *         MOVE COUNTER ITEMS INTO DATA BASE
```

```
MEMBER NAME  DOMCLRMT
 *      ELSE
 *        IF MACRO ID EQ XXXX0110 THEN
 *        ENDIF
 *      ENDIF
 *    ENDIF
 *  ENDIF
 *  END DOMCLRPC
 *
 *
 *
 *                   SUBROUTINE
 *
 *
 *  BEGIN DOMCLRPH SUBROUTINE
 *     /* PUT HISTORY PROCESSING SEGMENT */
 *  CASENTRY
 *    CASE ID=0
 *      INCLUDE PUT HISTORY RCB SEGMENT (DOMCLPHR)
 *    CASE ID=2
 *      INCLUDE PUT HISTORY ANALOG AND NAMES LIST SEGMENT (DOMCLPHA)
 *    CASE ID=4
 *      INCLUDE PUT HISTORY COUNTER SEGMENT (DOMCLPHC)
 *    CASE ID=6
 *      INCLUDE PUT HISTORY STATUS SEGMENT (DOMCLPHS)
 *  ENDCASE
 *  DO LOGHIST  SUBROUTINE USING HEADER    INFORMATION FROM FEEDBACK
 *     BLOCK TO RETRIEVE ARRAY
 *  REPLACE DATA INTO BUFFER USING DISPLACEMENT AND LENGTH FROM FDBCK BL
 *  ISSUE PUTLOG TO REPLACE ARRAY
 *  ISSUE FREEMAIN MACRO TO FREE BUFFER AREA
 *  END DOMCLRPH
 *
 *
 *
 *  BEGIN DOMCLPHR SUBROUTINE
 *     /* PUT LOGGED HISTORY RCB */
 *  DO GETBUFF SUBROUTINE USING LENGTH OF RCB ARRAY
 *  END DOMCLPHR
 *
 *
 *
 *  BEGIN DOMCLPHA SUBROUTINE
 *     /* PUT HISTORY ANALOG AND NAMES LIST DATA */
 *  DO GETBUFF SUBROUTINE USING LENGTH OF ANALOG ARRAY
 *  DO LOGHIST SUBROUTINE USING FEEDBACK HEADER INFO. FOR ANALOG ARRAY
 *  REPLACE ANALOG DATA IN LOGGED HISTORY DATA BASE
 *  END DOMCLPHA
 *
 *
 *
 *  BEGIN DOMCLPHC SUBROUTINE
 *     /* PUT LOGGED HISTORY COUNTER DATA */
 *  DO GETBUFF SUBROUTINE USING LENGTH OF COUNTER ARRAY
 *  REPLACE COUNTER DATA IN LOGGED HISTORY DATA BASE
 *  END DOMCLPHC
 *
 *
 *
```

```
MEMBER NAME  DOMCLRMT
 *   BEGIN DOMCLPHS SUBROUTINE
 *      /* PUT LOGGED HISTORY STATUS DATA */
 *   DO GETBUFF SUBROUTINE USING LENGTH OF STATUS ARRAY
 *   END DOMCLPHS
 *
 *
 *
 *   BEGIN DOMCLSCH SUBROUTINE
 *      /* THIS SUBROUTINE SEARCHES THE HISTORY DATA BASE ARRAYS FOR
 *         SPECIFIC DATA ITEMS.  */
 *   TURN FOUND BIT OFF
 *   END ADDRESS EQ START ADDRESS OF GETMAIN + RCB ARRAY LENGTH
 *   DO UNTIL RCB ARRAY ADDRESS GE END ADDRESS OR
 *        UNTIL FOUND BIT IS ON
 *     IF RCB S/7 ID EQ S/7 ID AND
 *     IF RCB RMT ID EQ RMT ID THEN
 *       TURN FOUND BIT ON
 *       /* INDEX ADDRESS(S) WILL POINT TO CORRECT LOCATION WITHIN
 *          ARRAY(S). */
 *     ELSE
 *       ADD RCB DSECT LENGTH TO RCB ARRAY ADDRESS
 *       IF ANALOG ARRAY ADDRESS NE ZERO THEN
 *         MULTIPLY (NO. ANALOG ITEMS)(LENGTH OF ANALOG DSECT)
 *         ADD RESULT TO ANALOG ARRAY ADDRESS
 *         MULTIPLY (NO. ANALOG ITEMS)(LENGTH OF NAMES LIST DSECT)
 *         ADD RESULT TO NAMES LIST ARRAY ADDRESS
 *       ELSE
 *         IF STATUS ARRAY ADDRESS NE ZERO THEN
 *           MULTIPLY (NO. OF STATUS ITEMS)(LENGTH OF STATUS DSECT)
 *           ADD RESULT TO STATUS ARRAY ADDRESS
 *         ELSE
 *           IF COUNTER ARRAY ADDRESS NE ZERO THEN
 *             MULTIPLY (NO. OF COUNTER ITEMS)(LENGTH OF COUNTER DSECT)
 *             ADD RESULT TO COUNTER ARRAY ADDRESS
 *           ENDIF
 *         ENDIF
 *       ENDIF
 *     ENDIF
 *   ENDDO
 *   ISSUE ERROR MACRO IF FOUND BIT IS NE ZERO
 *   END DOMCLSCH
 *
 *
 *
 *   BEGIN GETBUFF  SUBROUTINE
 *      /* ISSUE GETMAIN FOR BUFFER SPACE */
 *   ISSUE GETMAIN FOR LENTH PROVIDED
 *   ISSUE MACRO ERROR EXIT IF RETURN CODE NE ZERO
 *   SET UP SAVE AREA FOR SUBPOOL AND ADDRESS FOR USE IN FREEMAIN MACRO
 *   INITIALIZE AREA TO ZERO
 *   END GETBUFF
 *
 *
 *
 *   BEGIN BUFFGET  SUBROUTINE
 *      /* THIS SUBROUTINE ISSUES A GETWA MACRO FOR A SPECIFIED LENGTH
 *         AND RETURNS AN ADDRESS OF AN AREA IN REG 1 */
 *   ISSUE GETWA MACRO
```

```
MEMBER NAME  DOMCLRMT
 * ISSUE MACRO ERROR EXIT IF RETURN CODE NE ZERO
 *   END BUFFGET
 *
 *
 *
 *   BEGIN LOGHIST  SUBROUTINE
 *      /* ISSUE GETLOG MACRO */
 *   SAVE REGISTERS
 *   LOAD REGISTERS WITH ADDRESSES AND VALUES REQUIRED FOR MACRO
 *   ISSUE GETLOG MACRO
 *   IF RETURN CODE NE ZERO THEN
 *     ISSUE ERROREXIT MACRO
 *   ENDIF
 *   RESTORE REGISTERS
 *   END LOGHIST
 *
 *
 *
 *   BEGIN DOMCLERR SUBROUTINE
 *      /* THIS SUBROUTINE PROCESSES ERROR EXITS FROM OTHER SUBROUTINES*/
 *   CHECK FOR ERROR CONDITION AND SET APPROPRIATE ERROR RETURN CODE
 *   END DOMCLERR
```

```
MEMBER NAME  DOMCNCNV
 * BEGIN DOMCNCNV
 * /* THIS IS THE PROGRAM USED FOR EBCDIC TO FLOATING CONVERSION */
 * INITIALIZE NUMBER AREA
 * IF LENGTH EXCEEDS MAX THEN
 *   RETURN TO CALLER
 * ENDIF
 * IF LENGTH EQ ZERO THEN
 *   RETURN TO CALLER
 * ENDIF
 * MOVE NUMBER TO WORK AREA
 * IF NUMBER GT ALLOWABLE FLOATING POINT NO THEN
 *   RETURN CODE EQ 5
 *   RETURN TO CALLER
 * ENDIF
 * CONVERT NUMBER TO FLOATING POINT
 * RETURN TO CALLER
 * END DOMCNCNV
```

```
MEMBER NAME  DOMCPDC1
 *   DOMCPDC1 MAIN SEGMENT
 *     ESTABLISH ADDRESSABILITY FOR XCVT, EMSCVT, SYSGEN OPTIONS TABLE,
 *      RCB, AND PDC OPTIONS TABLE
 *     IF A HIERARCHY CHANGE OF STATUS
 *       ESTABLISH ADDRESSABILITY FOR STATUS ITEM
 *       DETERMINE DISPLACEMENT INTO PDC OPTIONS TABLE FOR DEVICE TYPE
 *       IF SWITCH CHANGE OF STATUS
 *         SET PROCESSING INDICATOR TO ZERO
 *         DO DOMOPT - ENTITY, EVENTING PROCESSING
 *       ELSE                        NOT A SWITCH CHANGE OF STATUS
 *         IF BREAKER CHANGE OF STATUS
 *           SET PROCESSING INDICATOR TO ZERO
 *           DO DOMOPT- ENTITY, EVENTING PROCESSING
 *         ELSE                      OTHER THAN SWITCH/BREAKER COS
 *           IF MOTOR OPERATED SWITCH CHANGE OF STATUS
 *             IF BOTH EVEN/ODD STATUS ITEM STATUS BITS ARE OFF
 *               ERREXIT ONE
 *             ELSE                  TEST EVEN/ODD STATUS BIT PAIR
 *               IF BOTH EVEN/ODD STATUS ITEM STATUS BITS ARE ON
 *                 ERREXIT ONE
 *               ELSE                STATUS BITS SETS PROPERLY
 *                 SET PROCESSING INDICATOR TO ONE
 *                 DO DOMOPT - ENTITY, EVENTING PROCESSING
 *               ENDIF
 *             ENDIF
 *           ELSE                    TCT2 OR TCT3 COS
 *             IF TAP CHANGING TRANSFORMER TYPE 2 CHANGE OF STATUS
 *               SET PROCESSING INDICATOR TO 2
 *               DO DOMOPT - ENTITY, EVENTING PROCESSING
 *             ELSE                  TCT3 CHANGE OF STATUS
 *               IF TAP CHANGING TRANSFORMER TYPE 3 CHANGE OF STATUS
 *                 SET PROCESSING INDICATOR TO 2
 *                 DO DOMOPT - ENTITY, EVENTING PROCESSING
 *               ENDIF
 *             ENDIF
 *           ENDIF
 *         ENDIF
 *       ENDIF
 *     ELSE                          PROCESSING FOR PDC PORTION OF RDA
 *       IF PROCESSING RAW DATA ARRAY PDC DATA
 *         ESTABLISH ADDRESSABILITY TO RDA PDC DATA
 *         DETERMINE NUMBER OF RDA PDC ENTRIES
 *         ESTABLISH ADDRESSABILITY TO RDA PDC ENTRIES
 *         UNTIL ALL RDA PDC ENTRIES ARE PROCESSED
 *           IF SYSTEM/370 CONTROL ACTION
 *             IF SUCCESSFUL RETURN CODE
 *               IF EITHER A TAG OR UNTAG REQUEST
 *                 DO DOMPTCH - PATCH DOMCDC01
 *               ELSE                NOT A TAG/UNTAG ACTION
 *                 IF EITHER A VERIFY OR EXECUTE TIMEOUT
 *                   DO DOMPTCH - PATCH DOMCDC01
 *                 ENDIF
 *               ENDIF
 *             ELSE                  UNSUCCESSFUL CONTROL ACTION
 *               DO DOMPTCH - PATCH DOMCDC01
 *             ENDIF
 *           ELSE                    CONTROL ORIGINATING FROM ANOTHER CPU
 *             IF SUCCESSFUL CONTROL ACTION
```

```
MEMBER NAME  DOMCPDC1
 *              CONVERT DEVICE TYPE FROM ASCII TO EBCDIC
 *              CONVERT DEVICE NUMBER FROM BINARY TO DECIMAL
 *              USE GETITEM TO LOCATE THE STATUS ITEM FOR THE DEVICE
 *              ERREXIT IF THE DEVICE STATUS ITEM IS NOT FOUND TO TWO
 *              IF EITHER A TAG OR UNTAG OPTION
 *                SET PROCESSING INDICATOR TO 3
 *                DO DOMOPT - ENTITY, EVENTING PROCESSING
 *              ELSE                NOT A TAG/UNTAG OPTION
 *                IF A VERIFY TIMEOUT
 *                  ESTABLISH ADDRESSABILITY FOR STATUS ITEM
 *                  TURN OFF STATUS ITEM EXECUTE BIT
 *                ELSE              NOT VERIFY TIMEOUT
 *                  SET STATUS ITEM EXECUTE BIT
 *                ENDIF
 *              ENDIF
 *            ELSE                  RETURN CODE IS NONZERO
 *              IF EXECUTE TIMEOUT
 *                TURN OFF STATUS ITEM EXECUTE BIT
 *              ENDIF
 *            ENDIF
 *          ENDIF
 *          GO TO NEXT RDA PDC ENTRY
 *        ENDDO
 *      ENDIF
 *    ENDIF
 *    RESTORE REGISTER 13
 *    EXIT
 *    ERRENTER  ONE                 STATUS BITS SET INCORRECTLY
 *    ERRENTER  TWO                 DEVICE NAME NOT FOUND BY GETITEM
 *    ERRENTER  THREE               STATUS ITEM ALREADY TAGGED
 *    ERRENTER  FOUR                STATUS ITEM ALREADY UNTAGGED
 *    ERRENTER  FIVE                RCB ADDRESS NOT FOUND BY GETITEM
 *    ERRETURN
 *    RESTORE REGISTER 13
 *    EXIT
 *  ENDSEGMENT DOMCPDC1
 *  DOMENT SUBROUTINE SEGMENT
 *    IF ENTITY ID IS ZERO
 *      ISSUE THE ENTITY
 *    ELSE                          NONZERO ENTITY ID
 *      DELETE THE ENTITY
 *    ENDIF
 *  ENDSEGMENT DOMENT
 *  DOMMESG SUBROUTINE SEGMENT
 *    SET UP PARAMETERS FOR MESSAGE MACRO
 *    RETRIEVE MESSAGE TEXT FOR MESSAGE 314
 *  ENDSEGMENT DOMMESG
 *  DOMEVNT SUBROUTINE SEGMENT
 *    SET UP PARAMETERS FOR SCEVENT MACRO
 *    ISSUE EVENT
 *  ENDSEGMENT DOMEVNT
 *  DOMOPT SUBROUTINE SEGMENT
 *    SAVE REGISTER 3 CONTENTS
 *    SAVE REGISTER 7 CONTENTS
 *    RETRIEVE ADDRESS OF PDC OPTIONS TABLE
 *    IF THE OPTION INDICATOR IS SET TO ZERO
 *      PICK UP DEVICE NAME
 *      SET STATUS ITEM EXECUTE BIT
```

```
MEMBER NAME  DOMCPDC1
 *      IF STATUS ITEM STATUS BIT IS OFF
 *        DO DOMENT - ISSUE OR DELETE ENTITY
 *        DO DOMMESG - RETRIEVE MESSAGE
 *        DO DOMEVNT - ISSUE EVENT
 *        SET ENTITY ID TO ZERO
 *      ELSE                          STATUS ITEM STATUS BIT ON
 *        SET ENTITY ID TO ONE
 *        DO DOMENT - ISSUE OR DELETE ENTITY
 *        DO DOMMESG - RETRIEVE MESSAGE
 *        DO DOMEVNT - ISSUE ENTITY
 *      ENDIF
 *    ELSE                            NONZERO OPTION INDICATOR
 *      IF OPTION INDICATOR IS ONE
 *        PICK UP DEVICE NAME
 *        SET BOTH EVEN/ODD STATUS ITEM EXECUTE BITS
 *        IF OPEN OPTION
 *          SET ENTITY ID TO ZERO
 *          DO DOMENT - ISSUE OR DELETE ENTITY
 *          DO DOMMESG - RETRIEVE MESSAGE
 *          DO DOMEVNT - ISSUE EVENT
 *        ELSE                        CLOSE OPTION
 *          SET ENTITY ID TO ONE
 *          DO DOMENT - ISSUE OR DELETE MESSAGE
 *          DO DOMMESG - RETRIEVE MESSAGE
 *          DO DOMEVNT - ISSUE EVENT
 *        ENDIF
 *      ELSE                          OPTION INDICATOR NOT ZERO OR ONE
 *        IF OPTION INDICATOR IS TWO
 *          PICK UP DEVICE NAME
 *          IF TAP CHANGING TRANSFORMER TYPE 3 CHANGE OF STATUS
 *            ADJUST PDC OPTION TABLE ADDRESS
 *            SET SECOND STATUS ITEM EXECUTE BIT
 *          ENDIF
 *          SET STATUS ITEM EXECUTE BIT
 *          IF STATUS ITEM STATUS BIT IS OFF
 *            SET ENTITY ID TO ONE
 *            DO DOMENT - ISSUE OR DELETE ENTITY
 *            DO DOMMESG - RETRIEVE MESSAGE
 *            DO DOMEVNT - ISSUE EVENT
 *          ELSE                      NOT AUTOMATIC OPTION
 *            SET ENTITY ID TO ZERO
 *            DO DOMENT - ISSUE OR DELETE ENTITY
 *            DO DOMMESG - RETRIEVE MESSAGE
 *            DO DOMEVNT - ISSUE EVENT
 *          ENDIF
 *        ELSE                        OPTION INDICATOR IS THREE
 *          IF A TAG OPTION
 *            ERREXIT IF STATUS ITEM TAG BIT ALREADY SET
 *            SET STATUS ITEM TAG BIT
 *            SET ENTITY ID TO ZERO
 *            DO DOMENT - ISSUE OR DELETE ENTITY
 *            DO DOMMESG - RETRIEVE MESSAGE
 *            DO DOMRCB - DETERMINE RCB ADDRESS, ISSUE EVENT
 *          ELSE                      UNTAG OPTION
 *            ERREXIT IF STATUS ITEM UNTAGGED ALREADY
 *            TURN OFF STATUS ITEM TAG BIT
 *            SET ENTITY ID TO ONE
 *            DO DOMENT - ISSUE OR DELETE ENTITY
```

```
MEMBER NAME  DOMCPDC1
 *             DO DOMMESG - RETRIEVE MESSAGE
 *             DO DOMRCB - DETERMINE RCB ADDRESS, ISSUE EVENT
 *           ENDIF
 *         ENDIF
 *       ENDIF
 *     ENDIF
 *     RESTORE REGISTER 3 CONTENTS
 *     RESTORE REGISTER 7 CONTENTS
 *   ENDSEGMENT DOMOPT
 *   DOMPTCH SUBROUTINE SEGMENT
 *     USE GETWA TO OBTAIN PARAMETER LIST AREA FOR PATCH
 *     PATCH DOMCDC01
 *   ENDSEGMENT DOMPTCH
 *   DOMRCB SUBROUTINE SEGMENT
 *     SET UP PARAMETERS FOR GETITEM
 *     USE GETITEM TO RESOLVE ADDRESS OF RCB
 *     ERREXIT IF RCB ADDRESS NOT RESOLVED
 *     DO DOMEVNT - ISSUE THE EVENT
 *   ENDSEG DOMRCB
```

```
MEMBER NAME  DOMCROUT
 *  CSECT   DOMCROUT                    /*  ROUTE ADMINISTRATIVE MESSAGES */
 *
 *    INSERT PATCH ID FOR USE AS INDEX INTO SEGMENT TABLE ADDRESS LIST
 *    EXECUTE REQUESTED SEGMENT
 *    EXIT PGM
 *
 *    SEGMENT   DOMCRTS7                /*  DISPLAY MESSAGES FROM S/7  */
 *      LOCATE INPUT TRANSACTION
 *      TRANSLATE TEXT FROM ASCII TO EBCDIC
 *      SAVE ORIGIN ID
 *      SAVE MESSAGE TEXT
 *      RELEASE INPUT BUFFER
 *      RETRIEVE FORMATTED DATA USING MESSAGE NUMBER DPP230I
 *      DISPLAY TEXT OF MESSAGE TO SYSTEM MESSAGE ZONE OF
 *       MASTER DISPATCHER ACCESS AND FUNCTION AREAS
 *    ENDSEG    DOMCRTS7
 *
 *    SEGMENT   DOMCRTRQ                /* ROUTE DISPATCHER MESSAGE */
 *      READ INPUT FROM DISPLAY
 *      IF ERROR ON READ
 *        FLAG READ ERROR
 *        BRANCH TO EXIT SEGMENT
 *      ENDIF
 *      STRTSRCH FOR FIRST SIGNIFICANT CHARACTER OF TEXT
 *      EXITIF  NON-BLANK/NON-UNDERSCORE CHARACTER FOUND
 *        STRTSRCH FOR LAST SIGNIFICANT CHARACTER OF TEXT (RT TO LEFT)
 *        EXITIF  NON-BLANK/NON-UNDERSCORE CHARACTER FOUND
 *          CALCULATE TEXT LENGTH
 *        ORELSE
 *          CHECK NEXT CHARACTER TO LEFT
 *        ENDLOOP
 *          SET NO TEXT FLAG
 *        ENDSRCH
 *      ORELSE
 *        CHECK NEXT CHARACTER TO RIGHT
 *      ENDLOOP
 *        SET NO TEXT FLAG
 *      ENDSRCH
 *      IF TEXT PRESENT, THEN
 *        REPLACE UNDERSCORES WITH BLANKS IN ACCESS, FUNCTION & CPU FLDS
 *        IF ACCESS OR FUNCTION PRESENT, THEN
 *          SET  ACCESS/FUNCTION FLAG
 *        ENDIF
 *        IF CPU PRESENT, THEN
 *          SET CPU FLAG
 *        ENDIF
 *        IF ACCESS/FUNCTION FLAG AND CPU FLAG ON, OR
 *        IF ACCESS/FUNCTION FLAG AND CPU FLAG OFF, THEN
 *          SET PARM ERROR FLAG
 *        ELSE
 *          IF  CPU ID PRESENT, THEN
 *            IF CPU ID NOT NUMERIC, THEN
 *              SET CPU ID ERROR FLAG
 *            ELSE
 *              CONVERT CPU ID TO BINARY
 *              TRANSLATE TEXT TO ASCII
 *              BUILD ADMINISTRATIVE MESSAGE TRANSACTION
 *              WRITE MESSAGE TO S/7
```

```
MEMBER NAME  DOMCROUT
 *              IF ERROR ON WRITE, THEN
 *                SET ERROR FLAG AS TO TYPE OF ERROR
 *              ENDIF
 *            ENDIF
 *          ELSE
 *            DISPLAY MESSAGE TO SYSTEM MESSAGE ZONE OF THE
 *             ACCESS AND FUNCTION AREA GIVEN
 *            IF ERROR ON WRITE
 *              SET ACCESS/FUNCTION ERROR FLAG
 *            ENDIF
 *          ENDIF
 *        ENDIF
 *      ELSE
 *        SET NO TEXT FLAG
 *      ENDIF
 *      LOAD THE REPLY MESSAGE NUMBER USING THE ERROR FLAGS AS AN INDEX
 *      RETRIEVE APPROPRIATE MESSAGE
 *      DISPLAY RESULTS OF REQUEST TO DISPATCHER SCRATCH PAD
 *    ENDSEG    DOMCRTRQ
 *
 *    SEGMENT   DOMCRTRF            /*  REFRESH REQUEST FIELDS   */
 *      REWRITE DATA ENTRY FIELDS
 *    ENDSEG    DOMCRTRF
 *
 *  END     DOMCROUT
```

```
MEMBER NAME  DOMCSANA
* BEGIN DOMCSANA
*  /* THIS IS THE CONTROL PROGRAM FOR THE ANALOG DATA DISPLAY */
* IF PATCH ID NE TO 40 THEN
*   TURN ON LITES OF PROGRAM FUNCTION KEYS USED FOR THIS DISPLAY
* ENDIF
* IF PATCH ID EQ 10 THEN
*   DO INITIALIZATION PROCESSING (DOMCSPIN)
* ELSE
*   IF PATCH ID EQ 15 THEN
*     DO RE-INITIALIZATION PROCESSING (DOMCSAIN)
*   ELSE
*     IF PATCH ID EQ 20 THEN
*       DO PAGE FORWARD PROCESSING (DOMCSAPF)
*     ELSE
*       IF PATCH ID EQ 25 THEN
*         DO PAGE BACKWARD PROCESSING (DOMCSAPB)
*       ELSE
*         IF PATCH ID EQ 30 THEN
*           DO REFRESH PROCESSING (DOMCSARF)
*         ELSE
*           IF PATCH ID EQ 35 THEN
*             DO UPDATE PROCESSING (DOMCSAUP)
*           ELSE
*             IF PATCH ID EQ 40 THEN
*               DO CANCEL PROCESSING-DOMCSANL-
*             ENDIF
*           ENDIF
*         ENDIF
*       ENDIF
*     ENDIF
*   ENDIF
* ENDIF
* END DOMCSANA
*
* BEGIN SEGMENT DOMCSAIN
*                               /* INITIALIZE ANALOG DISPLAY */
* IF CEATAB SENSOR BASED DATA ADDRESS EQ TO ZERO THEN
*   GET BUFFER FOR RCE
*   STORE ADDRESS IN CEATAB
* ENDIF
* GET ADDRESS OF PARTIAL SCREEN READ ADDRESS
* PUT RDACS ID IN RCE
* PUT S/7 ID IN RCE
* PUT ACCESS AREA IN RCE
* DO INITIAL SEGMENT(SET UP PAGE ONE OF DISPLAY)
* END SEGMENT DOMCSAIN
*
* BEGIN SEGMENT DOMCSARN
* /* RE-INITIALIZE ANALOG DISPLAY */
* DO INITIAL SEGMENT (SET UP PAGE ONE OF DISPLAY)
* END SEGMENT DOMCSARN
*
* BEGIN SEGMENT DOMCSAPF
* /* PAGE FORWARD PROCESSING SECTION */
* GET RCE ADDRESS
*   IF DIRECT PAGE REQUEST AREA FILLED THEN
*     IF CHARACTERS NOT VALID THEN
*       PUT OUT ERROR MESSAGE TO SCREEN
```

```
MEMBER NAME  DOMCSANA
*      ELSE
*        IF NEW PAGE NO. GREATER THAN TOTAL THEN
*          PUT OUT ERROR MESSAGE
*        ELSE
*          CONVERT DIGITS TO BINARY
*          CALCULATE CURRENT ITEM ADDRESS
*          PLACE NEW PAGE NO. IN RCE
*          DO PCBUILD SEGMENT (BUILD PC DATA PAGE & DISPLAY)
*        ENDIF
*      ENDIF
*    ELSE
*      IF CURRENT PAGE NO EQ TOTAL PAGE NO. THEN
*        SET CURRENT PAGE NO. TO ZERO
*      ENDIF
*      INCREASE PAGE NUMBER BY ONE
*      CALCULATE CURRENT ITEM ADDRESS
*      DO ANBUILD SECTION (BUILD NEXT PAGE OF DISPLAY)
*    ENDIF
* END SEGMENT DOMCSAPF
*
* BEGIN SEGMENT DOMCSAPB
* /* PAGE BACKWARD ROUTINE */
*    IF DIRECT PAGE REQUEST AREA FILLED THEN
*      IF CHARACTERS NOT VALID THEN
*        PUT OUT ERROR MESSAGE TO SCREEN
*      ELSE
*        IF NEW PAGE NO. GREATER THAN TOTAL THEN
*          PUT OUT ERROR MESSAGE
*        ELSE
*          CONVERT DIGITS TO BINARY
*          CALCULATE CURRENT ITEM ADDRESS
*          PLACE NEW PAGE NO. IN RCE
*          DO PCBUILD SEGMENT (BUILD PC DATA PAGE & DISPLAY)
*        ENDIF
*      ENDIF
*    ELSE
*      IF CURRENT PAGE NO. EQ ONE THEN
*        CURRENT PAGE NUMBER EQUAL TOTAL PAGES PLUSE ONE
*        SUBTRACT ONE FROM PAGE NO.
*        CALCULATE CURRENT ITEM ADDRESS
*        DO ANBUILD SECTION (BUILD ANALOG DATA PAGE)
*      ENDIF
*    ENDIF
* END SEGMENT DOMCSAPB
*
* BEGIN SEGMENT DOMCSARF
* /* THIS SECTION REFRESHES  THE CURRENT DISPLAY PAGE */
*    CALCULATE CURRENT ITEM ADDRESS
*    DO ANBUILD SECTION (BUILD DISPLAY PAGE)
* ENDIF
* END SEGMENT DOMCSARF
*
* BEGIN SEGMENT DOMCSAUP
* /* UPDATE DATA BASE ITEMS FROM DISPLAY */
* GET BUFFER FOR STATUS INFORMATION OF UPDATES
* IF RDACS ACCESS AREA EQ TO ONE OF DISPLAY UNIT ACCESS AREAS THEN
*    IF PARTIAL SCREEN READ DATA PRESENT THEN
*      IF SCREEN READ DATA LENGTH NE ZERO THEN
```

```
MEMBER NAME  DOMCSANA
*       PUT BACKGROUND STATUS HEADER INFORMATION IN BUFFER
*       DO UNTIL ALL DATA PROCESSED
*         IF ITEM NO. IS NOT BLANK    /* ALL DATA FROM SCREEN READ */
*           IF FIRST CHARACTER NE UNDERSCORE THEN
*             GET ADDRESS OF ANALOG ITEM
*             IF ITEM IS NOT OFFLINE (OUT OF SERVICE)
*               PLACE ITEM NO. IN MESSAGE LINE IN BUFFER
*             ENDIF
*             IF FUNCTION CODE IS NE TO DISPLA UNIT FUNCTION CODES THEN
*               PLACE ITEM NO. IN MESSAGE LINE IN BUFFER
*               GENERATE SECURITY EVENT TO FUNCTION CODE & ACCESS AREA
*                  OF DISPLAY UNIT AND TO ACCESS AREA FUNCTION CODE OF
*                  ITEM
*             ENDIF
*             CHECK EACH CHARACTER AGAINST CHAR. STRING FOR VALIDITY
*             IF CHAR. NOT VALID THEN
*               PLACE ITEM NO. IN MESSAGE LINE IN BUFFER
*             ENDIF
*             IF NO ERROR CONDITION FOUND THEN
*               PLACE NO. IN BUFFER
*               BRANCH TO CONVERT ROUTINE TO CONVERT EBCDIC NO TO
*                  FLOATING POINT
*               IF FLOATING POINT NO LARGER THAN ALLOWED PUT ERROR
*                  MESSAGE TO SCREEN
*               ENDIF
*               IF VALUE PRESENT THEN
*                 EVENT UPDATE OF ANALOG VALUE
*                 PLACE NO. IN MESSAGE BUFFER UNDER SUCCESSFUL UPDATES
*               ENDIF
*             ENDIF
*           ENDIF
*         ENDIF
*         BUMP ADDRESSES TO NEXT ITEM ADDRESS
*       ENDDO
*       IF AN UPDATE WAS SUCCESSFUL THEN
*         DO DOMCSARF SECTION (REFRESH SCREEN)
*         PUT STATUS INFORMATION TO SCREEN
*       ENDIF
*     ELSE
*   GENERATE EVENT FOR ATTEMPTED UPDATE TO NON MATCHING ACCESS AREA
*   EVENT TO ACCESS AREA AND FUNCTION CODE OF DISPLAY UNIT
*   EVENT TO ACCESS AREA OF RDACS ITEM
*   PUT MESSAGE TO SCREEN
*   PUT MESSAGE TO SCREEN 'PSREAD INCORRECT'
*     ENDIF
*   ELSE
*     PUT MESSAGE TO SCREEN 'PSREAD INCORRECT'
*   ENDIF
* ELSE
* ENDIF
* END SEGMENT DOMCSAUP
*
* BEGIN SEGMENT INITIAL
* /* INITIALIZE DISPLAY TO PAGE ONE */
* SET CURRENT PAGE NO. TO ONE.
* CALCULATE TOTAL NO. OF PAGES FOR ANALOG DATA
* PLACE START DATA ITEM ADDRESS IN RCE
* CURRENT ITEM ADDR. EQ START ITEM ADDRESS
```

```
MEMBER NAME  DOMCSANA
* CALCULATE ADDR. OF LAST DATA ITEM
* DO ANBUILD SEGMENT (BUILD PAGE ONE AND DISPLAY)
* END SEGMENT INITIAL
*
* BEGIN SEGMENT ANBUILD
* /* BUILD AND DISPLAY ANALOG PAGE */
* GET BUFFER FOR DISPLAY
* SET POINTERS INTO BUFFER
* CALCULATE START NO.
* PLACE NO. IN TEMP. STORAGE
* DO 16 TIMES
*   PLACE NAME IN BUFFER
*   GET FUNCTION AREA AND PLACE IN BUFFER
*   GET VALUE AND PLACE IN BUFFER
*   GET TYPE AND PLACE IN BUFFER
*   PLACE STATUS CONDITION IN BUFFER
*   IF ITEM EQ LAST ITEM THEN
*     SET CONDITION FLAG TO LEAVE LOOP
*   ENDIF
* ENDDO
* IF DISPLAY ON SCREEN IS NOT DOMDSAN2 THEN
*   CALL DISPLAY BACKGROUND TO SCREEN
*   INCREASE TASK COUNTER BY ONE
* ENDIF
* PUT TIME AND DATE ON SCREEN
* WRITE DISPLAY BUFFER TO SCREEN
* END ANBUILD SEGMENT
*
```

Former copy has been moved or deleted.

```
MEMBER NAME  DOMCSANL
 * BEGIN  DOMCSANL
 * /* THIS CSECT CONTROLS THE ANALOG DETAIL DISPLAY DOMDSANL */
 * SET UP STAE PARAMATER LIST TO ZERO TASK RESOURCE AREA AND CEATAB
 *   ENTRY
 * IF PATCH ID IS LESS THAN 5 THEN  (PATCH FROM ANALOG DISPLAY)
 *   GET BUFFER FOR PARTIAL SCREEN READ DATA
 *   PLACE PAGE NO. IN BUFFER
 *   PLACE ACCESS AREA NAME IN BUFFER
 *   DO UNTIL ALL ITEMS PROCESSED
 *     PLACE ITEM NO. IN BUFFER
 *     PLACE ITEM NAME IN BUFFER
 *   ENDDO
 * ENDIF
 * IF PATCH ID NE 10 THEN (CANCEL QUEUE PATCH)
 *   TURN ON BACKLITES OF PROGRAM FUNCTION KEYS
 * ENDIF
 * RETRIEVE SYSGEN PER CENT VALUE   (A)
 * CALCULATE N2       (100-A)
 * CALCULATE N1       (A-N2)
 * SAVE N2, N1, AND A
 * IF PATCH ID EQ 2 OR
 * IF PATCH ID EQ 6 THEN
 *   DO DOMCFIRS (FIRST HALF PAGE PROCESSING)
 * ELSE
 *   IF PATCH ID EQ 4 OR
 *   IF PATCH ID EQ 8 THEN
 *     DO DOMCSCND (SECOND HALF PAGE PROCESSING)
 *   ELSE
 *     IF PATCH ID EQ 10 THEN
 *       DO DOMCANCL (CANCEL PROCESSING)
 *     ELSE
 *       IF PATCH ID EQ 12 THEN
 *         DO DOMCUPDT (UPDATE SECTION)
 *       ELSE
 *         IF PATCH ID EQ 14 OR 16 THEN
 *           DO DOMCMOD1 (CONTROL OF STATUS SETTINGS)
 *         ENDIF
 *       ENDIF
 *     ENDIF
 *   ENDIF
 * ENDIF
 * EXIT         RETURN TO SYSTEM
 * END DOMCSANL
 *
 *
 * BEGIN DOMCFIRS        /* FIRST HALF OF PAGE */
 * SET DATA POINTERS TO POINT TO FIRST HALF PAGE OF DATA READ INTO
 *    BUFFER
 * SET INDICATOR THAT PROCESSING IS FOR FIRST HALF PAGE
 * DO PAGEBLD (BUILD AND DISPLAY PAGE)
 * END DOMCFIRS
 *
 * BEGIN DOMCSCND      /* SECOND  HALF OF PAGE */
 * SET DATA POINTERS TO POINT TO SECOND HALF OF DATA READ INTO BUFFER
 * SET INDICATOR THAT PROCESSING IS FOR SECOND HALF PAGE
 * DO PAGEBLD (BUILD AND DISPLAY PAGE)
 * END DOMCSCND
 *
```

```
MEMBER NAME  DOMCSANL
 * BEGIN DOMCANCL       /* CANCEL */
 * IF DISPLAY NAME NE TO 'DOMDSANL' THEN
 *   FREE INPUT DATA BUFFER AREA
 * ENDIF
 * SUBTRACT ONE FROM TASK COUNTER
 * IF DISPLAY NAME NE TO A SENSOR BASED DATA DISPLAY THEN
 *   FREE AREA USED FOR REMOTE CONTROL ELEMENT (RCE)
 *   IF TASK COUNTER IS ZERO THEN
 *     DPATCH TASK IF WORK QUEUE IS EMPTY
 *   ENDIF
 * ENDIF
 * END DOMCANCL
 *
 *
 * BEGIN DOMCUPDT
 * IF    DATA READ FROM SCREEN OR
 * IF LENGTH OF DATA READ IS NZERO THEN
 *   BUILD RCB NAME
 *   GET RCB ADDRESS
 *   GET NAMES LIST ADDRESS
 *   GET AND SAVE DISPLAY UNIT ACCESS AREA AND FUNCTION CODES
 *   GET AND SAVE ITEM ACCESS AREA CODE
 *   IF MASTER DISPATCHER ACCESS AREA CODE EQ UNIT ACCESS AREA CODE OR
 *   IF ITEM ACCESS AREA CODE EQ UNIT ACCESS AREA CODE THEN
 *     GET BUFFER FOR STATUS MESSAGES
 *     GET STATUS MESSAGES
 *     DO UNTIL ALL ITEMS PROCESSED
 *       IF FIRST BYTE OF VALUES NE  UNDERSCORE THEN
 *         IF FIRST BYTE OF VALUES NE TO BLANK THEN
 *           GET ADDRESS OF ITEM
 *           TURN ERROR FLAG OFF
 *           FIND OFFSET INTO NAMES LIST AND RETRIEVE FUNCTION CODE OF
 *              ITEM
 *           IF MASTER DISPATCHER FUNCTION CODE EQ DISPLAY UNIT FUNCTION
 *              CODE OR
 *           IF ITEM FUNCTION CODE EQ DISPLAY UNIT FUNCTION CODE
 *           ELSE
 *             TURN ERROR FLAG ON
 *             GENERATE SECURITY EVENT TO DISPLAY UNIT ACCESS AREA AND
 *                FUNCTION AND TO ITEM ACCESS AREA AND FUNCTION
 *             PLACE NO UNDER STATUS MESSAGE FOR FUNCTION CODE ERROR
 *           ENDIF
 *           IF ITEM IS NOT OFFLINE THEN
 *             TURN ERROR FLAG ON
 *             PLACE NO. WITH STATUS MESSAGE FOR OFFLINE ERROR
 *           ENDIF
 *           IF INPUT DATA IS NOT VALID THEN
 *             TURN ERROR FLAG ON
 *             PLACE NO. WITH STATUS MESSAGE FOR BAD INPUT DATA
 *           ENDIF
 *           IF ERROR FLAG ON THEN
 *           ELSE
 *             TURN UPDATE FLAG OFF
 *             IF NEW A COEFF ENTERED THEN
 *               CONVERT TO FLOATING POINT
 *               PLACE VALUE IN DATA BASE
 *               TURN UPDATE FLAG ON
 *               GENERATE AN EVENT
```

```
MEMBER NAME  DOMCSANL
 *            ENDIF
 *            IF NEW B COEFF ENTERED THEN
 *              CONVERT TO FLOATING POINT
 *              PLACE VALUE IN DATA BASE
 *              TURN UPDATE FLAG ON
 *              GENERATE AN EVENT
 *            ENDIF
 *            IF NEW HIGH LIMIT ENTERED THEN
 *              CONVERT TO FLOATING POINT
 *              PLACE VALUE IN BUFFER
 *              TURN UPDATE FLAG ON
 *              GENERATE AN EVENT
 *            ENDIF
 *            IF NEW LOW LIMIT ENTERED THEN
 *              CONVERT TO FLOATING POINT
 *              PLACE VALUE IN BUFFER
 *              TURN UPDATE FLAG ON
 *              GENERATE AN EVENT
 *            ENDIF
 *            IF HIGH OR LOW OPERATING LIMITS ENTERED THEN
 *              IF NEW HIGH LIMIT ENTERED THEN
 *                CALCULATE     HIGH OPERATING LIMIT IN RAW DATA VALUE
 *                  (HOLRV = HOLEV - B/A) /* A= SYSGEN PER CENT */
 *                                        /* B= B COEFF        */
 *                                        /*RV= RAW VALUE      */
 *                                        /*EU= ENG? UNITS     */
 *                                        /*N2= 100-A          */
 *                                        /*N1= A-N2           */
 *                                        /*RW= WARNING RANGE  */
 *                IF HIGH OPER LIMIT GT TRANSDUCER LIMIT THEN
 *                  INSERT TRANSDUCER VALUE IN PLACE OF HOL
 *                  PLACE ITEM NO. IN STATUS MESSAGE OF HOL VALUE B
 *                      REPLACED BY TRANSDUCER VALUE
 *                ENDIF
 *              ELSE
 *                CALCULATE RAW VALUE FROM HIGH WARNING LIMIT
 *                  (HOLRV = HWL + N2(RW/N1))
 *              ENDIF
 *              IF LOW OPERATING LIMIT NEW THEN
 *                CALCULATE LOW OPERATING LIMIT IN RAW VALUE FORM
 *                  (LOLRV = LOLEU - B/A)
 *                IF LOW OPER. LIMIT LT TRANSDUCER VALUE THEN
 *                  INSERT TRANSDUCER VALUE IN PLACE OF LOL
 *                  PLACE ITEM NO. IN STATUS MESSAGE OF LOL VALUE
 *                      REPLACED BY TRANSDUCER VALUE
 *                ENDIF
 *              ELSE
 *                CALCULATE RAW VALUE FROM LOW WARNING LIMIT
 *                   (LOLRV = LWL-N2(RW/N1))
 *              ENDIF
 *              CALCULATE HIGH WARNING LIMIT AND REPLACE IN DATA BASE
 *               (HW = A(H-L)+L)
 *              CALCULATE LOW WARNING LIMIT AND REPLACE IN DATA BASE
 *               (LW = N2(H-L)+L
 *            ENDIF
 *            IF UPDATE FLAG ON THEN
 *              PLACE ITEM NO. IN STATUS MESSAGE FOR SUCCESSFUL UPDATES
 *            ENDIF
```

```
MEMBER NAME  DOMCSANL
 *              ZERO UPDATE FLAG
 *            ENDIF
 *          ENDIF
 *        ENDIF
 *        UPDATE COUNTERS AND CONTROL TO NEXT ITEM
 *      ENDDO
 *      IF ANY STATUS MESSAGE HAS NO ITEM NO'S THEN
 *        PLACE-NO ERRORS FOUND- AFTER MESSAGE
 *      ENDIF
 *      IF ANY ITEM UPDATED SUCCESSFULLY THEN
 *        IF FIRST HALF OF PAGE ON SCREEN THEN
 *          DO DOMCFIRS (REFRESH FIRST HALF OF PAGE)
 *        ELSE
 *          DO DOMCSCND (REFRESH SECOND HALF OF PAGE)
 *      ENDIF
 *      PUT STATUS MESSAGES TO SCREEN
 *      PLACE TEXT OF LAST MESSAGE IN SCRATCH PAD ZONE
 *      FREE MESSAGE BUFFER
 *    ELSE
 *      GENERATE SECURITY EVENT TO UNIT ACCESS AREA AND UNIT FUNCTION
 *         CODE
 *      GENERATE SECURITY EVENT TO ITEM ACCESS AREA AND FUNCTION CODE
 *      GET MESSAGE FOR SCRATCH PAD ZONE
 *    ENDIF
 *  ELSE
 *    GET MESSAGE FOR SCRATCH PAD ZONE THAT PSREAD WAS INCORRECT
 *  ENDIF
 *  WRITE MESSAGE TO SCRATCH PAD ZONE
 *  END DOMCUPDT
 *
 *
 *  BEGIN DOMCMOD1 (MODIFY STATUS CONDITIONS)
 *  GET ITEM ADDRESS
 *  GET RCB ADDRESS
 *  GET AND SAVE ITEM ACCESS AREA ID
 *  GET AND SAVE DISPLAY UNIT ACCESS AREA ID
 *  GET AND SAVE DISPLAY UNIT FUNCTION CODE
 *  IF MASTER DISPATCHER ACCESS AREA EQ DISPLAY UNIT ACCESS AREA OR
 *  IF ITEM ACCESS AREA EQ DISPLAY UNIT ACCESS AREA THEN
 *    IF MASTER DISPATCHER FUNCTION CODE EQ DISPLAY UNIT FUNCTION CODE OR
 *    IF ITEM FUNCTION CODE EQ DISPLAY UNIT FUNCTION CODE THEN
 *      IF ITEM IS OFFLINE (OUT OF SERVICE) THEN
 *        IF PATCH ID IS 14 THEN
 *          REVERSE ALARM BIT SETTING
 *        ELSE
 *          REVERSE USER CONVERSION FLAG BIT
 *        ENDIF
 *        IF FIRST HALF OF PAGE ON SCREEN THEN
 *          DO DOMCFIRS (REFRESH FIRST HALF OF PAGE)
 *        ELSE
 *          DO DOMCSCND (REFRESH SECOND HALF OF PAGE)
 *        ENDIF
 *        GENERATE DATA EVENT INDICATING UNID, ITEM NAME, AND NEW
 *            STATUS CONDITION
 *        PLACE SUCCESSFUL UPDATE MESSAGE IN SCRATCH PAD ZONE
 *      ELSE
 *        PLACE MESSAGE FOR NOT OFFLINE IN SCRATCH PAD ZONE
 *      ENDIF
```

```
MEMBER NAME  DOMCSANL
 *    ELSE
 *      GENERATE SECURITY EVENT TO ACCESS AREA AND FUNCTION CODE OF
 *         DISPLAY UNIT AND ACCESS AREA AND FUNCTION CODE OF ITEM
 *      PLACE FUNCTION CODE-NO MATCH MESSAGE IN SCRATCH PAD ZONE
 *    ENDIF
 *  ELSE
 *    GENERATE SECURITY EVENT TO ACCESS AREA AND FUNCTION CODE OF DISPLAY
 *       UNIT AND ACCESS AREA AND FUNCTION CODE OF ITEM
 *    PLACE NO MATCH ACCESS AREA MESSAGE IN SCRATCH PAD ZONE
 *  ENDIF
 *  WRITE SCRATCH PAD ZONE TO SCREEN
 *  END DOMCMOD1
 *
 *
 *  BEGIN PAGEBLD
 *  GET BUFFER TO BUILD PAGE OF DATA
 *  SET UP ADDRESS TO MONITOR LOCATION IN BUFFER FOR EACH ITEM
 *  IF FIRST HALF OF PAGE  THEN
 *    PLACE FIRST IN HEADER
 *  ELSE
 *    PLACE SECOND IN HEADER
 *  ENDIF
 *  IF ITEM NO.'S NOT BLANK THEN
 *    DO UNTIL ITEM NO'S BLANK OR EIGHT ITEMS PROCESSED
 *      CALCULATE HIGH OPERATING LIMIT /*SEE DOMCUPDT FOR SYMBOL EQUATES
 *         (HOL =((HWL)+(((HWL-LWL)/N1)(N2)))) THEN AX+B
 *      CALCULATE LOW WARNING LIMIT
 *         (LOL =((LWL)-(((HWL-LWL)/N1)(N2)))) THEN AX+B
 *      PLACE BOTH OPERATING LIMITS IN BUFFER
 *      PLACE A AND B COEFFICIENTS IN BUFFER
 *      PLACE ALARMABLE/NOT ALARMABLE INDICATORS IN BUFFER
 *      PLACE USER CONVERSION/NO USER CONVERSION IN BUFFER
 *      INCREASE ADDRESSESS TO NEXT ITEM NO AND NAME
 *    ENDDO
 *  ENDIF
 *  IF DISPLAY NAME NE DOMDSANL THEN
 *    INCREASE TASK COUNTER BY ONE
 *    BRING BACKGROUND TO SCREEN
 *  ENDIF
 *  DISPLAY FIRST OR SECOND IN HEADER
 *  DISPLAY PAGE NO.
 *  DISPLAY ACCESS AREA NAME
 *  DISPLAYSUBSTATION/REMOTE NAME
 *  IF ITEM NOS NOT EQUAL TO BLANK THEN
 *    DISPLAY ITEM NO.S
 *            ITEM NAMES
 *            CHARACTERS 'HIGH'
 *            CHARACTERS 'LOW'
 *            TAB CHARACTERS
 *            UNDERSCORE CHARACTERS
 *           HIGH OPERATING LIMIT
 *           LOW OPERATING LIMITS
 *           END CHARACTERS
 *           CHARACTER 'A'
 *           CHARACTER 'B'
 *           TAB CHARACTERS
 *           UNDERSCORE CHARACTERS
 *           A COEFFICIENT
```

```
MEMBER NAME  DOMCSANL
 *          B COEFFICIENT
 *          END CHARACTERS
 *          TAB CHARACTER
 *          N/A
 *          N/C
 *    BLANK STATUS AREA AND SCRATCH PAD ZONE
 * ELSE
 *    DISPLAY MESSAGE IN SCRATCH PAD ZONE OF NO DATA DEFINED FOR THIS PG.
 * ENDIF
 * FREE BUFFER
 * END PAGEBLD
```

```
MEMBER NAME  DOMCSENT
 * DOMCSENT MAIN SEGMENT
 *    POINT TO CASYS ARRAY WITH ADDRESS FROM EMSCVT
 *    GET ADDRESS OF CASTATUS ARRAY USING GETARRAY MACRO
 *    ERROR EXIT TO ERR1 IF RETURN CODE IS NOT ZERO
 *    CALCULATE ADDRESS OF END OF ARRAY
 *    UNTIL ENTIRE ARRAY PROCESSED DO
 *      BYPASS HEADER IN STATUS GROUP
 *      FOR NUMBER OF ITEMS IN GROUP DO -(UNTIL-DO LOOP)
 *        SAVE TYPE INDICATORS
 *        IF TAG FLAG IS ON THEN
 *          BUILD ENTITY NAME
 *          ISSUE ENTITY WITH DOMTAGN ATTRIBUTE FROM CASYS ARRAY USING
 *           DISPENT MACRO
 *        ENDIF
 *        IF TYPE IS TCT TYPE1 THEN
 *        ELSE
 *          IF TYPE IS TCT TYPE3 THEN
 *            BUILD ENTITY NAME
 *            IF THE STATUS BIT IS ON THEN     (MANUAL)
 *              ISSUE ENTITY WITH DOMDEVC FROM CASYS USING DISPENT MACRO
 *            ENDIF
 *            ADJUST THE LOOP CONTROL COUNT TO BYPASS 2ND ITEM IN PAIR
 *          ELSE
 *            IF TYPE IS MOS THEN
 *              POINT TO SECOND ITEM IN EVEN-ODD PAIR
 *              IF STATUS BIT IS ON IN FIRST ITEM AND
 *              IF STATUS BIT IS ON IN SECOND ITEM THEN
 *                POINT TO DOMMOSH (ALARM ATTRIBUTE-HARDWARE)
 *              ELSE
 *                IF STATUS BIT IS OFF IN FIRST ITEM AND
 *                IF STATUS BIT IS OFF IN SECOND ITEM THEN
 *                  POINT TO DOMMOSS (ALARM ATTRIBUTE-STUCK)
 *                ELSE
 *                  IF STATUS BIT IS ON IN FIRST ITEM AND
 *                  IF STATUS BIT IS OFF IN SECOND ITEM THEN
 *                    POINT TO DOMDEVC (OPEN ATTRIBUTE)
 *                  ELSE
 *                    ZERO ENTITY POINTER
 *                  ENDIF
 *                ENDIF
 *              ENDIF
 *              IF ENTITY POINTER IS NOT ZERO THEN
 *                BUILD ENTITY NAME FROM FIRST ITEM
 *                ISSUE ENTITY USING DISPENT MACRO
 *              ENDIF
 *              ADJUST POINTER TO SECOND ITEM
 *              ADJUST CONTROL COUNT
 *            ELSE
 *              IF TYPE IS TCT TYPE 2 THEN
 *                IF STATUS BIT IS ON THEN
 *                  BUILD ENTITY NAME
 *                  ISSUE ENTITY USING DISPENT MACRO WITH DOMDEVC
 *                   ATTRIBUTE FROM CASYS ARRAY
 *                ELSE
 *                  IF STATUS BIT IS OFF AND
 *                  IF STATUS NAME IS NOT BLANK THEN
 *                    BUILD ENTITY NAME
 *                    ISSUE ENTITY USING DISPENT MACRO AND DOMDEVC
```

```
MEMBER NAME  DOMCSENT
 *                      ATTRIBUTE FROM CASYS ARRAY
 *                   ENDIF
 *                 ENDIF
 *               ENDIF
 *             ENDIF
 *           ENDIF
 *         ENDIF
 *         POINT TO NEXT STATUS ITEM
 *       ENDDO
 *     ENDDO
 *     EXIT WITH RETURN CODE SET TO ZERO
 *
 *     ERROR ENTER ERR1
 *       ISSUE MESSAGE 310 USING MESSAGE MACRO
 *       SET RETURN CODE TO 4
 *       EXIT WITH RETURN CODE SET TO FOUR
 * ENDSEGMENT DOMCSENT
```

```
MEMBER NAME  DOMCSGET
 * BEGIN DOMCSGET
 * /* THIS CSECT CONTROLS THE RCB MENU DISPLAY
 * IF PATCH ID NE 40 THEN
 *   TURN ON PROGRAM FUNCTION KEY LITES
 * ENDIF
 * IF PATCH ID EQ 10 THEN
 *   DO DOMCINIT (INITIALIZATION PROCESSING)
 * ELSE
 *   IF PATCH ID EQ 20 THEN
 *     DO DOMCPGFD (PAGE FORWARD PROCESSING)
 *   ELSE
 *     IF PATCH ID EQ 25 THEN
 *       DO DOMCPGBK (PAGE BACKWARD PROCESSING)
 *     ELSE
 *       IF PATCH ID EQ 40 THEN
 *         DO DOMCANCL (CANCEL PROCESSING
 *       ENDIF
 *     ENDIF
 *   ENDIF
 * ENDIF
 * EXIT
 * END DOMCSGET
 *
 * BEGIN DOMCINIT SEGMENT
 * CALCULATE NO. RCB ENTRIES
 * CALCULATE NO. PAGES FOR DISPLAY
 * STORE PAGE NO. IN RCE
 * STORE DATA ADDRESSES IN RCE (START,CURRENT,LAST)
 * ADD ONE TO TASK COUNTER
 * BRING DISPLAY BACKGROUND TO SCREEN
 * DO RMTBUILD (PAGE BUILD)
 * END SEGMENT DOMCINIT
 *
 * BEGIN DOMCPGFD SEGMENT
 * IF CURRENT PAGE NO. EQ LAST PAGE NO. THEN
 *   CURRENT PAGE NO. EQ ONE
 *   CURRENT DATA ADDR. EQ START DATA ADDR.
 * ELSE
 *   INCREASE CURRENT PAGE NO. BY ONE
 * ENDIF
 * DO RMTBUILD (PAGE BUILD)
 * END DOMCPGFD SEGMENT
 *
 * BEGIN DOMCPGBK SEGMENT
 * IF CURRENT PAGE NO. EQ 1 THEN
 *   CURRENT PAGE NO. EQ LAST PAGE NO.
 *   CALCULATE DATA ADDRESS
 * ELSE
 *   SUBTRACT ONE FROM PAGE NO.
 *   CALCULATE DATA ADDRESS
 * ENDIF
 * DO RMTBUILD (PAGE BUILD)
 * END SEGMENT DOMCPGBK
 *
 * BEGIN SEGMENT DOMCANCL
 *
 * SUBTRACT FROM TASK COUNTER
 * CANCEL TASK IF DISPLAY NAME ON SCREEN DOES NOT BELONG TO THIS TASK
```

```
MEMBER NAME  DOMCSGET
 * END SEGMENT DOMCANCL
 *
 * BEGIN SEGMENT RMTBUILD
 * GET BUFFER
 * PLACE DATA TO BE DISPLAYED IN BUFFER WITH POINTERS MAINTAINED
 * PUT TIME/DATE TO SCREEN
 * PUT ALL DATA FROM BUFFER TO SCREEN
 * FREE BUFFER AREA
 * END SEGMENT RMTBUILD
 *
```

```
MEMBER NAME  DOMCSLOG
 * DOMCSLOG MAIN SEGMENT
 *    SET END OF FILE INDICATOR IN STAE PARAMETER LIST
 *    GET ADDRESS OF CASTLOG ARRAY FROM EMSCVT
 *    STORE INFORMATION ABOUT PARAMETER LIST IN STAE LIST
 *    CALCULATE NUMBER OF ITEMS IN PARAMETER LIST
 *    FIND STARTING ADDRESS IN ARRAY
 *    IF ARRAY IS FULL AND WAS LOGGED THEN
 *      REINITIALIZE POINTERS
 *      POINT TO BEGINNING OF ARRAY PAST HEADER
 *    ENDIF
 *    MOVE TIME AND DATE TO ARRAY
 *    ADJUST COUNTS IN BEGINNING OF ARRAY
 *    POINT TO NEXT BUFFER IN CHAIN PASSED
 *    UNTIL ALL ITEMS IN BUFFER(S) PROCESSED DO
 *      CALCULATE HOW MANY ITEMS FIT IN ARRAY
 *      IF THE NUMBER OF ITEMS IS GREATER THAN THE REMAINING SPACE THEN
 *        SAVE THE REMAINING NUMBER AFTER SUBTRACTING
 *      ELSE
 *        ZERO REMAINING NUMBER COUNT
 *      ENDIF
 *      STORE THE NUMBER OF ITEMS IN THE ARRAY
 *      ADJUST POINTER TO ARRAY
 *      UNTIL ALL ITEMS WHICH FIT IN ARRAY PROCESSED DO
 *        MOVE STATUS ITEM FROM BUFFER TO ARRAY
 *        ADJUST POINTER TO ARRAY AND TO BUFFER
 *        ADJUST ARRAY CONTROL COUNTS
 *      ENDDO
 *      IF ANY ITEMS REMAIN THEN
 *        ADJUST CONTROL COUNTS
 *        IF ARRAY IS FULL THEN
 *          DO DOMBLOCK
 *          DO DOMLOG
 *        ENDIF
 *      ENDIF
 *    ENDDO
 *    IF ARRAY IS FULL THEN
 *      DO DOMBLOCK
 *      DO DOMLOG
 *    ELSE
 *      DO DOMBLOCK
 *    ENDIF
 *    WHILE THERE ARE BUFFERS TO BE FREED DO
 *      FREE BUFFER USING FREEWA MACRO
 *      POINT TO NEXT BUFFER
 *    ENDDO
 *    SET END OF LIST INDICATOR IN STAE LIST
 *    EXIT
 * ENDSEGMENT DOMCSLOG
 *
 * DOMBLOCK SUBROUTINE SEGMENT
 *    PUT THE ARRAY TO THE DASTLOG ARRAY ON DISK USING PUTBLOCK MACRO
 * ENDSEGMENT DOMBLOCK
 *
 * DOMLOG SUBROUTINE SEGMENT
 *    PUT CASTLOG USING THE PUTLOG MACRO
 *    PUT CASTATUS USING THE PUTLOG MACRO
 * ENDSEGMENT DOMLOG
```

```
MEMBER NAME  DOMCSPCD
* BEGIN DOMCSPCD
* /* THIS IS THE CONTROL PROGRAM FOR DISPLAYS DOMDSPC1 AND DOMDSPC2 */
* TAKE ERROR EXIT IF CEATAB (CONTROL ELEMENT ADDRESS TABLE) AREA IS
*    ZERO
* TURN ON BACKGROUND LITES FOR PROGRAM FUNCTION KEYS
* IF PATCH ID EQ 10 THEN
*   DO DOMCSPIN SECTION (INITIALIZATION PROCESSING)
* ELSE
*   IF PATCH ID EQ 15 THEN
*     DO DOMCSPRN SECTION (RE-INITIALIZATION PROCESSING)
*   ELSE
*     IF PATCH ID EQ 20 THEN
*       DO DOMCSPGF SECTION (PAGE FORWARD PROCESSING)
*     ELSE
*       IF PATCH ID EQ 25 THEN
*         DO DOMCSPGB SECTION (PAGE BACKWARD SECTION)
*       ELSE
*         IF PATCH ID EQ 30 THEN
*           DO DOMCSPRF SECTION (REFRESH PROCESSING)
*         ELSE
*           IF PATCH ID EQ 35 THEN
*             DO DOMCSPUP SECTION (UPDATE SECTION)
*           ELSE
*             IF PATCH ID EQ 40 THEN
*               DO DOMCSPCL SECTION (CANCEL PROCESSING)
*             ENDIF
*           ENDIF
*         ENDIF
*       ENDIF
*     ENDIF
*   ENDIF
* ENDIF
* END DOMCSPCD AND RETURN TO SYSTEM
*
*
* BEGIN DOMCSPIN SEGMENT
* GET ADDRESS OF PARTIAL SCREEN READ BUFFER
* SAVE RDACS ID
* SAVE S/7 ID
* SAVE ACCESS AREA NAME
* CONVERT BLANKS IN IDS TO ZEROS
* DO INITIAL SEGMENT (SET UP POINTERS FOR FIRST PAGE DOMDSPC1)
* END DOMCSPIN SEGMENT
*
*
* BEGIN DOMCSPRN SEGMENT
* SET UP POINTERS WITHOUT OTHER PROCESSING IN DOMCSPIN SEGMENT
* DO INTIAL SEGMENT (SET UP POINTERS FOR FIRST PAGE    DOMDSPC1)
* END DOMCSPRN SEGMENT
*
*
* BEGIN DOMCSPGF SEGMENT
*   IF DIRECT PAGE REQUEST AREA FILLED THEN
*     IF CHARACTERS NOT VALID THEN
*       PUT OUT ERROR MESSAGE TO SCREEN
*     ELSE
*       IF NEW PAGE NO. GREATER THAN TOTAL THEN
*         PUT OUT ERROR MESSAGE
```

```
MEMBER NAME  DOMCSPCD
*        ELSE
*          CONVERT DIGITS TO BINARY
*          CALCULATE CURRENT ITEM ADDRESS
*          PLACE NEW PAGE NO. IN RCE
*          DO PCBUILD SEGMENT (BUILD PC DATA PAGE & DISPLAY)
*        ENDIF
*      ENDIF
*    ELSE
*      IF CURRENT PAGE NO EQ TOTAL PAGE NO. THEN
*        SET CURRENT PAGE NO EQ ZERO
*      ENDIF
*      INCREASE CURRENT PAGE COUNT BY ONE
*      CALCULATE CURRENT ITEM ADDRESS
*      DO PCBUILD SEGMENT (BUILD PC DATA PAGE & DISPLAY)
*    ENDIF
*  END DOMCSPGF SEGMENT
*
*  BEGIN DOMCSPGB SEGMENT
*    IF DIRECT PAGE REQUEST AREA FILLED THEN
*      IF CHARACTERS NOT VALID THEN
*        PUT OUT ERROR MESSAGE TO SCREEN
*      ELSE
*        IF NEW PAGE NO. GREATER THAN TOTAL THEN
*          PUT OUT ERROR MESSAGE
*        ELSE
*          CONVERT DIGITS TO BINARY
*          CALCULATE CURRENT ITEM ADDRESS
*          PLACE NEW PAGE NO. IN RCE
*          DO PCBUILD SEGMENT (BUILD PC DATA PAGE & DISPLAY)
*        ENDIF
*      ENDIF
*    ELSE
*      IF CURRENT PAGE NO. EQ ONE THEN
*        CURRENT PAGE NO EQ TOTAL PAGE NO.
*      ENDIF
*      SUBTRACT ONE FROM CURRENT PAGE
*      CALCULATE START OF FIRST ITEM ADDRESS FOR CURRENT PAGE
*      DO PCBUILD SEGMENT (BUILD AND DISPLAY DATA PAGE)
*    ENDIF
*  END DOMCSPGB SEGMENT
*
*
*  BEGIN DOMCSPRF SEGMENT
*    CALCULATE START ADDRESS OF FIRST ITEM ON PAGE
*    DO PCBUILD SEGMENT (BUILD AND DISPLAY A DATA PAGE)
*  END DOMCSPRF SEGMENT
*
*
*  BEGIN DOMCSPUP SEGMENT
*  ZERO INDICATOR FLAGS
*  IF PARTIAL SCREEN READ DATA IS NOT BLANK THEN
*    IF LENGTH OF PARTIAL SCREEN READ NOT ZERO THEN
*      GET ADDRESS OF RCB ITEM
*      IF ACCESS AREA OF ITEM EQUALS PRIMARY OR SECONDARY ACCESS AREA OF
*         DISPLAY UNIT OR MATER DISPATCHER ACCESS AREA THEN
*        GET STATUS MESSAGES FOR UPDATE STATUS RESULTS AND PLACE IN
*           BUFFER
*        DO EACH ITEM WHICH HAS NEW DATA ENTERED
```

```
MEMBER NAME  DOMCSPCD
*        IF ITEM FUNCTION CODE NOT EQUAL TO FUNCTION CODE1 OF DISPLAY
*           UNIT OR FUNCTION CODE OF MASTER DISPATCHER THEN
*          GENERATE A SECURITE EVENT TO ACCESS AREA AND FUNCTION CODE OF
*             DISPLAY UNIT AND TO ACCESS AREA AND FUNCTION CODE OF ITEM
*          PLACE ITEM NO. IN STATUS MESSAGE FOR BAD FUNCTION CODES
*        ENDIF
*        IF ITEM IS NOT OFFLINE (OUT OF SERVICE) THEN
*          PLACE ITEM NO. IN STATUS MESSAGE FOR ITEMS NOT OFFLINE
*        ENDIF
*        IF NEW INPUT DATA FOR EACH ITEM IS NOT VALID THEN
*          PLACE ITEM NO IN STATUS MESSAGE FOR BAD INPUT DATA
*        ENDIF
*        IF ITEM FLAGGED IN ERROR (ONE OR MORE OF ABOVE CONDITIONS) THEN
*        ELSE
*          SET FLAG TO REFRESH
*          IF CURRENT VALUE NEW THEN
*            CONVERT EBCDIC NO. TO FLOATING POINT
*            EVENT UPDATE OF CURRENT VALUE.
*            PLACE CURRENT VALUE IN DATA BASE
*          ENDIF
*          IF ACCUMULATED VALUE NEW THEN
*            CONVERT EBCDIC NO. TO FLOATING POINT
*            EVENT UPDATE OF ACCUMULATED VALUE
*            PLACE ACCUMULATED VALUE IN DATA BASE
*          ENDIF
*          IF INCREMENT VALUE IS NEW THEN
*            CONVERT EBCDIC NO. TO FLOATING POINT
*            EVENT UPDATE OF INCREMENT VALUE
*            PLACE INCREMENT VALUE IN DATA BASE
*          ENDIF
*        ENDIF
*        IF UPDATE FLAG IS ON THEN
*          PLACE ITEM NO IN STATUS MESSAGE FOR SUCCESSFUL ITEM UPDATES.
*        ENDIF
*        ENDDO
*        IF ANY ITEM UPDATED SUCCESSFULLY THEN
*          DO DOMCSPRF SEGMENT (REFRESH SCREEN
*        ENDIF
*        PUT STATUS MESSAGES TO SCREEN
*      ELSE
*        GENERATE A SECURITY EVENT FOR ATTEMPTED UPDATE TO ACCESS AREA
*           NOT ALLOWED - EVENT TO ACCESS AREA AND FUNCTION CODE OF
*           DISPLAY UNIT AND TO ACCESS AREA AND FUNCTION CODE OF
*           ITEM
*        PLACE MESSAGE IN SCRATCH PAD ZONE
*      ENDIF
*    ELSE
*      GENERATE MESSAGE FOR SCRATCH PAD ZONE FOR INCORRECT PARTIAL
*         SCREEN READ
*    ENDIF
*  ELSE
*    GENERATE MESSAGE FOR SCRATCH PAD ZONE FOR INCORRECT PARTIAL SCREEN
*       READ
*  ENDIF
*  WRITE MESSAGE TO SCRATCH PAD ZONE
*  END DOMCSPUP SEGMENT
*
*
```



```
MEMBER NAME  DOMCSPCD
* BEGIN DOMCSPCL SEGMENT
* IF DISPLAY NAME IS NE TO A SENSOR BASED DATA DISPLAY NAME
*   FREE RCE AREA
* ENDIF
* END DOMCSPCL SEGMENT
*
*
* BEGIN INITIAL SEGMENT
* CALCULATE NO. OF PAGES FOR DISPLAY
* INCREASE PAGE NO. BY ONE FOR FIRST PAGE
* GET START DATA ITEM ADDR.
* CURRENT ITEM ADDRESS EQ START ITEM ADDR
* CALCULATE ADDR. OF START OF LAST DATA ITEM
* DO PCBUILD SEGMENT (BUILD PAGE ONE DATA AND DISPLAY)
* END INITIAL SEGMENT
*
*
* BEGIN PCBUILD SEGMENT
* DO FOR EACH ITEM ON PAGE
*   PLACE NAME IN BUFFER
*   PLACE FUNCTION NAME IN BUFFER
*   PLACE CURRENT VALUE IN BUFFER
*   PLACE ACCUMULATED VALUE IN BUFFER
*   PLACE INCREMENT VALUE IN BUFFER
*   PLACE STATUS INFORMATION IN BUFFER
*   PLACE TYPE IN BUFFER
*   BUMP ITEM ADDRESS TO NEXT ITEM
* ENDDO
* IF DISPLAY NAME NE DOMDSPC2 THEN
*   INCREASE TASK COUNTER BY ONE
*   PUT BACKGROUND TO SCREEN
* ENDIF
* WRITE DATA IN BUFFER TO SCREEN
* END PCBUILD SEGMENT
*
*
```

Former copy has been moved or deleted.

```
MEMBER NAME  DOMCSOMD
 * BEGIN DOMCSOMD
 * IF LFC IN SYSTEM THEN
 *   TURN ON LITES FOR KEY 10
 * ENDIF
 * IF EDC IN SYSTEM THEN
 *   TURN ON LITE FOR KEY 9
 * ENDIF
 * TURN ON OTHER SYSTEM KEY LITES
 * IF USE- FIELD IS ZERO THEN
 *   GETMAIN AREA FOR CEATAB
 *   PLACE ADDRESS IN DCE USER FIELD
 * ENDIF
 * END DOMCSOMD
```

```
MEMBER NAME  DOMCSSTA
* BEGIN DOMCSSTA
* /* STATUS DATA DISPLAY CONTROL PROGRAM */
* IF CONTROL ELEMENT ADDRESS TABLE IS ZERO (CEA) THEN
*   ERREXIT - UNABLE TO BRING UP DISPLAY
* ENDIF
* IF PATCH ID NE 40 THEN (CANCEL PATCH)
*   TURN ON BACK LITES FOR PROGRAM FUNCTION KEYS
* ENDIF
* IF PATCH ID EQ 10 THEN
*   DO DOMCSSIN (INITIALIZATION SEGMENT)
* ELSE
*   IF PATCH ID EQ 15 THEN
*     DO DOMCSSRN (RE-INITIALIZATION SEGMENT)
*   ELSE
*     IF PATCH ID EQ 20 THEN
*       DO DOMCSSPF (PAGE FORWARD SEGMENT)
*     ELSE
*       IF PATCH ID EQ 25 THEN
*         DO DOMCSSPB (PAGE BACKWARD SEGMENT)
*       ELSE
*         IF PATCH ID EQ 30 THEN
*           DO DOMCSSRF (REFRESH SEGMENT)
*         ELSE
*           IF PATCH ID EQ 36 OR
*           IF PATCH ID EQ 35 THEN
*             DO DOMCSSUP (DATA UPDATE PROCESSING SEGMENT)
*           ELSE
*             IF PATCH ID EQ 40 THEN
*               DO DOMCSSCL (CANCEL SEGMENT)
*             ENDIF
*           ENDIF
*         ENDIF
*       ENDIF
*     ENDIF
*   ENDIF
* ENDIF
* EXIT AND RETURN TO SYSTEM
* END DOMCSSTA
*
*
* BEGIN DOMCSSIN SEGMENT
* PUT RDACS ID IN RCE FROM SCREEN READ DATA
* PUT S/7 ID IN RCE FROM SCREEN READ DATA
* PLACE ACCESS AREA IN RCE FROM SCREEN READ DATA
* DO INITIAL (PAGE ONE INITIAIZATION)
* END DOMCSSIN
*
*
* BEGIN DOMCSSRN
* SET UP POINTERS FOR AID, CEA, AND RCE ADDRESSES
* DO INITIAL (PAGE ONE INITIALIZATION)
* END DOMCSSRN
*
*
* BEGIN DOMCSSPF
*   IF DIRECT PAGE REQUEST AREA FILLED THEN
*     IF CHARACTERS NOT VALID THEN
*       PUT OUT ERROR MESSAGE TO SCREEN
```

```
MEMBER NAME  DOMCSSTA
*      ELSE
*        IF NEW PAGE NO. GREATER THAN TOTAL THEN
*          PUT OUT ERROR MESSAGE
*        ELSE
*          CONVERT DIGITS TO BINARY
*          CALCULATE CURRENT ITEM ADDRESS
*          PLACE NEW PAGE NO. IN RCE
*          DO PCBUILD SEGMENT (BUILD PC DATA PAGE & DISPLAY)
*        ENDIF
*      ENDIF
*    ELSE
*      IF CURRENT PAGE NO. EQ TOTAL PAGE NO. THEN
*        SET CURRENT PAGE NO. TO ZERO
*      ENDIF
*      INCREASE PAGE COUNT BY ONE
*      CALCULATE CURRENT ITEM ADDRESS
*      DO STBUILD (BUILD STATUS DATA DISPLAY PAGE)
*    ENDIF
*  END DOMCSSPF
*
*
*  BEGIN DOMCSSPB
*    IF DIRECT PAGE REQUEST AREA FILLED THEN
*      IF CHARACTERS NOT VALID THEN
*        PUT OUT ERROR MESSAGE TO SCREEN
*      ELSE
*        IF NEW PAGE NO. GREATER THAN TOTAL THEN
*          PUT OUT ERROR MESSAGE
*        ELSE
*          CONVERT DIGITS TO BINARY
*          CALCULATE CURRENT ITEM ADDRESS
*          PLACE NEW PAGE NO. IN RCE
*          DO PCBUILD SEGMENT (BUILD PC DATA PAGE & DISPLAY)
*        ENDIF
*      ENDIF
*    ELSE
*      IF CURRENT PAGE NO EQ ONE
*        SET CURRENT PAGE NUMBER = TOTAL PAGES +1
*      ENDIF
*      SUBTRACT ONE FROM CURRENT PAGE
*      CALCULATE CURRENT ITEM ADDRESS
*      DO STBUILD (BUILD STATUS DATA DISPLAY PAGE)
*    ENDIF
*  END DOMCSSPB
*
*
*  BEGIN DOMCSSRF
*    CALCULATE START ADDRESS FOR FIRST ITEM ON PAGE
*    DO STBUILD (BUILD STATUS DATA DISPLAY PAGE)
*  END DOMCSSRF
*
*
*  BEGIN DOMCSSUP
*  ISSUE GETITEM BASED ON NAME RECEIVED FROM SCREEN
*  IF GETITEM SUCCESSFUL THEN
*    BUILD RCB NAME
*    ISSUE GETITEM
*    GET AND STORE RCB ACCESS AREA CODE
```

```
MEMBER NAME  DOMCSSTA
*   GET AND STORE STATUS ITEM FUNCTION CODE
*   GET AND STORE DISPLAY UNIT ACCESS AREA CODE AND FUNCTION CODE
*   IF DISPLAY UNIT ACCESS AREA CODES EQUALS MASTER DISPATCHER ACCESS
*       AREA CODE OR THE RCB ACCESS AREA CODE THEN
*     IF DISPLAY UNIT FUNCTION CODES EQUALS MASTER DISPATCHER FUNCTION
*         CODE OR THE ITEM FUNCTION CODE THEN
*       IF ITEM IS OFFLINE SELF OR OTHER THEN
*         IF PATCH ID EQ 35 THEN
*           REVERSE ALARMABLE BIT SETTING
*         ELSE
*           REVERSE CONTROLLABLE BIT SETTING
*         ENDIF
*         DO DOMCSSRF (REFRESH DISPLAY PAGE)
*         EVENT SUCCESSFUL UPDATE WITH A DATA EVENT
*         GET MESSAGE THAT UPDATE WAS SUCCESSFUL
*       ELSE
*         GET MESSAGE THAT ITEM NOT OFFLINE
*       ENDIF
*     ELSE
*       GENERATE A SECURITY EVENT FOR ATTEMPTED UPDATE TO FUNCTION AREA
*          OTHER THAN OWN
*       GET MESSAGE THAT FUNCTION CODE NOT CORRECT
*     ENDIF
*   ELSE
*     GENERATE SECURITY EVENT THAT UPDATE ATTEMPTED TO ACCESS AREA
*        OTHER THAN OWN
*     GET MESSAGE THAT ACCESS AREA NOT CORRECT
*   ENDIF
* ELSE
*   GET MESSAGE THAT DATA ITEM RETRIEVAL WAS UNSUCCESSFUL
* ENDIF
* DISPLAY MESSAGE TO SCREEN
* END DOMCSSUP
*
*
* BEGIN STBUILD
* DO FOR ALL ITEMS (16) IN GROUP
*   PLACE NAME IN BUFFER
*   PLACE FUNCTION NAME IN BUFFER
*   IF ITEM NAME EQ BLANKS THEN
*     PLACE'NOT WIRED' IN TYPES FIELD IN BUFFER
*   ELSE
*     PLACE TYPE NAME IN BUFFER
*   ENDIF
*   PLACE ASTERISKS IN COLUMNS FOR APPLICABLE STATUS CONDITIONS
* ENDDO
* IF DISPLAY NAME NE DOMDSST2 THEN
*   BRING BACKGROUND TO SCREEN
* ENDIF
* DISPLAY: PAGE NOS
*          DIRECT PAGING DATA
*          ACCESS AREA
*          SUBSTATION REMOTE NAME
*          ITEM NAMES
*          FUNCTION NAMES
*          TYPE
*          STATUS INFORMATION
* END STBUILD
```



```
MEMBER NAME  DOMCSSTA
*
*
* BEGIN INITIAL
* CALCULATE NUMBER PAGES FOR DATA AND STORE IN RCE
* SET CURRENT PAGE NUMBER EQUAL ONE
* PUT START ITEM ADDRESS IN RCE
* CURRENT ITEM ADDR. EQ START ITEM ADDRESS
* CALCULATE LAST ITEM ADDRESS AND PLACE IN RCE
* DO STBUILD (BUILD PAGE ONE DISPLAY DATA AND DISPLAY)
* END INITIAL
*
*
* BEGIN DOMCSSCL
* IF DISPLAY NAME NE SENSOR BASED DATA DISPLAY
*   FREE RCE BUFFER AREA
* ENDIF
* END DOMCSSCL
```

Former copy has been moved or deleted.

```
MEMBER NAME  DOMCSTAE
          BEGIN DOMCSTAE         ENERGY MANAGEMENT SYSTEM STAE PROCESSOR
          SEARCH SRTOS INPUT FOR ABENDING MODULE
          IF PARAMETER LIST PROVIDED BY OS,THEN
            DO SEGMENT MESSAGE
            DO SEGMENT CORRECT
            DO SEGMENT RETASK
            DO SEGMENT RETURN
          ELSE
            IF NO LIST BUT REGISTERS POINT TO INFO, THEN
              ESTABLISH POINTERS FROM REGISTERS
              DO SEGMENT MESSAGE
              DO SEGMENT CORRECT
              DO SEGMENT RETASK
              DO SEGMENT RETURN
            ELSE
              DO SEGMENT RETASK
              DO SEGMENT RETURN
            ENDIF
          ENDIF
          EXIT
 *
 **  FOLOWING ARE ALL OF THE SEGMENTS WHICH PERFORM FUNCTIONS
 *
          SEGMENT MESSAGE
          STORE ABEND CODE IN MESSAGE
          STORE ABENDING MODULE NAME AND TASK NAME IN MESSAGE
          COMPUTE DISPLACEMENT AND STORE IN MESSAGE
          WTO MESSAGE
          END SEGMENT MESSAGE
 *
          SEGMENT CORRECT
          PROCESS STAE PARAMETER LIST PROVIDED BY ABENDING TASK
            IF INDICATORS ARE ON,THEN
              ZERO AREA
              TURN OFF BITS
              TURN ON BITS
              FREEWA AREA (RUN CHAIN IF REQUESTED)
              FREEMAIN CORE (RUN CHAIN IF REQUESTED)
              RE-INITIALIZE DATA
            ENDIF
          ENDDO
          END SEGMENT CORRECT
 *
          SEGMENT RETASK
          PATCH DOMCTCBI UNDER ABENDING TASK NAME TO RE-ESTABLISH TASK
          END SEGMENT RETASK
 *
          SEGMENT RETURN
          IF ABEND WAS REQUESTED,THEN
            ALLOW ABEND
          ELSE
            ESTABLISH RETRY
          ENDIF
          END SEGMENT RETURN
 *
          SEGMENT RETRY
          FREEMAIN STAE WORK AREA
          END SEGMENT RETRY
```

```
MEMBER NAME  DOMCSTAE
 *
          END DOMCSTAE
```

```
MEMBER NAME  DOMCTCBI
 *   CSECT    DOMCTCBI                    /* TASK STAE INITIALIZATION   */
 *
 *     IF ENTRY TO CREATE THE TASK, THEN
 *       MOVE ADDRESS OF STAE PARAMETER LIST TO FIRST WORD RESOURCE TABLE
 *       IF THIS IS THE DATA ACQUISITION TASK, THEN
 *         PAGE FIX THE FIRST SAVE AREA
 *       ENDIF
 *     ELSE
 *       NO-OP STAE PARAMETER LIST
 *       ISSUE TASK STAE PASSING PARAMETER LIST TO EXIT PGM
 *     ENDIF
 *     EXIT
 *
 *   END      DOMCTCBI
```

```
MEMBER NAME  DOMCTIME
 *   DOMCTIME MAIN SEGMENT
 *     ESTABLISH ADDRESSABILITY FOR PATCH INPUT LIST, XCVT, PATCH PARAM
 *      LIST, AID INPUT, DCE AND EMSCVT
 *     ERREXIT TO THREE IF MORE THAN ONE PRIMARY S/7
 *     IF NOT AT TOP OF HIERARCHY
 *       ERREXIT ONE
 *     ELSE
 *       RETRIEVE MASTER DISPATCHER ACCESS AREA AND FUNCTION CODE
 *       IF REQJESTORS PRIMARY AA/FC DO NOT MATCH MASTER DISPATCHER AA/FC
 *         IF REQUESTORS SECONDARY AA/FC DO NOT MATCH MASTER DISPATCHER
 *          AA/FC
 *           IF REQUESTORS TERTIARY AA/FC DO NOT MATCH MASTER DISPATCHER
 *            AA/FC
 *             ERREXIT TWO
 *           ENDIF
 *         ENDIF
 *       ENDIF
 *       ESTABLISH ADDRESSABILITY TO PARTIAL SCREEN READ AREA
 *       IF FIRST POSITION OF INCREMENT IS NOT AN UNDERLINE CHARACTER
 *         IF FIRST POSITION OF INCREMENT IS BLANK
 *           ERREXIT TO THREE IF SECOND POSITION OF INCREMENT IS NON-
 *            NUMERIC
 *           ERREXIT TO THREE IF SECOND POSITION OF INCREMENT IS NON-
 *            INTEGER
 *           ERREXIT TO THREE IF FIRST POSITION OF DECREMENT IS NOT
 *            UNDERLINE CHARACTER
 *           ERREXIT TO THREE IF SECOND POSITION OF DECREMENT IS NOT
 *            BLANK
 *           SET INCREASE INDICATOR
 *           DO BUILD            BUILD S/7 TCR TRANSACTION
 *         ELSE                  FIRST POSITION INCREMENT NONBLANK
 *           IF SECOND POSITION OF INCREMENT IS BLANK
 *             SWITCH INCREMENT VALUE INTEGERS
 *           ENDIF
 *           ERREXIT TO THREE IF NONNUMERIC VALUE IN FIRST POSITION OF
 *            INCREMENT
 *           ERREXIT TO THREE IF FIRST POSITION OF INCREMENT IS NON-
 *            INTEGER
 *           ERREXIT TO THREE IF NONNUMERIC VALUE IN SECOND POSITION OF
 *            INCREMENT
 *           ERREXIT TO THREE IF SECOND POSITION OF INCREMENT IS NON-
 *            INTEGER
 *           ERREXIT TO THREE IF FIRST POSITION OF DECREMENT IS NOT
 *            UNDERLINE CHARACTER
 *           ERREXIT TO THREE IF SECOND POSITION OF DECREMENT IS NOT
 *            BLANK
 *           SET INCREASE INDICATOR
 *           DO BUILD            BUILD S/7 TCR TRANSACTION
 *         ENDIF
 *       ELSE                    FIRST POSITION INCREMENT UNDERLINE
 *        ERREXIT TO THREE IF SECOND POSITION OF INCREMENT IS NOT BLANK
 *        ERREXIT TO THREE IF FIRST POSITION OF DECREMENT IS UNDERLINE
 *         CHARACTER
 *        IF FIRST POSITION OF DECREMENT IS BLANK
 *          ERREXIT TO THREE IF SECOND POSITION OF DECREMENT IS NON-
 *           NUMERIC
 *          ERREXIT TO THREE IF SECOND POSITION OF DECREMENT IS NON-
 *           INTEGER
```

```
MEMBER NAME  DOMCTIME
 *          SET DECREASE INDICATOR
 *          DO BUILD                  BUILD S/7 TCR TRANSACTION
 *        ELSE                        FIRST POSITION DECREMENT NONBLANK
 *          IF SECOND POSITION OF DECREMENT IS BLANK
 *            SWITCH DECREMENT VALUE INTEGERS
 *          ENDIF
 *          ERREXIT TO THREE IF FIRST POSITION OF DECREMENT NONNUMERIC
 *          ERREXIT TO THREE IF FIRST POSITION OF DECREMENT NONINTEGER
 *          ERREXIT TO THREE IF SECOND POSITION OF DECREMENT NONNUMERIC
 *          ERREXIT TO THREE IF SECOND POSITION OF DECREMENT NONINTEGER
 *          SET DECREASE INDICATOR
 *          DO BUILD                  BUILD S/7 TCR TRANSACTION
 *        ENDIF
 *      ENDIF
 *    ENDIF
 *    EXIT
 *    ERRENTER ONE                    NOT TOP OF HIERARCHY
 *    MOVE MESSAGE NUMBER 257
 *    DO MESG                         RETRIEVE MESSAGE
 *    DO ZONE                         WRITE TO SCRATCH PAD ZONE
 *    ERRENTER TWO                    INVALID ACCESS AREA, FUNCTION CODE
 *    MOVE MESSAGE NUMBER 256
 *    DO MESG                         RETRIEVE MESSAGE
 *    DO ZONE                         WRITE TO SCRATCH PAD ZONE
 *    ERRENTER THREE                  INVALID OPERAND
 *    MOVE MESSAGE NUMBER 255
 *    DO MESG                         RETRIEVE MESSAGE
 *    DO ZONE                         WRITE TO SCRATCH PAD ZONE
 *    ERRETURN
 *    EXIT
 *    ENDSEGMENT DOMCTIME
 *    MESG SUBROUTINE SEGMENT
 *      BLANK MESSAGE AREA
 *      RETRIEVE REQUESTED MESSAGE NUMBER VIA MESSAGE MACRO
 *    ENDSEGMENT MESG
 *    ZONE SUBROUTINE SEGMENT
 *      SET UP PARAMETERS FOR DWZONE MACRO
 *      WRITE TO DISPLAY UNIT SCRATCH PAD ZONE USING DWZONE MACRO
 *    ENDSEGMENT ZONE
 *    BUILD SUBROUTINE SEGMENT
 *      MOVE IN TCR TRANSACTION DATA
 *      IF INCREASE TCR REQUEST
 *        MOVE INCREMENT INDICATOR INTO TRANSACTION
 *        CONVERT INCREMENT TO BINARY
 *      ELSE        DECREMENT TCR REQUEST
 *        MOVE DECREMENT INDICATOR INTO TRANSACTION
 *        CONVERT DECREMENT TO BINARY
 *      ENDIF
 *      CONVERT MESSAGE TEXT TO ASCII USING ASCICONV MACRO
 *      SHIP TCR TRANSACTION TO S/7 USING S7WRITE MACRO
 *      BLANK MESSAGE AREA
 *      DO ZONE                         WRITE TO SCRATCH PAD ZONE
 *    ENDSEGMENT BUILD
```

```
MEMBER NAME  DOMCTLCM
*  BEGIN DOMCTLCM
*  /* THIS MODULE (CSECT) GENERATES AND/OR PROCESSES CTL TRANSFER
*     OF CONTROL COMMANDS.*/
*  IF PATCH ID EQUALS TWO OR
*  IF PATCH ID EQUALS ONE THEN
*    CONVERT CTL COMMAND TEXT FROM ASCII TO EBCDIC
*    CONVERT CPU ID'S TO BINARY
*    IF TOP OF HIERARCHY THEN
*      DO DESEARCH (LOCATE DESTINATION CPU)
*      ERREXIT IF SEARCH IS UNSUCCESSFUL
*      DO TABLEMOD (MODIFY DATA BASE AND PDC ARRAY IF REQUIRED)
*      CONVERT CTL COMMAND FROM EBCDIC TO ASCII
*      WRITE CTL COMMAND TO DESTINATION CPU
*    ELSE
*      DO TABLEMOD (MODIFY DATA BASE AND PDC ARRAY IF REQUIRED
*    ENDIF
*    IF PATCH ID EQ ONE THEN
*      FREEWA BUFFER
*    ELSE
*      IF PATCH ID EQUALS TWO THEN
*        RELEASE S/7 BUFFER
*      ENDIF
*   ENDIF
*  ELSE
*    IF PATCH ID EQUALS 36 THEN
*      RETRIEVE ITEM IN DATA BASE
*      IF TOP OF HIERARCHY AND
*      IF MASTER DISPATCHER THEN
*        DO BUILDCTL (GENERATE A CTL COMMAND)
*      ELSE
*        IF ITEM IS IN DATA BASE THEN
*          SEARCH RCB'S FOR ONE THAT BELONGS TO STATUS ITEM
*          IF UNIT IS MASTER DISPATCHER OR
*          IF UNIT FUNCTION CODE AND ACCESS AREA CODE MATCH THOSE OF
*            STATUS ITEM THEN
*            DO BUILDCTL (GENERATE A CTL COMMAND)
*          ELSE
*            ISSUE MESSAGE TO DISPLAY
*          ENDIF
*        ELSE
*          ISSUE MESSAGE TO DISPLAY
*        ENDIF
*      ELSE
*        ISSUE MESSAGE TO DISPLAY
*      ENDIF
*    ENDIF
*  ENDIF
*  EXIT DOMCTLCM
*  END OF DOMCTLCM
*
*
*  BEGIN TABLEMOD SEGMENT
*  /* THIS SEGMENT MODIFYS THE DATA BASE AND/OR PDC ARRAY */
*  IF ITEM KNOWN IN DATA BASE THEN
*    IF TO-CPU EQUALS OWN FRONT END CPU THEN
*      MAKE ITEM CONTROLLABLE
*      EVENT OCCURRENCE
*      PLACE CONTROLLING ID IN PDC TABLE
```

```
MEMBER NAME  DOMCTLCM
*      ELSE
*        MAKE ITEM NOT CONTROLLABLE (IF REQUIRED)
*        EVENT OCCURRENCE
*        CHANGE CONTROLLING CPU ID IN PDC TABLE
*      ENDIF
*    ELSE
*      DISPLAY MESSAGE THAT ITEM NOT FOUND
*    ENDIF
*    END SEGMENT TABLEMOD
*
*
*    BEGIN BUILD CTL SEGMENT
*    /* BUILD A CTL COMMAND AND SHIP */
*    RETRIEVE INPUT DATA FROM SCREEN
*    IF CPU ID INPUT IS VALID THEN
*      BUILD CTL COMMAND AND HEADER
*      CONVERT TEXT OF CTL COMMAND FROM EBCDIC TO ASCII
*      IF TOP OF HIERARCHY THEN
*        PATCH DOMCTLCM WITH AN ID OF 1
*      ELSE
*        SHIP CTL COMMAND TO TOP OF HIERARCHY
*      ENDIF
*    ELSE
*      WRITE ERROR MESSAGE TO DISPLAY
*      FREE CTL AREA
*    ENDIF
*    END SEGMENT BUILDCTL
*
*
*    BEGIN SEGMENT DESEARCH
*    /* THIS SEGMENT LOCATES THE DESTINATION CPU ID */
*    STRTSRCH WHILE ENTRIES IN REMOTE LIST,DO
*      EXITIF INPUT CPU ID EQ LID IN REMOTE LIST,THEN
*        SAVE THE LOCAL CPU ID TO ROUTE THE COMMAND TO
*      ORELSE
*        RETRIEVE POINTER TO NEXT REMOTE LID
*      ENDLOOP
*        STRTSRCH UNTIL ALL LOCAL ID'S EXAMINED,DO
*          EXITIF INPUT CPU ID EQ LID IN LOCAL LIST,THEN
*            SAVE CPU ID/DESTINATION SAME AS TO CPU ID
*          ORELSE
*            RETRIEVE POINTER TO NEXT LOCAL ID
*          ENDLOOP
*            SAVE LOCAL FRONT END CPU ID
*        ENDSRCH
*    ENDSRCH
*    END SEGMENT DESEARCH
```

```
MEMBER NAME  DOMCWBIN
 * DOMCWBIN MAIN SEGMENT
 *    ESTABLISH ADDRESSABILITY FOR CVT AND EMSCVT DSECTS
 *    USE GETARRAY TO RESOLVE ADDRESSES FOR THE WALLBOARD NAME ARRAY
 *     AND WALLBOARD CONFIGURATION TABLE
 *    ERROR EXIT TO ONE IF AT LEAST ONE OF THE ADDRESSES COULD NOT BE
 *     RESOLVED BY GETARRAY
 *    TURN ON EMSCVT WALLBOARD PROCESS BIT
 *    ESTABLISH ADDRESSABILITY FOR BOTH WALLBOARD NAME ARRAY(CAWBNAME)
 *     AND WALLBOARD CONFIGURATION TABLE(CAWBCONF)
 *    IF MANUAL/AUTO ITEM NAME IS NOT BLANK
 *      USE GETITEM TO RESOLVE THE ADDRESS OF THE MANUAL/AUTO ITEM NAME
 *      IF GETITEM RETURNS A NONZERO RETURN CODE
 *        ZERO THE MANUAL/AUTO ITEM ADDRESS
 *        DO WBMESG - ISSUE MESSAGE
 *      ELSE
 *        OVERLAY MANUAL/AUTO ITEM NAME WITH ITS ADDRESS
 *      ENDIF
 *    ELSE
 *      ZERO THE MANUAL/AUTO ITEM ADDRESS
 *    ENDIF
 *    PICK UP NUMBER OF CAWBNAME ARRAY ENTRIES FROM THE ARRAY HEADER
 *    IF THE NUMBER OF CAWBNAME ARRAY ENTRIES IS VALID
 *      ESTABLISH ADDRESSABILITY TO CAWBNAME ARRAY ENTRIES
 *      ZERO EMSCVT COMMAND LIST POINTER AND FLAG BIT
 *      DO LOOP THRU CAWBNAME ARRAY ENTRIES
 *        USE GETITEM TO RESOLVE THE ADDRESS OF WALLBOARD ITEM NAME
 *        IF GETITEM RETURNS A NONZERO RETURN CODE
 *          DO WBMESG - ISSUE MESSAGE
 *          DECREMENT NUMBER OF CAWBNAME ARRAY ENTRIES IN ARRAY HEADER
 *           BY ONE
 *          IF NOT PROCESSING LAST WALLBOARD NAME ARRAY ENTRY
 *            DECREMENT THE LOOP COUNT BY ONE
 *            PREPARE FOR DATA SHIFT
 *            DO LOCATE - DATA FOR SHIFT LOCATED
 *            DO LOOP THRU CAWBNAME ARRAY ENTRIES FOR DATA SHIFT
 *              IF ONE LAMP ADDRESS CAWBNAME ARRAY ENTRY
 *                OVERLAY CAWBNAME ARRAY ENTRY WITH NEXT ENTRY
 *                SET  POINTERS FOR NEXT DATA OVERLAY
 *              ELSE
 *                IF TWO LAMP ADDRESS CAWBNAME ARRAY ENTRY
 *                  OVERLAY CAWBNAME ARRAY ENTRY WITH NEXT ENTRY
 *                  SET POINTERS FOR NEXT DATA OVERLAY
 *                ELSE
 *                  IF THREE LAMP ADDRESS CAWBNAME ARRAY ENTRY
 *                    OVERLAY CAWBNAME ARRAY ENTRY WITH NEXT ENTRY
 *                    SET POINTERS FOR NEXT DATA OVERLAY
 *                  ENDIF
 *                ENDIF
 *              ENDIF
 *            ENDDO
 *            RESTORE CAWBNAME ARRAY ENTRY PROCESS ADDRESS
 *          ENDIF
 *        ELSE
 *          OVERLAY CAWBNAME ARRAY ENTRY WITH ITS ADDRESS
 *          DO LOCATE - LOCATE POINTER TO NEXT CAWBNAME ARRAY ENTRY
 *        ENDIF
 *      ENDDO
 *    ENDIF
```

```
MEMBER NAME  DOMCWBIN
 *    EXIT
 *    ERROR ENTER ONE
 *      IF CAWBNAME ARRAY ADDRESS COULD NOT BE RESOLVED
 *        ZERO EMSCVT WALLBOARD NAME ARRAY ADDRESS ENTRY
 *        DO WBMESG - ISSUE MESSAGE
 *      ENDIF
 *      IF CAWBCONF ARRAY ADDRESS COULD NOT BE RESOLVED
 *        ZERO EMSCVT WALLBOARD CONFIGURATION TABLE ADDRESS ENTRY
 *        DO WBMESG - ISSUE MESSAGE
 *      ENDIF
 *    ERROR ENTER TWO
 *    ERROR RETURN
 *      TURN OFF WALLBOARD PROCESS BIT
 *      EXIT
 * ENDSEGMENT DOMCWBIN
 *
 *
 * LOCATE SUBROUTINE SEGMENT
 *    IF ONE LAMP ADDRESS CAWBNAME ARRAY ENTRY
 *      SET THE POINTER TO NEXT CAWBNAME ARRAY ENTRY
 *    ELSE
 *      IF TWO LAMP ADDRESS CAWBNAME ARRAY ENTRY
 *        SET THE POINTER TO NEXT CAWBNAME ARRAY ENTRY
 *      ELSE
 *        IF THREE LAMP ADDRESS CAWBNAME ARRAY ENTRY
 *          SET THE POINTER TO NEXT CAWBNAME ARRAY ENTRY
 *        ELSE
 *          ERROR EXIT TO TWO
 *        ENDIF
 *      ENDIF
 *    ENDIF
 * ENDSEGMENT LOCATE
 * WBMESG SUBROUTINE SEGMENT
 *    IF MESSAGE 241
 *      ISSUE MESSAGE MACRO
 *    ELSE
 *      PICK UP VARIABLE AND ISSUE MESSAGE MACRO
 *    ENDIF
 * ENDSEGMENT WBMESG
```

```
MEMBER NAME  DOMCWSTA
 * DOMCWSTA MAIN SEGMENT
 *    GET STATUS ARRAY USING GETARRAY MACRO
 *    ERROR EXIT TO ERR1 IF RETURN CODE IS NOT ZERO
 *    CALCULATE SIZE OF ARRAY
 *    DIVIDE TO FIND NUMBER OF GROUPS IN ARRAY
 *    MULTIPLY BY NUMBER OF ITEMS IN GROUP = TOTAL NUMBER OF ITEMS
 *    GET STATUS CHANGE ARRAY USING GETARRAY MACRO
 *    ERROR EXIT TO ERR3 IF RETURN CODE IS NOT ZERO
 *    SET UP INFORMATION FOR GETBLOCK MACRO
 *    GET DASTLOG BLOCK USING GETBLOCK MACRO INTO CASTLOG ARRAY
 *    ERROR EXIT TO ERR2 IF RETURN CODE IS NOT ZERO
 *    IF ARRAY WAS NOT LOGGED THEN
 *      UNTIL ALL ENTRIES IN CASTLOG PROCESSED DO
 *        UNTIL ALL ENTRIES IN DATE/TIME GROUP PROCESSED DO
 *          START SEARCH FOR NUMBER OF ITEMS IN STATUS ARRAY
 *            EXIT IF MATCH ON ITEM NAME FOUND THEN
 *              UPDATE STATUS ARRAY ITEM FROM CASTLOG ARRAY
 *              ADJUST POINTER TO CASTLOG ARRAY
 *            OR ELSE
 *              POINT TO NEXT STATUS ITEM IN STATUS ARRAY
 *              SUBTRACT 1 FROM STATUS GROUP CONTROL COUNT
 *              IF CONTROL COUNT IS ZERO THEN
 *                BYPASS STATUS GROUP HEADER
 *                RESET CONTROL COUNT
 *              ENDIF
 *            ENDLOOP
 *            ADJUST POINTER TO LOG
 *          END SEARCH
 *          POINT TO START OF STATUS ARRAY
 *          RESET CONTROL COUNT FOR STATUS ARRAY
 *        ENDDO
 *        IF ALL ENTRIES IN CASTLOG NOT PROCESSED THEN
 *          POINT TO NEXT DATE/TIME GROUP IN LOG
 *          SET NUMBER OF ITEMS IN DATE/TIME GROUP CONTROL COUNT
 *        ENDIF
 *      ENDDO
 *    ENDIF
 *    ZERO RETURN CODE
 *    EXIT
 *
 *    ERROR ENTER ERR1
 *      SET RETURN CODE TO 4
 *    ERROR ENTER ERR2
 *      SET RETURN CODE TO 8
 *    ERROR ENTER ERR3
 *      SET RETURN CODE TO 12
 *    ERRETURN
 *      EXIT
 * ENDSEGMENT DOMCWSTA
```

```
MEMBER NAME  DOMTAGCI
 *        DOMTAGCI MAIN SEGMENT
 *          ***************************************************************
 *          **  THIS ROUTINE IS CALLED BY SUPERVISORY CONTROL            **
 *          *   INITIALIZATION.                                           *
 *          *   ITS FUNCTION IS TO PATCH THE AGC AND EDC INITIALIZATION   *
 *          **  ROUTINES IF THEY HAVE BEEN SYSGENED IN THE SYSTEM.       **
 *          ***************************************************************
 *
 *          IF AGC HAS BEEN SYSGENED THEN
 *            PATCH EP=DOMALFCI
 *            WAIT ON PATCH
 *            IF EDC HAS BEEN SYSGENED THEN
 *              PATCH EP=DOMAEDCI
 *              WAIT ON PATCH
 *            ENDIF
 *          ENDIF
 *        ENDSEGMENT DOMTAGCI
```

```
MEMBER NAME  DOMTAPLD
 * BEGIN DOMTAPLD
 * /* THIS IS THE CONTROL PROGRAM FOR THE PERFORMANCE LOG CONTROL
 *     DISPLAY.  THE PATCH ID DETERMINES WHAT TYPE OF PROCESSING
 *     WILL TAKE PLACE. */
 * IF TASK RESOURCE TABLE IS ZERO THEN
 *    SET PATCH ID TO 2
 *    GET TASK RESOURCE TABLE FOR TASK
 *    DO HOUSEKEEPING AND LINKAGE
 *    IF CONTROL ELEMENT ADDRESS POINTER TO RCE IS NOT ZERO THEN
 *      ZERO POINTER
 *    ENDIF
 * ELSE
 *    DO HOUSEKEEPING AND LINKAGE
 * ENDIF
 * TURN PROGRAM FUNCTION KEY BACKLITES ON
 * IF PATCH ID IS 2 THEN
 *    DO INITIAL PROCESSING  (DOMTAINI)
 * ELSE
 *    IF PATCH ID IS 4 THEN
 *      DO PAGE FORWARD PROCESSING  (DOMTAPGF)
 *    ELSE
 *      IF PATCH ID IS 6 THEN
 *        DO PAGE BACKWARD PROCESSING (DOMTAPGB)
 *      ELSE
 *        IF PATCH ID IS 10 THEN
 *          DO ADD NEW NAME SECTION (DOMTAADD)
 *        ELSE
 *          IF PATCH ID IS 12 THEN
 *            DO DELETE OLD NAME SECTION (DOMTADEL)
 *          ELSE
 *            IF PATCH ID IS 14 THEN
 *              DO CANCEL PROCESSING  (DOMTACNL)
 *            ENDIF
 *          ENDIF
 *        ENDIF
 *      ENDIF
 *    ENDIF
 * ENDIF
 * RETURN TO SYSTEM
 * END DOMTAPLD
 *
 * /* INITIAL SECTION */
 * BEGIN DOMTAINI SUBROUTINE
 * CALCULATE TOTAL NUMBER OF PAGES FOR DISPLAY
 * PLACE RCB ENTRY ADDRESSES IN DSECT   START,STOP,CURRENT,NEXT
 *               -USED IN RCB SEARCH-
 * DO PAGE BUILD SUBROUTINE
 * END DOMTAINI SUBROUTINE
 *
 * /* PAGE FORWARD SUBROUTINE */
 * BEGIN  DOMTAPGF SUBROUTINE
 * GET CURRENT PAGE NO.
 * GET LAST PAGE NO.
 * IF CURRENT PAGE EQ LAST PAGE THEN
 *    SET PAGE NO. TO ONE (CURRENT)
 * ELSE
 *    INCREASE CURRENT PAGE NO. BY ONE
 * ENDIF
```

```
MEMBER NAME  DOMTAPLD
 * DO PAGE BUILD SUBROUTINE
 * END DOMTAPGF SUBROUTINE
 *
 * /* PAGE BACKWARD SUBROUTINE
 * BEGIN DOMTAPGB SUBROUTINE
 * GET CURRENT PAGE NO.
 * GET LAST PAGE NO.
 * IF CURRENT PAGE NO. EQUALS LAST PAGE NO. THEN
 *   CURRENT PAGE EQUALS ONE
 * ELSE
 *   SUBTRACT ONE FROM CURRENT PAGE NO.
 * ENDIF
 * DO PAGE BUILD SUBROUTINE
 * END DOMTAPGB SUBROUTINE
 *
 * /* ADD ITEM TO CAPLNAME ARRAY */
 * BEGIN DOMTAADD SUBROUTINE
 * GET NAME TO BE ADDED FROM SCREEN READ
 * IF NAME NOT IN CAPLNAME ARRAY THEN
 *   IF CAPLNAME ARRAY NOT FULL THEN
 *     IF ITEM NAME VALID THEN
 *       DO SEARCH TO FIND CORRESPONDING RCB
 *       CALCULATE NAMES LIST ADDRESS AND OFFSET
 *       PLACE ITEM NAME IN CAPLNAME ARRAY
 *       INSERT ANALOG DATA ADDRESS
 *       INSERT NAMES LIST DATA ADDRESS
 *       INCREASE NO. VALID ENTRIES BY ONE
 *       DECREASE NO. OF VOID ENTRIES BY ONE
 *       RECALCULATE NO.OF PAGES FOR DISPLAY
 *       REFRESH CURRENT PAGE
 *       LOG CAPLNAME ARRAY AFTER UPDATE
 *     ELSE
 *       WRITE ERROR MESSAGE THAT ITEM NAME NOT IN DATA BASE
 *     ENDIF
 *   ELSE
 *     WRITE ERROR MESSAGE THAT CAPLNAME ARRAY IS FULL
 *   ENDIF
 * ELSE
 *   WRITE MESSAGE THAT ITEM NAME ALREADY IN FILE
 * ENDIF
 * END DOMTAADD SUBROUTINE
 *
 * /* DELETE ITEM FROM CAPLNAME ARRAY */
 * BEGIN DOMTADEL SUBROUTINE
 * GET NAME TO BE DELETED FROM PARTIAL SCREEN READ
 * SEARCH CAPLNAME ARRAY TOFIND MATCH
 * IF NAME IS IN FILE THEN
 *   MOVE ALL FOLLOWING ENTRIES UP ONE SLOT
 *   ZERO LAST SLOT
 *   ADD ONE TO NO. OF VOID ENTRIES
 *   SUBTRACT ONE FROM NO. OF VALID ENTRIES
 *   CALCULATE NO. OF PAGES FOR DISPLAY
 *   REDISPLAY CURRENT PAGE
 * ELSE
 *   PRINT MESSAGE THAT ITEM TO BE DELETED NOT IN FILE
 * ENDIF
 * END DOMTADEL SUBROUTINE
 *
```

```
MEMBER NAME  DOMTAPLD
 * */ CANCEL SUBROUTINE */
 * BEGIN DOMTACNL SUBROUTINE
 * SUBTRACT ONE FROM TASK COUNTER
 * IF DISPLAY NOT EQUAL-DOMTAPLT- THEN
 *   DPATCH TASK
 * ENDIF
 * END DOMTACNL SUBROUTINE
 *
 * /* BUILD DISPLAY PAGE SUBROUTINE */
 * BEGIN DOMTAPAG SUBROUTINE
 * CALCULATE FIRST ITEM TO BE DISPLAYED
 * GET DISPLAY BUFFER AREA
 * DO 16 TIMES OR UNTIL TOTAL NO. OF ITEMS PROCESSED OR UNTIL
 *   ENTRY EQUALS ZERO
 *   PLACE NAME IN BUFFER
 *   PLACE SUBSTATION/REMOTE NAME IN BUFFER
 *   PLACE TYPE NAME IN BUFFER
 * ENDDO
 * IF LAST ITEM THEN
 *   PLACE 'END' AFTER LAST ENTRY
 * ENDIF
 * IF CURRENT DISPLAY NAME NE TO DOMDAPLD THEN
 *   BRING UP BACKGROUND FOR DOMDAPLD
 *   ADD ONE TO TASK COUNTER
 * ENDIF
 * DISPLAY BUFFER
 * FREE BUFFER AREA
 * END DOMTAPAG SUBROUTINE
 *
 * /* THIS SECTION SEARCHES TO FIND THE RCB ASSOCIATED WITH AN ANALOG
 *    ITEM. */
 * BEGIN SEARCH SUBROUTINE
 * WHEN RCB ANALOG ADDRESS IS EQUAL TO OR LESS THAN ITEM ADDRESS AND
 *    NEXT RCB ANALOG ADDRESS IS GREATER THAN ITEM ADDRESS THE END
 *    SEARCH
 * END SEARCH SUBROUTINE
 *
```

```
MEMBER NAME  DOMTAPLI
 * BEGIN DOMTAPLI
 * /* THIS IS THE PERFORMANCE LOG CONTROL PROGRAM INITIALIZATION */
 * GET ADDRESS OF CAPLNAME, CAPTLOG, AND CALOGTIM ARRAYS AND PLACE
 *     IN EMSCVT TABLE
 * UNTIL ALL VALID ENTRIES PROCESSED DO
 *   IF ITEM NAME NOT VALID THEN
 *     REPLACE ENTRY WITH BLANKS AND ADJUST COUNTS
 *   ELSE
 *   GET ANALOG ITEM ADDRESS AND PLACE IN CAPLNAME ARRAY
 *   DO SEARCH TO FIND RCB ASSOCIATED WITH ANALOG POINT
 *   USING ANALOG NAMES LIST ADDRESS IN ASSOCIATED RCB CALCULATE ADDRESS
 *      OF NAMES LIST DATA IN DATA BASE
 *   PLACE ADDRESS IN CAPLNAME ARRAY
 *   ENDIF
 * ENDDO
 * IF ITEM IN CAPLNAME IS BLANK THEN
 *   MOVE ALL ITEMS FOLLOWING UP ONE POSITION
 * ENDIF
 * ZERO ALL VOID ENTRIES
 * END DOMTAPLI
 *
 *
 *
 *
```

```
MEMBER NAME  DOMTAPLM
 * BEGIN DOMTAPLM
 * /* THIS PROGRAM CONTROLS THE PERFORMANCE LOG RETRIEVAL */
 * GET TIME OF PATCH
 * GET BUFFER TO PLACE RETRIEVED ARRAY IN     CAPTLOG
 * RETRIEVE CURRENT LOGGED COPY
 * ERREXIT IF UNSUCCESSFUL
 * IF CALOGTIM UPDATE FLAG IS OFF THEN
 *   TURN FLAG ON
 *   IF TIME COUNT IS ZERO THEN
 *     PLACE HEADER TIME IN FIRST SLOT
 *     UPDATE COUNT BY ONE
 *     WRITE EVENT & MESSAGE TO SYSTEM MESSAGE ZONE
 *   ELSE
 *     COMPARE TIME TO OTHERS IN TIME SLOT
 *     IF TIME EQUALS ANY ENTRY THEN
 *       SET NOT PROCESSED FLAG ON
 *     ELSE
 *       IF COUNT LESS THAN TEN
 *         PLACE TIME IN NEXT VACANT FIELD
 *         EVENT LOGGING RETRIEVAL TIME ACCEPTED
 *       ENDIF
 *     ENDIF
 *     IF NOT PROCESSED FLAG IS OFF THEN
 *       IF RETRIEVAL ACTIVE FLAG IS OFF THEN
 *         TURN ACTIVE FLAG ON
 *         CALCULATE NO. OF ENTRIES IN LOG FILE
 *         TURN MINUS MODE FLAG ON
 *         SET STEP RETRIEVAL COUNT EQ TO NO OF LOGGED COPPIES IN NINE
 *            SECONDS OR TWO, WHICHEVER IS GREATER
 *         ADD SAME NUMBER TO RETRIEVAL COUNT.
 *         GET LOG DATA ARRAY FROM LOGGED ARRAY
 *         PLACE TIME OF RETRIEVED RECORD IN CALOGTIM ARRAY
 *         INCREACE RETRIEVAL COUNT BY ONE
 *         SHIP ARRAY TO DOMTUSER
 *         PATCH DOMTAPLR
 *       ELSE
 *         IF PLUS MODE COUNT GT ZERO THEN
 *           IF HEADER    TIME GREATER THAN SAVED RETRIEVAL TIME THEN
 *             TURN PLUS FLAG OFF
 *             TURN MINUS MODE FLAG ON
 *             SET PLUS MODE COUNTER TO ZERO
 *           ELSE
 *             SET PLUS MODE COUNTER TO ZERO
 *             TURN PLUS MODE FLAG ON
 *             TURN MINUS MODE FLAG OFF
 *           ENDIF
 *         ENDIF
 *       ENDIF
 *       ELSE
 *       TURN NOT PROCESSED FLAG OFF
 *     ENDIF
 *   ENDIF
 *   TURN UPDATE FLAG OFF
 * ELSE
 *   PATCH DOMTAPLM ID=4
 * ENDIF
 * FREE BUFFER
 * EXIT
```

```
MEMBER NAME  DOMTAPLM
 * END DOMTAPLM
```

```
MEMBER NAME  DOMTAPLP
 * BEGIN DOMTAPLP
 * /* PROCESS PROGRAM FOR PERFORMANCE LOG */
 * GET TIME OF LAST SCAN CYCLE
 * PLACE TIME IN CAPTLOG ARRAY
 * IF LOGGING COUNT IS LESS THAN RETRIEVAL COUNT THEN
 *   SUSPEND LOGGING AND PRINT MESSAGE
 * ELSE
 *   DEMAND LOG CAPTLOG ARRAY
 * ENDIF
 * DO FOR EACH VALID ENTRY IN CAPLNAME ARRAY
 *   PLACE ANALOG DATA IN CAPTLOG ARRAY
 *   PLACE NAMES LIST DATA IN CAPTLOG ARRAY
 * ENDDO
 * ZERO VOID ENTRY SPACES IN CAPTLOG ARRAY
 * END DOMTAPLP
```

```
MEMBER NAME  DOMTAPLR
 * BEGIN DOMTAPLR
 * /* THIS IS THE PERFORMANCE LOG RETRIEVAL PROGRAM */
 * GET AND STORE DOMTUSER MODULE ADDRESS
 * IF PATCH ID EQUALS FOUR THEN
 *   DO APLRMM (MINUS MODE) SEGMENT
 * ELSE
 *   CALCULATE SIZE NEEDED FOR BUFFER TO HOLD ONE LOGGED CAPTLOG ARRAY
 *   ADD LOG HEADER SIZE + LENGTH FIELD + SCAN TIME FIELD
 *   GETMAIN AREA
 *   DO UNTIL PLUS MODE RETRIEVAL COUNT EQ END COUNT OR
 *     CANCEL BIT IS ON OR
 *     PTIME ACTIVE BIT IS ON
 *     GET LOGGED ARRAY ONE PAST LAST RETRIEVED ARRAY TIME
 *     IF MINUS MODE BIT IS ON THEN
 *       IF CONDITION CODE EQUALS ZERO OR EIGHT THEN
 *         IF RETRIEVAL TIME EQUALS EARLIEST TIME THEN
 *           DO APLRMM (MINUS MODE) SEGMENT
 *         ENDIF
 *         PLACE HEADER TIME IN RETRIEVED TIME
 *         ADD ONE TO RETRIEVAL COUNTER
 *
 *         DO APLRSDT (SHIP DATA) SECTION
 *       ELSE
 *         EVENT FACT THAT LOGGED RETRIEVAL CANCELLED BECAUSE
 *           RETRIEVAL FROM LOGGED ARRAYS FAILED
 *         TURN CANCEL BIT ON
 *       ENDIF
 *     ELSE
 *       IF HEADER TIME IS GREATER THAN LAST RETRIEVED TIME THEN
 *         PLACE HEADER TIME IN RETRIEVED TIME SLOT
 *         ADD ONE TO RETRIEVAL COUNT
 *         ADD ONE TO RETRIEVAL PLUS COUNTER
 *         DO APLRSDT (SHIP DATA)
 *         IF PLUS MODE COUNTER EQ END COUNT THEN
 *           IF UPDATE FLAG IS OFF THEN
 *             TURN UPDATE FLAG ON
 *             TURN ALL OTHER FLAGS OFF
 *             MOVE ALL 'F-FS' TO FIRST WORD OF BUFFER
 *             DO APLRSDT (SHIP DATA) SEGMENT
 *             TURN UPDATE FLAG OFF
 *             CLEAR CALOGTIM ARRAY
 *           ENDIF
 *         ENDIF
 *       ELSE
 *         PATCH DOMTAPLR ID=2
 *         TURN PTIME BIT ON
 *       ENDIF
 *     ENDIF
 *   ENDDO
 * ENDIF
 * TURN PTIME BIT OFF
 * FREE BUFFER USED TO SHIP DATA IN
 * DELETE TASK IF ACTIVE FLAG IS OFF
 * END     DOMTAPLR
 *
 * BEGIN SEGMENT APLRMM
 * /* THIS IS A SUBROUTINE OF DOMTAPLR */
 * IF UPDATE FLAG IS OFF THEN
```

```
MEMBER NAME  DOMTAPLR
 *    TURN UPDATE FLAG IS ON
 *    IF COUNT EQ 1 THEN
 *      ZERO FIRST AND ONLY MARK TIME
 *      TURN PLUS MODE FLAG ON
 *      TURN MINUS MODE FLAG OFF
 *      SET COUNTER TO ZERO
 *    ELSE
 *      SUBTRACT ONE FROM COUNT
 *      CALCULATE LENGTH AND MOVE ALL MARK TIMES UP ONE
 *      ZERO LAST ENTRY
 *    ENDIF
 * ELSE
 *    PATCH DOMTAPLR ID=4 ONE SECOND PTIME
 * ENDIF
 * END SEGMENT APLRMM
 *
 * BEGIN SEGMENT APLRSDT
 * /* THIS IS A SUBROUTINE OF DOMTAPLR USED TO SHIP DATA */
 * SHIP BUFFER ADDRESS TO DOMTUSER MODULE
 * END    SEGMENT APLRSDT
 *
```

```
MEMBER NAME  DOMTBSET
 *        DOMTBSET MAIN SEGMENT (BUILD SCAN EXCEPTION TABLE ENTRIES)
 *          IF CURRENT BUFFER POINTER IS EQUAL TO OR PAST BUFFER END    X
 *                THEN
 *            GET 136 BYTE BUFFER
 *            SET ENTRY COUNT FOR THIS BUFFER TO ZERO
 *            SET BUFFER END POINTER
 *            INITIALIZE CURRENT BUFFER POINTER
 *            IF THIS IS THE FIRST BUFFER IN A CHAIN THEN
 *              SET FIRST BUFFER POINTER TO CURRENT BUFFER
 *              STORE S/7 ID AND TERMINAL ID IN BUFFER
 *            ELSE
 *              SET FORWARD BUFFER CHAIN POINTER IN LAST BUFFER
 *            ENDIF
 *          ENDIF
 *          IF PREVIOUS ENTRY HAS DIFFERENT S/7 AND TERMINAL ID THEN
 *            STORE NEW S/7 ID AND TERMINAL ID IN BUFFER
 *            BUMP OLD S/7 AND TERMINAL POINTER IN BUFFER TO NEW ENTRY
 *          ENDIF
 *          STORE DATABASE ADDRESS OF ALARMED POINT IN BUFFER
 *          STORE DATA VALUE IN BUFFER
 *          STORE ERROR AND TYPE FLAGS IN BUFFER
 *          BUMP ENTRY COUNT BY 1
 *          BUMP CURRENT BUFFER POINTER TO NEXT AVAILABLE POSITION
 *        ENDSEGMENT DOMTBSET
```

```
MEMBER NAME  DOMTCACS
 *        BEGIN DOMTCACS
 *
 *        /* THE ANALOGUE CONVERSION SEGMENT DOMTCACS IS ENTERED FROM
 *        DOMTCONV FOR EACH ANALOGUE POINT.  LOGIC ORDER ASSUMES GOOD
 *        DATA IS THE PREDOMINANT CASE */
 *
 *        IF POINT IS INACTIVE IN RCB  AND RDA SENT  THEN
 *          INDEX TO NEXT ACTIVE ENTRY IN RCB  /*POINT NOT YET WIRED*/
 *        ENDIF
 *        IF POINT IS ONLINE IN RCB  THEN
 *          CASENTRY
 *            CASE  RDA FROM S/7
 *              INCLUDE CONVERT RDA SEGMENT (DOMTCRDA)
 *            CASE     USER CONVERSION BIT ON IN ANALOGUE BLOCK
 *              INCLUDE USER CONVERSION SEGMENT (DOMTUSER)
 *          ENDCASE
 *          IF SET FLAGS INDICATES CONVERTABLE VALUE  THEN
 *            CONVERT FIXED POINT TO FLOATING POINT
 *            STORE DIRECTLY IN RCB
          ENDIF
 *          IF SET INDICATOR HAS ERROR
 *            CALL BUILD SET SEGMENT (DOMTBSET) GIVING SET INDICATOR
 *          ENDIF
 *        ENDIF
 *        INDEX TO NEXT ENTRY IN RCB
 *        INDEX TO NEXT ENTRY IN RDA
 *
 *        END DOMTCACS
 *
 *        DOMTUSER SUBROUTINE SEGMENT
 *        /* USER CONVERSION SEGMENT IS ENTERED FOR EACH ANALOGUE
 *        POINT THAT HAS THE USER CONVERSION BIT ON IN THE ANALOGUE
 *        BLOCK
 *        THIS ROUTINE HAS ACCESS TO ALL CONTROL FIELDS IN PARTICULAR
 *        POINTER TO VALUE IN RDA
 *        POINTER TO RCB
 *        POINTER TO ANALOGUE BLOCK INCLUDING LIMITS AND CONVERSION DATA
 *        USER RETURNS THE ONE BYTE SET ENTRY WHICH MAY REQUEST ALARM
 *        GENERATION AND FULLWORD FIXED POINT CONVERTED DATA.    /*
 *        ENDSEGMENT DOMTUSER
 *
 *        SUBROUTINE SEGMENT DOMTCRDA
 *        /* RDA ANALOGUE CONVERSION SEGMENT*/
 *        IF BIT 14 & 15 OFF IN RDA VALUE  THEN
 *          /* ADC GAVE CONVERTIBLE DATA*/
 *          IF RDA VALUE HIGHER THAN 90% WARNING LIMIT  THEN
 *            CALCULATE 100% LIMIT
 *            IF HIGHER THAN OPERATING LIMIT  THEN
 *              IF LOWER THAN OFFSCALE LIMIT THEN
 *                MAKE SET INDICATOR TO OPERATING LIMIT VIOLATED
 *              ELSE
 *                MAKE SET INDICATOR TO OFFSCALE
 *              ENDIF
 *            ELSE
 *              MAKE SET INDICATOR TO 90% WARNING
 *            ENDIF
 *          ELSE
 *            IF RDA VALUE LOWER THAN 10% WARNING LIMIT  THEN
```

```
MEMBER NAME  DOMTCACS
 *              CALCULATE LOWER OPERATING LIMIT
 *              IF LOWER THAN OPERATING LIMIT  THEN
 *                IF HIGHER THAN OFFSCALE LIMIT  THEN
 *                  MAKE SET INDICATOR TO OPERATING LIMIT
 *                ELSE
 *                  MAKE SET INDICATOR TO OFFSCALE
 *                ENDIF
 *              ELSE
 *                MAKE SET INDICATOR TO 10% WARNING
 *              ENDIF
 *            ENDIF
 *          ENDIF
 *          /* GUESS 10% OF 10% OF CONVERSIONS REDUNDANT FOR OFFSCALE*/
 *          PERFORM SCALING AND CONVERSION TO ENGINEERING UNITS
 *          IF VALUE IS WITHIN 10/90% LIMITS IN SET INDICATOR  THEN
 *            /* ALARM CLEARANCE*/
 *            IF POINT IS ALARMED IN RCB  THEN
 *              IF BACK IN LIMITS COUNT IS THREE  THEN
 *                MAKE SET INDICATOR TO CLEAR ALARM
 *              ELSE
 *                INCREMENT ALARM COUNT BY ONE
 *              ENDIF
 *            ENDIF
 *          ELSE
 *            BACK IN LIMITS COUNT SET ZERO /*TO TRAP OSCILLATION*/
 *          ENDIF
 *        ELSE
 *          /*DETERMINE WHY RDA VALUE CANNOT BE CONVERTED*/
 *          IF BIT 14 ON IN RDA  THEN
 **           IF BIT 15 ON  THEN
 *              MAKE SET INDICATOR TO MISSING DATA
 *              /* S/7 HAS SET BITS 14 & 15 ON*/
 *            ELSE
 *              MAKE SET ADC FAILURE  /* PRESENTLY SAME AS OFFSCALE*/
 *            ENDIF
 *          ELSE
 *            /* BIT 15 WAS ON*/
 *            MAKE SET INDICATOR TO OFFSCALE
 *          ENDIF
 *        ENDIF
 *        ENDSEGMENT DOMTCRDA
```

```
MEMBER NAME  DOMTCHRT
 * DOMTCHRT MAIN SEGMENT
 *   ESTABLISH ADDRESSABILITY
 *   GET AREA FOR PASSING PARAMETERS, USING GETWA MACRO
 *   IF MACRO FAILS THEN
 *     SET RETURN CODE TO 56
 *   ELSE
 *     MOVE MACRO LIST INTO AREA GOTTEN
 *     ZERO BOTH ECBS
 *     STORE SECOND ECB ADDRESS IN PARAMETER LIST AREA
 *     PATCH DOMCHRTA USING PATCH MACRO
 *     IF RETURN CODE IS NOT ZERO THEN
 *       SET RETURN CODE TO 60
 *       FREE AREA USING FREEWA MACRO
 *     ELSE
 *       WAIT ON FIRST ECB
 *       IF ECB POST CODE IS NOT GOOD THEN
 *         SET RETURN CODE TO 60
 *       ELSE
 *         CHECK RETURN CODE IN ECB
 *         IF IT IS ZERO THEN
 *           WAIT ON SECOND ECB
 *           SET RETURN CODE TO THAT PASSED IN THE ECB
 *         ENDIF
 *       ENDIF
 *     ENDIF
 *   ENDIF
 *   EXIT WITH RETURN CODE SET
 * ENDSEGMENT DOMTCHRT
```

```
MEMBER NAME  DOMTCLOK
 *        DOMTCLOK MAIN SEGMENT
 *          DO UNTIL S/7 TIME HAS BEEN PLACED IN DATA BASE
 *            ISSUE A WAIT FOR N NUMBER OF SECONDS
 *          ENDDO
 *          GET S/370 TIME STORED BY EXTERNAL INTERRUPT HANDLER ON THE  X
 *                EVEN MINUTE PULSE
 *          GET CURRENT S/370 TIME
 *          CURRENT TIME MINUS EVEN MINUTE PULSE TIME = ELAPSED TIME    X
 *                SINCE LAST EVEN MINUTE
 *          GET S/7 TIME FROM S/370 DATA BASE
 *          ROUND TIME TO LAST EVEN MINUTE
 *          CURRENT TIME = ROUNDED TIME + ELAPSED TIME
 *          GET CURRENT SRTOS TIME
 *          DELTA TIME = CURRENT TIME MINUS SRTOS TIME
 *          PATCH SRTOS MODULE DPPCUPCF PASSING DELTA TIME
 *        ENDSEGMENT DOMTCLOK
```

```
MEMBER NAME  DOMTCONV
*        DOMTCONV MAIN SEGMENT (DATA CONVERSION DIRECTOR)
*          USE S/7 ID TO FIND ASSOCIATED RCB LIST AND RDA MAP
*          DO UNTIL ALL DATA IN THIS RDA HAS BEEN PROCESSED
*            INDEX TO RCB FOR THIS TERMINAL
*            IF PSEUDO PDC TERMINAL AND
*            IF ANY STATUS POINTS EXIST ON THIS TERM THEN
*              CALL PDC PROCESSOR (DOMCPDC1)
*            ENDIF
*              IF THIS EMT IS INSERVICE THEN
*                BUMP TO STATUS DATA IN RDA FOR THIS TERMINAL
*                  IF STATUS DATA COUNT IS NOT ZERO THEN
*                    CALL STATUS DATA CONVERSION DOMTCSES
*                  ENDIF
*                  BUMP TO ANALOG DATA IN RDA FOR THIS TERMINAL
*                  IF ANALOG DATA COUNT IS NOT ZERO THEN
*                  CALL ANALOG DATA CONVERSION DOMTCACS
*                  ENDIF
*                  BUMP TO PULSE COUNTER DATA IN RDA FOR THIS TERMINAL
*                    IF PC DATA COUNT IS NOT ZERO THEN
*                      CALL PC DATA CONVERSION DCMTCPC
*                    ENDIF
*              ENDIF
*            BUMP TO NEXT TERMINAL IN RDA
*          ENDDO
*          IF ANY SCAN EXCEPTION ENTRIES WERE BUILT THEN
*            PATCH DOMCALR1 TO ISSUE ALARMS
*          ENDIF
*        ENDSEGMENT DOMTCONV
```

```
MEMBER NAME  DOMTCPC
 *DOMTCPC MAIN SEGMENT
 */* ENTRY FROM DOMTCONV TO TRANSFER PULSE COUNTER DATA FROM RDA TO
 *THE RCD.  ERROR CHECKING FOR THE FOLLOW MISSING DATA, BCH ERROR,
 *OVERRUN, PARITY ERROR, AND ROLLOVER OF COUNTER */
 *
 *USE THE RAD ADDRESS PASSED IN GENERAL REGISTER NINE TO FIND THE FIRST
 *      PC ENTRY
 *
 *   IF THIS IS TERMINAL ZERO  THEN
 *     DO UNTIL PC POINTS ARE PROCESSED
 *       DO SEGMENT PCPROCES
 *     ENDDO
 *   ENDIF
 *DO UNTIL ALL PC POINTS ARE PROCESSED
 *   IF THE IDT FLAG IS ON THEN
 *     IF THE PC MISSING DATA FLAG IS ON THEN
 *       SET THE FLAGES IN THE FIRST BYTE OF THE S.E.T. FOR AN ALARM
 *       DO SEGMENT BUILDALM
 *     ELSE
 *       IF THE PC DATA BCH ERROR FLAG IS ON THEN
 *         SET THE FLAGES IN THE FIRST BYTE OF THE S.E.T. FOR AN ALARM
 *         DO SEGMENT BUILDALM
 *       ELSE
 *         WRITE A MESSAGE TO OPERATOR STATING IDT BIT ON BUT NETHER
 *               PC MISSING DATA FLAG OR BCH ERROR FLAG ARE ON
 *       ENDIF
 *     ENDIF
 *   ELSE
 *     IF THE PC OVERRUN FLAG IS ON THEN
 *       SET THE FLAG OVERUN FLAG IN THE S.E.T. ARRAY FOR AN ALARM
 *       DO SEGMENT PCPROCES
 *       DO SEGMENT BUILDALM
 *     ELSE
 *       /* PROCESS THE PC DATA NORMALLY */
 *         CLEAR DATA BASE ERROR FLAGS AND SET FLAGS TO CLEAR ALARM
 *         DO SEGMENT BUILDALM
 *       DO SEGMENT PCPROCES
 *     ENDIF
 *   ENDIF
 *   IF THE LAST IDT FLAG OF THIS SET HAS BEEN TESTED THEN
 *     MOVE NEXT SET OF IDT FLAGS
 *   ENDIF
 *   SET NEXT IDT FLAG FOR TESTING
 *ENDDO
 *EXIT
 *
 *BUILDALM SUBROUTINE SEGMENT
 *   /* BUILD THE S.E.T. FOR DOMTBSET TO SET AN ALARM.  REGISTOR 1 WILL
 *   CONTAIN THE ADDRESS OF A 2 WORD FIELD.  THE FIELD CONTAINS ERROR
 *   FLAGS, DATA BASE ADDRESS, AND VALUE RECEIVED IF SENT TO DATA BASE /*
 *   STORE ADDRESS OF DATA BASE INTO SET
 *   IF THE OVERRUN OR ROLLOVER ERROR FLAG IS NOT ON THEN
 *     STORE ZERO IN THE S.E.T. VALUE FIELD
 *   ELSE
 *     STORE VALUE THAT GOES TO DATA BASE
 *   ENIF
 *   CALL DOMTBSET  THIS BUILD THE ALARM S.E.T.
 *ENDSEGMENT BUILDALM
```

```
MEMBER NAME  DOMTCPC
 *
 *PCPROCES SUBROUTINE SEGMENT
 *   /* PROCESS THE DATA FROM RDA.  TEST FOR A ROLLOVERE OF THE
 *   ACCUMULATED COUNTER.  STORE ALL PROCESSED DATA IN THE RCD. */
 *   CONVERT PC DATA TO FLOATING POINT DATA
 *   MULTIPLY PC DATA BY COEFFICIENT FOR FINISHED COUNTER
 *   IF A ROLLOVER HAS OCCURRED THEN
 *     REINITIALIZE THE ACCUMULATED COUNTER
 *     SET ERROR FLAG IN THE RCD AND S.E.T.
 *     DO SEGMENT BUILDALM
 *   ENDIF
 *   STORE VALUE PROCESSED INTO THE RCB
 *ENDSEGMENT PCPROCES
 *END DOMTCPC
```

```
MEMBER NAME  DOMTCRDA
*        BEGIN DOMTCRDA
*        /* RDA ANALOGUE CONVERSION SEGMENT*/
*        IF BIT 14 & 15 OFF IN RDA VALUE  THEN
*          /* ADC GAVE CONVERTIBLE DATA*/
*          IF RDA VALUE HIGHER THAN 90% WARNING LIMIT  THEN
*            CALCULATE 100% LIMIT
*            IF HIGHER THAN OPERATING LIMIT  THEN
*              IF LOWER THAN OFFSCALE LIMIT THEN
*                MAKE SET INDICATOR TO OPERATING LIMIT VIOLATED
*              ELSE
*                MAKE SET  INDICATOR TO OFFSCALE
*              ENDIF
*            ELSE
*              MAKE SET  INDICATOR TO 90% WARNING
*            ENDIF
*          ELSE
*            IF RDA VALUE LOWER THAN 10% WARNING LIMIT  THEN
*              CALCULATE LOWER OPERATING LIMIT
*              IF LOWER THAN OPERATING LIMIT  THEN
*                IF HIGHER THAN OFFSCALE LIMIT  THEN
*                  MAKE SET INDICATOR TO OPERATING LIMIT
*                ELSE
*                  MAKE  SET INDICATOR TO OFFSCALE
*                ENDIF
*              ELSE
*                MAKE SET INDICATOR  TO 10% WARNING
*              ENDIF
*            ENDIF
*          ENDIF
*          /* GUESS 10% OF 10% OF CONVERSIONS REDUNDANT FOR OFFSCALE*/
*          PERFORM SCALING AND CONVERSION TO ENGINEERING UNITS
*          IF VALUE IS WITHIN 10/90% LIMITS IN SET INDICATOR  THEN
*            /* ALARM CLEARANCE*/
*            IF POINT IS ALARMED IN RCB  THEN
*              IF BACK IN LIMITS COUNT IS THREE  THEN
*                MAKE SET INDICATOR TO CLEAR ALARM
*              ELSE
*                INCREMENT ALARM COUNT BY ONE
*              ENDIF
*            ENDIF
*          ELSE
*            BACK IN LIMITS COUNT SET ZERO /*TO TRAP OSCILLATION*/
*          ENDIF
*        ELSE
*          /*DETERMINE WHY RDA VALUE CANNOT BE CONVERTED*/
*          IF BIT 14 ON IN RDA  THEN
*            IF BIT 15 ON  THEN
*              MAKE SET INDICATOR TO MISSING DATA
*              /* S/7 HAS SET BITS 14 & 15 ON*/
*            ELSE
*              MAKE SET ADC FAILURE  /* PRESENTLY SAME AS OFFSCALE*/
*            ENDIF
*          ELSE
*            /* BIT 15 WAS ON*/
*            MAKE SET INDICATOR TO OFFSCALE
*          ENDIF
*        ENDIF
*        END DOMTCRDA
```

```
MEMBER NAME  DOMTCSES
 * DOMTCSES MAIN SEGMENT
 *    ZERO SAVE FIELDS FOR STATUS LOG
 *    SAVE DATE AND TIME FROM RDA FOR STATUS LOG
 *    SAVE ADDRESS OF TERMINAL HEADER
 *    UNTIL ALL ITEMS FOR CURRENT TERMINAL ARE PROCESSED DO
 *      POINT TO STATUS DATA USING ADDRESS FROM RCB
 *      SEARCH FOR MATCHING CARD ADDRESS OR SEQUENCE NUMBER IN DATA BASE
 *      EXIT IF MATCH FOUND
 *        IF LATCHED STATUS GROUP THEN
 *          IF LATCH BIT IS ON THEN
 *            SET UP MASK
 *            UNTIL ALL ITEMS IN GROUP ARE COMPARED DO
 *              IF STATUS ITEM CHANGED THEN
 *                IF ITEM CHANGED EVEN NUMBER OF TIMES THEN
 *                  IF DEVICE CONTROL ACTION IN PROGRESS THEN
 *                    DO DOMCDVC - DEVICE CONTROL SUBROUTINE
 *                  ENDIF
 *                  DO DOMSETB - SUBROUTINE TO CALL SET BUILDER(DOMTBSET)
 *                ELSE
 *                  IF DEVICE CONTROL ACTION THEN
 *                    DO DOMCDVC
 *                  ELSE
 *                    DO DOMSETB
 *                  ENDIF
 *                  DO FLIP
 *                ENDIF
 *              ELSE - LATCH BIT IF OFF
 *                IF ITEM HAS CHANGED THEN
 *                  IF DEVICE CONTROL ACTION IS IN PROGRESS THEN
 *                    DO DOMCDVC
 *                  ELSE
 *                    DO DOMSETB
 *                  ENDIF
 *                  DO FLIP
 *                ENDIF
 *              ENDIF
 *              POINT TO NEXT STATUS ITEM IN GROUP
 *              ADJUST MASK
 *            ENDDO
 *            CHANGE GROUP WORD IN DATA BASE
 *          ENDIF
 *        ELSE  -STATUS GROUP IS UNLATCHED
 *          IF ANY CHANGES IN STATUS THEN
 *            SET UP MASK
 *            UNTIL ALL ITEMS IN GROUP ARE COMPARED DO
 *              IF ITEM HAS CHANGED STATUS THEN
 *                DO FLIP
 *                IF DEVICE CONTROL ACTION IN PROGRESS THEN
 *                  DO DOMCDVC
 *                ELSE
 *                  DO DOMSETB
 *                ENDIF
 *              ENDIF
 *              POINT TO NEXT STATUS ITEM IN GROUP
 *              ADJUST MASK
 *            ENDDO
 *            CHANGE GROUP WORD IN DATA BASE
 *          ENDIF
```

```
MEMBER NAME  DOMTCSES
 *        ENDIF
 *      ORELSE
 *        POINT TO NEXT STATUS GROUP
 *      ENDLOOP
 *        IF NO MATCH FOUND THEN
 *          SET UP MESSAGE NUMBER 465 -UNMATCHED ADDRESS/SEQUENCE #
 *          DO DOMERROR
 *        ENDIF
 *      END SEARCH
 *      POINT TO NEXT STATUS ENTRY IN RDA
 *    ENDDO
 *    IF THERE IS A STATUS LOG BUFFER THEN
 *      PATCH DOMCSLOG  -STATUS LOG PROCESSOR
 *      IF PATCH FAILED THEN
 *        FREE STATUS LOG AREA USING FREEWA MACRO
 *      ENDIF
 *      SET STAE FLAG TO ZERO
 *    ENDIF
 *    EXIT
 *
 *    ERROR ENTER ERR1
 *      SET UP MESSAGE NUMBER 464 - NO GETWA AVAILABLE
 *      DO DOMERROR
 *      EXIT
 * ENDSEGMENT DOMTCSES
 *
 *
 * DOMCDVC SUBROUTINE SEGMENT
 *    IF SYSTEM/370 IS CONTROLLING CPU THEN
 *      GET AREA FOR PASSING PARAMETERS USING GETWA
 *      BUILD PARAMETER LIST WITH S/7 ID, TERMINAL ID, STATUS GROUP AND
 *       ITEM ADDRESSES
 *      PATCH DOMCDC01 WITH PATCH ID OF 12
 *      IF PATCH FAILS THEN
 *        FREE PARAMETER LIST ADDRESS USING FREEWA MACRO
 *      ENDIF
 *    ELSE - 370 IS NOT CONTROLLING
 *      SET UP PARAMETERS
 *      CALL DOMCPDC1
 *      IF WALLBOARD FLAG IS ON THEN
 *        IF WALLBOARD BUFFER ADDRESS IS ZERO THEN
 *          GET AREA FOR BUFFER USING GETWA MACRO
 *          ERROR EXIT TO ERR1 IF RETURN CODE IS NOT ZERO
 *          SAVE ADDRESS OF BUFFER
 *        ELSE
 *          UNTIL CURRENT POSITION IN BUFFER FOUND DO
 *            CHAIN THROUGH BUFFERS
 *          ENDDO
 *        ENDIF
 *        IF WALLBOARD BUFFER IS FULL THEN
 *          GET AREA FOR NEXT BUFFER USING GETWA MACRO
 *          ERROR EXIT TO ERR1 IF RETURN CODE IS NOT ZERO
 *          CHAIN NEW BUFFER ONTO OLD
 *        ENDIF
 *        IF MOTOR OPERATED SWITCH OR
 *        IF TYPE 3 TCT THEN
 *          CALCULATE IF ITEM IF EVEN OR ODD
 *          POINT TO EVEN ITEM
```

```
MEMBER NAME  DOMTCSES
 *        ENDIF
 *        ADD ENTRY TO WALLBOARD BUFFER
 *        ADJUST BUFFER COUNT
 *      ENDIF
 *    ENDIF
 * ENDSEGMENT DOMCDVC
 *
 * DOMSETB SUBROUTINE SEGMENT
 *   CALL DOMTBSET PASSING ADDRESS OF STATUS ITEM AND STATUS GROUP
 * ENDSEGMENT DOMSETB
 *
 * DOMERROR SUBROUTINE SEGMENT
 *   ISSUE MESSAGE USING MESSAGE MACRO
 * ENDSEGMENT DOMERROR
 *
 * DOMLOG SUBROUTINE SEGMENT
 *   IF LOG BUFFER HAS NOT BEEN OBTAINED OR
 *   IF LOG BUFFER IS FULL THEN
 *     GET AN AREA USING GETWA MACRO CHAINING WHEN NECESSARY
 *   ENDIF
 *   MOVE THE STATUS ITEM TO THE BUFFER
 *   INCREMENT THE COUNT OF ITEMS IN THE BUFFER
 * ENDSEGMENT DOMLOG
 *
 * FLIP SUBROUTINE SEGMENT
 *   IF BIT IS ON IN RDA THEN
 *     SET ON STATUS BIT IN DATA BASE ITEM
 *   ELSE
 *     SET OFF STATUS BIT IN DATA BASE ITEM
 *   ENDIF
 * ENDSEGMENT FLIP
```

```
MEMBER NAME  DOMTDISK
 *   CSECT   DOMTDISK          /*  S/7 DISK LOAD  */
 *     IF S/7 HAS DISK, THEN
 *       IF ALL DISK SECTORS TRANSMITTED, THEN
 *         S7WRITE DISK INITIALIZATION COMPLETE
 *       ELSE                      DISK SECTORS REMAIN TO TRANSMIT
 *         READ THE CURRENT MEMBER
 *         FORMAT THE DESIRED SECTOR
 *         S7WRITE THE DESIRED SECTOR
 *         FORMAT THE DESIRED SECTOR
 *         INCREMENT SECTOR BY 1
 *         IF NEW MEMBER REQUIRED, THEN
 *           INCREMENT MEMBER COUNT BY 1
 *           ZERO SECTOR COUNT
 *         ENDIF
 *       ENDIF
 *     ELSE                       SYSTEM/7 HAS NO DISK
 *       S7WRITE DISK INITIALIZATION COMPLETE
 *     ENDIF
 *   EXIT  DOMTDISK
```

```
MEMBER NAME  DOMTEIPL
 *   CSECT   DOMTEIPL          /*  IPL S/7  /*
 *     UPDATE ARRAY TAS7COMM THAT IPL IS IN PROGRESS
 *     FIND  THE MEMBER  BOOTSTRP
 *     READ BOOTSTRP
 *     WHILE RECORDS ARE 'TX' RECORDS, DO
 *       STORE DATA IN FORMAT AREA
 *       READ NEXT RECORD
 *     ENDDO
 *     S7WRITE BOOTSTRP VIA IPL ADDRESS
 *     FIND THE MEMBER S7LOAD
 *     READ  S7LOAD
 *     WHILE RECORDS ARE 'TX' RECORDS, DO
 *       S7WRITE RECORD
 *       READ NEXT RECORD
 *     ENDDO
 *     S7WRITE 'EN' RECORD
 *     UPDATE ARRAY TAS7COMM THAT IPL IS COMPLETE
 *   END DOMTEIPL
```

```
MEMBER NAME  DOMTFAIL
 *  CSECT   DOMTFAIL                  /*  S/7 FAILOVER  */
 *
 *    IF VALID ID, THEN
 *      IF PRIMARY FAILURE, THEN
 *        VARY ALL TERMINALS OUT OF SERVICE THIS LID
 *      ELSE
 *        IF NOT BACKUP FAILURE, THEN
 *          ISSUE MESSAGE DPP207I
 *          EXIT PROGRAM
 *        ENDIF
 *      ENDIF
 *      PURGE ALL I/O TO S/7
 *      CLEAR DISK LOAD CONTRO ENTRY
 *      CLEAR S/7 CONTROL BYTE
 *      UPDATE TAS7COMM OF FAILURE
 *      EVENT THE FAILURE
 *      ALARM THE FAILURE
 *      IF S/370 SUPPORT DISK OPERATIVE, THEN
 *        IF PRIMARY FAILURE, THEN
 *          IF CURRENT BACKUP, THEN
 *            VARY BACKUP TO PRIMARY
 *          ELSE
 *            IF ANY S/7 ABLE THIS LID, AND
 *            IF THAT S/7 IS NOT IPLED, THEN
 *              VARY THE S/7 PRIMARY
 *            ELSE
 *              IF ANY S/7 ABLE THIS LID, AND
 *              IF THAT S/7 IPLED AS BACKUP OTHER LID, THEN
 *                VARY THE S/7 AS PRIMARY
 *              ELSE
 *                EVENT NO BACKUP AVAILABLE
 *              ENDIF
 *            ENDIF
 *          ENDIF
 *        ENDIF
 *      ENDIF
 *    ELSE
 *      ISSUE MESSAGE DPP207I
 *    ENDIF
 *    EXIT PROGRAM
 *
 *  END     DOMTFAIL
```

```
MEMBER NAME  DOMTHSMD
 *  CSECT    DOMTHSMD                  /*  REQUEST TO ENTER HIERARCHY  */
 *
 *    IF REQUEST FROM MASTER DISPATCHER, THEN
 *      STRTSRCH  TAS7COMM ARRAY
 *      EXITIF  INPUT LOGICAL ID FOUND
 *        IF LID ACTIVE AND
 *        IF UNIT WAITING FOR DISPATCHER ACTION, THEN
 *          INFORM DISPATCHER REQUEST ACCEPTED
 *          IF S/370 AT TOP, THEN
 *            WRITE STOP SCAN COMMAND TO S/7
 *          ELSE
 *            WRITE TOP/NOT-TOP, NODE UP, TO THE TOP
 *            UPDATE TAS7COMM ARRAY THAT NODE IS UP
 *          ENDIF
 *        ELSE
 *          FLAG S/7 NOT CONTROLLABLE
 *        ENDIF
 *      ORELSE
 *        CHECK NEXT LOCAL LOGICAL ID
 *      ENDLOOP
 *        FLAG INVALID LOGICAL ID
 *      ENDSRCH
 *    ELSE
 *      FLAG INVALID REQUEST/ACCESS-FUNCTION
 *    ENDIF
 *    IF ERROR, THEN
 *      DISPLAY ERROR MESSAGE TO DISPATCHER
 *    ENDIF
 *    EXIT THE PROGRAM
 *
 *  END      DOMTHSMD
```

```
MEMBER NAME  DOMTHS8B
 *  CSECT   DOMTHS8B                   /*  INITIAL SCAN TIMEOUT  */
 *
 *     STRTSRCH  REMOTE LID LIST IN TAS7COMM ARRAY
 *     EXITIF    ORIGIN ID IS A REMOTE
 *       USE LOCAL ID REMOTE REPORTS THROUGH
 *     ORELSE
 *       CHECK NEXT REMOTE
 *     ENDLOOP
 *       ASSUME LOCAL ORIGINATED MESSAGE
 *     ENDSRCH
 *     EVENT SCAN ABORT CONDITION
 *     NOTIFY DISPATCHER TO RETRY START SCANNING
 *     RELEASE THE INPUT BUFFER
 *     EXIT THE PROGRAM
 *
 *  END     DOMTHS8B
```

```
MEMBER NAME  DOMTHS8D
 *  CSECT   DOMTHS8D                /*  SCANNING HAS STOPPED  */
 *
 *    RELEASE THE INPUT BUFFER
 *    UPDATE TAS7COMM ARRAY
 *    TURN OFF DATA ACQUISITION SCAN PERFORMANCE BIT THIS LID
 *    IF S/370 AT THE TOP, THEN
 *      STRTSRCH  TAS7COMM ARRAY
 *      EXITIF  LOCAL LOGICAL ID FOUND
 *        LOCK  TAS7COMM ARRAY
 *        IF STOP IS TO CHANGE SCAN MODES, THEN
 *          INFORM LEG OF SCAN MODE
 *          RESUME CYCLIC SCANNING
 *          UNLOCK TAS7COMM ARRAY
 *        ELSE
 *          UNLOCK TAS7COMM ARRAY
 *          WRITE THE YEAR, DAY, HOUR, MINUTE, SECONDS AND HUNDRETHS
 *           OF A SECOND TO LOCAL S/7
 *          LOCK CASCAN ARRAY
 *          UNTIL ALL DEFINE CYCLIC SCANS ARE PROCESSED, DO
 *            WRITE TO THE LEG STATUS OF SCAN, ACTIVE/INACTIVE
 *            CHECK NEXT SCAN
 *          ENDDO
 *          WRITE SCANNING MODE TO LEG
 *          UNLOCK CASCAN ARRAY
 *          COMMAND THE LEG TO PERFORM INITIAL SCAN & START CYCLIC
 *          UPDATE TAS7COMM ARRAY THAT THE NODE IS UP
 *          CLEAR UNIT CONTROL FLAGS
 *        ENDIF
 *      ORELSE
 *        CHECK NEXT LOCAL LOGICAL ID
 *      ENDLOOP
 *      ENDSRCH
 *
 *    ENDIF
 *    EXIT THE PROGRAM
 *
 *  END     DOMTHS8D
```

```
MEMBER NAME  DOMTHS83
*  CSECT    DOMTHS83                  /*  INITIALIZATION COMPLETE  */
*
*    IF  S/7 INITIALIZATION FAILED, THEN
*      PATCH S/7 FAILOVER
*    ELSE
*      IF S/7 TO BECOME A PRIMARY, THEN
*        CHANGE FROM UTILITY TO SCAN BUFFERS
*        COMMAND S/7 TO BECOME PRIMARY
*      ELSE
*        FLAG CONTROL COMPLETE, S/7 NOW IN DESIRED STATE
*        EVVENT S/7 NOW OPERATIONAL AS BACKUP
*        DISPLAY SYSTEM MESSAGE THAT S/7 NOW OPERATIONAL AS BACKUP
*      ENDIF
*    ENDIF
*    RELEASE THE INPUT BUFFER
*    EXIT THE PROGRAM
*
*  END      DOMTHS83
```



```
MEMBER NAME  DOMTHS84
*   CSECT    DOMTHS84                 /*  TOP/NOT-TOP OF HIERARCHY  */
*
*     IF  INITIALIZATION FAILED, THEN
*       PATCH FAILOVER
*     ELSE
*       IF THERE IS A TOP/NOT-TOP STATUS CHANGE, THEN
*         SET NEW POSITION
*         EVENT THE NEW POSITION
*       ENDIF
*       IF INITIALIZATION TYPE TRANSACTION, THEN
*         VARY ALL TERMINALS ONLINE THIS LEG (LOCAL LID)
*         FLAG CONTROL COMPLETE
*         IF NO DISPATCHER ACTION REQUIRED, THEN
*           IF S/370 AT THE TOP OF HIERARCHY, THEN
*             COMMAND LEG TO STOP SCANNING
*           ELSE
*             INFORM TOP THAT THIS NODE READY TO ENTER HIERARCHY
*           ENDIF
*             FLAG NODE IS UP IN TAS7COMM ARRAY
*         ELSE
*           INFORM DISPATCHER THAT NODE READY TO ENTER THE HIERARCHY
*         ENDIF
*       ELSE
*         IF A NEW NODE IS UP, THEN
*           IF NODE IS UP FROM BELOW, THEN
*             IF VALID CONFIGURATION, VERSION AND MOD LEVEL, THEN
*               EVENT APPEARANCE OF NODE
*               NOTIFY DISPATCHER OF ENTRANCE OF NODE IN HIERARCHY
*               IF LOCAL S/7 NOT WAITING FOR DISPATCHER ACTION, THEN
*                 COMMAND LEG TO STOP SCANNING
*               ENDIF
*             ELSE
*               IGNORE TRANSACTION
*             ENDIF
*           ENDIF
*         ELSE
*           IF NODE IS DOWN, THEN
*             IF DOWN NODE IS BELOW, THEN
*               EVENT NODE DOWN(OFFLINE)
*             ENDIF
*           ENDIF
*         ENDIF
*       ENDIF
*     ENDIF
*     RELEASE THE INPUT BUFFER
*     EXIT THE PROGRAM
*
*   END      DOMTHS84
```

```
MEMBER NAME  DOMTHS97
 *  CSECT   DOMTHS97                 /*  S/7 OFFLINE  */
 *
 *    RELEASE INPUT BUFFER
 *    FLAG OFFLINE CONTROL COMPLETE
 *    VARY ALL TERMINAL OUT OF SERVICE FOR THIS LID
 *    UPDATE TAS7COMM ARRAY THAT S/7 OFFLINE
 *    EVENT THE S/7 OFFLINE
 *    EXIT  PROGRAM
 *
 *  END     DOMTHS97
```

```
MEMBER NAME  DOMTINFO
 *   CSECT      DOMTINFO            /*  S/7 SUPPORT INITIALIZATION  */
 *     CALCULATE REQUIRED SPACE
 *     GETMAIN REQUIRED SPACE
 *     BUILD FCVT
 *     BUILD ALL DECBS AND DCBS FOR S/7 DISK SUPPORT
 *     OPEN ALL S/7 DISK SUPPORT DCBS
 *     OPEN ALL S/7 UNIT DCBS
 *     IF S/7 CHECKPOINT SUPPORTED, THEN
 *       CALCULATE SIZE OF IN CORE TABLES FOR S/7 CHECKPOINT
 *       GETMAIN NECESSARY SPACE
 *       CHAIN AREA TO FCVT
 *       BUILD DCBS AND ARRAY ID TABLES
 *       IF CPINIT SPECIFIED, THEN
 *         INITIALIZE FILE TO ZEROS
 *       ENDIF
 *       OPEN REAL-TIME S/7 CHECKPOINT DCBS
 *     ENDIF
 *     EXIT PROGRAM
 *
 *   END        DOMTINFO
```

```
MEMBER NAME  DOMTIOER
 *  CSECT  DOMTIOER            PROCESS S/7 I/O ERRORS
 *     PATCH  S/7 FAILOVER PROGRAM OF S/7 I/O ERROR
 *     PURGE ALL I/O TO FAILING CPU
 *     POST REQUESTOR OF FAILURE IF OUTPUT I/O
 *     CLEAN UP IOB EXTENSION FLAGS
 *     EXIT BACK TO MAIN ROUTINE
 *  END   DOMTIOER
```

```
MEMBER NAME  DOMTPCLG
 *        DOMTPCLG MAIN SEGMENT
 *          IF THIS QUEUE IS ON THE EVEN HOUR THEN
 *            GET NUMBER OF PULSE COUNTER DATA POINTS IN CACOUNT
 *            DO UNTIL ALL POINTS HAVE BEEN PROCESSED
 *              IF THIS POINT IS IN SERVICE THEN
 *                IF THIS POINT IS ALARMED NOW THEN
 *                  DETERMINE NUMBER OF MINUTES ELAPSED SINCE LAST GOOD X
 *                        READING
 *                  MULTIPLY NUMBER OF ELAPSED MINUTES TIMES PC         X
 *                        EXTRAPALATION VALUE
 *                  ADD THIS AMOUNT TO THE ACCUMULATED AMOUNT
 *                  STORE NEW ACCUMULATED AMOUNT
 *                ENDIF
 *              ENDIF
 *            ENDDO
 *            PUTBLOCK FOR ARRAY CACOUNT
 *            PUTLOG FOR ARRAY CACOUNT
 *          ELSE
 *            PUTBLOCK FOR ARRAY CACOUNT
 *          ENDIF
 *        ENDSEGMENT DOMTPCLG
```

```
MEMBER NAME  DOMTPCNT
*  CSECT DOMTPCNT
*       'WARM START PULSE COUNTER INITIALIZATION ROUTINE'
*   GETARRAY FOR CACOUNT ARRAY
*   ERREXIT TO BADA IF GETARRAY FAILED
*   PICK UP NUMBER OF PC POINTS IN CACOUNT ARRAY
*   UNTIL ALL PULSE COUNTER POINTS CHECKED DO
*     RETRIEVE TIME OF LAST GOOD PC DATA
*     IF LAST GOOD DATA WAS GT CURRENT TIME THEN
*       SAVE THE MOST CURRENT TIME
*     ENDIF
*     RETRIEVE ADDRESS OF NEXT PC POINT
*   ENDDO
*   IF ANY GOOD PULSE COUNTER DATA THEN
*     SAVE THE HOUR DOWN
*     PICK UP MINUTES/DAY AND YEAR
*     CONVERT THE VALUES
*     IF SYSTEM WAS DOWN FOR MORE THAN ONE HOUR THEN
*       SET INDICATOR FOR MORE THAN ONE HOUR DOWN
*     ELSE
*       SET INDICATOR FOR LESS THAN ONE HOUR DOWN
*     ENDIF
*     BRING ALL PC POINTS UP TO THE FIRST EVEN HOUR
*     SET THE NUMBER OF MINUTES TO PROCESS
*     FORCE A CONTINOUS LOOP
*     UNTIL ALL PC ITEMS PROCESSED DO
*       IF DOWN A FULL HOUR THEN
*         DO HOURPC SEGMENT
*       ELSE
*         DO PCHRPART SEGMENT
*       ENDIF
*         PICK UP THE ADDRESS OF THE NEXT PC POINT
*     ENDDO
*       DO LOGPC SEGMENT
*   ENDIF
*   EXIT AND RETURN TO SYSTEM
*
* /* HOURPC SEGMENT 'USED FOR EVEN HOUR PC DATA PROCESSING'
*      COMPUTE CHANGES FOR ONE HOUR
*      SAVE THE CHANGED PC VALUES
* /* END HOURPC SEGMENT
*
* /* PCHRPART SEGMENT 'USED FOR PC DATA UP TO ONE HOUR'
*      COMPUTE CHANGES THAT OCCURED UP TO FIFTY NINE MINUTES
*      SAVE THE CHANGED PC VALUES
* /* END PCHRPART SEGMENT
*
* /* LOGPC SEGMENT 'USED TO LOG THE CHANGED PC VALUES TO THE DATA BASE'
*      LOG THE PULSE COUNTER CHANGES
* /* END LOGPC SEGMENT
*
* /* ERRENTER BADA
*      ISSUE MESSAGE MACRO TO INDICATE GETARRAY FAILED
* /* END
* END OF DOMTPCNT
```

```
MEMBER NAME  DOMTPSUM
 *        DOMTPSUM MAIN SEGMENT
 *        /* THE FUNCTION OF THIS PROGRAM IS TO DO POINT SUMMATION FOR  X
 *           ALL GROUPS IN THE POINT SUMMATION TABLE THAT HAVE A SCAN   X
 *           ID EQUAL TO THE CURRENT SCAN ID PASSED */
 *          GET THE SCAN ID PASSED
 *          DO UNTIL ALL SCAN ID'S IN THE POINT SUMMATION TABLE HAVE    X
 *                BEEN SEARCHED
 *            IF THIS SCAN ID IS SAME AS CURRENT SCAN ID THEN
 *              DO UNTIL ALL POINTS IN THIS GROUP HAVE BEEN PROCESSED
 *                GET THE DATA BY USING THE ADDRESS IN THE TABLE
 *                ADD THIS DATA TO THE TOTAL
 *              ENDDO
 *              IF DATA INDICATOR BIT FOR SUM POINT IS ON THEN
 *                ADD THE OLD SUM DATA VALUE TO THE NEW SUM VALUE AND   X
 *                      REPLACE THE NEW SUM VALUE WITH IT
 *              ENDIF
 *              PERFORM NORMAL ANALOG PROCESSING ON THE NEW SUM VALUE   X
 *                    (LIMIT CHECKING,ALARMS,ETC.)
 *              STORE THE NEW SUM VALUE IN DATA BASE SLOT POINTED TO BY X
 *                    SUM POINT ADDRESS IN POINT SUMMATION TABLE
 *            ENDIF
 *          ENDDO
 *        ENDSEGMENT DOMTPSUM
```

```
MEMBER NAME  DOMTPUNT
 *  CSECT  DOMTPUNT   -   ROUTE INPUT DATA FROM S/7
 *
 *     CALCULATE LENGTH OF INPUT = REQUESTED LENGTH - RESIDUAL COUNT
 *     STORE  LENGTH OF TEXT (BYTES) IN HEADER = TOTAL LENGTH - L'HEADER
 *     IF S/7 IN SCAN MODE, THEN
 *       IF  S/7 IN CYCLIC SCAN MODE, OR
 *       IF S/7 IN FIRST CYCLIC SCAN MODE, THEN
 *         IF INPUT IS SCAN DATA, OR
 *         IF INPUT IS POLLOVERLOAD, THEN
 *           CANCEL TIMEOUTCOND SECURITY REPLY COMPLETE
 *         ENDIF
 *       ELSE
 *         CANCEL TIMEOUT
 *       ENDIF
 *     ELSE
 *       CANCEL TIMEOUT
 *     ENDIF
 *     IF  INPUT IS SECURITY MESSAGE, THEN
 *       RELEASE THE BUFFER
 *     ELSE
 *       IF SCANNING STOPPED, OR
 *       IF INITIAL SCAN TIMEOUT, THEN
 *         FLAG S/7 IN NON-SCAN MODE
 *       ELSE
 *         IF SCAN DATA, THEN
 *           IF THIS SCAN IS INITIAL SCAN, THEN
 *             SET FIRST CYCLIC SCAN FLAG
 *             CLEAR CYCLIC SCAN FLAG
 *           ELSE
 *             CLEAR FIRST CYCLIC SCAN FLAG
 *             SET CYCLIC SCAN FLAG
 *           ENDIF
 *         ELSE
 *           IF  S/7 OFFLINE REPLY, THEN
 *             FLAG IOB S/7 OFFLINE, NO MORE CYCLIC READS
 *           ENDIF
 *         ENDIF
 *       ENDIF
 *       STRTSRCH     SEARCH ROUTING TABLE FOR TRANSACTION CODE (TC)
 *       EXITIF   TC FOUND
 *         IF  INPUT NOT SCAN DATA, AND
 *         IF  DATA IN SCAN BUFFER, THEN
 *           GET A UTILITY BUFFER
 *           IF UTILITY BUFFER AVAILABLE, THEN
 *             MOVE DATA FROM SCAN BUFFER TO UTILITY BUFFER (L'UTILITY)
 *             RELEASE SCAN BUFFER
 *             IF DATA TRUNCATED IN MOVE, THEN
 *               STORE NEW TEXT LENGTH IN HEADER
 *             ENDIF
 *             BRANCH TO ROUTE TRANSACTION
 *           ELSE
 *             ISSUE MESSAGE  DPP204I - NO UTILITY BUFFER AVAILABLE TO
 *                                      ROUTE TC FROM S/7
 *             RELEASE SCAN BUFFER
 *           ENDIF
 *         ELSE                        ROUTE THE TRANSACTION
 *           IF GENERAL COMMAND TRANSACTION, THEN
 *             LOAD COMMAND CODE
```

```
MEMBER NAME  DOMTPUNT
 *            MULTIPLY BY LENGTH OF ENTRY IN ROUTING TABLE
 *            ACCESS CORRECT PATCH FORM
 *          ENDIF
 *          FORMAT PARAM  -  UNIT INDEX + A(TRANSACTION)
 *          PATCH  PROCESSING PROGRAM
 *          IF  PATCH FAILED, THEN
 *            ISSUE MESSAGE DPP231I - PATCH FAILED ROUTING TRANSACTION
 *            RELEASE BUFFER
 *          ENDIF
 *        ENDIF
 *      ORELSE                      NEXT ENTRY IN ROUTING TABLE
 *        INCREMENT TO NEXT ENTRY IN ROUTING TABLE
 *      ENDLOOP                     NO MATCH
 *        ISSUE MESSAGE DPP205I - INVALID TRANSACTION CODE
 *        RELEASE BUFFER
 *      ENDSRCH
 *    ENDSEGMENT DOMTPUNT
 *
```

```
MEMBER NAME  DOMTP1IN
*   CSECT    DOMTP1IN            /*  PHASE ONE EMS INITIALIZATION   */
*
*     CALCULATE SIZE OF EMS DATA/WORK AREA
*     GETMAIN FOR REQUIRED SPACE
*     IF  GETMAIN NOT SATISFIED
*       ABEND THE JOBSTEP
*     ENDIF
*     CHAIN EMSCVT TO DPPXCVT
*     ADD WORK AREA TO DPPXFIX ARRAY
*     BUILD THE S/7 CONTROL TABLE
*     BUILD AND INITIALIZE S/7 COMMUNICATION TASK AREAS
*     PATCH  DOMTSINT - DAQ INITIALIZATION
*     IF PROCESSING SUCCESSFUL, THEN
*       CALL  DOMCSENT
*     ENDIF
*     EXIT PROGRAM
*
*   END      DOMTP1IN
```

```
MEMBER NAME  DOMTP2IN
 *  CSECT   DOMTP2IN
 *    PATCH  S/7 I/O TASK
 *    DO UNTIL ALL LOGICAL IDS ARE PROCESSED .
 *      IF BACKUP ASSIGNED, THEN
 *        VARYS7  S/7 TO BACKUP MODE
 *      ENDIF
 *      IF PRIMARY ASSIGNED, THEN
 *        VARYS7  S/7 TO PRIMARY MODE
 *      ENDIF
 *    ENDDO
 *  END  DOMTP2IN
```

```
MEMBER NAME  DOMTRESI
*  CSECT  DOMTRESI        /*  EMS INITIALIZATION  */
*
*    IF  PATCH IS PRE-RESTART WRITE, THEN
*      PAGE FIX ALL ITEMS IN PAGE FIX ARRAY
*      SET SRTOS STAE PROCESSOR TO ABEND JOB STEP IF S/7 I/O PGM ABENDS
*      PROCESS ALL PATCH PARAM INPUT
*      INITIALIZE THE EMS TASK STRUCTURE, INCLUDING STAE PROCESSING,
*      PATCH DISM INITIALIZATION PHASE 1 WITH DESIRED PRIORITY
*      CALL PHASE 1 DATA ACQUISITION INITIALIZATION
*      IF SUCCESSFUL, THEN
*        PATCH  DOMTINFO TO OPEN S/7 DCBS AND BUILD FCVT AND S/7
*          CHECKPOINT CONTROL AREAS
*        IF SUCCESSFUL, THEN
*          PATCH DOMCINIT - SUPERVISORY CONTROL INITIALIZATION PROGRAM
*        ENDIF
*      ENDIF
*    ELSE                         POST RESTART WRITE PATCH
*      PAGE FIX ALL ITEMS IN PAGE FIX ARRAY
*      PATCH PHASE TWO DISM INITIALIZATION
*      REINITIALIZE TAS7COMM ARRAY
*      PATCH PHASE TWO SUPERVISORY CONTROL INITIALIZATION
*      CALL DOMTALPI - PERFORMANCE LOG INITILIZATION
*      CALL DOMTP2IN - PHASE TWO DAQ INITIALIZATION
*      IF WALLBOARDS PRESENT IN SYSTEM, THEN
*        CALL DOMCWBIN - WALLBOARD INITIALIZATION
*      ENDIF
*      EVENT  INITIALIZATION COMPLETE
*    ENDIF
*    EXIT PROGRAM
*
*  END   DOMTRESI
```

```
MEMBER NAME  DOMTRLBF
 *  CSECT   DOMTRLBF                  /*  RELEASE INPUT BUFFER  */
 *
 *    LOCATE BUFFER PREFIX
 *    IF BUFFER IS NOT A DYNAMIC BUFFER, THEN
 *      ZERO ALLOCATED FLAG TO RELEASE THE BUFFER
 *    ELSE
 *      FREEWA DYNAMIC BUFFER
 *    ENDIF
 *    EXIT
 *
 *  END     DOMTRLBF
```

```
MEMBER NAME  DOMTSCAN
 *        BEGIN  DOMTSCAN
 *        /*  THIS CSECT IS ENTERED FROM INITSCAN MACRO.  THE CSECT IS
 *        USED TO CHANGE BETWEEN INITIAL,STANDARD,AND EMERGENCY MODES*/
 *
 *        CASE1 CYCLIC MODE,DO
 *          MOVE FLAG TO MESSAGE TO NOTE STANDARD MODE
 *        ENDDO
 *
 *        CASE2 INITIAL MODE,DO
 *          MOVE FLAG TO MESSAGE TO NOTE INITIAL MODE
 *        ENDDO
 *
 *        CASE3 EMERGENCY MODE,DO
 *          MOVE FLAG TO MESSAGE TO NOTE EMERGENCY MODE
 *        ENDDO
 *
 *        IF SYS=ALL WAS SPECIFIED BY THE INITSCAN THEN
 *          DO UNTIL ALL SYS/7 IDS ARE USED
 *            S7WRITE MACRO FOR EACH SYS/7
 *            IF THE RETURN CODE WAS BAD FROM S7WRITE
 *              SAVE ERROR CODE AND KEEP TOTAL COUNT OF ERROR
 *            ELSE
 *              IF THIS IS AN EMERGENCY SCAN THEN
 *                SET FLAG IN THE S7CT
 *              ENDIF
 *            ENDIF
 *            NEXT SYS/7 S7CT
 *          ENDDO
 *          LOAD TOTAL NUMBER OF ERRORS AND ERROR CODE
 *        ELSE
 *          S7WRITE MACRO WITH SYS/7 ID FROM INITSCAN MACRO
 *          IF THE RETURN FROM THE S7WRITE WAS ZERO THEN
 *            IF THIS IS AN EMERGENCY SCAN THEN
 *              SET FLAG IN THE S7CT
 *            ENDIF
 *          ENDIF
 *        ENDIF
 *        RETURN
 *
 *        END  DOMTSCAN
```

```
MEMBER NAME  DOMTSINT
 *        DOMTSINT MAIN SEGMENT (DATA ACQUISITION INITIALIZATION)
 *          GET ADDRESS OF WORK AREA (ECVTDQWA) FROM EMSCVT
 *          INITIALIZE WORK AREA
 *          GETARRAY TYPE=ADDR FOR CADSPA,CADBIND,CASPECLM
 *          PUT ADDRESSES IN WORK AREA
 *          DEFINE LOCKS FOR 'CONSISTENT DATA' AND 'REAL TIME DATA'
 *          BUILD LIST FOR PUTBLOCK OF CACOUNT ARRAY
 *          FIND ADDRESS OF PULSE DURATION OUTPUT ARRAY AALFCPDO AND     X
 *                NUMBER OF GENERATOR ENTRIES IN IT.
 *          STORE ADDRESS AND COUNT IN EMSCVT
 *        ENDSEGMENT DOMTSINT
```

```
MEMBER NAME  DOMTSIOA
 *   CSECT    DOMTSIOA              START I/O APPENDAGE
 *
 *     SET BIT X'20', DEBRSIOA, OF BYTE X'0E', DEBFLGS1, ON IN THE DEB
 *    */ THIS IS NECESSARY TO HAVE THIS APPENDAGE REENTERED AT EVERY  /*
 *    */ SIO RATHER THAN ONLY ONCE PRIOR TO THE INITIAL SIO          /*
 *     INCREMENT BY 1 THE NUMBER OF TIMES THIS APPENDAGE ENTERED
 *     IF  THE LIMIT REACHED, THEN
 *       POST THE I/O COMPLETE WITH A X'61'
 *       RETURN - SKIP THE I/O OPERATION
 *     ELSE
 *       RETURN - NORMAL
 *     ENDIF
 *   END      DOMTSIOA
```

```
MEMBER NAME  DOMTSSYN
*          DOMTSSYN MAIN SEGMENT (SCAN SYNC AND CONVERSION DIRECTOR)
*            PICK UP ADDRESSES OF S/7 CONTROL TABLE & DATABASE INDICATOR
*             ARRAYS
*            DO DEGMENT DOMTSYN2
*            IF S/7 SCANNING CONSISTENT DATA
*              DO SEGMENT DOMTSTIM
*            ENDIF
*            ENDIF
*            RELEASE INPUT BUFFER
*            EXIT
*          ENDSEGMENT DOMTSSYN
*
*          DOMTSYN2 SUBROUTINE SEGMENT(S/7 RDA SYNCHRONIZATION)
*            IF RDA TRANSACTION CODE NOT 8A(SCAN DATA) OR 90(POLL OVRLD)
*              ISSUE ERROR MESSAGE
*            ELSE
*              SEARCH THE S/7 COMM TABLE FOR RDA ORIGINATING S/7
*                IF THE LOGICAL ID IS NOT SCANNING
*                  INITIATE SCANNING
*                ENDIF
*                FLAG THE S/7 AS ACTIVE IN THE EMSCVT
*                IF THE PATCH ID IS 4
*                  SAVE THE SCAN TIME FROM THE RDA
*                  DO SEGMENT DOMTSBUF
*                ELSE
*                  IF PATCH ID IS 8
*                    DO SEGMENT DOMTSS7S
*                  ELSE
*                    ISSUE INVALID PATCH ID MESSAGE
*                  ENDIF
*                ENDIF
*              ORELSE
*                BUMP TO NEXT S/7 CONTROL TABLE ENTRY
*              ENDLOOP
*              ISSUE UNMATCHED S/7 ID IN S/7 CONTROL TABLE AND RDA
*               MESSAGE
*          ENDSEGMENT DOMTSYN2
*
*          DOMTSTIM SUBROUTINE SEGMENT(UPDATE SPA AND DETECT FAILING S/7)
*           IF S/7 HAS INPUT VALID SYSTEM TIME
*             TURN THE INDICATOR OFF
*           ELSE
*             IF FIRST COMPLETE DATA SCAN
*               UPDATE CATIMES ARRAY WITH TIME IN 10 MIL UNITS
*               IF INITIAL SCAN
*                 SET FLAG FOR PROGRAM DOMTCLOK
*               ENDIF
*               COMPUTE POWER SYSTEM TIME
*             ENDIF
*           ENDIF
*           IF S/7 FAILED
*             DO UNTIL S/7 FOUND THAT DID NOT FAIL ON LAST SCAN
*              CYCLE
*               TURN OFF INDICATOR FOR ACTIVE S/7S THAT SHIPPED DATA
*                THIS SCAN
*             ENDDO
*             DO SEGMENT DOMTSRST
*           ENDIF
```

```
MEMBER NAME  DOMTSSYN
*          IF INITIAL SCAN
*            IF PC DATA HAS BEEN RECEIVED WITHIN LAST MINUTE CYCLE
*              TURN PC INDICATOR OFF
*              PATCH PROGRAM DOMTPCLG FOR HOURLY PC LOG
*              ISSUE ERROR MESSAGE FOR INVALID DOMTPCLG RETURN CODES
*             SAVE FAILING INDICATORS
*           ENDIF
*           BRANCH TO POINT SUMMATION ROUTINE DOMTPSUM
*           PATCH USER CYCLIC PROCESSOR DOMUSERC
*           ISSUE ERROR MESSAGE FOR INVALID DOMUSERC RETURN CODES
*           PATCH ANALOG LOGGING PROCESSOR DOMTAPLP
*           ISSUE ERROR MESSAGE FOR INVALID DOMTAPLP RETURN CODES
*           ENDIF
*         ENDSEGMENT DOMTSTIM
*
*         DOMTSBUF SUBROUTINE SEGMENT (UPDATE SPA,SET LOCKS,CALL CONV )
*           IF THIS S/7 IS ACTIVE THEN
*             IF DATA BASE IS OPEN THEN
*               LOCK CONSISTENT DATA RESOURCE
*               SET NORMAL MODE IN SCAN ARRAY AND S/7 CONTROL TABLE
*             ENDIF
*             IF THE UPDATE CYCLE NOT IN PROGRESS
*               SET THE UPDATE CYCLE IN PROGRESS INDICATOR
*               LOCK THE CONSISTENT DATA RESOURCE
*               ISSUE ERROR MESSAGE IF LOCK FAILED
*             ENDIF
*           ENDIF
*           LOCK REAL TIME DATA RESOURCE
*           ISSUE ERROR MESSAGE IF LOCK FAILED
*           CALL DATA CONVERSION ROUTINE DOMTCONV
*           UNLOCK REAL TIME DATA RESOURCE
*           ISSUE ERROR MESSAGE IF UNLOCK FAILED
*           IF THE INITIAL SCAN
*             IF THERE IS A WALLBOARD S/7
*               IF THE ORIGINATING RDA S/7 IS THE WALLBOARD S/7
*                 PATCH WALLBOARD PROCESSOR DOMCWBPR
*               ENDIF
*             ENDIF
*           ENDIF
*           IF THE S/7 ACTIVE
*             SAVE THE SCAN ID
*             IF NOT THE INITIAL SCAN
*               SEARCH UNTIL SCAN ID IS FOUND IN SPA
*               IF INDICATOR IS NONZERO
*                 CALCULATE SCAN COMPLETE DURATION,DAMPED MEAN ABS.
*                   DEVIATION, LAST BASIC SCAN CYCLE BOUNDARY
*               ENDIF
*               ISSUE ERROR MESSAGE IF SCAN ID IS NONZERO AND IS NOT
*                FOUND IN SPA
*             ENDIF
*             IF FIRST DATA SCAN HAS NOT OCCURRED
*               SET FIRST DATA SCAN INDICATOR
*               PICK UP SYSTEM DEVIATION FROM RDA
*               IF RDA DEVIATION INDICATOR IS ON
*                 CONVERT SYSTEM TIME RDA DEVIATION TO 10 MIL UNITS
*               ENDIF
*               PICK UP HOURS FROM EXTERNAL TIME STANDARD INDICATOR
*               IF HOURS OR SECONDS FROM EXTERNAL TIME STANDARD PLUS
```

```
MEMBER NAME  DOMTSSYN
*                   CONVERT TIME TO 10 MIL UNITS
*                    IF INITIAL SCAN
*                      SET FLAG FOR PROGRAM DOMTCLOK
*                    ENDIF
*                   UPDATE TIMES IN CATIMES ARRAY
*                   INDICATE THAT S/7 HAS INPUT A VALID SYSTEM TIME
*                 ENDIF
*               ENDIF
*               IF HOUR ROLLOVER OCCURRED
*                 SET PC DATA HAS BEEN OUPUT INDICATOR
*               ELSE
*                 TURN OFF PC DATA HAS BEEN OUPUT INDICATOR
*                 SET PC DATA PROCESSING HAS BEEN DOWN INDICATOR
*               ENDIF
*               ISSUE ERROR MESSAGE IF SCAN DATA LOST
*               DO SEGMENT DOMTSRST
*             ENDSEGMENT DOMTSBUF
*
*           DOMTSS7S SUBROUTINE SEGMENT (PROCESS POLL OVERLOAD)
*             ISSUE POLL OVERLOAD MESSAGE AND GENERATE EVENT
*             DO SEGMENT DOMTSRST
*           ENDSEGMENT DOMTSS7S
*
*           DOMTSRST SUBROUTINE SEGMENT (SET INDICATORS,CALL DOMTPSUM)
*             TURN ANY BUFFER ARRIVED INDICATOR ON FOR THIS S/7
*             TURN OFF DATA CONSISTENCY INDICATORS FOR INACTIVE S/7S
*             IF UPDATE CYCLE IS IN PROGRESS AND SOME S/7 DATA INCONSIS.
*               UNLOCK CONSISTENT DATA RESOURCE
*               ISSUE ERROR MESSAGE IF UNLOCK FAILED
*               TURN OFF UPDATE CYCLE IN PROGRESS INDICATOR
*             ENDIF
*             TURN OFF INDICATORS THAT DATA HAS BEEN SHIPPED FROM S/7
*              FOR AGC PROCESSING FOR INACTIVE S/7S
*             IF SOME S/7S ARE ACTIVE AND SOME DATA HAS BEEN SHIPPED
*              FROM S/7 FOR AGC PROCESSING
*              SET INDICATOR FOR NEW AGC DATA TO USE AND NEW DATA FOR USE
*             ENDIF
*           ENDSEGMENT DOMTSRST
```

```
MEMBER NAME  DOMTS7HS
*  CSECT   DOMTS7HS                /*  S/7 HIERARCHY SUPPORT  */
*
*    IF  PATCH ID VALID, THEN
*      RESTORE ALL REGISTERS
*      BRANCH TO PROCESSING SUBROUTINE
*    ENDIF
*    EXIT THE PROGRAM
*
*  END     DOMTS7HS
```

```
MEMBER NAME  DOMTS7IO
*       CSECT   DOMTS7IO    S/7 I/O SUPERVISOR
*
*          UNTIL TASK TERMINATES, DO
*            WAIT  FOR ANY ONE ECB TO BE POSTED
*
*    /* PROCESS REQUESTS FOR I/O PURGE */
*            IF   PURGE REQUEST ECB POSTED, THEN
*              IF  BUFFER CHANGE REQUEST, THEN
*                FLAG IOB TO USE SCAN BUFFERS
*              ELSE
*                DO SEGMENT PURGE7IO
*              ENDIF
*              POST REQUESTOR OF COMPLETION
*            ENDIF
*
*    /* PROCESS OUTSTANDING READ ECB'S */
*            UNTIL ALL READ ECBS PROCESSED, DO
*              IF ECB POSTED, THEN
*                IF POST CODE = X'7F', THEN
*                  CALL DOMTPUNT TO ROUTE THE TRANSACTION
*                  DO SEGMENT CHEKTIME
*                  ISSUE READ ENTRY POINT FOR BRANCH
*                  IF S/7 ONLINE, THEN
*                    DO SEGMENT GETBUFFR
*                    IF BUFFER OBTAINED, THEN
*                      BUILD CHANNEL PROGRAM
*                      DO SEGMENT SETIME
*                    ENDIF
*                  ENDIF
*                ELSE
*                  CALL SUBROUTINE DOMTIOER - I/O ERROR ANALYZER
*                ENDIF
*              ELSE
*                IF  CYCLIC READ FLAG SET, THEN
*                  IF I/O OUTSTANDING, THEN
*                    DO SEGMENT CHEKTIME
*                  ELSE
*                    DO SEGMENT GETBUFFR
*                    IF BUFFER OBTAINED, THEN
*                      DO SEGMENT SETIME
*                    ENDIF
*                  ENDIF
*                  ELSE
*                    BRANCH TO I/O ERROR ROUTINE DOMTIOER
*                ENDIF
*              ELSE
*                IF CYCLIC READ
*                  IF I/O OUTSTANDING
*                    DO SEGMENT CHEKTIME
*                  ELSE
*                    BRANCH TO ISSUE NEXT READ ENTRY POINT
*                  ENDIF
*                ENDIF
*              ENDIF
*            ENDDO
*
*   /*  PROCESS OUTSTANDING WRITE ECBS  */
*           UNTIL ALL WRITE ECBS PROCESSED, DO
```

```
MEMBER NAME  DOMTS7IO
*             IF ECB POSTED, THEN
*               FLAG I/O COMPLETE, CANCEL TIMEOUT
*               IF POST CODE = X'7F', THEN
*                 IF OUTPUT TRANSACTION IS OFFLINE COMMAND, THEN
*                   FLAG IOB OFFLINE
*                 ENDIF
*                 IF  USER DECB REQUESTING CYCLIC READ, THEN
*                   STORE CYCLIC READ BIT
*                 ENDIF
*                 POST USER ECB WITH RETURN CODE = 0
*               ELSE
*                 CALL SUBROUTINE DOMTIOER
*               ENDIF
*             ELSE
*               DO SEGMENT CHEKTIME - CHECK FOR EXCP TIMEOUT
*             ENDIF
*           ENDDO
*
*  /*  PROCESS OUTSTANDING EIPL ECBS  */
*           UNTIL ALL EIPL ECBS PROCESSED, DO
*             IF ECB POSTED, THEN
*               FLAG I/O COMPLETE, CANCEL TIMEOUT
*               IF POST CODE = X'7F', THEN
*                 POST USER ECB WITH RETURN CODE = 0
*               ELSE
*                 CALL SUBROUTINE DOMTIOER
*               ENDIF
*             ELSE
*               DO SEGMENT CHEKTIME - CHECK FOR EXCP TIMEOUT
*             ENDIF
*           ENDDO
*
*  /*  PROCESS REQUESTS FOR WRITE ECBS  */
*           UNTIL ALL WRITE REQUEST ECBS PROCESSED, DO
*             IF ECB POSTED, THEN
*               IF  IOB ONLINE, THEN
*                 BUILD CHANNEL PROGRAM
*                 DO SEGMENT SETIME
*               ELSE
*                 POST USER S/7 OFFLINE
*               ENDIF
*             ENDIF
*           ENDDO
*
*  /*  PROCESS REQUESTS FOR IPL  */
*           UNTIL ALL EIPL REQUEST ECBS PROCESSED, DO
*             IF ECB POST, THEN
*               RESET ALL IOB,FLAGS THIS S/7 TO NCMINAL VALUES
*               BUILD CHANNEL PROGRAM
*               FLAG UCBS FOR WRITE AND EIPL UNITS READY & NOT BUSY
*               FLAG UCB FOR READ UNIT AS READY & DEVICE BUSY
*               GO INTO KEY ZERO STATE
*               SET READ UNIT BUSY TO QUEUE 1ST REQUEST UNTIL S/7 READY
*               COME OUT OF KEY ZERO STATE
*               DO SEGMENT SETIME
*             ENDIF
*           ENDDO
*
```



```
MEMBER NAME  DOMTS7IO
*           /* PROCESS STIMER ECB */
*           ZERO STIMER ECB
*           STIMER - REAL TIME OF BASIC SCAN CYCLE EXIT=DOMTIMEX
*         ENDDO
*
*
*
*       GETBUFFR  SEGMENT  -  GET A BUFFER
*         FLAG IOB WITH BUFFER AVAILABLE
*         IF  SCAN BUFFERS NEEDED, THEN
*           STRTSRCH SCAN BUFFER POOL ADDRESS LIST
*           EXITIF  CORRECT BUFFER POOL FOUND
*             STRTSRCH UNTIL END OF LIST
*             EXITIF MOST RECENT BUFFER FOUND
*               ALLOCATE THE NEXT BUFFER
*               SET NEW BUFFER AS MOST RECENT
*               ZERO BUFFER
*             ORELSE
*               INCREMENT TO NEXT LIST ENTRY
*             ENDLOOP                END OF LIST
*         /*  THIS IS MOST RECENT BUFFER  */
*               ALLOCATE FIRST BUFFER IN LIST
*               SET NEW BUFFER AS MOST RECENT
*               ZERO BUFFER
*             ENDSRCH
*           ORELSE
*             INCREMENT TO NEXT BUFFER POOL ADDRESS
*           ENDLOOP
*       /*  INVALID LID  */
*             RETURN BUFFER NOT AVAILABLE
*           ENDSRCH
*         ELSE
*           STRTSRCH  UNTIL ALL UTILITY BUFFERS SEARCHED
*           EXITIF  BUFFER AVAILABLE
*             ZERO BUFFER
*           ORELSE
*             NEXT BUFFER
*           ENDLOOP
*             GETWA A DYNAMIC BUFFER
*             IF AVAILABLE, THEN
*               ZERO THE BUFFER
*             ELSE
*               RETURN NO BUFFER AVAILBALE
*             ENDIF
*           ENDSRCH
*         ENDIF
*       END SEGMENT  GETBUFFR
*
*
*
*       CHEKTIME SEGMENT -  DETERMINE IF EXCP TIMED OUT
*         IF I/O OUTSTANDING THIS IOB, THEN
*           IF CURRENT TIME > TIMEOUT TIME, THEN
*             IF  FIRST TIMEOUT, THEN
*               SET TIMEOUT FLAG ON
*               CALC. NEW TIMEOUT TIME
*             ELSE                2ND TIMEOUT
*               SET IOB - NO OUTSTANDING I/O
```

```
MEMBER NAME  DOMTS7IO
*               FAIL THE S/7
*               PATCH TASK TO PROCESS S/7 TIMEOUT FAILURE
*               SET ALL THREE UNIT IOBS OFFLINE
*             ENDIF
*           ENDIF
*         ENDIF
*       END SEGMENT CHEKTIME
*
*
*
*       SETIME SEGMENT  -   SET TIMEOUT INTERVAL
*         ISSUE I/O COMMAND - EXCP
*         IF INTERVAL NOT ALREADY SET, THEN
*           OBTAIN CURRENT TIME
*           IF READ EXCP, THEN
*             SET TIMEOUT BIT
*           ELSE
*             IF S/7 IN SCAN MODE, THEN
*               IF CYCLIC SCAN FLAG ON, THEN
*                 ADD PRIMARY INTERVAL TO CURRENT TIME
*               ELSE
*                 IF FIRST CYCLIC SCAN FLAG ON, THEN
*                   ADD 3 * PRIMARY INTERVAL TO CURRENT TIME
*                 ELSE
*                   ADD BACKUP INTERVAL TO CURRENT TIME
*                 ENDIF
*               ENDIF
*             ELSE
*               ADD BACKUP INTERVAL TO CURRENT TIME
*             ENDIF
*           ELSE
*             ADD WRITE INTERVAL TO CURRENT TIME
*           ENDIF
*           STORE TIMEOUT TIME IN IOB
*           TURN ON TIME SET BIT
*         ENDIF
*       END SEGMENT SETIME
*
*
*       DOMTIMEX SEGMENT -  STIMER ASYNCHRONOUS EXIT
*         LOCATE  STIMER ECB ADDRESS
*         POST STIMER ECB CODE=0
*         EXIT  DOMTIMEX
*       END SEGMENT DOMTIMEX
*
*       PURGE7IO SEGMENT - PURGE ALL I/O TO S/7
*         UNTIL  ALL 3 DEVICE ADDRESSES PURGED
*           IF  IOB ONLINE, THEN
*             FLAG IOB OFFLINE
*             IF  EXCP OUTSTANDING, THEN
*               PURGE I/O
*               FLAG IOB NO EXCP OUTSTANDING
*               IF I/O NOT COMPLETED BEFORE PURGE, THEN
*                 IF READ IOB, THEN
*                   RELEASE THE BUFFER
*                 ELSE              OUTPUT OR IPL IOB
*                   POST REQUESTOR - S/7 FAILED
*                 ENDIF
```

```
MEMBER NAME  DOMTS7IO
*              ENDIF
*            ENDIF
*          ENDIF
*          PROCESS NEXT DEVICE ADDRESS
*        ENDDO
*      END SEGMENT  PURGE7IO
*
*
*
*      END DOMTS7IO
```

```
MEMBER NAME  DOMTUPD7
 *    CSECT  DOMTUPD7                 /*  UPDATE AND LOG TAS7COMM ARRAY */
 *
 *    IF LOGICAL ID VALID, THEN
 *      IF UNIT AVAILABLE THIS LID, THEN
 *        IF TYPE OF UPDATE VALID, THEN
 *          LOCK TAS7COMM ARRAY
 *          UNTIL ALL LOCAL LID PROCESSED, DO
 *            IF S/7 AVAILABLE THIS LID, THEN
 *              IF S/7 IPLED THIS LID, THEN
 *                PERFORM DESIRED LFLAG UPDATE
 *              ENDIF
 *              UPDATE PFLAG ENTRY THIS S/7
 *            ENDIF
 *          ENDDO
 *          IF ARRAY TO BE LOGGED, THEN
 *            LOG TAS7COMM ARRAY
 *          ENDIF
 *          UNLOCK TAS7COMM ARRAY
 *        ELSE
 *          SET INVALID TYPE RETURN CODE
 *        ENDIF
 *      ELSE
 *        SET UNIT UNAVAILABLE RETURN CODE
 *      ENDIF
 *    ELSE
 *      SET INVALID LID RETURN CODE
 *    ENDIF
 *    IF AN EVENT IS TO BE ISSUED, THEN
 *      EVENT THE UPDATE
 *    ENDIF
 *    EXIT THE PROGRAM
 *
 *
 *    /*  LFLAG UPDATE SEGMENTS  */
 *    NEW-PRIMARY SEGMENT
 *      INSERT ROUTING INDEX
 *      FLAG LFLAG THAT PRIMARY IS ASSIGNED
 *      EVENT UPDATE
 *      LOG ARRAY
 *    END SEGMENT NEW-PRIMARY
 *
 *    NEW-BACKUP SEGMENT
 *      INSERT ROUTING INDEX
 *      FLAG LFLAG THAT BACKUP IS ASSIGNED
 *      EVENT THE UPDATE
 *      LOG THE ARRAY
 *    END SEGMENT NEW-BACKUP
 *
 *    INACTIVE/FAILURE-OF-S/7 SEGMENT
 *      DEALLOCATE S/7 AS BACKUP OR PRIMARY
 *      LOG THE ARRAY
 *    END INACTIVE/FAILURE-OF-S/7 SEGMENT
 *
 *    LID-SCANNING SEGMENT
 *      INDICATE LOGICAL ID NOW SCANNING
 *      EVENT THE ACTION
 *    END LID-SCANNING SEGMENT
 *
```

```
MEMBER NAME   DOMTUPD7
 *    LID-STOPPED-SCANNING SEGMENT
 *      INDICATE LOGICAL ID NOT SCANNING
 *      EVENT SCANNING STOPPED
 *    END LID-STOPPED-SCANNING SEGMENT
 *
 *    S/370-DISK-ERROR SEGMENT
 *      FLAG S/370 DISK SUPPORT ERROR
 *      PERFORM INACTIVE/FAILURE-OF-S/7 SEGMENT
 *    END S/370-DISK-ERROR SEGMENT
 *
 *    NODE-IS-UP SEGMENT
 *      FLAG LID AS ENTERED THE HIERARCHY
 *    END NODE-IS-UP-SEGMENT
 *
 *    END     DOMTUPD7
```

```
MEMBER NAME  DOMTUSER
 * BEGIN DOMTUSER
 * /* THIS PROGRAM PROCESSES RETRIEVED PERFORMANCE LOG ARRAYS. THIS IS
 *     ONLY A STUB AND MUST BE EXPANDED BY THE USER. */
 * END DOMTUSER
```

```
MEMBER NAME  DOMTVARY
*      DOMTVARY
*         DOMTVARY MAIN SEGMENT
*         /*  THIS MODULE SUPPORTS THE VARYCONF AND VARYSCAN MACROS.
*         IF SCAN IS TO BE CHANGED THEN
*           IF MODE IS PRESENT THEN
*             IF MODE IS STANDARD THEN
*               SET NORMAL MODE INDICATOR
*             ELSE
*               IF MODE IS INITIAL THEN
*                 SET INITIAL MODE INDICATOR
*               ELSE
*                 IF MODE IS EMERGENCY THEN
*                   SET EMERGENCY MODE INDICATOR
*                 ELSE
*                   SET BAD MODE ERROR INDICATOR
*                 ENDIF
*               ENDIF
*             ENDIF
*           ELSE
*             IF CARD ADDRESS PRESENT THEN
*               CONVERT CARD ADDRESS TO EBCDIC
*             ENDIF
*             IF POINT ADDRESS PRESENT THEN
*               CONVERT POINT ADDRESS TO EBCDIC
*             ENDIF
*             IF ITEM COUNT PRESENT THEN
*               CONVERT ITEM COUNT TO EBCDIC
*             ENDIF
*             IF TERMINAL IS PRESENT
*               CONVERT TERMINAL TO EBCDIC
*             ENDIF
*             CONVERT SCAN ID TO EBCDIC
*           ENDIF
*         ELSE
*           IF SCAN IS TO BE DEACTIVATED
*             SEARCH FOR SCAN ID
*               ISSUE ERROR MESSAGE IF SCAN ID IS NOT FOUND
*           ELSE
*             IF SCAN IS TO ACTIVATED
*               SEARCH FOR SCAN ID
*                 ISSUE ERROR MESSAGE IF SCAN ID NOT FOUND
*             ENDIF
*           ENDIF
*           CONVERT SCAN ID TO EBCDIC
*           BUILD S/7 TRANSACTION
*         ELSE
*           SET UP MESSAGE FOR MODE CHANGE
*         ENDIF
*         IF SYS IS PRESENT
*           DO WRITETO7 SEGMENT
*         ELSE
*           LOCK TAS7COMM CONTROL BLOCK
*           UNTIL ALL S/7S PROCESSED
*             IF S/7 IS ACTIVE AND NODE IS UP
*               IF MODE IS TO BE CHANGED
*                 DO WRITETO7 SEGMENT
*                 IF WRITE SUCCESSFUL
*                   SET MODE CHANGE BIT
```

```
MEMBER NAME  DOMTVARY
*                   ENDIF
*                 ELSE
*                   DO WRITETO7 SEGMENT
*                 ENDIF
*               ENDIF
*             ENDDO
*             UNLOCK TAS7COMM CONTROL BLOCK
*           ENDIF
*         ELSE
*           GETWA FOR PARAMETER LIST
*           IF VARY TERMINAL REQUEST THEN
*             UNTIL ALL S/7S PROCESSED
*               UNTIL ALL RCB'S PROCESSED FOR THIS S/7
*                 IF PROPER S/7 AND REQUESTED TERMINAL THEN
*                   TURN ON FIRST PASS INDICATOR
*                   IF TERMINAL IS MANUAL AND NOT PCINT SUMMATION
*                     ISSUE ERROR MESSAGE
*                   ENDIF
*                   IF VARY TERMINAL ONLINE REQUEST THEN
*                     EXIT IF TERMINAL ALREADY IN REQUESTED STATE
*                   ELSE
*                     EXIT IF TERMINAL ALREADY IN REQUESTED STATE
*                   ENDIF
*                   PATCH DOMTVRPT TO VARY TERMINAL
*                   EXIT IF PATCH DID NOT WORK
*                 ELSE
*                   INCREMENT TO NEXT RCB TERMINAL ENTRY
*                 ENDIF
*               ENDDO
*               INCREMENT TO NEXT S/7 CONTROL TABLE ENTRY
*             ENDDO
*             EXIT IF RCB NOT FOUND
*           ELSE                      VARYING A POINT
*             GET THE POINT ADDRESS
*             EXIT IF POINT NOT FOUND
*             GET ADDRESSES OF THE PULSE COUNTER,ANALOG,AND STATUS ARRAYS
*             EXIT IF ONE OF THE ARRAYS NOT FOUND
*             LOCATE THE ARRAY IN WHICH POINT LIES
*             EXIT IF POINT DOES NOT LIE IN ANY OF THE ARRAYS
*             IF STATUS POINT THEN
*               IF OUT OF SERVICE REQUEST THEN
*                 EXIT IF POINT ALREADY IN REQUESTED STATE
*               ELSE
*                 IF POINT STATUS IS OUT OF SERVICE THEN
*                   EXIT IF POINT ALREADY IN REQUESTED STATE
*                 ELSE
*                   EXIT IF POINT ALREADY IN REQUESTED STATE
*                 ENDIF
*               ENDIF
*             ELSE
*               IF ANALOG POINT THEN
*                 IF OUT OF SERVICE REQUEST THEN
*                   EXIT IF POINT ALREADY IN REQUESTED STATE
*                 ELSE
*                   IF POINT STATUS IS OUT OF SERVICE THEN
*                     EXIT IF POINT ALREADY IN REQUESTED STATE
*                   ELSE
*                     EXIT IF POINT ALREADY IN REQUESTED STATE
```

```
MEMBER NAME  DOMTVARY
*                  ENDIF
*                ENDIF
*              ELSE
*                IF OUT OF SERVICE REQUEST THEN
*                  EXIT IF POINT ALREADY IN REQUESTED STATE
*                ELSE
*                  IF POINT STATUS IS OUT OF SERVICE THEN
*                    EXIT IF POINT ALREADY IN REQUESTED STATE
*                  ELSE
*                    EXIT IF POINT ALREADY IN REQUESTED STATE
*                  ENDIF
*                ENDIF
*              ENDIF
*            ENDIF
*            TURN OF FIRST PASS INDICATOR
*            UNTIL ALL S/7S PROCESSED
*              UNTIL ALL RCB'S PROCESSED FOR THIS S/7
*                IF ANALOG POINT THEN
*                  IF POINT ADDRESS WITHIN LAST RCB THEN
*                   TURN ON FIRST PASS INDICATOR
*                  ENDIF
*                ELSE
*                  IF PC POINT THEN
*                    IF POINT ADDRESS WITHIN LAST RCB THEN
*                      TURN ON FIRST PASS INDICATOR
*                    ENDIF
*                  ELSE
*                    IF POINT ADDRESS WITHIN LAST RCB THEN
*                      TURN ON FIRST PASS INDICATOR
*                    ENDIF
*                  ENDIF
*                ENDIF
*                INCREMENT TO NEXT RCB ENTRY
*              ENDDO
*              INCREMENT TO NEXT S/7 CONTROL TABLE ENTRY
*            ENDDO
*            EXIT IF POINT ADDRESS NOT FOUND
*            IF POINT OFFLINE
*              PICK UP PATCH ID FOR POINT OFFLINE
*            ELSE
*              PICK UP PATCH ID FOR POINT ONLINE
*            ENDIF
*            IF NOT POINT SUMMATION TERMINAL
*              EXIT IF MANUAL DATA
*              IF STATUS POINT
*                EXIT IF MANUAL POINT
*              ELSE
*                IF ANALOG POINT
*                  EXIT IF MANUAL POINT
*                ELSE
*                  IF PC POINT
*                    EXIT IF MANUAL POINT
*                  ENDIF
*                ENDIF
*              ENDIF
*              PATCH DOMTVRPT TO VARY POINT
*              ISSUE ERROR MESSAGE IF PATCH FAILED
*            ENDIF
```

```
MEMBER NAME  DOMTVARY
*           EXIT
*         END DOMTVARY
```

```
MEMBER NAME  DOMTVRPT
*        DOMTVRPT MAIN SEGMENT
*          EVENT THE COMMAND
*          IF PATCHID WAS FOR 3707 ONLINE,OR
*          IF PATCHID WAS FOR A 3707 OFFLINE,THEN
*            WRITE THE STATUS CHANGE TO THE SM ZONE OF THE DISPLAY
*          ENDIF
*          CASENTRY PATCHID
*            CASE 1 . VARY POINT OFFLINE
*              DO SEGMENT PIQCONV
*              IF POINT IS NOT OFFLINE SELF THEN
*                IF TERMINAL IS MANUAL AND SUMMATION
*                  SET NO WRITE TO S/7 INDICATOR
*                ELSE
*                  FIND POINT NUMBER
*                  SEND MESSAGE TO /7
*                  IF POINT IS ONLINE AND
*                  IF ALARM IS OUTSTANDING THEN
*                    DELETE ALARM
*                  ENDIF
*                  SET POINT OFFLINE SELF
*                ENDIF
*            CASE 2 _ VARY POINT ONLINE
*              DO SEGMENT PIQCONV
*              IF POINT IS OFFLINE SELF THEN
*                IF TERMINAL IS ONLINE (SELF AND OTHER) THEN
*                  FIND POINT NUMBER
*                  SEND MESSAGE TO THE /7
*                  SET THE POINT ONLINE
*                ELSE
*                  SET THE POINT OFFLINE OTHER
*                ENDIF
*              ENDIF
*            CASE 3 _ VARY TERMINAL OFFLINE
*              IF TERMINAL IS OFFLINE OTHER THEN
*                SET TERMINAL OFFLINE SELF
*              ELSE
*                IF TERMINAL IS  ONLINE SELF THEN
*                  IF WRITE TO S/7 INDICATOR OFF THEN
*                    DO SEGMENT WRTS7
*                    WRITE TRANSACTION 08 TO S/7
*                  ELSE
*                    SET TERMINAL OFFLINE SELF
*                    SET ALL ONLINE POINTS UNDER THE TERMINAL OFFLINE
*                     OTHER
*                  ENDIF
*                ENDIF
*              ENDIF
*            CASE 4 _ VARY TERMINAL ONLINE
*              IF TERMINAL IS OFFLINE OTHER THEN
*                IF WRITE TO S/7 INDICATOR OFF THEN
*                  DO SEGMENT WRTS7
*                  WRITE TRANSACTION 08
*                ELSE
*                  TURN OFF TERMINAL OFFLINE SELF INDICATOR
*                  SET. ALL OFFLINE OTHER POINTS UNDER THE TERMINAL
*                  ONLINE
*                ENDIF
*              ELSE
```

```
MEMBER NAME  DOMTVRPT
*                 TURN OFF TERMINAL OFFLINE SELF INDICATOR
*                 SET TERMINAL STATUS TO OFFLINE OTHER
*               ENDIF
*             CASE 5 _ VARY SYSTEM/7 OFFLINE
*               DO FOR EACH TERMINAL UNDER THIS S/7
*                 IF TERMINAL IS SUMMATION OR ACTIVE THEN
*                   SET THE TERMINAL OFFLINE OTHER
*                   SET ALL ONLINE POINTS TO OFFLINE OTHER
*                 ENDIF
*               ENDDO
*
*             CASE 6 _ VARY SYSTEM/7 ONLINE
*               DO FOR EACH TERMINAL UNDER THIS S/7
*                 IF TERMINAL IS SUMMATION OR ACTIVE THEN
*                   SET TERMINAL OFFLINE OTHER
*                   IF TERMINAL IS OFFLINE SELF THEN
*                     SET ALL OFFLINE OTHER POINTS UNDER THE TERMINAL
*                     TO ONLINE
*                   ENDIF
*                 ENDIF
*               ENDDO
*         ENDSEGMENT DOMTVRPT
*
*         WRTS7 SUBROUTINE SEGMENT(BUILD S/7 TRANSACTION)
*           IF VARY POINT REQUEST
*             INSERT TRANSACTION POINT NUMBER
*             IF VARY POINT OFFLINE
*               INSERT TRANSACTION OFFLINE INDICATOR
*             ELSE
*               INSERT TRANSACTION ONLINE INDICATOR
*             ENDIF
*           ELSE
*             IF VARY TERMINAL
*               INSERT TRANSACTION TEXT FOR ONLINE
*               IF VARY TERMINAL OFFLINE
*                 INSERT TRANSACTION OFFLINE INDICATOR
*               ENDIF
*               CONVERT TEXT TO ASCII
*             ENDIF
*           ENDIF
*           INSERT TRANSACTION ORIGIN AND DESTINATION IDS
*           IF TERMINAL REQUEST
*             REVERSE TRANSACTION ORIGIN AND DESTINATION IDS
*           ENDIF
*           ENDSEGMENT WRTS7
*
*           PIQCONV SUBROUTINE SEGMENT(SET ALARM,MANUAL,& IN/OUT OF
*                                      SERVICE INDICATORS)
*           IF ANALOG POINT
*             IF ALARM OUTSTANDING
*               SET ALARM INDICATOR
*             ENDIF
*             IF MANUAL POINT
*               SET MANUAL POINT INDICATOR
*             ENDIF
*             IF POINT OUT OF SERVICE-OTHER
*               SET OUT OF SERVICE-OTHER INDICATOR
*             ELSE
```

```
MEMBER NAME  DOMTVRPT
*                 IF POINT OUT OF SERVICE-SELF
*                   SET OUT OF SERVICE-SELF INDICATOR
*                 ELSE
*                   SET INSERVICE INDICATOR
*                 ENDIF
*               ENDIF
*             ELSE
*               IF STATUS PT
*                 IF ALARM OUTSTANDING
*                   SET ALARM INDICATOR
*                 ENDIF
*                 IF MANUAL POINT
*                   SET MANUAL POINT INDICATOR
*                 ENDIF
*                 IF POINT OUT OF SERVICE-OTHER
*                   SET OUT OF SERVICE-OTHER INDICATOR
*                 ELSE
*                   IF POINT OUT OF SERVICE-SELF
*                     SET OUT OF SERVICE-SELF INDICATOR
*                   ELSE
*                     SET INSERVICE INDICATOR
*                   ENDIF
*                 ENDIF
*               ELSE
*                 IF PULSE COUNTER POINT
*                   IF ALARM OUTSTANDING
*                     SET ALARM INDICATOR
*                   ENDIF
*                   IF MANUAL POINT
*                     SET MANUAL POINT INDICATOR
*                   ENDIF
*                   IF POINT OUT OF SERVICE-OTHER
*                     SET OUT OF SERVICE-OTHER INDICATOR
*                   ELSE
*                     IF POINT OUT OF SERVICE-SELF
*                       SET OUT OF SERVICE-SELF INDICATOR
*                     ELSE
*                       SET INSERVICE INDICATOR
*                     ENDIF
*                   ENDIF
*                 ENDIF
*               ENDIF
*          ENDSEGMENT PIQCONV
```

```
MEMBER NAME  DOMTVS7
*   DOMTVS7 MAIN SEGMENT
*
*   IF PRIMARY REQUEST, THEN
*     SET PRIMARY CODE
*   ELSE
*     IF BACKUP REQUEST, THEN
*       SET BACKUP CODE
*     ELSE
*       IF OFFLINE REQUEST, THEN
*         SET OFFLINE FLAG
*       ELSE
*         SET RETURN CODE INVALID PARAMETER
*         BRANCH TO EXIT PROGRAM CODE
*       ENDIF
*     ENDIF
*   ENDIF
*   SEARCH FOR UNIT REQUESTED
*     EXITIF REQUESTED UNIT FOUND
*       ORELSE
*        INCREMENT TO NEXT UNIT
*     ENDLOOP
*       SET INVALID PARAMETER RETURN CODE
*       BRANCH TO EXIT PROGRAM CODE
*   ENDSRCH
*   SEARCH FOR LID MATCH
*     EXITIF LID MATCH FOUND
*       ORELSE
*       INCREMENT TO NEXT LID
*     ENDLOOP
*       SET INVALID PARAMETER RETURN CODE
*       BRANCH TO EXIT PROGRAM CODE
*   ENDSRCH
*   IF UNIT NOT ASSIGNED THIS LID, THEN
*     SET INVALID PARAMETER RETURN CODE
*     BRANCH TO EXIT PROGRAM CODE
*   ENDIF
*   IF S/7 NOT CONTROLLABLE, THEN
*     SET NOT CONTROLLABLE RETURN CODE
*     BRANCH TO EXIT PROGRAM CODE
*   ENDIF
*   LOCK VARYS7 CONTROL BLOCK
*   IF PRIMARY REQUEST, THEN
*     BRANCH TO PRIMARY PROCESSOR
*     IF BACKUP REQUEST, THEN
*       BRANCH TO BACKUP PROCESSOR
*       IF OFFLINE REQUEST, THEN
*         BRANCH TO OFFLINE PROCESSOR
*       ELSE
*         SET INVALID PARAMETER RETURN CODE
*         BRANCH TO EXIT PROGRAM CODE
*       ENDIF
*     ENDIF
*   ENDIF
*
*   /*  VARY S/7 PRIMARY  */
*   PRIMARY SEGMENT
*     IF LID HAS NO CURRENT PRIMARY, THEN
*       IF S/7 IPLED THIS LID, THEN
```

```
MEMBER NAME  DOMTVS7
*          UPDATE TAS7COMM NEW-PRIME
*          SWITCH TO SCAN BUFFERS
*          WRITE TRANSACTION TO S/7 TO ASSUME PRIMARY
*        ELSE
*          IF S/7 IPLED SOME OTHER LID, THEN
*            IF S/7 IPLED AS BACKUP, THEN
*              UPDATE TAS7COMM S/7 INACTIVE
*              UPDATE TAS7COMM NEW-PRIME
*              PATCH TASK TO IPL S/7
*            ELSE
*              SET RETURN CODE LID NOT CONTROLLABLE
*            ENDIF
*          ELSE
*            UPDATE TAS7COMM NEW-PRIME
*            PATCH TASK TO IPL S/7
*          ENDIF
*        ENDIF
*      ELSE
*        IF S/7 IS CURRENT PRIMARY
*          SET RETURN CODE S/7 ALREADY IN REQUESTED STATE
*        ELSE
*          SET RETURN CODE S/7 NOT CONTROLLABLE
*        ENDIF
*      ENDIF
*    END PRIMARY SEGMENT
*
*    /*  VARY S/7 BACKUP  */
*    BACKUP SEGMENT
*      IF NO CURRENT BACKUP THIS LID, THEN
*        IF UNIT IPLED THIS LID, THEN
*          SET RETURN CODE NOT CONTROLLABLE
*        ELSE
*          IF UNIT IPLED OTHER LID, THEN
*            IF UNIT IPLED AS BACKUP, THEN
*              UPDATE TAS7COMM S/7 INACTIVE
*              UPDATE TAS7COMM S/7 NEW-BACKUP
*              PATCH TASK TO IPL S/7
*            ELSE
*              SET RETURN CODE LID NOT CONTROLLABLE
*            ENDIF
*          ELSE
*            UPDATE TAS7COMM S/7 NEW-BACKUP
*            PATCH TASK TO IPL S/7
*          ENDIF
*        ENDIF
*      ELSE
*        SET RETURN CODE NOT CONTROLLABLE
*      ENDIF
*    END BACKUP SEGMENT
*
*    /*  VARY A S/7 OFFLINE  */
*    OFFLINE SEGMENT
*      IF S/7 IPLED THIS LID, THEN
*        WRITE OFFLINE COMMAND TO S/7
*      ELSE
*        IF S/7 IPLED OTHER LID, THEN
*          SET RETURN CODE NOT CONTROLLABLE
*        ELSE
```

```
MEMBER NAME  DOMTVS7
*          SET RETURN CODE ALREADY IN STATE
*        ENDIF
*      ENDIF
*    END OFFLINE SEGMENT
*
*    /*  EXIT PROGRAM CODE  */
*      UNLOCK VARYS7 CONTROL BLOCK
*      EXIT
*
*    END     DOMTVS7
```

```
MEMBER NAME  DOMTWRIT
 *  CSECT   DOMTWRIT                 /*  REQUEST OUTPUT TO S/7   */
 *
 *     IF TYPE OF REQUEST IS VALID, THEN
 *       IF UNIT NOT SPECIFIED, THEN
 *         STRTSRCH  REMOTE LID SECTION OF TAS7COMM ARRAY
 *         EXITIF REQUESTED LID IS A REMOTE
 *           SAVE LOCAL LID IN PATH
 *         ORELSE
 *           NEXT REMOTE
 *         ENDLOOP
 *         ENDSRCH
 *         STRTSRCH  LOCAL LID IN TAS7COMM
 *         EXITIF  LOCAL LID FOUND TO ROUTE MESSAGE
 *         ORELSE
 *           CHECK NEXT LID
 *         ENDLOOP
 *           SET INVALID ID RETURN CODE
 *           EXIT PGM
 *         ENDSRCH
 *         STORE UNIT (S/7) MESSAGE ROUTED TO
 *         IF LOCAL LID IS INACTIVE, THEN
 *           SET INACTIVE RETURN CODE
 *           EXIT PGM
 *         ENDIF
 *       ENDIF
 *       IF IPL REQUEST, THEN
 *         LOCATE IPL REQUEST ECB THIS UNIT
 *       ELSE
 *         IF STANDARD TRANSACTION, THEN
 *           IF UNIT NOT SPECIFIED, THEN
 *             IF INITIALIZATION NOT COMPLETE, THEN
 *               SET INACTIVE RETURN CODE
 *               EXIT PROGRAM
 *             ENDIF
 *           ENDIF
 *           CALCULATE LENGTH OF TRANSACTION TO THE HALFWORD
 *           IF USER EXIT ROUTINE SPECIFIED, THEN
 *             CALL USER EXIT ROUTINE
 *             IF NON-ZERO RETURN CODE, THEN
 *               SET USER ERROR RETURN CODE
 *               EXIT PROGRAM
 *             ENDIF
 *           ENDIF
 *           CONVERT LENGTH IN HEADER FROM TEXT LENGTH IN BYTES TO
 *            TRANSACTION LENGTH IN HALFWORDS(S/7 WORDS)
 *         ENDIF
 *         LOCATE WRITE REQUEST ECB THIS UNIT
 *       ENDIF
 *       LOCK REQUEST RESOURCE FOR THIS UNIT
 *       POST REQUEST ECB
 *       WAIT FOR I/O COMPLETION
 *       UNLOCK RESOURCE
 *       IF STANDART TRANSACTION, THEN
 *         RESTORE TRANSACTION LENGTH TO ENTRY VALUE
 *       ENDIF
 *       IF I/O UNSUCCESSFUL, THEN
 *         SET I/O ERROR RETURN CODE
 *       ELSE
```

```
MEMBER NAME  DOMTWRIT
 *        SET ZERO RETURN CODE
 *      ENDIF
 *    ELSE
 *      SET INVALID TYPE RETURN CODE
 *    ENDIF
 *    EXIT PROGRAM
 *
 *  END     DOMTWRIT
```

```
MEMBER NAME  DOMTXEND
 *  CSECT   DOMTXEND             ABNORMAL END APPENDAGE
 *    IF  ENTRY DUE TO INTERVENTION REQUIRED, THEN
 *      SET UCB NOT READY FLAG ON
 *      RETURN - RETRY THE OPERATION
 *    ELSE
 *      RETURN - NORMAL
 *    ENDIF
 *  END     DOMTXEND
```

```
MEMBER NAME  DOMUCONV
*      DOMUCONV MAIN SEGMENT
*         **************************************************************
*         **  THIS ROUTINE IS A STUB TO BE REPLACED BY A USER WRITTEN *
*         *  ROUTINE TO DO ANALOG DATA CONVERSION IF DESIRED BY THE   *
*         *  USER.IT RECEIVES CONTROL FROM THE ANALOG DATA CONVERSION *
*         *  ROUTINE DOMTCACS IF THE USER CONVERSION BIT IS ON IN THE *
*         ** ANALOG DATA BASE ITEM.                                   *
*         **************************************************************
*      END OF SEGMENT DOMUCONV
```

```
MEMBER NAME  DOMUSERC
 *        DOMUSERC MAIN SEGMENT
 *          ************************************************************
 *          **  THIS ROUTINE IS A STUB TO BE REPLACED BY A USER      **
 *          *   WRITTEN ROUTINE.  IT RECEIVES CONTROL VIA PATCH       *
 *          *   FROM SCAN PROCESSING AT THE END OF EACH SCAN CYCLE.   *
 *          **  IT IS PASSED THE CURRENT SCAN ID AS THE PATCH ID.    **
 *          ************************************************************
 *
 *          RETURN WITH A ZERO RETURN CODE
 *        ENDSEGMENT DOMUSERC
```

```
MEMBER NAME  DOMUSERI
 *        DOMUSERI MAIN SEGMENT
 *          ************************************************************
 *          **  THIS ROUTINE IS A STUB TO BE REPLACED BY A USER       **
 *          *   WRITTEN ROUTINE.  IT RECEIVES CONTROL VIA PATCH        *
 *          *   AFTER ALL ACTIVE S/7'S HAVE REPORTED IN WITH THEIR     *
 *          **  INITIAL SCAN.                                         **
 *          ************************************************************
 *
 *          RETURN WITH A ZERO RETURN CODE
 *        ENDSEGMENT DOMUSERI
```

```
MEMBER NAME  DOMCWBS7
 * BEGIN DOMCWBS7
 * /* IO INTERFACE PROGRAM BETWEEN S/370 AND S/7 FOR WALLBOARDS*/
 * IF PATCH ID=2 THEN
 *   IF COMMAND LIST POINTER IN EMSCVT IS ZERO THEN
 *     STORE ADDRESS OF PASSED COMMAND LIST IN EMSCVT
 *     IF EMSCVT INITIALIZATION FLAG IS OFF AND
 *     IF COMMAND LIST INITIALIZATION FLAG IS OFF THEN
 *       DO LIST  /*GENERATE FINAL COMMAND LIST TO BE SENT TO S/7*/
 *     ELSE
 *       IF COMMAND LIST INITIALIZATION FLAG IS ON THEN
 *         TURN INITIALIZATION FLAG ON IN EMSCVT
 *         DO LIST  /*GENERATE FINAL COMMAND LIST TO BE SENT TO S/7*/
 *       ENDIF
 *     ENDIF
 *   ELSE
 *     PLACE ADDRESS OF COMMAND LIST AT END OF CHAIN POINTED TO BY
 *          ECVTWBCL FIELD IN EMSCVT
 *   ENDIF
 * ELSE
 *    IF PATCH ID = 4 THEN
 *     IF EMSCVT COMMAND LIST POINTER EQUALS ZERO THEN
 *       PRINT END OF INITIALIZATION MESSAGE AND ISSUE AN EVENT
 *       TURN OFF INITIALIZATION FLAG IN EMSCVT
 *     ELSE
 *       IF INITIALIZATION FLAG IN COMMAND LIST IS ON THEN
 *         DO LIST /*GENERATE FINAL COMMAND LIST TO BE SENT TO S/7*/
 *       ELSE
 *         PRINT END OF INITIALIZATION MESSAGE AND ISSUE AN EVENT
 *         TURN OFF INITIALIZATION FLAG IN EMSCVT
 *         UNTIL COMMAND LIST POINTER IN EMSCVT IS ZERO OR
 *         UNTIL INITIALIZATION FLAG IN EMSCVT IS ON DO
 *           IF COMMAND LIST INITIALIZATION FLAG IS ON THEN
 *             TURN ON INITIALIZATION FLAG IN EMSCVT
 *           ENDIF
 *           DO LIST /*GENERATE FINAL COMMAND LIST TO BE SENT TO S/7*/
 *         ENDDO
 *       ENDIF
 *     ENDIF
 *   ENDIF
 * ENDIF
 * EXIT DOMCWBS7 MODULE
 *
 *
 * START LIST SEGMENT
 * PLACE S/7 DESTINATION ID IN HEADER
 * PLACE ORIGIN S/370 ID IN HEADER
 * PLACE TRANSACTION CODE 14 AND FLAGS IN HEADER
 * IF S/7 WALLBOARD CONTROLLER IS ACTIVE AND SCANNING THEN
 *   ISSUE S7WRITE MACRO TO SEND TRANSACTION CODE 14 TO S/7
 * ENDIF
 * IF NEXT COMMAND LIST POINTER IS ZERO THEN
 *   PLACE ZEROS IN COMMAND LIST POINTER IN EMSCVT
 * ELSE
 *   PLACE ADDRESS OF NEXT COMMAND LIST IN COMMAND LIST POINTER FIELD
 *        IN EMSCVT
 * ENDIF
 * FREE COMMAND LIST BUFFER JUST SENT TO S/7
 *
```

```
MEMBER NAME  DOMCWBS7
 *
 * END DOMCWBS7 MODULE
```

```
MEMBER NAME  DOMCWBPR
 * BEGIN DOMCWBPR
 * /* THIS IS THE WALLBOARD CONTROL PROGRAM */
 * IF WALLBOARD S/7 CONTROLLER IS ACTIVE AND SCANNING THEN
 *   IF MANUAL/AUTO SWITCH IS DEFINED THEN
 *     IF WALLBOARD IS IN AUTOMATIC MODE THEN
 *       IF AUTOMATIC MODE FLAG IS OFF THEN/*ALWAYS ON INITIALLY*/
 *         TURN ON AUTOMATIC MODE FLAG
 *         TURN ON INITIALIZATION FLAG
 *         DO EVNT1  /*MESSAGE AND EVENT THAT WALLBOAR IS IN AUTOMATIC
 *                     MODE*/
 *       ENDIF
 *     ELSE
 *       IF AUTOMATIC MODE FLAG IS ON THEN
 *         TURN OFF AUTOMATIC MODE FLAG
 *         DO EVNT1  /*MESSAGE AND EVENT THAT WALLBOARD IS IN MANUAL
 *                     MODE*/
 *       ENDIF
 *     ENDIF
 *   ENDIF
 *   IF WALLBOAR AUTOMATIC MODE FLAG IS ON THEN
 *     IF INITIALIZATION FLAG IS ON OR
 *     IF PATCH ID EQUALS 4 THEN
 *       TURN ON COMMAND LIST INITIALIZATION FLAG
 *       DO FOR EACH ENTRY IN CAWBNAME ARRAY
 *         BRANCH TO CONFIGURATION ROUTINE BASED ON CONFIGURATION
 *             NUMBER FOR EACH ENTRY
 *             /* CASE1,CASE2,----CASE10*/
 *         BUMP TO NEXT ENTRY
 *       ENDDO
 *       TURN OFF COMMAND LIST INITIALIZATION FLAG
 *       TURN ON LAST ITEM PROCESSED FLAG
 *     ELSE
 *       IF PATCH ID EQUALS EIGHT THEN
 *         DO FOR EACH ITEM ADDRESS PASSED IN PATCH PARM LIST
 *           SEARCH CAWBNAME ARRAY FOR MATCH
 *           BRANCH TO CONFIGURATION ROUTINE BASED ON CONFIGURATION
 *               NUMBER FOR EACH ENTRY
 *               /*CASE1,CASE2,....CASE10*/
 *           BUMP TO NEXT ENTRY
 *         ENDDO
 *         TURN LAST ITEM PROCESSED FLAG ON
 *       ENDIF
 *     ENDIF
 *     IF LAST ITEM PROCESSED FLAG IS ON THEN
 *       DO CLISTPR  /*COMMAND LIST PROCESSING*/
 *     ENDIF
 *   ENDIF
 * ENDIF
 * EXIT
 * END DOMCWBPR
 *
 *
 * START CASE1 SEGMENT    /*STATUS*/
 * IF STATUS ITEM HAS NOMINAL VALUE THEN
 *   LAMP1 = STATE1
 * ELSE
 *   LAMP1 = STATE2
 * ENDIF
```

```
MEMBER NAME  DOMCWBPR
 * DO CLISTPR    /*COMMAND LIST PROCESSING*/
 * END CASE1 SEGMENT
 *
 *
 * START CASE2 SEGMENT    /*STATUS*/
 * IF ALARM BIT FOR ITEM IS OFF THEN
 *   LAMP1 = STATE1
 * ELSE
 *   IF ALARM IS ACKNOWLEDGED
 *     LAMP1 = STATE2
 *   ELSE       /*UNACKNOWLEDGED*/
 *     LAMP1 = STATE3
 *   ENDIF
 * ENDIF
 * DO CLISTPR    /*COMMAND LIST PROCESSING*/
 * END CASE2 SEGMENT
 *
 *
 * START CASE3 SEGMENT    /*STATUS*/
 * IF VALUE IS NOMINAL THEN
 *   LAMP1 = STATE2
 *   LAMP2 = STATE1
 * ELSE
 *   LAMP1 = STATE1
 *   LAMP2 = STATE2
 * ENDIF
 * DO CLISTPR    /*COMMAND LIST PROCESSING*/
 * END CASE3 SEGMENT
 *
 *
 * START CASE4 SEGMENT    /*STATUS*/
 * IF NO ALARM EXISTS THEN
 *   LAMP1 = STATE2
 *   LAMP2 = STATE1
 * ELSE
 *   IF ALARM IS ACKNOWLEDGED
 *     LAMP1 = STATE1
 *     LAMP2 = STATE2
 *   ELSE   /* UNACKNOWLEDGED */
 *     LAMP1 = STATE1
 *     LAMP2 = STATE3
 *   ENDIF
 * ENDIF
 * DO CLISTPR  /*COMMAND LIST PROCESSING*/
 * END CASE4 PROCESSING
 *
 *
 * START CASE5 PROCESSING  /*STATUS*/
 * IF NO ALARM EXISTS THEN
 *   LAMP1 = STATE1
 *   LAMP2 = STATE1
 *   LAMP3 = STATE2
 * ELSE
 *   IF ALARM IS ACKNOWLEDGED THEN
 *     LAMP1 = STATE2
 *     LAMP2 = STATE1
 *     LAMP3 = STATE1
 *   ELSE  /*UNACKNOWLEDGED*/
```

```
MEMBER NAME  DOMCWBPR
*      LAMP1 = STATE1
*      LAMP2 = STATE2
*      LAMP3 = STATE3
*    ENDIF
*  ENDIF
*  DO CLISTPR    /*COMMAND LIST PROCESSING
*  END CASE5 SEGMENT
*
*
*  START CASE6 SEGMENT    /*ANALOG*/
*  IF NO ALARM EXISTS THEN
*    LAMP1 = STATE1
*  ELSE
*    IF ALARM IS ACKNOWLEDGED THEN
*      LAMP1 = STATE2
*    ELSE   /*UNACKNOWLEDGED*/
*      LAMP1 = STATE3
*    ENDIF
*  ENDIF
*  DO CLISTPR    /*COMMAND LIST PROCESSING*/
*  END CASE6 SEGMENT
*
*
*  START CASE7 SEGMENT    /*ANALOG*/
*  IF NOT OUT OF LIMITS HIGH THEN
*    LAMP1 = STATE1
*  ELSE
*    IF OUT OF HIGH WARNING LIMIT THEN
*      LAMP1 = STATE2
*    ELSE
*      IF OUT OF HIGH OPERATING LIMIT THEN
*        LAMP1 = STATE 3
*      ELSE
*        IF OFFSCALE HIGH OR LOW THEN
*          LAMP1 = STATE 4
*        ENDIF
*      ENDIF
*    ENDIF
*  ENDIF
*  DO CLISTPR SEGMENT     /*COMMAND LIST PROCESSING*/
*  END CASE7 SEGMENT
*
*
*  START CASE8 SEGMENT    /*ANALOG*/
*  IF OUT OF LIMITS WARNING OR
*  IF OUT OF LIMITS OPERATING THEN
*    IF OUT OF LIMITS LOW THEN
*      IF OUT OF LIMITS WARNING THEN
*        LAMP1 = STATE 2
*      ELSE
*        LAMP1 = STATE 3
*      ENDIF
*    ELSE
*      LAMP1 = STATE 1
*    ENDIF
*  ELSE
*    IF OFFSCALE HIGH OR LOW THEN
*      LAMP1 = STATE 4
```

```
MEMBER NAME  DOMCWBPR
*   ELSE
*     LAMP1 = STATE 1
*   ENDIF
* ENDIF
* DO CLISTPR   /*COMMAND LIST PROCESSING
* END CASE8 SEGMENT
*
*
* START CASE9 SEGMENT
* IF OUT OF LIMITS WARNING OR
* IF OUT OF LIMITS OPERATING OR
* IF OFFSCALE HIGH OR LOW THEN
*   IF OUT OF LIMITS WARNING THEN
*     IF OUT OF WARNING HIGH THEN
*       LAMP1 = STATE 1
*       LAMP2 = STATE 2
*     ELSE          ..............LOW WARNING
*       LAMP1 = STATE2
*       LAMP2 = STATE1
*     ENDIF
*   ELSE
*     IF OUT OF OPERATING LIMITS THEN
*       IF OUT HIGH THEN
*         LAMP1 = STATE 1
*         LAMP2 = STATE 3
*       ELSE      .............OUT LOW OPERATING
*         LAMP1 = STATE 3
*         LAMP2 = STATE 1
*       ENDIF
*     ELSE        .............OFFSCALE HIGH OR LOW
*       LAMP1 = STATE 3
*       LAMP2 = STATE 3
*     ENDIF
*   ENDIF
* ELSE              .............NOMINAL CONDITION
*   LAMP1 = STATE 1
*   LAMP2 = STATE 1
* ENDIF
* DO CLISTPR  /*COMMAND LIST PROCESSING*/
* END CASE9 SEGMENT
*
*
* START CASE 10 SEGMENT  /*ANALOG */
* IF NOT OUT OF LIMITS THEN
*   LAMP1 = STATE1
*   LAMP2 = STATE1
*   LAMP3 = STATE1
* ELSE
*   IF OUT OF LIMITS WARNING OR
*   IF OUT OF LIMITS OPERATING OR
*   IF OFFSCALE THEN
*     IF OUT OF WARNING LIMIT THEN
*       IF OUT HIGH THEN
*         IF ALARM ACKNOWLEDGED THEN
*           LAMP1 = STATE 1
*           LAMP2 = STATE 2
*           LAMP3 = STATE 2
*         ELSE        ............OUT HIGH UNACKNOWLEDGED
```



```
MEMBER NAME  DOMCWBPR
*            LAMP1 = STATE 1
*            LAMP2 = STATE 3
*            LAMP3 = STATE 2
*          ENDIF
*        ELSE          ............OUT LOW  WARNING
*          IF ALARM ACKNOWLEDGED THEN
*            LAMP1 = STATE 2
*            LAMP2 = STATE 2
*            LAMP3 = STATE 1
*          ELSE        ............OUT LOW UNACKNOWLEDGED
*            LAMP1 = STATE 2
*            LAMP2 = STATE 3
*            LAMP3 = STATE 1
*          ENDIF
*        ENDIF
*      ELSE
*        IF OUT OF OPERATING LIMITS THEN
*          IF OUT HIGH THEN
*            IF ALARM ACKNOWLEDGED THEN
*              LAMP1 = STATE 1
*              LAMP2 = STATE 2
*              LAMP3 = STATE 3
*            ELSE      ............OUT HIGH UNACKNOWLEDGED
*              LAMP1 = STATE 1
*              LAMP2 = STATE 3
*              LAMP3 = STATE 3
*            ENDIF
*          ELSE        ............OUT LOW OPERATING LIMITS
*            IF ALARM ACKNOWLEDGED THEN
*              LAMP1 = STATE 3
*              LAMP2 = STATE 2
*              LAMP3 = STATE 1
*            ELSE      ............OUT LOW OPERATING UNACKNOWLEDGED
*              LAMP1 = STATE 3
*              LAMP2 = STATE 3
*              LAMP3 = STATE 1
*            ENDIF
*          ENDIF
*        ELSE          ............OFFSCALE HIGH OR LOW
*          LAMP1 = STATE 3
*          LAMP2 = STATE 3
*          LAMP3 = STATE 3
*        ENDIF
*      ENDIF
*    ENDIF
* ENDIF
* DO CLISTPR /*COMMAND LIST PROCESSING*/
* END CASE10 SEGMENT
*
*
* START CLISTPR SEGMENT  /*PROCESS COMMAND LIST*/
* IF COMMAND LIST BUFFER RETRIEVED FLAG IS NOT ON THEN
*   DO GETBUFSG SEGMENT  /*RETRIEVE AREA FOR COMMAND LIST BUFFER*/
* ENDIF
* IF LAST ENTRY PROCESSED FLAG IS ON THEN
*   DO PATCHSEG SEGMENT  /*SHIP COMMAND LIST TO DOMCWBS7*/
*   TURN LAST ENTRY PROCESSED FLAG OFF
* ELSE
```

```
MEMBER NAME  DOMCWBPR
*    IF COMMAND LIST BUFFER LENGTH (USED) EQ OR IS GT 256 THEN
*      DO PATCHSEG SEGMENT /*SHIP COMMAND LIST TO DOMCWBS7*/
*      DO GETBUFSG SEGMENT /*RETRIEVE AREA FOR COMMAND LIST BUFFER*/
*    ENDIF
*    PLACE COMMAND(S) IN BUFFER    /*1-3 COMMAND DEPENDING ON CASE*/
*    ADJUST LENGTH FIELD ACCORDINGLY /*COMMAND LIST LENGTH CHECKED
*                                     FOLLOWING EACH COMMAND PLACED
*                                     IN BUFFER*/
* ENDIF
* END CLISTPR SEGMENT
*
*
* START GETBUFSG SEGMENT
* GET BUFFER AREA
* ERREXIT IF BUFFER UNAVAILABLE
* ZERO AREA
* TURN ON COMMAND LIST RETRIEVED FLAG
* IF INITIAL FLAG IS ON THEN
*    SET INITIAL FLAG BYTE INDICATOR ON IN BUFFER TEXT
* ENDIF
* PLACE LENGTH OF FLAG BYTES IN LENGTH FIELDS
* END GETBUFSG SEGMENT
*
*
* START PATCHSEG SEGMENT /*SEND COMPLETED COMMAND LIST TO DOMCWBS7*/
* PLACE END FLAG IN BUFFER TEXT
* INCREASE BUFFER TEXT LENGTH BY END FLAG LENGTH AND REPLACE IN
*      LENGTH FIELDS
* PATCH DOMCWBS7 ID=2,TASK=DOM+UTIL
* IF PATCH FAILS THEN
*    FREE BUFFER AREA
* ENDIF
* END PATCHSEG SEGMENT
*
*
* END DOMCWBPR MODULE
```

```
MEMBER NAME  DOMTABLE
 *   THIS CSECT IS A TABLE OF ADDRESS CONSTANTS THAN POINT TO MACRO
 *   PROCESSING ROUTINES.  THESE MACRO PROCESSING ROUTINES ARE LINK-
 *   EDITED WITH THIS CSECT TO FORM ONE LOAD MODULE, EACH OF WHICH
 *   IS DOCUMENTED INDIVIDUALLY.
 *
 *   CSECT    DOMTABLE                  MACRO PROCESSOR ADDRESS TABLE
 *         DC     V(DOMTWRIT)           ADDRESS S7WRITE  PROCESSOR
 *         DC     V(DOMTSCAN)           ADDRESS VARYSCAN PROCESSOR
 *         DC     V(DOMTVS7)            ADDRESS VARYS7   PROCESSOR
 *         DC     V(DOMCLRMT)           ADDRESS DOMCLGET/DOMCLPUT PROCESSOR
 *         DC     V(DOMCEVNT)           ADDRESS SCEVENT  PROCESSOR
 *         DC     V(DOMCALM2)           ADDRESS DOMCALRM PROCESSOR
 *         DC     V(DOMCDC10)           ADDRESS SCDEVICE PROCESSOR
 *         DC     V(DOMCLFRE)           ADDRESS DOMCFREE PROCESSOR
 *         DC     V(DOMTCODE)           ADDRESS ASCICONV PROCESSOR
 *         DC     V(DOMCNCNV)           ADDRESS PGM TO CONVERT
 *         DC                           EBCDIC TO FLOATING OR FIXED
 *         DC     V(DOMTUPDT)           ADDRESS UPD7COMM PROCESSOR
 *         DC     V(DOMTCHRT)           ADDRESS STRPCHRT PROCESSOR
 *   END      DOMTABLE
```

```
MEMBER NAME  DOMTPDSG
 * CSECT     DOMTPDSG              /* READ LOGICAL RECORD FROM PDS */
 *
 *    LOAD DISPLACEMENT INTO BUFFER OF LAST LOGICAL RECORD FROM
 *      DECB EXTENSION
 *    ADD LENGTH OF LOGICAL RECORD
 *    IF NEW PHYSICAL RECORD IS REQUIRED, THEN
 *      SET END OF DATA EXIT - EODAD
 *      SET SYNAD ERROR EXIT - SYNAD
 *      READ NEXT PHYSICAL RECORD
 *      CALCULATE ACTUAL SIZE OF RECORD READ
 *      SET STORE LENGTH OF ACTUAL RECORD IN DECB EXTENSION
 *      SET DISPLACEMENT TO LOGICAL RECORD TO ZERO
 *    ENDIF
 *    STORE DISPLACEMENT INTO DECB EXTENSION
 *    LOAD ADDRESS OF LOGICAL RECORD IN INPUT BUFFER
 *    RETURN TO CALLER
 *
 *    EODAD SEGMENT
 *      SET RETURN ADDRESS TO ZERO - INDICATES END OF DATA
 *    END SEGMENT EODAD
 *
 *    SYNAD SEGMENT
 *      SET RETURN ADDRESS TO NEGATIVE ONE - INDICATES I/O ERROR
 *    END SEGMENT SYNAD
 *
 *   END      DOMTPDSG
```

```
MEMBER NAME  DOMTROUT
 *  THIS LOAD MODULE IS A TABLE OF ALL S/7 TO S/370 TRANSACTION CODES,
 *  PATCH ID AND PATCH SUPL LIST FOR FOR USE IN ROUTING THE TRANSACTION
 *  THE LENGTH OF THE TABLE AND NUMBER OF ENTRIES ARE IN THE FIRST TWO
 *  HALFWORDS. THE FORMAT OF EACH ENTRY IS AS FOLLOWS :
 *        TRANSACTION CODE - 1 BYTE
 *        PATCH ID - 1 BYTE
 *        PATCH  SUPL=L   WITH NECESSARY PARAMETERS
 *  THERE IS ONE ENTRY PER TRANSACTION WITH THE FOLLOWING EXCEPTIONS:
 *    TRANSACTION    EXCEPTION
 *        X'8F'        NOT ROUTED
 *        X'88'        THERE IS MORE THAN ONE ENTRY. THEY ARE ORDERED
 *                       AS FOLLOWS -
 *                         POWER DEVICE CONTROL
 *                         SCAN COMMAND
 *                         STRIPCHART REPLY
```

```
MEMBER NAME  DOMTRS7V
 * DOMTRS7V MAIN SEGMENT
 *    POINT TO HIERARCHY TABLE (TAS7HIER) AND EMSCVT
 *    ERROR EXIT TO ERR1 IF 370 IS NOT TOP OF HIERARCHY
 *    ERROR EXIT TO ERR8 IF ACCESS AREA AND FUNCTION DO NOT MATCH
 *     THOSE OF MASTER DISPATCHER
 *    POINT TO DATA READ FROM SCREEN
 *    ERROR EXIT TO ERR2 IF NO ID ENTERED
 *    ERROR EXIT TO ERR3 IF NO OPTION CHOSEN
 *    ERROR EXIT TO ERR4 IF BOTH OPTIONS CHOSEN
 *    ERROR EXIT TO ERR5 IF ID ENTERED IS NOT NUMERIC
 *    CONVERT ID TO BINARY
 *    START SEARCH ON HIERARCHY TABLE FOR NUMBER OF LOCAL IDS
 *    EXIT IF MATCH FOUND ON LOCAL ID
 *      ERROR EXIT TO ERR6 SINCE ID IS NOT REMOTE
 *    OR ELSE
 *      POINT TO NEXT ID IN TABLE
 *      ADJUST CONTROL COUNT
 *    END LOOP
 *    END SEARCH
 *    START SEARCH ON HIERARCHY TABLE FOR REMAINING ENTRIES
 *    EXIT IF MATCH FOUND ON ID
 *    OR ELSE
 *      POINT TO NEXT ID IN TABLE
 *    END LOOP
 *      ERROR EXIT TO ERR5 SINCE ID IS NOT VALID
 *    END SEARCH
 *    BUILD TRANSACTION CODE X'08' VARY MESSAGE
 *    POINT TO PARENT ENTRY FOR DESTINATION ID
 *    CONVERT MESSAGE TEXT TO ASCII USING ASCICONV MACRO
 *    WRITE MESSAGE TO SYSTEM/7 USING S7WRITE MACRO
 *    ERROR EXIT TO ERR7 IF MACRO FAILS
 *    SET UP FOR MESSAGE # 405
 *    DO MESG
 *    EXIT WITH RETURN CODE SET TO ZERO
 *
 *    ERROR ENTER ERR1 - NOT TOP OF HIERARCHY
 *      SET UP FOR MESSAGE # 257
 *    ERROR ENTER ERR2 - NO ID ENTERED
 *      SET UP FOR MESSAGE # 261
 *    ERROR ENTER ERR3 - NO OPTION CHOSEN
 *      SET UP FOR MESSAGE # 262
 *    ERROR ENTER ERR4 - BOTH OPTIONS CHOSEN
 *      SET UP FOR MESSAGE # 263
 *    ERROR ENTER ERR5 - INVALID SYSTEM/7 ID
 *      SET UP FOR MESSAGE # 264
 *    ERROR ENTER ERR6 - ID IS NOT REMOTE
 *      SET UP FOR MESSAGE # 265
 *    ERROR ENTER ERR7 - UNABLE TO COMMUNICATE WITH S/7
 *      SET UP FOR MESSAGE # 244
 *    ERROR ENTER ERR8 - NO MATCH ON ACCESS AREA AND FUNCTION
 *      SET UP FOR MESSAGE # 308
 *    ERROR RETURN
 *    DO MESG
 *    EXIT WITH RETURN CODE SET TO ZERO
 * ENDSEGMENT DOMTRS7V
 *
 * MESG SUBROUTINE SEGMENT
 *    GET MESSAGE USING MESSAGE MACRO
```

```
MEMBER NAME  DOMTRS7V
 *   WRITE MESSAGE TO SCRATCH PAD ZONE USING DWZONE MACRO
 *   PATCH DOMCFGD1 WITH ID OF 32 TO REFRESH DISPLAY
 * ENDSEGMENT MESG
```

```
MEMBER NAME  DOMTS7CP
 *  CSECT    DOMTS7CP       PROCESS S/7 CHECKPOINT REQUESTS
 *    OBTAIN A(INPUT BUFFER) FROM PROBL
 *    IF  ANY CHECKPOINT DATA SETS, THEN
 *      IF  CHECKPOINT DATA SET THIS LOGICAL ID, THEN
 *        OBTAIN A(DCB)
 *        OBTAIN A(ARRAY ID LIST)
 *        IF REQUESTED ARRAY ID EXISTS, THEN
 *          INITIALIZE DECB FOR I/O TO CHECKPOINT DATA SET
 *          SET SYNAD ROUTINE ADDRESS IN DCB
 *          SET ZERO CONDITION CODE
 *        ELSE
 *          SET CONDITION CODE = INVALID ARRAY ID
 *        ENDIF
 *      ELSE
 *        SET CONDITION CODE = NO CHECKPOINT DATA SET
 *      ENDIF
 *    ELSE
 *      SET CONDITION CODE = NO CHECKPOINT DATA SET
 *    ENDIF
 *    IF CONDITION CODE IS ZERO, THEN
 *      IF PATCH ID IS ONE, THEN LOG CHECKPOINT DATA
 *        CONVERT OFFSET TO RELATIVE SECTOR # AND SECTOR DISPLACEMENT
 *        ADD RELATIVE BLOCK NUMBER OF RELATIVE SECTOR ZERO
 *        IF VALID RELATIVE SECTOR, THEN
 *          IF BIT CHANGE, THEN
 *            IF VALID BIT # (0-15), THEN
 *              CONVERT SECTOR DISPLACEMENT TO BYTES
 *              STORE RELATIVE BLOCK NUMBER
 *              DO  SEGMENT READREC
 *              OBTAIN A MASK WITH PROPER BIT # SET TO ONE
 *              IF BIT TO BE SET TO ONE, THEN
 *                OR THE MASK INTO THE DESIRED BYTE
 *              ELSE
 *                INVERT MASK BITS
 *                AND THE MASK INTO DESIRED BYTE
 *              ENDIF
 *              DO SEGMENT WRITEREC
 *              ZERO CONDITION CODE
 *            ELSE
 *              SET CONDITION CODE = INVALID BIT NUMBER
 *            ENDIF
 *          ELSE
 *            CALCULATE # OF SECTORS INVOLVED IN CHANGE
 *            IF LAST SECTOR VALID THIS ARRAY, THEN
 *              T = # WORDS TO CHANGE
 *              CONVERT # S/7 WORDS TO BYTES - T=T*2
 *              CONVERT SECTOR DISPLACEMENT TO BYTES
 *              STORE INITIAL RELATIVE BLOCK #
 *              WHILE MORE BYTES TO CHANGE, DO
 *                DO SEGMENT READREC
 *                LOCATE BYTES TO CHANGE
 *                R = # BYTES REMAINING IN THIS RECORD
 *                IF T LESS THAN R, THEN
 *                  R = T
 *                ENDIF
 *                DECREMENT # BYTES TO CHANGE, T=T-R
 *                MOVE R BYTES INTO RELATIVE SECTOR
 *                INCR TO NEXT WORD TO CHANGE
```

```
MEMBER NAME  DOMTS7CP
 *                 DO SEGMENT WRITEREC
 *                 INCREMENT RELATIVE BLOCK # BY ONE
 *                 SET SECTOR DISPLACEMENT = ZERO
 *               ENDDO
 *               SET CONDITION CODE = ZERO
 *             ELSE
 *               CONDITION CODE = INVALID RELATIVE SECTOR
 *             ENDIF
 *           ENDIF
 *         ELSE
 *           CONDITION CODE = INVALID RELATIVE SECTOR
 *         ENDIF
 *       ELSE                           PATCH ID = 2, RETRIEVE CHECKPOINT
 *         IF LAST SECTOR REQUESTED IS VALID, THEN
 *           GETWA SPACE TO BUILD TRANSACTION
 *           IF  GETWA OBTAINED, THEN
 *             BUILD HEADER
 *             INSERT ARRAY ID
 *             INSERT RELATIVE SECTOR REQUESTED
 *             CALCULATE RELATIVE BLOCK # OF FIRST SECTOR
 *             UNTIL ALL SECTORS OBTAINED, DO
 *               STORE RELATIVE BLOCK #
 *               DO SEGMENT READREC
 *               MOVE SECTOR DATA INTO POSITION
 *               INCR RELATIVE BLOCK # BY ONE
 *             ENDDO
 *             S7WRITE DATA TO REQUESTING S/7
 *             FREEWA SPACE
 *             CONDITION CODE = ZERO
 *           ELSE
 *             CONDITION CODE = NO GETWA AVAILABLE
 *           ENDIF
 *         ELSE
 *           CONDITION CODE = INVALID SECTOR REQUESTED
 *         ENDIF
 *       ENDIF
 *     ENDIF
 *     RELEASE INPUT BUFFER
 *     RESET STAE PARAMETER LIST
 *     IF CONDITION CODE IS NOT ZERO
 *       MESSAGE  288 S/7 CHECKPOINT ERROR TC=XX, AID=XX, ARRAY=XX, CC=XX
 *       CONDITION CODE = ZERO
 *     ENDIF
 *     EXIT
 *
 *
 *     BEGIN SEGMENT - READREC
 *       CLEAR ECB
 *       CLEAR SYNAD ERROR BIT
 *       READ S/7 CHECKPOINT DATA SET (USE RELATIVE BLOCK #)
 *       CHECK I/O
 *       ERREXIT  IF,SYNAD ERROR BIT ON TO IOER
 *       RETURN
 *     END SEGMENT - READREC
 *
 *
 *     BEGIN SEGMENT - WRITEREC
 *       CLEAR ECB
```

```
MEMBER NAME  DOMTS7CP
 *       CLEAR SYNAD ERROR BIT
 *       WRITE THE RECORD (USING RELATIVE BLOCK NUMBER)
 *       CHECK I/O
 *       ERREXIT  IF SYNAD ERROR BIT IS ON
 *       RETURN
 *     END SEGMENT - WRITEREC
 *
 *     BEGIN SEGMENT - SYNADRTN (SYNAD ERROR EXIT)
 *       TURN ON SYNAD ERROR BIT
 *       RETURN (NEXT INSTRUCTION AFTER CHECK)
 *     END SEGMENT - SYNADRTN
 *
 *
 *     BEGIN SEGMENT - IOER
 *       MESSAGE  206  S7XXCKPT ERROR CC=XXXXXX
 *       CONDITION CODE = ZERO
 *     END SEGMENT - IOER
 *
 *   END   DOMTS7CP
```

```
MEMBER NAME  DOMTVDB
 * DOMTVDB MAIN SEGMENT
 *    SAVE MESSAGE RECEIVED IN WORK AREA
 *    RELEASE MESSAGE BUFFER USING RLSEBUFF MACRO
 *    ERROR EXIT TO ERR1 IF TRANSACTION CODE IS NOT ZERO
 *    START SEARCH ON SYSTEM/7 CONTROL TABLE FOR MATCH ON S/7 ID
 *    EXIT IF MATCH FOUND
 *      POINT TO RCB LIST FOR SYSTEM/7
 *      SET UP REFERENCE NUMBER FROM MESSAGE
 *      UNTIL THE POINT IS FOUND DO
 *        POINT TO RCB
 *        IF RCB IS NOT MANUAL THEN
 *          IF ANY ANALOG POINTS FOR RCB THEN
 *            DO ANSRCH
 *          ENDIF
 *          IF THE POINT HAS NOT BEEN FOUND THEN
 *            IF ANY STATUS POINTS FOR TERMINAL THEN
 *              IF REFERENCE # GREATER THAN # OF STATUS POINTS THEN
 *                SUBTRACT STATUS COUNT FROM REFERENCE #
 *              ELSE
 *                DO STSRCH
 *              ENDIF
 *            ENDIF
 *          ENDIF
 *          IF POINT HAS NOT BEEN FOUND THEN
 *            IF ANY PULSE COUNTER POINTS FOR TERMINAL THEN
 *              DO PCSRCH
 *            ENDIF
 *          ENDIF
 *        ENDIF
 *        IF RCBS HAVE NOT ALL BEEN SEARCHED THEN
 *          POINT TO NEXT RCB
 *        ELSE
 *          ERROR EXIT TO ERR2 IF POINT HAS NOT BEEN FOUND
 *        ENDIF
 *      ENDDO
 *      GET AREA FOR PARAMETER LIST USING GETWA MACRO
 *      ERROR EXIT TO ERR3 IF MACRO FAILS
 *      MOVE PARAMETER LIST TO AREA GOTTEN
 *      IF COMMAND IS ONLINE THEN
 *        SET PATCH ID TO X'81'
 *      ELSE
 *        IF COMMAND IS OFFLINE THEN
 *          SET PATCH IF TO X'82'
 *        ELSE
 *          ERROR EXIT TO ERR4
 *        ENDIF
 *      ENDIF
 *      PATCH VARY PROCESSOR (DOMTVRPT)
 *      ERROR EXIT TO ERR5 IF PATCH FAILS
 *    OR ELSE
 *      POINT TO NEXT S/7 CONTROL TABLE ENTRY
 *    END LOOP
 *      ERROR EXIT TO ERR6
 *    END SEARCH
 *    EXIT WITH RETURN CODE SET TO ZERO
 *
 *    ERROR ENTER ERR1
 *      SET CODE TO 4
```

```
MEMBER NAME  DOMTVDB
 *    ERROR ENTER ERR2
 *      SET CODE TO 8
 *    ERROR ENTER ERR3
 *      SET CODE TO 12
 *    ERROR ENTER ERR4
 *      SET CODE TO 16
 *    ERROR ENTER ERR5
 *      SET CODE TO 20
 *    ERROR ENTER ERR6
 *      SET CODE TO 24
 *    ERROR RETURN
 *      SET UP VARIABLES FOR MESSAGE
 *      ISSUE MESSAGE # 260 USING MESSAGE MACRO
 *    EXIT WITH RETURN CODE SET TO ZERO
 *  ENDSEGMENT DOMTVDB
 *
 *  ANSRCH SUBROUTINE SEGMENT
 *    POINT TO ANALOG DATA AND NAMES LIST
 *    UNTIL ALL ANALOG POINTS SEARCHED DO
 *      WHILE THE ANALOG POINT IS MANUAL DO
 *        POINT TO THE NEXT ANALOG POINT AND NAMES LIST ENTRY
 *      ENDDO
 *      IF REFERENCE NUMBER IS ZERO THEN
 *        SET UP PARAMETER LIST
 *        FORCE END OF LOOP
 *      ELSE
 *        POINT TO THE NEXT ANALOG POINT AND NAMES LIST ENTRY
 *        SUBTRACT ONE FROM THE REFERENCE NUMBER
 *      ENDIF
 *    ENDDO
 *  ENDSEGMENT ANSRCH
 *
 *  STSRCH SUBROUTINE SEGMENT
 *    POINT TO STATUS DATA
 *    WHILE REFERENCE # IS LARGER THAN GROUP COUNT DO
 *      IF ITEM COUNT IS NZERO THEN
 *        SUBTRACT GROUP COUNT FROM REF # AND FROM ITEM COUNT
 *        POINT TO NEXT STATUS GROUP
 *      ELSE
 *        FORCE END OF LOOP
 *      ENDIF
 *    ENDDO
 *    IF END OF LOOP NOT FORCED THEN
 *      UNTIL GROUP PROCESSED DO
 *        IF REFERENCE # IS ZERO THEN
 *          SET UP PARAMETER LIST
 *          FORCE END OF LOOP
 *        ELSE
 *          POINT TO THE NEXT STATUS POINT
 *          SUBTRACT ONE FROM THE REFERENCE NUMBER
 *        ENDIF
 *      ENDDO
 *    ENDIF
 *  ENDSEGMENT STSRCH
 *
 *  PCSRCH SUBROUTINE SEGMENT
 *    POINT TO PULSE COUNTER DATA FOR THIS TERMINAL
 *    UNTIL ALL PC POINTS PROCESSED DO
```

```
MEMBER NAME  DOMTVDB
 *      WHILE THE PC POINT IS MANUAL DO
 *        POINT TO THE NEXT PC POINT
 *      ENDDO
 *      IF THE REFERENCE NUMBER IS ZERO THEN
 *        SET UP PARAMETER LIST
 *        FORCE END OF LOOP
 *      ELSE
 *        POINT TO THE NEXT PC POINT
 *      ENDIF
 *    ENDDO
 *  ENDSEGMENT PCSRCH
```

CHAPTER 5. DIRECTORY

The Directory lists the modules that comprise the System/370 Energy Management System Program Product. The modules are listed in three tables:

1. Real time modules listed under their operating tasks.
2. Load modules that execute under calling program's task.
3. Short-lived modules and modules used as tables.

TASK (Relative priority) / • Load Module / Object Module	TYPE	Method of Operation Figure	Description
Real Time Modules Listed Under Their Operating Tasks			
• DOMXS7IO (0)	I		
• DOMTS7IO	RENT	2.6.3.0	System/7 Intercommunication
DOMTS7IO			System/7 I/O Processing
DOMTIOER			System/7 I/O Error Processing
DOMTPUNT			Route System/7 Input
• DOMCINIT	NON-REUS	2.1.1	Supervisoty Control Initialization
DOMCINIT			Initialization Controller
DOMCIALM			Regenerate Status Alarms
DOMCWSTA			Update Status Array
DOMTAGCI			AGC Initialization
DOMTPCNT			Pulse Counter Initialization
• DOMTINFO	NON-REUS	2.1.1.0	System/7 Failover Initialization
DOMTINFO			Initialize for System I/O
DOMTPDSG			Retrieve System Load for IPL
DOMXDAQ (1)	I		
• DOMTSSYN	RENT	2.2.1.1	Scan Processing
DOMTSSYN			Scan Processing Control
DOMTCONV			Sensor Based Data Conversion
DOMTPSUM			Point Summation
DOMTCACS			Analog Data Conversion
DOMTCPC			Pulse Counter Data Conversion
DOMTCSES			Status Scan Processor
DOMTPSET			Scan Exception Processing
DOMCPDC1			PDC Hierarchy Processor
• DOMTSINT	RENT	2.1.1	Data Acquisition Initialization
DOMTSINT			Data Acquisition Initialization
DOMXUSER (2)	I		

TASK (Relative priority) • Load Module Object Module	TYPE TYPE	Method of Operation Figure	Description Description
• DOMUSERC	NA	NA	User Cyclic Processor
DOMUSERC			User Cyclic Processor
• DOMTAPLP	RENT	2.2.3.2.1	Performance Logging
DOMTAPLP			Cyclic Processor
• DOMUSERI	NA	NA	User Initialization
DOMUSERI			User Initialization
DOMXAGC (4)	I		
• DOMAAGCB	REUS	2.4.1.0	Auto Generation Control Cyclic Processor
DOMAAGCB			Auto Generation Control Cyclic Processor
DOMAAGCO			AGC Output Interface Processor
DOMCGENO			AGC Output Processor
• DOMAAGCI	REUS	2.1.3	Auto Generation Control Initialization
DOMAAGCI			Auto Generation Control Initialization
• DOMCAGCK	REUS	2.4.1.3	Allow/Disallow AGC Output
DOMCAGCK			Allow/Disallow AGC Output
DOMXALRM (5)	I		
• DOMCALR1	RENT	2.3.1.1	Alarm Management
DOMCALR1			Alarm Supervisor
DOMCALR3			Build Status Alarm Record
DOMCALR4			Build Analog Alarm Record
DOMCALR5			Build Counter Alarm Record
DOMCALR6			Chain Update, Messages
• DOMCEVT1	REUS	2.3.3.1	Event Logging Controller
DOMCEVT1			Log Event and Add Comments
DISPLAYP			Output Messages
• DOMCEVT5	RENT	2.3.3.1	Event Logging Processor
DOMCEVT5			Process Event List from Alarms
DOMXPDC (6)	I		
• DOMCDCO1	RENT	2.3.2.0	Device Control Processor
DOMCDCO1			Device Control Process Controller
DOMCDCO6			Device Control Initialization
DOMCDCO7			Arm and Execure Processor
DOMCDCO8			PDC Reply Processing
DOMCDCO9			COS Receives, Time-Out Cancel Processor
DOMCDCO5			Device Control System/7 Message Controller
DOMCDC11			Option Checking, Logging
• DOMCDCO2	RENT	2.3.2.0	Device Control Execution Time-Out
DOMCDCO2			Device Control Execution Time-Out
• DOMCDCO3	RENT	2.3.2.0	Device Control Arm

TASK (Relative Priority) • Load Module Object Module	TYPE TYPE	Method of Operation Figure	Description
			Time-Out
DOMCDC03			Device Control Arm Time-Out
DOMXPCC (7)	I		
• DOMTS7HS	RENT	2.6.5	System/7 Hierarchy Support
DOMTS7HS			Routing Processor
DOMTHS83			Transaction Code 83 Processing
DOMTHS84			Transaction Code 84 Processing
DOMTHS8B			Transaction Code 8B Processing
DOMTHS8D			Transaction Code 8D Processing
DOMTHS97			Transaction Code 97 Processing
DOMTHSMD			Enter Hierarchy Processing
• DOMTVRPT	RENT	2.2.6.1.1	Vary a Point in or Out of Service
DOMTVRPT			Vary a Point in or Out of Service
• DOMCDC04	RENT	2.3.2.2	Device Control Macro Processor
DOMCDC04			Device Control Macro Processor
• DOMTFAIL	RENT	2.6.4	System/7 Failover Task
DOMTFAIL			System/7 Failover Task
• DOMCWBPR	RENT	2.3.5.1	Wallboard Processor
DOMCWBPR			Wallboard Processor
• DOMTS7CP	RENT	2.6.6	System/7 Checkpoint
DOMTS7CP			System/7 Checkpoint Processing
• DOMTVDB	RENT	2.2.5	System/7 I/O Data Base Process
DOMTVDB			System/7 Vary Data Base Process
DOMXEMD1 (8)	I		
• DOMCEVD1	RENT	2.3.3.2	Event Log Menu Display
DOMCEVD1			Read Events Menu
DISPLAYP			Write Messages
• DOMCEVD2	RENT	2.3.3.2	Events Log Display
DOMCEVD2			Display Events Log
DISPLAYP			Write Messages
• DOMCEVD4	RENT	2.3.3.2	Events Log Display Updating
DOMCEVD4			Add Comment to Event
DISPLAYP			Write Messages
DOMXEMD2 (8)	I		
• DOMCALD1	RENT	2.3.1.2	Alarm Display
DOMCALD1			Alarm Display Control Processor
DOMCALD2			Build the Dynamic Display
DOMCALD3			Retrieve Alarms
DOMCALD4			Process Acknowledge/Delete
• DOMCHART	RENT	2.3.4.3	Stripchart
DOMCHART			Stripchart Control
DOMCHRT1			Build the Dynamic Display
DOMCHRT2			Process Changes

TASK (Relative priority) • Load Module Object Module	TYPE TYPE	Method of Operation Figure	Description Description
DOMXEMD3 (8)	I		
• DOMCDGD1	RENT	2.2.6.2	Power Configuration Control (PCC)
DOMCFGD1			Control PCC Displays
DOMCBLD1			Build PCC-I Display
DOMCBLD2			Build PCC-II Display
DOMCBLD3			Build PCC-III Display
• DOMCFGD2	RENT	2.2.6.1	PCC Change Processor
DOMCFGD2			PCC Change Processor
DISPLAYP			Write Messages
• DOMCAND1	RENT	2.2.1.3	Scan Display
DOMCAND1			Scan Display
• DOMCDIG1	RENT	2.7.3	Online Diagnostics
DOMCDIG1			Online Diagnostics Processing
DISPLAYP			Write Messages
• DOMCTIME	RENT	2.2.6.1	Time Correction
DOMXEMD4 (8)	I		
• DOMCSANA	RENT	2.3.6.2	Analog Data Display Control
DOMCSANA			Analog Data Display Control
• DOMCSSTA	RENT	2.3.6.5	Status Data Display Control
DOMCSSTA			Status Data Display Control
DOMCTLCM			Transfer of PSC Control
• DOMCSPCD	RENT	2.3.6.4	Pulse Counter Data Display Control
DOMSPCD			Pulse Counter Data Display Control
• DOMCSANL	RENT	2.3.6.3	Analog Detail Data Display Control
DOMSPANL			Analog Detail Data Display Control
• DOMCSGET	RENT	2.3.6.1	Remote Control Block Menu Display
DOMCSGET			Remote Control Block Menu Display
• DOMTAPLD	RENT	2.2.3.2.3	Performance Log Display
DOMTAPLD			Performance Log Display
• DOMTAPLM	RENT	2.2.3.2.2 2.2.3.2.1	Performance Log Retrieval Maintenance
DOMTAPLM			Performance Log Retrieval Maintenance
DOMTUSER			Performance Log Retrieval Processor
• DOMCSOMD	RENT	2.1.2	Sign on Display Control
DOMCSOMD			Sign on Display Control
DOMXAGCD (8)	I		
• DOMAGPUP	REUS	2.4.2.6	AGC General Parameter Display
DOMAGPUP			AGC General Parameter Display
• DOMAGSPG	REUS	2.4.2.5	Generator Control Status Matrix Display
DOMAGSPG			Generator Control Status

TASK (Relative priority) • Load Module Object Module	TYPE TYPE	Method of Operation Figure	Description Description
			Matrix Display
• DOMAICUP	REUS	2.5.3.4	EDC Intervals/Costs/Losses
DOMAICUP			Display Processor
• DOMALMUP	REUS	2.4.3.2 2.4.2.1	AGC/EDC Menu Display
DOMALMUP			AGC/EDC Menu Display
• DOMAATPG	REUS	2.5.3.2	Automatic EDC Output Table Display
DOMAATPG			Automatic EDC Output Table Display
• DOMACTPG	REUS	2.5.3.1	EDC Output Table (complete) Display
DOMACTPG			EDC Output Table (complete) Display
• DOMAMTPG	REUS	2.4.3.1.1	Manual EDC Output
DOMAMTPG			Manual EDC Output
• DOMAESUS	REUS	2.4.2.5	Activate/Suspend EDC
DOMAESUS			Activate/Suspend EDC
• DOMALTPG	REUS	2.4.2.2	AGC Output Table Display
DOMALTPG			AGC Output Table Display
• DOMAMTUP	REUS	2.4.2.3	Mode Change/Time Correction Display
DOMAMTUP			Mode Change/Time Correction Display
• DOMATCOR	REUS	2.4.2.3	Start/Stop AGC Time Correction
DOMATCOR			Start/Stop AGC Time Correction
• DOMATLPG	REUS	2.4.2.4	Interchange Schedule Matrix Display
DOMATLPG			Interchange Schedule Matrix Display
• DOMACEPG	REUS	2.4.3.1.2	Change Entry Display
DOMACEPG			Change Entry Display
DOMXEDC (9)			
• DOMAEDCB	REUS	2.5.1	Economic DisPATCH Control (EDC)
DOMAEDCB			EDC Cyclic Processor
• DOMAEDCI	REUS	2.1.4	EDC Initialization
DOMAEDCI			EDC Initialization
DOMXUTIL (10)	I		
• DOMTEIPL	RENT	2.6.4	System/7 IPL Processor
DOMTEIPL			System/7 IPL Processor
DOMTPDSG			Read PDS Logical Record
• DOMTAPLR	RENT	2.2.3.2.2.1 2.2.3.2.2.2	Performance Log Retrieval
DOMTAPLR			Retrieval Processor
DOMTUSER			Record Processor
• DOMTPCLG	REUS	2.2.3.1	Pulse Counter Data Log
DOMTPCLG			Pulse Counter Data Log
• DOMCSLOG	RET	2.2.3.3	Change of Status Log
DOMCSLOG			Change of Status Log
• DOMTDISK	RENT	2.6.2	System/7 Disk Load
DOMTDISK			Transaction Code 92 Processing

TASK (Relative priority) TYPE

• Load Module Object Module	TYPE	Method of Operation Figure	Description Description
DOMTPDSG			Read PDS Logical Record
• DOMCHRTA	RENT	2.3.4.2	Stripchart Change Processor
DOMCHRTA			Stripchart Change Processor
• DOMCWBS7	RENT	2.3.5.2	Wallboard I/O
DOMCWBS7			S/370-S/7 Interface Processor
• DOMCHRTC	RENT	2.3.4.1	Stripchart Cyclic Processor
DOMCHRTC			Stripchart Cyclic Processor

<u>Load Modules That Execute Under Calling Programs' Task</u>

• DOMTABLE	RENT	2.7.2.0	Macro Support Routines
DOMTABLE			Macro Support Routine Address
DOMCALM2			DOMCALRM
DOMCDC10			SCDEVICE
DOMCEVNT			SCEVENT
DOMCLFRE			DOMCFREE
DOMCLRMT			DOMCLGET/DOMCLPUT
DOMTCODE			ASCICOMV Table
DOMTSCAN			INITSCAN
DOMTVARY			VARYCONF
DOMTVS7			VARYS7
DOMTWAIT			S7WRITE
DOMCNCNV			Sensor Based Data Conversion Routines
DOMTCHRT			DOMCSCHT
DOMTUPD7			UPD7COMM
• DOMCSTAE	RENT	2.7.1	Task Recovery Processor
DOMCSTAE			Task Recovery Processor
• DOMCROUT	RENT	2.7.4	Input Data Routing Processor
DOMCROUT			Input Data Routing Processor

<u>Short Lived Modules and Modules Used as Tables</u>

Short Lived Modules

• DOMTCLOK	REUS	2.2.2	Time Synchronization
DOMTCLOK			Time Synchronization
• DOMCTCBI	RENT	2.1.1.0	Task Stae Initialization
DOMCTCBI			Task Stae Initialization
• DOMTRESI	NON-REUS	2.1.1.0	DAQ Initialization
DOMTRESI			Resident DAQ Initialization
DOMCWBIN			Wallboard Table Initialization
DOMTAPLI			Performance Log Table Initialization
DOMTP1IN			DAQ Precheckpoint Initialization

TASK (Relative priority) • Load Module Object Module	TYPE TYPE	Method of Operation Figure	Program Organization Page	Description Description
DOMTP2IN				DAQ Post-checkpoint Initialization
DOMCSENT				Initialization of Entities
				Modules Used as Tables
• DOMTROUT	NA	NA		Transaction Routing Table
DOMTROUT				Transaction Routing Table
• DOMTASKS	NA	NA		Define Task Structure
DOMTASKS				Define Task Structure

CHAPTER 6. DATA AREAS

This section shows the formats of the data base arrays, major control blocks, tables, data sets, and macro parameter DSECTS used by the System/370 Energy Management System. Descriptions of each array, control block, table, data set, or macro parameter DSECT precedes the format illustration. Each array and table description contains the purpose, the creating identifier, and the pointer that locates the array or table. Format illustrations are arranged alphabetically in each section. The subsections in this section are as follows:

- Arrays
- Tables
- Control blocks
- Data sets
- Macro parameters DSECTS

ARRAYS

This section describes all the data base arrays used by the System/370 Energy Management System. The array names are identified by the title of each description.

AAACTADR (Actual addresses of analog input readings)

TOTAL SIZE: 4x (number of generators + number of tie lines + number of non-conforming loads) bytes

CREATED BY: System generation

PURPOSE: DOMALFCI, the AGC initialization program, resolves the data base addresses of the generator, tie line, and nonforming load analog input readings and stores them in this array for use by the cyclic AGC processor, DOMALFCB, and the cyclic EDC processor DOMAEDCB.

Storage Map of AAACTADR:

Offset	TYPE	Description
0	X	Address of analog input for Generator #1
4	X	•• Generator #2
•	•	•• ••
•	•	•• ••
•	•	•• ••
4(NG-1)	X	•• Generator #NG
4(NG)	X	•• Tie Line #1
4(NG+1)	X	•• Tie Line #2
•	•	•• ••
•	•	•• ••
•	•	•• ••
4(NG+NT-1)	X	Tie Line #NT
4(NG+NT)	X	non-conforming load #1
4(NG+NT+1)	X	non-conforming load #2
•	•	•• ••
•	•	•• ••
•	•	•• ••
4(NG+NT+NNCL-1)	X	non-conforming load #NNCL

AACURVEA (Source Cost Curve Matrix)

TOTAL SIZE: (100 x number of generators in the system) bytes

CREATED BY: Customer (may use off line utility DPPXUTIL)

PURPOSE: This array is one of two arrays which may be used by the EDC cyclic processor DOMAEDCB as a source of generator cost curve information. The selection of which array is to be used is under power system operator control via the 5985.

Storage Map of AACURVEA: See system definition section of the System/370 program reference manual.

AACURVEB (Source Cost Curve Matrix)

TOTAL SIZE: (100 number of generators in the system) bytes

CREATED BY: Customer (may use off line utility DPPXUTIL)

PURPOSE: This array is one of two arrays which may be used by the EDC cyclic processor DOMAEDCB as a source of generator cost curve information. The selection of which array is to be used is under power system operator control via the 5985.

Storage Map of AACURVEB: See system definition section of the System/370 program reference manual.

AAEDCODD (Miscellaneous EDC parameters - Logged)

TOTAL SIZE: 32 bytes

CREATED BY: System generation

PURPOSE: Contains EDC parameters that are set to their last logged values during restart.

Storage Map of AAEDCODD:

Offset	Field	Type	Length	Description
0	AIAEDINT	F	4	Automatic EDC Interval (in .01 sec)
4	AICEDINT	F	4	Complete EDC Interval (in .01 sec)
8	AI00NROW	F	4	#rows in BMN matrix
12	AI0CCPTR	C	8	Cost Curve Pointer
20	AIBMNPTR	C	8	BMN Matrix Pointer
28	AIEDCSSP	F	4	EDC Suspension Control

AALFCACT (Actual power readings of generators and tie lines)

TOTAL SIZE: 4 (number of generators + number of tie lines) bytes

CREATED BY: System generation

PURPOSE: Used by the AGC cyclic processor DOMALFCB as temporary storage.

Storage Map of AALFCACT:

Offset	Type	Description		
0	E	Actual power reading in MW	for generator	#1
4	E	••	for generator	#2
•	•	••	••	•
•	•	••	••	•
•	•	••	••	•
4(NG-1)	E	••	for generator	#NG
4NG	E	••	for tie line	#1
4(NG+1)	E	••	for tie line	#2
•	•	••	••	•
•	•	••	••	•
•	•	••	••	•
4(NG+NT-1)	E	••	for tie line	#NT

AALFCEXC (AGC generator exclusicn array)

TOTAL SIZE: (4 x number of generators in the system) bytes

CREATED BY: System generation

PURPOSE: Specifies generators to be excluded from further AGC computations. Used by the AGC cyclic processor DOMALFCB as temporary storage.

Storage Map of AALFCEXC:

Offset	Type	Description	
0	F	Exclusion indicator for generator	#1
4	F	••	#2
•	•	••	•
•	•	••	•
•	•	••	•
4(NG-1)	F	••	#NG

AALFCODD (Miscellaneous AGC parameters - Logged)

TOTAL SIZE: 52 bytes

CREATED BY: System generation

PURPOSE: Contains AGC parameters that are set to their last logged values during restart.

Storage Map of AALFCODD:

Offset	Field	Type	Description	
0	AILFCINT	F	AGC interval	(.01 sec)
4	AILFCOOK	E	Frequency Bias	(MW/.1 Hertz)
8	AIPROSOK	E	Prospective Frequency Bias	(MW/.1 Hertz)
12	AIOFREQS	E	Scheduled Frequency	Hertz
16	AIOGAINA	E	Assist Mode Gain Control	
20	AIOGAINB	E	Economy Gain Control	
24	AIOGAINC	E	Normal Mode Gain Control	
28	AILFCOON	E	Proportion of Instantaneous Value in smoothed area requirement	
32	AILFCOOT	E	LFC mode adjustment	
36	AILFCOOX	E	Dead Band Limit	(MW)
40	AILFCOOY	E	Normal Mode Limit	(MW)
44	AILFCCRL	E	Lower Reserve Range Limit	(MW)

48	AILFCCRR	E	Raise Reserve Range Limit (MW)

AALFCODR (Miscellaneous AGC parameters - Not Logged)

TOTAL SIZ: 48 bytes

CREATED BY: System generation

PURPOSE: Contains AGC parameters that are set to their original initialization values during restart.

Storage Map of AALFCODR:

Offset	Field	Type	Length	Description	Init
0	AILEINIT	F	4	AGC/EDC Initialization Control Item	0
4	AIEDCINP	F	4	Successful EDC Indicator	0
8	AILFCSSP	F	4	Control item to suspend AGC	1
12	AI0FREQO	E	4	Frequency Offset	0
16	AI0000NG	F	4	Number of Generators in System (NG)	--
20	AI0000NT	F	4	Number of Tie Lines in System (NT)	--
24	AI00NCIS	F	4	Number of Interchange Companies (NCIS)	--
28	AI00NNCL	F	4	Number of non-conforming loads (NNCL)	--
32	AI0000NP	F	4	Number of plants (NP)	--
36	AI000NPE	F	4	NG+NT+NNCL	--
40	AILFCGO	H	2	Complete Data Scan Control Item	0
42	AIPLFCGO	H	2	Previous value of AILFCGO	0
44	AIDADFID	A	2	Access Area/Function Area ID	--
46	AICMFORL	H	2	Continuous Monitor Test Location	0

AALFCOTT (Temporary Storage used to build AALFCOUT)

TOTAL SIZE: 4 (number of generators in system + 4) bytes

CREATED BY: System generation

PURPOSE: Used by the AGC cyclic processor DOMALFCB as temporary storage.

Storage Map of AALFCOTT:

Offset	Type	Description	
0	F	AGC Running Mode*	
4	F	AGC Dynamic Mode**	
8	E	Instantaneous Area Requirement	(MW)
12	E	Smoothed Area Requirement	(MW)
16	E	Unit correction for generator #1	(MW)
20	E	Unit correction for generator #2	(MW)
•	•	••	•
•	•	••	•
•	•	••	•
4(NG+3)	E	Unit correction for generator #NG	(MW)

*AGC Running Modes
0 = LFC suspended
1 = Tie Line Bias Control
2 = Flat Frequency Control
3 = Flat Tie Line Control

**AGC Dynamic Mode
1 = Dead Band Mode
2 = Normal Mode
3 = Emergency Assist Mode

AALFCOUT (AGC Output Table)

TOTAL SIZE: 4 (number of generators + 4) bytes

CREATED BY: System generation

PURPOSE: Contains the output of the cyclic AGC processor, DOMALFCB

Storage Map of AALFCOUT:

Offset	Field	Type	Description	
0	AIRUNMOD	F	AGC Running Mode*	
4	AIDYNMOD	F	AGC Dynamic Mode**	
8	AI0000AR	E	Instantaneous Area Requirement	(MW)
12	AI000SAR	E	Smoothed Area Requirement	(MW)
16	AIUCOR01	E	Unit correction for generator #1	(MW)
20	AIUCOR02	E	Unit correction for generator #2	(MW)
•	•	•	••	
•	•	•	••	
•	•	•	••	
4(NG+3)	•	E	Unit correction for generator #NG	(MW)

*AGC Running Modes
0 = LFC Suspended
1 = Tie Line Bias Control
2 = Flat Frequency Control
3 = Flat Tie Line Control

**AGC Dynamic Modes
1 = Dead Band Mode
2 = Normal Mode
3 = Emergency Assist Mode

AALFCPDA (A coefficients for pulse duration computation)

TOTAL SIZE: (4x number of generators in system) bytes

CREATED BY: System generation (The customer may replace or initialize this array using the offline utility (DPPXUTIL.)

PURPOSE: This array is used by the version of the AGC output processor DOMALFCO which is supplied with the program product. It is used when converting unit corrections from megawatts to a pulse interval in accordance with the formula: pulse interval in 100 millisecond units = A(MW) + B. Since DOMALFCO is normally customized for each user it is likely that this array must also be customized.

Storage Map of AALFCPDA:

Offset	Field	Type	Init	Description	
0	AILPDA01	E	0.5	A coefficient for generator	#1
4	AILPDA02	E	0.5	••	#2
•	•	•	•	••	•
•	•	•	•	••	•
•	•	•	•	••	•
4(NG-1)	•	E	0.5	••	#NG

AALFCPDB (B coefficients for pulse duration computation)

TOTAL SIZE: (4 x number of generators in system) bytes

CREATED BY: System generation (The customer may replace or initialize this array using the offline utility DPPXUTIL.)

PURPOSE: This array is used by the version of the AGC output processor DOMALFCO which is supplied with the program product. It is used when converting unit corrections from megawatts to a pulse interval in accordance with the formula: pulse interval in 100 millisecond units = A(MW) + B. Since DOMALFCO is normally customized for each user it is likely that this array must also be customized.

Storage Map of AALFCPDB:

Offset	Field	Type	Init.	Description
0	AILPDB01	E	0.0	A coefficient for generator #1
4	AILPDB02	E	0.0	" #2
•	•	•	•	" •
•	•	•	•	" •
•	•	•	•	" •
4(NG-1)	•	E	0.0	" #NG

AALFCPDO (Pulse duration output array)

TOTAL SIZE: (10 x number of generators in the system) bytes

CREATED B: Systen generation

PURPOSE: Used as output table by the AGC processor program DOMALFCO and as input by the supervisory control cyclic processor DOMCGENO.

Storage Map of AALFCPDO:

Offset	Field	Type	Length	Description
0	AIPDON01	C	4	Character portion of device name for gen. #1 output control point in ASCII
4	--	H	2	Numerical portion of device name for gen. #1 output control point in binary
6	AIPDOO01	H	2	Pulse length for generator #1 in 100 millisecond units (+ = raise, - = lower)
8	AIPDOD01	H	2	System/7 destination code for gen. #1
10	AIPDON02	C	4	Character portion of device name for gen. #2 in ASCII
14	--	H	2	Numerical portion of device name for gen. #2 output control point in binary
16	AIPDOO02	H	2	Pulse length for generator #2 in 100 millisecond units (+ = raise, - = lower)
18	AIPDON02	H	2	System/7 destination code for gen. #2

AALFCPDP (Temporary Storage array for pulse duration output)

TOTAL SIZE: (2 x number of generators in the system) bytes

CREATED BY: Systen generation

PURPOSE: Used as temporary storage by the AGC processor DOMALFCO.

Storage Map of AALFCPDP:

Offset	Type	Description
0	H	Length of pulse in 100 millisecond units for generator #1
2	H	" #2
•	•	" •
•	•	" •
•	•	" •
2(NG-1)	H	" #NG

AALFCRHO (Participation Factors as Modified by DOMALFCB)

TOTAL SIZE: (4 x number of generators in the system) bytes

CREATED BY: System generation

PURPOSE: Used by AGC cyclic processor DOMALFCB as temporary storage.

Storage Map of AALFCRHO:

Offset	Type	Description	
0	E	Participation factor for generator	#1
4	E	••	#2
•	•	••	•
•	•	••	•
•	•	••	•
4(NG-1)	E	••	#NG

AALFCUNC (Unit correction array)

TOTAL SIZE: (4 x number of generators in the system) bytes

CREATED BY: System generation

PURPOSE: Used by the AGC cyclic processor DOMALFCB as temporary storage.

Storage Map of AALFCUNC:

Offset	Type	Description	
0	E	Unit correction (in MW) for generator	#1
4	E	••	#2
•	•	••	•
•	•	••	•
•	•	••	•
4(NG-1)	E	••	#NG

AAPLANTS (Plants Array)

TOTAL SIZE: 12 bytes/plant

CREATED BY: System Generation

PURPOSE: This array contains a plant name and the number of generators at that plant for each plant defined in the system. This array is used for display purposes.

Storage Map of AAPLANTS:

DEC	HEX	
0	0	AIPLT001
8	8	AINGAP01
12	C	AIPLT002
20	14	AINGAP02

Alphabetical List of Fields in AAPLANTS:

Field	DEC	HEX

AINGAP01	8	8
AINGAP02	20	14
AIPLT001	0	0
AIPLT002	12	C

Data Area Layout of AAPLANTS:

Offset	Bytes and Bit Pattern	Field	Description
0 (0)	8	AIPLT001	Plant name of 1st plant
8 (8)	4	AINGAP01	Number of generators at 1st plant
12 (C)	8	AIPLT002	Plant name of 2nd plant
20 (14)	4	AINGAP02	Number of generators at 2nd plant

AAPMAXEC (Maximum Economic Power Limits)

TOTAL SIZE: (4 x number of generators in the system) bytes

CREATED B: System generation

PURPOSE: Used by AGC cyclic prccessor DOMALFCB as temporary storage.

Storage Map of AAPMAXEC:

Offset	Type	Description		
0	E	Maximum economic power limit for generator	#1	(MW)
4	E	••	#2	(MW)
•	•	••	•	•
•	•	••	•	•
•	•	••	•	•
4(NG-1)	E	••	#NG	(MW)

AAPMAXEM (Maximum Emergency Power Limits)

TOTAL SIZE: (4 x number of generators in the system) bytes

CREATED BY: System generation

PURPOSE: Used by AGC cyclic prccessor DOMALFCB as temporary storage.

Storage Map of AAPMAXEM:

Offset	Type	Description		
0	E	Maximum emergency power limit for generator	#1	(MW)
4	E	••	#2	(MW)
•	•	••	•	•
•	•	••	•	•
•	•	••	•	•
4(NG-1)	E	••	#NG	(MW)

AAPMINEC (Minimum Economic Power Limits)

TOTAL SIZE: (4 x number of generators in the system) bytes

CREATED B: System generation

PURPOSE: Used by AGC cyclic prccessor DOMALFCB as temporary storage.

Storage Map of AAPMINEC:

Offset	Type	Description		
0	E	Minimum economic power limit for generator	#1	(MW)
4	E	••	#2	(MW)
•	•	••	•	•
•	•	••	•	•
•	•	••	•	•
4(NG-1)	E	••	#NG	(MW)

AAPMINEM (Minimum Emergency Power Limits)

TOTAL SIZE: (4 x number of generators in the system) bytes

CREATED BY: System generation

PURPOSE: Used by AGC cyclic processor DOMALFCB as temporary storage.

Storage Map of AAPMINEM:

Offset	Type	Description		
0	E	Minimum emergency power limit for generator	#1	(MW)
4	E	••	#2	(MW)
•	•	••	•	•
•	•	••	•	•
•	•	••	•	•
4(NG-1)	E	••	#NG	(MW)

AATIMCOR (Time Correction Array)

TOTAL SIZE: 41 bytes

CREATED BY: System generation

PURPOSE: This array contains time correction parameters used for display purposes only.

Storage Map of AATIMCOR:

DEC	HEX	
0	0	AIDSTRTC
10	A	AITSTRTC
14	E	AIDSTOPC
24	18	AITSTOPC
28	1C	AISPDSLW
37	25	AIDFREQO

Alphabetical List of Fields in AATIMCOR:

Field	DEC	HEX
AIDFREQO	37	25
AIDSTOPC	14	E
AIDSTRTC	0	0
AISPDSLW	28	1C
AITSTOPC	24	18
AITSTRTC	10	A

Data Area Layout of AATIMCOR:

Offset	Bytes and Bit Pattern	Field	Description
0(0)	10	AIDSTRTC	Date to start time correction
10(A)	4	AITSTRTC	Time to start time correction
14(E)	10	AIDSTOPC	Date to stop time correction
24(18)	4	AITSTOPC	Time to stop time correction
28(1C)	9	AISPDSLW	Speed up/Slow down indication
37(25)	4	AIDFREQO	Frequency offset

AATLFCIC (Integrated Values of AGC Commands Since Last Automatic EDC)

TOTAL SIZE: (4 x number of generators in system) bytes

CREATED BY: System generation

PURPOSE: This array is used to accumulate the sum of the AGC commands to each generator between successive automatic EDC executions. This sum is used in a calculation which determines whether or not each generator is obeying the commands sent to it.

Data Area Layout of AATLFCIC:

Offset	Description	
0	Sum of Commands (in Flt. Pt. MW) for generator with internal	#1
4	••	2
8	••	3
12	••	4
etc		

AATTGCSM (Temporary Copy of AA0TGCSM)

TOTAL SIZE: (68 x number of generators in system) bytes

CREATED B: System generation

PURPOSE: Used as temporary storage by the EDC cyclic processor, DOMAEDCB.

Storage Map of AATTGCSM: See "System Definition" section ot the System/370 Energy Management System Program Reference Manual for a storage map of AA0TGCSM. AATTGCSM is identical to it except for field names. These field names (item names) are not used when accessing AATTGCSM.

AATTTGCS (Temporary Copy of AA0TGCSM)

TOTAL SIZE: (68 x number of generators in system) bytes

CREATED B: System generation

PURPOSE: Used as temporary storage by the AGC cyclic processor, DOMALFCB.

Storage Map of AATTTGCS: See "System Definition" section of the System/370 Energy Management System Program Reference Manual for a storage map of AA0TGCSM. AATTTGCS is identical to it except for field names. These field names (item names) are not used when accessing AATTTGCS.

AAWTEDCA (Work table for generating manual EDC output table)

TOTAL SIZE: (16 x number of generators in system + 20) bytes

CREATED BY: System generation

PURPOSE: This table exists only in a system generated to contain AGC but not EDC. It is used as an intermediate storage table when the operator is engaged in changing the contents of array AAOTEDCA for use as input by the AGC cyclic processor DOMALFCB.

Storage Map of AAOTEDCA:

Offset	Field	Type	Description
0	AIWTI001	F	Generator #1 inclusion indicator (0 if not included)
4	AIWTI002	F	•• #2 •• •• (1 if included)
•	•	•	•• • •• ••
•	•	•	•• • •• ••
•	•	•	•• • •• ••
4(NG-1)	•	F	•• #NG •• ••
4NG	AIWTA001	E	Actual Power Output of Generator #1 (MW)
4(NG+1)	AIWTA002	E	•• #2 (MW)
•	•	•	•• •
•	•	•	•• •
•	•	•	•• •
4(2NG-1)	•	E	•• #NG (MW)
4(2NG)	AIWTD001	E	Desired Power Output for Generator #1 (MW)
4(2NG+1)	AIWTD002	E	•• #2 (MW)
•	•	•	•• • •
•	•	•	•• • •
•	•	•	•• • •
4(3NG-1)	•	E	•• #NG (MW)
4(3NG)	AIWTF001	E	Desired Participation Factor for Generator #1
4(3NG+1)	AIWTF002	E	•• #2
•	•	•	•• •
•	•	•	•• •
•	•	•	•• •
4(4NG-1)	•	E	#NG
16NG	AIWTOTSG	E	Total System Generation at Time of Dispatch (MW)
16NG+4	AIWTOTPD	E	Total Power Dispatched (MW)
16NG+8	AIWTOOSL	E	System Lambda ($/MWH)
16NG+12	AIWTODOD	P	Date of Dispatch (Julian)
16NG+16	AIWTOTOD	F	Time of Dispatch (.01 sec)

AAOCURVE (Cost curve matrix)

TOTAL SIZE: 100 (number of generators in system) bytes

CREATED BY: System generation

PURPOSE: Used by the EDC cyclic processor DOMAEDCB as temporary storage.

Storage Map Of AAOCURVE: See "System Definition" section of the System/370 Energy Management System Program Reference Manual for a map of AACURVEA. The storage map of AAOCURVE is identical except for field names. These field names (item names) are not used when accessing AAOCURVE.

AA0DELTP (Change in desired power settings since last automatic EDC)

TOTAL SIZE: (4 x number of generators in the system) bytes

CREATED BY: System generation

PURPOSE: Used by AGC cyclic processor DOMALFCB as temporary storage.

Storage Map of AA0PDLFC:

Offset	Type	Description
0	E	Change in desired power setting for generator #1 (MW)
4	E	•• #2 (MW)
•	•	•• • •
•	•	•• • •
•	•	•• • •
4(NG-1)	E	•• #NG (MW)

AA0GAMMA (Inverse of penalty factor)

TOTAL SIZE: (4 x number of generators in the system) bytes

CPEATED B: System generation

PURPOSE: Used as temporary storage by the EDC cyclic processor DOMAEDCB

Storage Map of AA0GAMMA:

Offset	Type	Description
0	E	Inverse of penalty factor for generator #1
4	E	•• #2
•	•	•• •
•	•	•• •
•	•	•• •
4(NG-1)	E	•• #NG

AA0ICEDC (EDC control array)

TOTAL SIZE: (4 x number of generators in the system) bytes

CREATED BY: System generation

PURPOSE: Specifies generators on which an economic disPATCH is to be performed. Used by DOMAEDCB as temporary storage.

Storage Map of AA0ICEDC:

Offset	Type	Description
0	F	Economic DisPATCH Control indicator for generator #1
4	F	•• #2
•	•	•• •
•	•	•• •
•	•	•• •
4(NG-1)	F	•• #NG

AA0ICLFC (AGC control array)

TOTAL SIZE: (4 x number of generators in the system) bytes

CREATED BY: System generation

PURPOSE: Specifies generators being automatically controlled by the AGC cyclic processor DOMALFCB (1 if on control, 0 if not). Used by the AGC cyclic processor DOMALFCB as temporary storage.

Storage Map of AA0ICLFC:

Offset	Type	Description	
0	F	Automatic control indication for generator	#1
4	F	••	#2
•	•	••	•
•	•	••	•
•	•	••	•
4(NG-1)	F	••	#NG

AA0INDEX (Control array used to compact BMN matrix)

TOTAL SIZE: 4 x (number of generators + number of tie lines +number of non-conforming loads) bytes

CREATED BY: System generation

PURPOSE: The BMN matrix contains coefficients related to the line losses experienced between power plants, tie points, and non-conforming loads. The index array specifies which coefficients are to be used with readings from individual generators within the plants, tie points, and non-conforming loads. This device enables us to compact the BMN matrices by storing coefficients for plants rather than individual generators.

Storage Map of AA0INDEX:

Offset	Type	Description	
0	F	Generator # of 1st generator at plant of gen.	#1
4	F	••	#2
•	•	••	•
•	•	••	•
•	•	••	•
4(NG-1)	F	••	#NG
4(NG)	F	(Generator # of 1st generator at plant of gen	#NG)+1
4(NG+1)	F	(••	)+2
•	•	(••	)•
•	•	(••	)•
•	•	(••	)•
4(NG+NT-1)	F	(••	)+NT
4(NG+NT)	F	(••	)+NT+1
4(NG+NT+1)	F	(••	)+NT+2
•	•	••	•
•	•	••	•
•	•	••	•
4(NG+NT+ NNCL-1)	F	(••	)+NT+NNCL

AA0INGEN (Analog Input Item Names for Generators)

TOTAL SIZE: (8 x number of generators in system) bytes

CREATED BY: System generation

PURPOSE: Used by DOMALFCI to establish data base addresses of analog input data for each generator.

Storage Map of AA0INGEN:

Offset	Field	Description	
0	AIGEN001	8 character Item name of generator	#1
8	AIGEN002	••	2
16	AIGEN003	••	3
24	AIGEN003	••	4
etc			

AA0INNCL (Analog Input Item Names for Non-Conforming Loads)

TOTAL SIZE: (8 x number of non-conforming loads in system) bytes

CREATED B: System generation

PURPOSE: Used by DOMALFCI to establish data base addresses of analog input data for each non-conforming load

Storage Map of AA0INNCL:

Offset	Field	Description	
0	AIINCL01	8 character item name of non-conforming load	#1
8	AIINCL02	••	#2
16	AIINCL03	••	#3
24	AIINCL04	••	#4
etc			

AA0PDEDC (Desired Power Settings Calculated on Last Automatic EDC)

TOTAL SIZE: (4 x number of generators in the sytem) bytes

CREATED BY: System generation

PURPOSE: Used by AGC cyclic processor DOMALFCB as temporary storage.

Storage Map of AA0PDEDC:

Offset	Type	Description		
0	E	Desired Power Setting for generator	#1	(MW)
4	E	••	#2	(MW)
•	•	••	•	•
•	•	••	•	•
•	•	••	•	•
4 (NG-1)	E	••	#NG	(MW)

AA0PDLFC (Desired Power Settings as modified by DOMALFCB)

TOTAL SIZE: (4 x number of generators in the system) bytes

CREATED BY: System generation

PURPOSE: Used by AGC cyclic processor DOMALFCB as temporary storage.

Storage Map of AA0PDLFC:

Offset	Type	Description		
0	E	Desired power setting for generator	#1	(MW)
4	E	••	#2	(MW)
•	•	••	•	•
•	•	••	•	•
•	•	••	•	•
4(NG-1)	E	••	#NG	(MW)

AA0TEDCA (Automatic EDC Output Table)

TOTAL SIZE: (16 x number of generators in system + 20) bytes

CREATED B: System generation

PURPOSE: Contains the output produced by the EDC cyclic processor, DOMAEDCB, when it is PATCHed for an automatic type EDC. This array is used as an input by the AGC cyclic processor DOMALFCB.

Storage Map of AA0TEDCA:

Offset	Field	Type	Description
0	AIADI001	F	Generator #1 inclusion indicator (0 if not inc.)
4	AIADI002	F	•• 2 •• •• (1 if inc.)
•	•	•	•• • •• ••
•	•	•	•• • •• ••
•	•	•	•• • •• ••
4(NG-1)	•	F	•• NG •• ••
4(NG)	AIADA001	E	Actual Power Output of Generator #1 (MW)
4(NG+1)	AIADA002	E	•• #2 (MW)
•	•	•	•• •
•	•	•	•• •
•	•	•	•• •
4(2NG-1)	•	E	•• #NG (MW)
4(2NG)	AIADD001	E	Desired Power Output for Generator #1 (MW)
4(2NG+1)	AIADD002	E	•• #2 (MW)
•	•	•	•• •
•	•	•	•• •
•	•	•	•• •
4(3NG-1)	•	E	•• #NG (MW)
4(3NG)	AIADF001	E	Desired Participation Factor for Generator #1
4(3NG+1)	AIADF002	E	•• #2
•	•	•	•• •
•	•	•	•• •
•	•	•	•• •
4(NG-1)	•	E	•• #NG
16 NG	AIAD0TSG	E	Total System Generation at Time of Dispatch (MW)
16NG+4	AIAD0TPD	E	Total Power Dispatched (MW)
16NG+8	AIAD00SL	E	System Lambda ($/MWH)
16NG+12	AIAD0DOD	P	Date of Dispatch (Julian)
16NG+16	AIAD0TOD	F	Time of Dispatch (.01 sec)

AA0TEDCC (Complete EDC Output Table)

TOTAL SIZE: (16 x number of generators in system + 20) bytes

CREATED BY: System generation

PURPOSE: Contains the output produced by the EDC cyclic processor, DOMAEDCB when it is PATCHed for a complete type EDC.

Storage Map of AA0TEDCC:

Offset	Field	Type	Description
0	AICDI001	F	Generator #1 inclusion indicator (0 if not inc.)
4	AICDI002	F	•• 2 •• •• (1 if inc.)
•	•	•	•• • •• ••
•	•	•	•• • •• ••
•	•	•	•• • •• ••
4(NG-1)	•	F	•• NG •• ••
4(NG)	AICDA001	E	Actual Power Output of Generator #1 (MW)
4(NG+1)	AICDA002	E	•• #2 (MW)
•	•	•	•• •
•	•	•	•• •
•	•	•	•• •
4(2NG-1)	•	E	•• #NG (MW)
4(2NG)	AICDD001	E	Desired Power Output for Generator #1 (MW)
4(2NG+1)	AICDD002	E	•• #2 (MW)
•	•	•	•• •
•	•	•	•• •
•	•	•	•• •
4(3NG-1)	•	E	•• #NG (MW)
4(3NG)	AICDF001	E	Desired Participation Factor for Generator #1
4(3NG+1)	AICDF002	E	•• #2
•	•	•	•• •
•	•	•	•• •
•	•	•	•• •
4(4NG-1)	•	E	•• #NG
16 NG	AICD0TSG	E	Total System Generation at Time of Dispatch (MW)
16NG+4	AICD0TPD	E	Total Power Dispatched (MW)
16NG+8	AICD0OSL	E	System Lambda ($/MWH)
16NG+12	AICD0DOD	P	Date of Dispatch (Julian)
16NG+16	AICD0TOD	F	Time of Dispatch (.01 sec)

AA0TEXTA (Language text array)

TOTAL SIZE: 1394 bytes

CREATED BY: System generation

PURPOSE: This array contains text items that are used for display purposes. Each item may be modified for language independence. The only restrictions to modification are length and item name specifications must remain unchanged. Some items in this array are used as dynamic parameters in display foregrounds, while others are used as event text.

Alphabetical List of Fields in AA0TEXTA:

Field	Dec	Hex
AIACCFUN	1116	45C
AIAMUCGN	636	27C
AIAUTOIN	556	22C
AIBLANKS	376	178
AIBMEVNT	1172	494
AIBSLDPT	156	9C
AICCEVNT	1162	48A
AICOMPIN	576	240
AICONVER	1241	4D9
AICURSCV	816	330

AIDBMDLM	716	2CC
AIDELETE	1360	550
AIDESSV1	836	344
AIDESSV2	956	3BC
AIDTOSRT	436	1B4
AIDTOSTP	476	1DC
AIDTSSC1	876	36C
AIDTSSC2	976	3D0
AIDTSTS1	896	380
AIDTSTS2	1016	3F8
AIECPLHI	196	C4
AIECPLML	176	B0
AIEMASIN	116	74
AIEMLMHI	276	114
AIEMLMLO	256	100
AIEVTACT	1258	4EA
AIEVTEDC	1136	470
AIEVTFFC	1309	51D
AIEVTFTL	1326	52E
AIEVTLFC	1139	473
AIEVTSUS	1275	4FB
AIEVTTLB	1292	50C
AIFRBIAS	416	1A0
AIFROFFS	536	218
AIFUELCS	356	164
AIGENSTA	96	60
AIINCLHI	236	EC
AIINCLML	216	D8
AIINTERV	796	31C
AIINVAID	1343	53F
AILFCRMD	396	18C
AILTEMRR	316	13C
AIMNUSRT	336	150
AIMRCSC1	936	3A8
AIMRCSC2	1056	420
AINMDLIM	736	2E0
AINMSARG	676	2A4
AINMUCGN	656	290
AINONCCR	1377	561
AINOTINL	1224	4C8
AIOUTLMH	1190	4A6
AIOUTLML	1207	4B7
AIPINSAR	696	2B8
AIPOINTT	596	254
AIREQTED	1142	476
AIRMPRTE	136	88
AISCHEDF	616	268
AISCH1DL	1076	434
AISCH2DL	1096	448
AISRRLRC	756	2F4
AISRRRRC	776	308
AISTEMRR	296	128
AITIMCEV	1182	49E
AITMSUSD	516	204
AITMTOSP	496	1F0
AITMTOST	456	1C8
AITTSSC1	876	36C
AITTSSC2	996	3E4
AITTSTS1	916	394
AITTSTS2	1036	40C
AIVIOLTE	1152	480
AI000NOC	6	6
AI00GENC	0	0
AI00UNKC	9	9
AI00YESC	3	3

AI0GNST0	68	44
AI0GNST1	54	36
AI0GNST2	40	28
AI0GNST3	26	1A
AI0GNST4	12	C
AI0GNSTU	82	52

Storage Map of AA0TEXTA:

Offset	Bytes and Bit Pattern	Field	Description
0 (0)	3	AI00GENC	GEN for generator.
3 (3)	3	AI00YESC	Yes.
6 (6)	3	AI000NOC	No.
9 (9)	3	AI00UNKC	UNK for unknown.
12 (C)	14	AI0GNST4	Automatic Control.
26 (1A)	14	AI0GNST3	Economically variable.
40 (28)	14	AI0GNST2	Base loaded.
54 (36)	14	AI0GNST1	Off control.
68 (44)	14	AI0GNST0	Out of service.
82 (52)	14	AI0GNSTU	Unknown status.
96 (60)	20	AIGENSTA	Generator status.
116 (74)	20	AIEMASIN	Emergency assist indicator.
136 (88)	20	AIRMPRTE	Ramp rate.
156 (9C)	20	AIBSLDPT	Base load point.
176 (B0)	20	AIECPLML	Economic power limit - low.
196 (C4)	20	AIECPLHI	Economic power limit - high.
216 (D8)	20	AIINCLML	Incremental cost limit - low.
236 (EC)	20	AIINCLHI	Incremental cost limit - high.
256 (100)	20	AIEMLMLO	Emergency limit - low.
276 (114)	20	AIEMLMHI	Emergency limit - high.
296 (128)	20	AISTEMRR	Short term ramp rate.
316 (13C)	20	AILTEMRR	Long term ramp rate.
336 (150)	20	AIMNUSRT	Minimum usable rate.
356 (164)	20	AIFUELCS	Fuel cost.
376 (178)	20	AIBLANKS	Blanks.
396 (18C)	20	AILFCRMD	LFC running mode.
416 (1A0)	20	AIFRBIAS	Frequency bias.
436 (1B4)	20	AIDTOSRT	Date to start.
456 (1C8)	20	AITMTOST	Time to start.
476 (1DC)	20	AIDTOSTP	Date to stop.
496 (1F0)	20	AITMTOSP	Time to stop.
516 (204)	20	AITMSUSD	Speed up/slow down.
536 (218)	20	AIFROFFS	Frequency offset.
556 (22C)	20	AIAUTOIN	Automatic interval.
576 (240)	20	AICOMPIN	Complete interval.
596 (254)	20	AIPOINTT	Pointer to.
616 (268)	20	AISCHEDF	Scheduled frequency.
636 (27C)	20	AIAMUCGN	Assist mode unit correction gain.
656 (290)	20	AINMUCGN	Normal mode unit correction economic gain.
676 (2A4)	20	AINMSARG	Normal mode unit correction smoothed area requirement gain.
696 (2B8)	20	AIPINSAR	Percent instantaneous area requirement in smoothed area requirement.
716 (2CC)	20	AIDBMDLM	Deadband mode limit.
736 (2E0)	20	AINMDLIM	Normal mode limit.
756 (2F4)	20	AISRRLRC	System response range lower generation capability criterion.
776 (308)	20	AISRRRRC	System response range raise generation capability criterion.
796 (31C)	20	AIINTERV	Interval.
816 (330)	20	AICURSCV	Current scheduled value.
836 (344)	20	AIDESSV1	Desired scheduled value for schedule

			one.
856(358)	20	AIDTSSC1	Date to start for schedule one.
876(36C)	20	AITTSSC1	Time to start for schedule one.
896(380)	20	AIDTSTS1	Date to stop for schedule one.
916(394)	20	AITTSTS1	Time to stop for schedule one.
936(3A8)	20	AIMRCSC1	Maximum rate of change for schedule one.
956(3BC)	20	AIDESSV2	Desired schedule value for schedule two.
976(3D0)	20	AIDTSSC2	Date to start for schedule two.
996(3E4)	20	AITTSSC2	Time to start for schedule two.
1016(3F8)	20	AIDTSTS2	Date to stop for schedule two.
1036(40C)	20	AITTSTS2	Time to stop for schedule two.
1056(420)	20	AIMRCSC2	Maximum rate of change for schedule two.
1076(434)	20	AISCH1DL	Schedule one.
1096(448)	20	AISCH2DL	Schedule two.
1116(45C)	20	AIACCFUN	Access/function area.
1136(470)	3	AIEVTEDC	EDC.
1139(473)	3	AIEVTLFC	LFC.
1142(476)	10	AIREQTED	Requested.
1152(480)	10	AIVIOLTE	Violation
1162(48A)	10	AICCEVNT	Cost curve.
1172(494)	10	AIBMEVNT	BMN matrix.
1182(49E)	8	AITIMCEV	Time correction.
1190(4A6)	17	AIOUTLMH	Out of limits - high.
1207(4B7)	17	AIOUTLML	Out of limits - low.
1224(4C8)	17	AINOTINL	Not in list.
1241(4D9)	17	AICONVER	Conversion error.
1258(4EA)	17	AIEVTACT	Activated.
1275(4FB)	17	AIEVTSUS	Suspended.
1292(50C)	17	AIEVTTLB	Tie line bias.
1309(51D)	17	AIEVTFFC	Flat frequency.
1326(52E)	17	AIEVTFTL	Flat tie line.
1343(53F)	17	AIINVAID	Invalid.
1360(550)	17	AIDELETE	Deleted.
1377(561)	17	AINONCCR	Not on cost curve.

AA0TGCSM (Generator Control Status Matrix)

TOTAL SIZE: (68 x number of generators in system) bytes

CREATED BY: Customer (may use off line utility DPPXUTIL)

PURPOSE: This array is used by the AGC cyclic processor DOMALFCB and by the EDC cyclic processor DOMAEDCB as a source of generator control status information. The contents of this array is changed by the power system operator via the 5985.

Storage Map of AA0TGCSM: See "System Definition" section of the System/370 Energy Management System Program Reference Manual.

AA0TICSM (Temporary Copy of AA00ICSM)

TOTAL SIZE: (60 x number of company interchange schedules) bytes

CREATED BY: System generation

PURPOSE: Used as temporary storage by the AGC cyclic processor, DOMALFCB, when updating the interchange schedule matrix AA00ICSM

Storage Map of AA0TICSM: Identical to AA00ICSM except for field names. These field names (item names) are not used when accessing AA0TICSM.

AA00BMNA (Source BMN coefficient matrix)

TOTAL SIZE: 4 (number of generating plants + number of ties lines + number of non-conforming loads) (number of generating plants) bytes

CREATED BY: Customer (may use off line utility DPPXUTIL)

PURPOSE: This array is one of four arrays which may be used by the EDC cyclic processor DOMAEDCB as a source of line loss coefficients. The selection of the array to be used is under power system operator control via the 5985.

Storage Map of AA00BMNA: See "System Definition" section of the System/370 Energy Management System Program Reference Manual.

AA00BMNB (Source BMN coefficient matrix)

TOTAL SIZE: 4 (number of generating plants + number of tie lines + number of non-conforming loads) (number of generating plants) bytes

CREATED BY: Customer (may use off line utility DPPXUTIL)

PURPOSE: This array is one of four arrays which may be used by the EDC cyclic processor DOMAEDCB as a source of line loss coefficients. The selection of the array to be used is under operator control via the 5985.

Storage Map of AA00BMNB: See "System Definition" section of the System 370 Energy Management System Program Reference Manual.

AA00BMNC (Source BMN coefficient matrix)

TOTAL SIZE: 4 (number of generating plants + number of tie lines + number of non-conforming loads) (number of generating plants) bytes

CREATED BY: Customer (may use off line utility DPPXUTIL)

PURPOSE: This array is one of four arrays which may be used by the EDC cyclic processor DOMAEDCB as a source of line loss coefficients. The selection of the array to be used is under operator control via the 5985.

Storage Map of AA00BMNC: See "System Definition" section of the System/370 Energy Management System Program Reference Manual.

AA00BMND (Source BMN coefficient matrix)

TOTAL SIZE: 4 (number of generating plants + number of tie lines + number of non-conforming loads) (number of generating plants) bytes

CREATED BY: Customer (may use off line utility DPPXUTIL)

PURPOSE: This array is one of four arrays which may be used by the EDC cyclic processor DOMAEDCB as a source of line loss coefficients. The selection of the array to be used is under operator control via the 5985.

Storage Map of AA00BMND: See "System Definition" section of the System/370 Energy Management System Program Reference Manual.

AA00IBMN (Control Array to Eliminate Redundant Penalty Factor Computations)

TOTAL SIZE: (4 x number of generators in system) bytes

CREATED BY: System generation

PURPOSE: Control array used by the EDC cyclic processor DOMAEDCB to eliminate redundant penalty factor computations. If the i-th word of the array is zero, a penalty factor is to be computed for i-th generator. If the i-th word is not zero it contains an integer which specifies the number of the generator whose penalty factor is to be used.

Storage Map of AA00IBMN:

Offset	Type	Description
0	F	Control word for generator #1
4	F	•• #2
•	•	•• •
•	•	•• •
•	•	•• •
4(NG-1)	F	•• #NG

AA00ICSM (Interchange Schedule Matrix)

TOTAL SIZE: (60 x number of company interchange schedules) bytes

CREATED BY: System generation

PURPOSE: Used by the operator to command interchange schedule changes and by the AGC cyclic processor, DOMALFCB, to implement them.

Storage Map of AA00ICSM:

Offset	Field	Type	Length	Description	
0	AIIC0101	C	8	Interchange Company Schedule Name #1	
8	AIIC0201	E	4	Current scheduled value	(MW)
12	AIIC0301	E	4	Desired scheduled value for change #1	(MW)
16	AIIC0401	P	4	Date to start change #1	(Julian)
20	AIIC0501	F	4	Time to start change #1	(.01 sec)
24	AIIC0601	P	4	Date to stop change #1	(Julian)
28	AIIC0701	F	4	Time to stop change #1	(.01 sec)
32	AIIC0801	E	4	Max. change rate #1	(MW/sec)
36	AIIC0901	E	4	Desired scheduled value for change #2	(MW)
40	AIIC1001	P	4	Date to start change #2	(MW)
44	AIIC1101	F	4	Time to start change #2	(Julian)
48	AIIC1201	P	4	Date to stop change #2	(.01 sec)
52	AIIC1301	F	4	Time to stop change #2	(Julian)
56	AIIC1401	E	4	Max. change rate #2	(MW/sec)
60	AIIC0102	C	8	Interchange Company Schedule Name #2	
68	AIIC0202	E	4	Current scheduled value	(MW)

72	AIIC0302	E	4	Desired scheduled value for change #1	(MW)
76	AIIC0402	P	4	Date to start change #1	(Julian)

AA00INSF (Analog Input Item Names for System Frequency)

TOTAL SIZE: 16 bytes

CREATED BY: System generation

PURPOSE: Enables processing programs to access primary and back up system frequency sensor readings.

Storage Map of AA00INSF:

Offset	Field	Description
0	AISYSF1	8 character item name for primary system frequency
8	AISYSF2	8 character item name for back up system frequency

AA00INTL (Analog Input Item Names for Tie-Lines)

TOTAL SIZE: (8 x number of tie-lines in system) bytes

CREATED B: System generation

PURPOSE: Used by DOMALFCI to establish data base addresses of analog input data for each tie-line.

Storage Map of AA00INTL:

Offset	Field	Description	
0	AITL0001	8 character item name of tie-line	#1
8	AITL0002	••	#2
16	AITL0003	••	#3
24	AITL0004	••	#4
etc			

AA00PACT (Actual power readings of generators, tie-lines, and non-conforming loads)

TOTAL SIZE: 4 (number of generators + number of tie-lines + number of non-conforming loads) bytes

CREATED BY: System generation

PURPOSE: Used by the EDC cyclic processor DOMAEDCB as temporary storage.

Storage Map of AA00PACT:

Offset	Type	Description		
0	E	Actual power reading in MW	for generator	#1
4	E	••	for generator	#2
•	•	••	••	•
•	•	••	••	•
•	•	••	••	•
4(NG-1)	E	••	for generator	#NG
4(NG)	E	••	for tie-line	#1
4(NG+1)	E	••	for tie-line	#2
•	•	••	••	•

•	•	•• •• •
•	•	•• •• •
4(NG+NT-1)	E	•• for tie-line #NT
4(NG+NT)	E	•• for non-conforming load #1
4(NG+NT+1)	E	•• for non-conforming load #2
•	•	•• •• •
•	•	•• •• •
•	•	•• •• •
4(NG+NT+ NNCL-1)	E	•• for non-conforming load #NNCL

AA00PDES (Desired power readings for generators, tie lines, and non-conforming loads)

TOTAL SIZE: 4 (number of generators + number of tie-lines + number of non-conforming loads) bytes

CREATED BY: System generation

PURPOSE: Used by the EDC cyclic processor DOMAEDCB as temporary storage.

Storage Map of AA00PDES:

Offset	Type	Description
0	E	Desired power reading in MW for generator #1
4	E	•• for generator #2
•	•	•• •• •
•	•	•• •• •
•	•	•• •• •
4(NG-1)	E	•• for generator #NG
4(NG)	E	Actual power reading in MW for tie-line #1
4(NG+1)	E	•• for tie-line #2
•	•	•• •• •
•	•	•• •• •
•	•	•• •• •
4(NG+NT-1)	E	•• for tie-line #NT
4(NG+NT)	E	Actual power reading in MW for non-conforming load #1
4(NG+NT+1)	E	•• for non-conforming load #2
•	•	•• •• •
•	•	•• •• •
•	•	•• •• •
4(NG+NT+ NNCL-1)	E	•• for non-conforming load #NNCL

AA00PMAX (Upper Economic Power Limits)

TOTAL SIZE: (4 x number of generators in the system) bytes

CREATED BY: System generation

PURPOSE: Used by the EDC cyclic processor DOMAEDCB as temporary storage.

Storage Map of AA00PMAX:

Offset	Type	Description
0	E	Upper economic power limit for generator #1 (MW)
4	E	•• #2 (MW)
•	•	•• • •
•	•	•• • •

•	•	••	•	•
4(NG-1)	E	••	#NG	(MW)

AA00PMIN (Lower Economic Power Limit)

TOTAL SIZE: (4 x number of generators in the system) bytes

CREATED BY: System generation

PURPOSE: Used by the EDC cyclic processor DOMAEDCB as temporary storage.

Storage Map of AA00PMIN:

Offset	Type	Description		
0	E	Lower economic power limit for generator	#1	(MW)
4	E	••	#2	(MW)
•	•	••	•	•
•	•	••	•	•
•	•	••	•	•
4(NG-1)	E	••	#NG	(MW)

AA00PTHI (Highest permitted power value on current EDC iteration)

TOTAL SIZE: (4 x number of generators in the system) bytes

CREATED BY: System generation

PURPOSE: Used by the EDC cyclic processor DOMAEDCB as temporary storage.

Storage Map of AA00PTHI:

Offset	Type	Description		
0	E	Highest permitted power value for generator	#1	(MW)
4	E	••	#2	(MW)
•	•	••	•	•
•	•	••	•	•
•	•	••	•	•
4(NG-1)	E	••	#NG	(MW)

AA00PTLO (lowest permitted power value on current EDC iteration)

TOTAL SIZE: (4 x number of generators in the system) bytes

CREATED BY: System generation

PURPOSE: Used by the EDC cyclic processor DOMAEDCB as temporary storage.

Storage Map of AA00PTLO:

Offset	Type	Description		
0	E	Lowest permitted power value for generator	#1	(MW)
4	E	••	#2	(MW)
•	•	••	•	•
•	•	••	•	•
•	•	••	•	•
4(NG-1)	E	••	#NG	(MW)

AA00RHO (Participation Factors Calculated on Last Automatic EDC)

TOTAL SIZE: (4 x number of generators in the system) bytes

CREATED BY: System generation

PURPOSE: Used by AGC cyclic processor DOMALFCB as temporary storage.

Storage Map of AA00RHOO:

Offset	Type	Description
0	E	Participation factor for generator #1
4	E	•• #2
•	•	•• •
•	•	•• •
•	•	•• •
4(NG-1)	E	•• #NG

AA000BMN (BMN matrix)

TOTAL SIZE: 4 (number of generating plants + number of tie-lines + number of non-conforming loads) (number of generating plants) bytes

CREATED BY: System generation

PURPOSE: Used as temporary storage by the EDC cyclic processor DOMAEDCB

Storage Map of AA000BMN: See system definition section of the System/370 program reference manual for a map of AA00BMNA and AA00BMNB. The storage map of AA000BMN is identical except for field names. These field names (item names) are not used when accessing AA000BMN.

AA000EXC (EDC generator exclusion array)

TOTAL SIZE: (4 x number of generators in the system) bytes

CREATED BY: System generation

PURPOSE: Specifies generators to be excluded from further EDC calculations. Used by the EDC cyclic processor DOMAEDCB as temporary storage.

Storage Map of AA000EXC:

Offset	Type	Description
0	F	Exclusion indicator for generator #1
4	F	•• #2
•	•	•• •
•	•	•• •
•	•	•• •
4(NG-1)	F	•• #NG

AA000DDP (Change in desired power settings between EDC iterations)

TOTAL SIZE: (4 x number of generators in the system) bytes

CREATED BY: System Generation

PURPOSE: Used by the EDC cyclic processor DOMAEDCB as temporary storage.

Storage Map of AA000DDP:

Offset	Type	Description		
0	E	Change in desired power setting for generator	#1	(MW)
4	E	••	#2	(MW)
•	•	••	•	•
•	•	••	•	•
•	•	••	•	•
4(NG-1)	E	••	#NG	(MW)

AA000RHO (Participation factors)

TOTAL SIZE: (4 x number of generators in the system) bytes

CREATED BY: System generation

PURPOSE: Used by the EDC cyclic processor DOMAEDCB as temporary storage.

Storage Map of AA000RHO:

Offset	Type	Description	
0	E	Participation factor for generator	#1
4	E	••	#2
•	•	••	•
•	•	••	•
•	•	••	•
4(NG-1)	E	••	#NG

AA00000A (Rate of Change of Incremental Costs)

TOTAL SIZE: (4x number of generators in system) bytes

CREATED BY: System generation

PURPOSE: Used as temporary storage by the EDC cyclic processor DOMAEDCB

Storage Map of AA00000A:

Offset	Type	Description	
0	E	Rate of change of incremental cost for generator	#1
4	E	••	#2
•	•	••	•
•	•	••	•
•	•	••	•
4(NG-1)	E	••	#NG

AA00000D (Main diagonal elements of BMN matrix in use)

TOTAL SIZE: (4 x number of generators in the system) bytes

CREATED BY: System generation

PURPOSE: Used as temporary storage by the EDC cyclic processor DOMAEDCB

Storage Map of AA00000D:

Offset	Type	Description
0	E	BMN coefficient that relates gen. #1 to itself B1,1
4	E	•• #2 •• B2,2
•	•	•• • ••
•	•	•• • ••
•	•	•• • ••
4(NG-1)	E	•• #NG •• BNG,NG

AA00000H (Incremental costs)

TOTAL SIZE: (4 x number of generators in system) bytes

CREATED BY: System generation

PURPOSE: Used as temporary storage by the EDC cyclic processor DOMAEDCB

Storage Map of AA00000H:

Offset	Type	Description
0	E	Incremental cost for generator #1
4	E	•• #2
•	•	•• •
•	•	•• •
•	•	•• •
4(NG-1)	E	•• #NG

AA00000M (Value of Cost Difference Function)

TOTAL SIZE: (4 x number of generators in system) bytes

CREATED BY: System generation

PURPOSE: Used as temporary storate by the EDC cyclic processor DOMAEDCB

Storage Map of AA00000M:

Offset	Type	Description
0	E	Value of cost difference function for generator #1
4	E	•• #2
•	•	•• •
•	•	•• •
•	•	•• •
4(NG-1)	E	•• #NG

AA00000T (Intermediate temporary storage)

TOTAL SIZE: (4 x number of generators in the system) bytes

CREATED BY: System generation

PURPOSE: Used by the EDC cyclic processor DOMAEDCB as temporary storage.

Storage Map of AA00000T:

Offset	Type	Description
0	E	Temporary storage for calculations related to generator #1
4	E	•• #2
•	•	•• •
•	•	•• •
•	•	•• •
4(NG-1)	E	•• #NG

ALEDCODD (Log copy of AAEDCODD)

TOTAL SIZE: 32 bytes

CREATED BY: System generation

PURPOSE: Log copy of AAEDCODD

Storage Map: See AAEDCODD

ALLFCODD (Log copy of AALFCODD)

TOTAL SIZE: 52 bytes

CREATED BY: System generation

PURPOSE: Log copy of AALFCODD

Storage Map: See AALFCODD

AL00ICSM (Log copy of AA00ICSM)

TOTAL SIZE: (60 x number of company interchange schedules) bytes

CREATED BY: System generation

PURPOSE: Log copy of AA00ICSM

Storage Map: See AA00ICSM

CAAATBL (ACCESS AREA TABLE)

TOTAL SIZE: 4 + (8 * A) + (A * (16 * F)) DSECT: AAD

CREATED BY: System generation

PURPOSE: The array contains an entry for each access area and has the controls for chaining alarm records, the routing codes for the typers and the information for audible alarming through the System/7. The array is accessed using the GETARRAY macro.

Storage Map of CAAATBL

DEC	HEX				
0	0	AA#AA (A)		AAR1	
4	4	AAID	AA#FC	AADISPL	
		AAAUDIB		AAR2	
4+ (8*A)		AAFC	AAR3	AAERC	AAARC
		AA#ALR		AANUMB	
		AAFIRST			
		AALAST			

Data Area Layout of CAAATBL

Offset	Bytes and Bit Pattern	Field	Description
0 (0)	2	AA#AA	Number of access areas in table
2 (2)	2	AAR1	Reserved
4 (4)	1	AAID	Access area ID
5 (5)	1	AA#FC	Number of function codes per access area
6 (6)	2	AADISPL	Displacement to start of function information for this access area
8 (8)	2	AAAUDIB	Audible alarm information formatted for System/7
10 (A)	2	AAR2	Reserved
0+ (2+ (8*A))	1	AAFC	Function code
1+ (2+ (8*A))	1	AAR3	Reserved
2+ (2+ (8*A))	1	AAERC	Event routing code for typers
3+ (2+ (8*A))	1	AAARC	Alarm routing code for typers
4+ (2+ (8*A))	2	AA#ALR	Number of alarms outstanding for this access area and function
6+ (2+ (8*A))	2	AANUMB	Last sequence number used for an alarm
8+ (2+ (8*A))	4	AAFIRST	Address of first detail alarm record in chain
12+ (2+ (8*A))	4	AALAST	Address of last detail alarm record in chain

CAAPLPT (Application point array)

TOTAL SIZE: 9 X number of entries DSECT: APPOINT

CREATED BY: System generation

PURPOSE: The non-sensor based alarms are related to an alarm control block and an application point so that they may be included in the applicable access area/function alarm chain. The application point entry is the equivalent of a status point for sensor based alarms and contains the APPOINT name, the alarm flag, and the function code.

POINTED TO BY: The alarm control block

Storage Map of CAAPLPT

DEC	HEX		
0	0	APALARM	
4	4	APNAME	
8	8	APFUNC	

Alphabetical List of Fields in CAAPLPT

Field	DEC	HEX
APALARM	0000	0000
APFUNC	0008	0008
APNAME	0001	0001

Data Area Layout of CAAPLPT

Offset	Bytes and Bit Pattern	Field	Description
0(0)	0	--	Flag byte
	1...	APALARM	Alarm outstanding - on
1(1)	7	APNAME	APPOINT name
8(8)	1	APFUNC	Function code

CAAPPL (Application alarm control block array)

TOTAL SIZE: 48 bytes per entry DSECT: RCBD

CREATED BY: System generation

PURPOSE: The non-sensor based alarms are related to an alarm control block and an application point so that they may be included in the applicable access area/function alarm chain. The alarm control block is the equivalent of an RCB for a sensor-based alarm and contains the ACB name, the count of alarms for the ACB, the access area ID, a pointer to the start of the CAAPLPT entries for the ACB, and the name of the display which is called up when the general alarm is achnowledged. The ACB name is used for the general alarm and appears in the general alarm zone of the display unit.

Storage Map of CAAPPL

DEC	HEX				
0	0	RCBACT	RCBS7I	RCBS7IT	RCBAA
4	4	RCBID	RCBDROP	RCBLINE	
8	8	RCBINDS			
12	C		RCBNAME		
16	10				
20	14				
24	18		RCBDISP		
28	1C		RCBNAMEL		
32	20	RCBANN		RCBPCN	RCBSN
36	24	RCBX1	RCBPC		
40	28	RCBX2	RCBANAL		
44	2C	RCBX3	RCBSTAT		

Alphabetical List of Fields in CAAPPL

Field	DEC	HEX
RCBAA	0003	0003
RCBACT	0000	0000
RCBANAL	0041	0029
RCBANN	0032	0020
RCBDISP	0021	0015
RCBDROP	0005	0005
RCBID	0004	0004
RCBINDS	0008	0008
RCBLINE	0006	0006
RCBNAME	0009	0009
RCBNAMEL	0029	001D
RCBPC	0037	0025
RCBPCN	0034	0022
RCBS7I	0001	0001
RCBS7IT	0002	0002
RCBSN	0035	0023
RCBSTAT	0045	002D
RCBX1	0036	0024
RCBX2	0040	0028
RCBX3	0044	002C

Data Area Layout of RCB

Offset	Bytes and Bit Pattern	Field	Description
0 (0)	1	RCBACT	Count of outstanding alarms
1 (1)	1	RCBS7I	Reserved
2 (2)	1	RCBS7IT	Reserved
3 (3)	1	RCBAA	Access area ID
4 (4)	1	RCBID	Reserved
5 (5)	1	RCBDROP	Reserved
6 (6)	2	RCBLINE	Reserved
8 (8)	Byte	RCBINDS	Flag byte as follows:
	1...	RCBACK	General alarm acknowledged if on

Offset	Bytes and Bit Pattern	Field	Description
	.111 1111		Reserved
9 (9)	12	RCBNAME	Alarm control block name
21 (15)	8	RCBDISP	Name of display associated with ACB
29 (1D)	3	RCBNAMEL	Reserved
32 (20)	2	RCBANN	Reserved
34 (22)	1	RCBPCN	Reserved
35 (23)	1	RCBSN	Count of appoint items for this ACB
36 (24)	1	RCBX1	Reserved
37 (25)	3	RCBPC	Reserved
40 (28)	1	RCBX2	Reserved
41 (29)	3	RCBANAL	Reserved
44 (2C)	1	RCBX3	Reserved
45 (2D)	3	RCBSTAT	Address of start of appoint data for this RCB

Note: The ACB uses the RCBD DSECT and has the same format as the RCB array.

CACOUNT (PULSE COUNTER DATA ARRAY)

TOTAL: 40 bytes/counter DSECT: PC

CREATED BY: System generation

PURPOSE: The array contains the count in engineering units, the accumulated value, the last point counter value, a filter value for estimating pulse counters, maximum pulse per scan, time of last good pulse counter value, and a scale multiplier.

POINTED TO BY: The remote control block (RCB) contains a pointer to the pulse counter data for that remote terminal in the RCBPC field.

Storage Map of CACOUNT

DEC	HEX				
0	0	PCINCR			
4	4	PCACCUM			
8	8	PCLAST			
12	C	PCDELTA			
16	10	PCMAX			
20	14	PCDAYTIM		PCHRTIME	PCMINTIM
24	18	PCACOEFF			
28	1C	PCFLAG	PCADDR	PCFORM	PCFCODE
32	20	PCTYPE	PCNAME		
36	24	PCNAME			

Alphabetical List of Field in CACOUNT

Field	DEC	HEX
PCACCUM	0004	0004
PCACOFF	0024	0018
PCADDR	0029	001D
PCDAYTIM	0020	0014
PCDELTA	0012	000C
PCFCODE	0031	001F
PCFLAG	0028	001C
PCFORM	0030	001E
PCHRTIME	0022	0016
PCINCR	0000	0000
PCLAST	0008	0008
PCMAX	0016	0010
PCMINTIM	0023	0017
PCNAME	0033	0021
PCTYPE	0032	0020

Data Area Layout of CACOUNT

Offset	Bytes and Bit Patterns	Field	Description
0 (0)	4	PCINCR	Increment from last scan period
4 (4)	4	PCACCUM	Accumulated count since last scan period
8 (8)	4	PCLAST	Last counter reading
12 (C)	4	PCDELTA	Pulse counter filter value
16 (10)	4	PCMAX	Maximum pulses per scan
20 (14)	2	PCDAYTIM	Day of last good pulse counter data reading
22 (16)	1	PCHRTIME	Hour of last good pulse counter data reading
23 (17)	1	PCMINTIM	Minute of last good pulse counter data reading
24 (18)	4	PCACOEFF	Scale multipler for converting to engineering units
28 (1C)	1	PCFLAG	Error flags as follows:

	0...		Always zero for FORTRAN
	.1..		Point defined but unwired and unscanned
	..X.		Reserved
	...1 11..	PCERROR	Error conditions as follows:
		PCGOOD	000-good data
		PCROLL	001-rollover occurred during last calculaticn
		PCOVER	010-transfer overrun
		PCMISS	011-missing data
		PCREASON	100 - outside of reasonableness check
		PCBCH	101-BCH error on last scan
		PCOUTS	110-out of service (self)
		PCOUTO	111-out of service (other)
	....1.	PCERRIND	Alarm bit – 1 = alarm outstanding 0 = no alarm outstanding
29(1D)	1	PCADDR	Address of point as follows:
	111.	PCPOINT	Point address of counter (0-7) on card
	...1 1111	PCCARD	Card address to which point count belongs
30(1E)	1	PCFORM	Format of display
31(1F)	1	PCFCODE	Function code, customer assigned to be used in conjunction with Display Management
32(20)	1	PCTYPE	Counter type - customer assigned
33(21)	7	PCNAME	7-character name assigned to point for display purposes

CADBIND (DATA BASE INDICATOR ARRAY)

TOTAL SIZE: 6 bytes DSECT: CADBIND

CREATED BY: System generation

PURPOSE: To indicate the progress of scan synchronization to application programs.

POINTED TO BY: ADRIND field of data acquisition work area DOMTWRKA.

Storage Map of CADBIND

DEC	HEX		
0	0	LFCGO	USERGO
4	4	DBCLOSED	

Alphabetical List of Fields in CADBIND

Field	DEC	HEX
DBCLOSED	0004	0004
LFCGO	0000	0000
USERGO	0002	0002

Data Area Layout of CADBIND

Offset	Bytes and Bit Pattern	Field	Description
0(0)	2	LFCGO	When 1, LFC has new data to use
2(2)	2	USERGO	When 1, user has new data to use
4(4)	2	DBCLOSED	When 1, the update cycle is in progress

CADSPA (SCAN PERFORMANCE ARRAY)

TOTAL: 10 byte header + 8* No. of Scan ID's DSECT: DOMTSPAD

CREATED BY: System generation

PURPOSE: The purpose of CADSPA is to keep data from scan synchronization to provide data about scan performance.

POINTED TO BY: ADRSPA field of data acquisition work area DOMTWRKA

Storage Map of CADSPA

Header

DEC	HEX		
0	0	CADSPAON	TIMECOMP
4	4	TIMEDEV	LASTSCAN

Multiple Entry Portion

Item Name	Dec	Hex		
	0	0	SPAS7ID	SPASCAN
CADXXYYY*	2	2	TIMESCAN	
	4	4	TIMEARIV	

*XX is logical System/7 ID, YYY is scan ID

Alphabetical List of Fields in CADSPA

Header

Field	Dec	Hex
CADSPAON	0000	0000
LASTSCAN	0006	0006
TIMECOMP	0002	0002
TIMEDEV	0004	0004

Multiple Entry Portion

Field	DEC	HEX
SPAS7ID	0000	0000
SPASCAN	0001	0001
TIMEARIV	0004	0004
TIMESCAN	0002	0002

Data Area Layout of CADSPA

Header

Offset	Bytes and Bit Patterns	Field	Description
0(0)	2	CADSPAON	Indicator - if zero, only LASTSCAN and TIMESCAN are calculated. If non-zero, all values are calculated.
2(2)	2	TIMECOMP	Scan complete duration
4(4)	2	TIMEDEV	Damped mean absolute deviation
6(6)	4	LASTSCAN	Last basic scan cycle boundary when any scan arrived

Multiple Entry Portion

Offset	Bytes and Bit Patterns	Field	Description
0(0)	1	SPAS7ID	System/7 ID
1(1)	1	SPASCAN	Scan ID
2(2)	2	TIMESCAN	Duration of this scan
4(4)	4	TIMEARIV	Time when scan was scheduled using Special Real Time Operating System clock

TITLE: CALOGTIM (Performance Log Retrieval Control Array)

TOTAL SIZE: 86 decimal bytes

CREATED BY: System generation DSECT: LGTM

PURPOSE: To provide control information of performance log retrieval.

POINTED TO BY: ECVTAPLT field of EMSCVT

Storage Map of CALOGTIM

DEC	HEX			
0	0	LGTMCNT	LGTMSPR1	LGTMFLAG
4	4	LGTMLSCT		
8	8	LGTMRPCT		
12	C	LGTMRECT		
16	10	LGTMENCT		
20	14	LGTMRETT		
24	18			
28	1C	LGTMT1		
32	20	LGTMT2		
36	24			
40	28	LGTMT3		
44	2C	LGTMT4		
48	30			
52	34	LGTMT5		
56	38	LGTMT6		
60	3C			
64	40	LGTMT7		
68	44	LGTMT8		
72	48			
76	4C	LGTMT9		
80	50	LGTMT10		
84	54			

Alphabetical List of Fields in CALOGTIM

Field	Dec	Hex	Field	Dec	Hex
LCTMCNT	0	0	LGTMT10	80	50
LGTMENCT	16	10	LGTMT2	32	20
LGTMFLAG	3	3	LGTMT3	38	26
LGTMLSCT	4	4	LGTMT4	44	2C
LGTMRECT	12	C	LGTMT5	50	32
LGTMRETT	20	14	LGTMT6	56	38
LGTMRPCT	8	8	LGTMT7	62	3E
LGTMSPR1	2	2	LGTMT8	68	44
LGTMT1	26	1A	LGTMT9	74	4A

Data Area Layout and CALOGTIM Array

Offset	Bytes and Bit Patterns	Field	Description
0	2	LGTMCNT	Count of entries T1 - T10
2	1	LGTMSPR1	Spare
3	1	LGTMFLAG	Flag byte
	1.......	LGTMRAMM	On = retrieval active, minus mode
	.1......	LGTMRAPM	On = retrieval active, plus mode
	..1.....	LGTMPTM	On = PTIME sent
	....1...	LGTMAUIP	On = CALOGTIM update in progress
	.....1..	LGTMNPRD	On = not processed

	......1.	LGTMCNCL	On = retrieval canceled
	1	LGTMRI	On = retrieval active
4	4	LGTMLSCT	logging suspension counter
8	4	LGTMRPCT	Retrieval count, plus mode
12	4	LGTMRECT	Retrieval count, all modes
16	4	LGTMENCT	No. of logged array entries
20	6	LGTMRETT	Time of currently retrieved log record
26	6	LGTMT1	Mark time no. one
32	6	LGTMT2	Mark time no. two
38	6	LGTMT3	Mark time no. three
44	6	LGTMT4	Mark time no. four
50	6	LGTMT5	Mark time no. five
56	6	LGTMT6	Mark time no. six
62	6	LGTMT7	Mark time no. seven
68	6	LGTMT8	Mark time no. eight
74	6	LGTMT9	Mark time no. nine
80	6	LGTMT10	Mark time no. ten

CANALOG (ANALOG DATA ARRAY)

TOTAL SIZE: 20 bytes/analog value DSECT: ANALOG

CREATED BY: System generation

PURPOSE: This array contains all of the analog data for all remote control blocks in remote order.

POINTED TO BY: Each group of analog points for each remote control block is pointed to by the RCBANAL pointer of the remote control block DSECT RCBD.

Storage Map of CANALOG

DEC	HEX				
0	0	ANLVALUE			
4	4	ANDAQFLG	ANXDUCER	ANFLG1	ANFLG2
8	8	ANHWARN		ANLWARN	
12	C	ANACOEFF			
16	10	ANBCOEFF			

Alphabetical List of Fields in CANALOG

Field	DEC	HEX
ANACOEFF	0012	000C
ANBCOEFF	0016	0010
ANDAQFLG	0004	0004
ANFLG1	0006	0006
ANFLG2	0007	0007
ANHWARN	0008	0008
ANLVALUE	0000	0000
ANLWARN	0010	000A
ANXDUCER	0005	0005

Data Area Layout of CANALOG

Offset	Bytes and Bit Patterns	Field	Description
0(0)	4	ANLVALUE	Latest scanned value in floating point
4(4)	Byte	ANDAQFLG	Flag byte as follows:
	1...		Always zero. Used to ensure the halfword is always positive.
	.1..	ANACTIVE	0 - point wired and scanned
			1 - point defined but unwired and unscanned
	..1.		0 - no limit checking to be done
			1 - limit checking performed
	...1 11..	ANEGOOD	000 - good data
		ANEWARN	001 - exceeded warning limits
		ANEOPER	010 - exceeded operating limits
		ANEMISS	011 - missing data
		ANEOSCL	100 - offscale
		ANEBCH	101 - BCH error on last scan
		ANEOFLS	110 - point out of service
		ANEOFLO	111 - terminal out of service
	11	ANRTNCNT	Return within limits count
5(5)	1	ANXDUCER	Relative position in the special transducer limit table
6(6)	1	ANFLG1	These two bytes are used together to form the flags and offset as follows:
7(7)	1	ANFLG2	
	1...	ANALARM	Alarm bit:
			0 = no alarm

Offset	Bytes and Bit Patterns	Field	Description
			1 = alarm outstanding
	.1..	ANNOALRM	Alarmability bit 0 = alarm generated 1 = alarm never generated
	..1.	ANUSRCON	User conversion flags - when non-zero can be used to direct the processing of a user conversion routine
	...1	ANWALLBD	When non-zero indicates this point is a wallboard point.
	.... 1...	ANALARMA	Outstanding alarm has been acknowledged
	.... .1..	ANOLIMHL	Alarm indicator 0 = low and 1 = high.
	.... ..1.	ANRESERV	Reserved
	Bits 7-15		Offset into analog names list. The address of the names list is in the RCB.
8 (8)	2	ANHWARN	Analog high warning limit
10 (A)	2	ANLWARN	Analog low warning limit
12 (C)	4	ANACOEFF	A conversion coefficient in floating point
16 (10)	4	ANBCOEFF	B conversion coefficient in floating point

CANAME (ANALOG NAMES LIST ARRAY)

TOTAL SIZE: 12 bytes/analog value DSECT: ANAMELST

CREATED BY: System generation

PURPOSE: This array in remote order contains the names list data associated with each analog point.

POINTED TO BY: Each group of names in this array for each remote control block is pointed to by the RCBNAMEL pointer of the remote control block DSECT RCBD.

Storage Map of CANAME

DEC	HEX				
0	0	ANFLG3	ANFLG4	ANFORM	ANFCODE
4	4	ANTYPE	ANNAME		
8	8				

Alphabetical List of Fields in CANAME

Field	Dec	Hex
ANFCODE	0003	0003
ANFLG3	0000	0000
ANFLG4	0001	0001
ANFORM	0002	0002
ANNAME	0005	0005
ANTYPE	0004	0004

Data Area Layout of CANAME

Offset	Bytes and Bit Patterns	Field	Description
0(0)	Byte	ANFLG3	Flag byte as follows:
	1...		Reserved
	.11.	ANSCANCL	Remote scan class to which data point belongs
	...1 1111	ANCADDR	Card address to which data point belongs
1(1)	Byte	ANFLG4	Flag byte as follows:
	1111		Reserved
	 1111	ANPADDR	Point address on ANCADDR card
2(2)	1	ANFORM	Bits 0-3 indicate index into units table CAUNIT. Bits 4-7 indicate number of decimal points used to display value.
3(3)	1	ANFCODE	Device function code
4(4)	1	ANTYPE	Analog type
5(5)	7	ANNAME	Seven-character point name

CAPDC (PDC Commands for PDC control)

TOTAL SIZE: Variable. There is one entry in this array for each status point that may be controllable from the System/370. If the point is attached to the System/7 (local), the entry is 15 bytes long. If the point is attached to a 3707 (remote), the entry is 13 bytes long.

CREATED BY: System Generation

PURPOSE: This array contains the PDC command information which is incorporated in the transaction code X'06' message to the System/7.

Pointed to By: This array is accessed using the GETARRAY or GETITEM macros.

NOTE: The item names for this array consit of the name of the data base item (The status item name) which is 7 characters long followed by a character 'C'.

Storage Map of CAPDC

The CAPDC consists of either 13 or 15 byte entries depending on whether the device to which the entry applies is remote or local.

Data Area Layout of CAPDC

The entries for this array are described in the Communications Formats section of this manual under transaction code X'06', item 3. These entries are exactly as they appear in the message but do not contain the X'8D', end of message, character, and the first byte of each entry contains the CPU ID which currently has control. This first byte precedes the text portion of the transaction code X'06' message.

CAPLNAME (SELECTED ANALOG POINT NAMES FOR PERFORMANCE LOGGING)

TOTAL SIZE: 4 byte header plus 16 bytes for each allocated entry

DSECT: APLN (header), APLNE (entries)

CREATED BY: System generation

PURPOSE: The analog point log array provides 8-character analog point names which are resolved at System/370 Energy Management System initialization to address pointers to analog data and names list data associated with the analog name. The array is formatted with 2 bytes containing the number of provided analog names followed by two bytes of the number of 16 bytes entries with no provided names. The purpose of this array is to identify those analog points which are to be written to the CAPTLOG array or cyclic intervals. The 8-character name is maintained along with the resolved addresses so that during a warm start the addresses may be re-resolved based on the new data base generation. This is necessary because this array is subject to realtime deletion and addition of analog item names.

POINTED TO BY: The start of CAPLNAME is pointed to by the ECVTAPLN field of the EMSCVT.

Storage Map of CAPLNAME

DEC	HEX		
0	0	APLNVALD	APLNVOID
4	4	APLNENM	
12	C	APLNEAAD	
16	10	APLNENLA	
20	14	APLNENM	

16-byte entries repeated for each valid and each void entry.

Alphabetical List of Fields in CAPLNAME Array

Field	Dec	Hex
APLNEAAD	12	C
APLNENLA	16	10
APLNENM	4	4
APLNVALD	0	0
APLNVOID	2	2

Data Area Layout of CAPLNAME

Offset	Bytes and Bit Patterns	Field	Description
0 (0)	2	APLNVALD	Number of valid entries
2 (2)	2	APLNVOID	Number of void entries
4 (4)	8	APLNENM	Analog point name
12 (C)	4	APLNEAAD	Address pointer to analog data in CANALOG array for point name
16 (10)	4	APLNENLA	Address pointer to names list data in CANAME array for point name

CAPTLOG (ARRAY OF SELECTED ANALOG POINT DATA AND ASSOCIATED NAMES LIST DATA)

TOTAL SIZE: 4 bytes of scan time plus (CANALOG DSECT plus names list CANAME DSECT lengths) for each entry allocated in the CAPLNAME array.

CREATED BY: System generation

PURPOSE: This array holds the selected analog data and associated names list data prior to logging. The array is demand logged following each scan cycle. The purpose of this array is to provide the means of generating a performance log based on selected analog points.

POINTED TO BY: The start of CAPTLOG is pointed to by the ECVTAPLL field of the EMSCVT.

Storage Map of CAPTLOG

DEC	HEX		
0	0	Scan time	
4	4	Analog data DSECT	This area is repeated for the number of total entries in the CAPLNAME array
4+length if analog DSECT	4+length if analog DSECT	names list data DSECT	

Data Area Layout of CAPTLOG

Offset	Bytes and Bit Patterns	Field	Description
0 (0)	4		Scan time
4 (4)	Length of CANALOG		Analog data for selected name
4+length of CANALOG (4+length of CANALOG)	Length of CANAME		Name list data for selected name

CARCB (REMOTE CONTROL BLOCK DATA ARRAY)

TOTAL SIZE: 48 bytes/remote control block DSECT: RCBD

CREATED BY: System generation

PURPOSE: This array contains all of the remote control blocks. Each control block is an item named CIR XX YYY where XX is System/7 ID and YYY is remote station ID.

POINTED TO BY: A list of pointers to each remote control block is pointed to by the S7CTRCBA field of the S7CT.

Storage Map of CARCB

DEC	HEX				
0	0	RCBACT	RCBS7I	RCBS7II	RCBAA
4	4	RCBID	RCBDROP	RCBLINE	
8	8	RCBINDS	RCBNAME		
12	C				
16	10				
20	14		RCBDISP		
24	18				
28	1C		RCBNAMEL		
32	20	RCBANN		RCBPCN	RCBSN
36	24	RCBX1	RCBPC		
40	28	RCBX2	RCBANAL		
44	2C	RCBX3	RCBSTAT		

Alphabetical List of Fields in CARCB

Field	DEC	HEX
RCBAA	0003	0003
RCBACT	0000	0000
RCBANAL	0041	0029
RCBANN	0032	0020
RCBDISP	0021	0015
RCBDROP	0005	0005
RCBID	0004	0004
RCBINDS	0008	0008
RCBLINE	0006	0006
RCBNAME	0009	0009
RCBNAMEL	0029	001D

Field	Dec	Hex
RCBPC	0037	0025
RCBPCN	0034	0022
RCBS7I	0001	0001
RCBS7II	0002	0002
RCBSN	0035	0023
RCBSTAT	0045	002D
RCBX1	0036	0024
RCBX2	0040	0028
RCBX3	0044	002C

Data Area Layout of CARCB

Offset	Bytes and Bit Patterns	Field	Description
0 (0)	1	RCBACT	Count of outstanding alarms
1 (1)	1	RCBS7I	ID of System/7 to which remote station is attached
2 (2)	1	RCBS7II	ID of next System/7 in path to remote station
3 (3)	1	RCBAA	Access area to which remote station belongs
4 (4)	1	RCBID	Unique ID by System/7 for this remote station
5 (5)	1	RCBDROP	Line drop number
6 (6)	2	RCBLINE	Line number to which remote station is attached
8 (8)	Byte	RCBINDS	Flag byte as follows:
	1...	RCBACK	General alarm acknowledged if on
	.1..	RCBTYPE	On - manual data; Off - live data
	..1.	RCBARM	Device on remote station is armed if on
	...1	RCBSSRV	On - out of service (self)
	.... 1...	RCBOSRV	On - out of service (other)
	XXX		Reserved
9 (9)	12	RCBNAME	Substation/remote station name
21 (15)	8	RCBDISP	Name of display associated with remote station
29 (1D)	3	RCBNAMEL	Address of analog names list
32 (20)	2	RCBANN	Count of analog data on remote station
34 (22)	1	RCBPCN	Count of pulse counter data on remote station

35(23)	1	RCBSN	Count of status groups on remote station
36(24)	1	RCBX1	Reserved
37(25)	3	RCBPC	Address of pulse counter data for remote station
40(28)	1	RCBX2	Reserved
41(29)	3	RCBANAL	Address of analog data for remote station
44(2C)	1	RCBX3	Reserved
45(2D)	3	RCBSTAT	Address of status data for remote station

CASCAN (SCAN STATUS ARRAY)

TOTAL SIZE: 4 bytes plus 8 bytes for each scan ID defined

CREATED BY: System generation DSECT: SCAN

PURPOSE: This array contains all of the system scan IDs. Each entry contains information about the scan for display and update purposes.

POINTED TO BY: ECVTSCAN field of the EMSCVT.

Storage Map of CASCAN

DEC	HEX			
0	0	SCAN#IDS		
4	4	SCANID	SCANFLAG	SCANFREQ
8	8	SCANOFST		(RESERVED)

Alphabetical List of Fields in CASCAN

Field	Dec	Hex
SCANFLAG	0005	0005
SCANFREQ	0006	0006
SCANID	0004	0004
SCANOFST	0008	0008
SCAN#IDS	0000	0000

Data Area Layout of CASCAN

Offset	Bytes and Bit Patterns	Field	Description
0(0)	4	SCAN#IDS	Number of scan IDs in this array.
4(4)	1	SCANID	Scan ID
5(5)	1	SCANFLAG	Flag byte as follows:
	XXXX X...	SCANRES	Reserved

	.... .XX.	SCANMODE	00=Normal mode 01=Initial mode 10=Emergency mode
	X	SCANACT	0=Scan active 1=Scan inactive
6 (6)	2	SCANFREQ	Scan frequency
8 (8)	2	SCANOFST	Scan offset
10 (A)	2	Reserved	

CASCHART (Stripchart Array)

TOTAL SIZE: 256 bytes DSECT: STRPCHRT

CREATED BY: System Generation

PURPOSE: This array contains information about up to 16 recorders and is used for determining which are available and which are in use.

Pointed to By: ECVTSTCH field of EMSCVT

Storage Map of CASCHART

DEC	HEX		
0	0	CHFLAGS	CHADDR
4	4	CHSCALEA	CHSCALEB
8	8	CHNAME	
12	C		CHTYPE
16	10		
252	FC		

Alphabetical List of Fields in CASCHART

Field	DEC	HEX
CHADDR	0001	0001
CHFLAGS	0000	0000
CHNAME	0008	0008
CHSCALEA	0004	0004
CHSCALEB	0006	0006
CHTYPE	0015	000F

Data Area Layout of CASCHART

Offset	Bytes and Bit patterns	Field	Description
0(0)	byte	CHFLAGS	Flags as follows:
	1...	CHNOTWR	On - recorder is not wired
	.1..	CHACTIVE	On - recorder is turned on
	..1.	CHTIME	On - time mark option in effect
	...1	CHANALG	On - analog value being recorded
	.... 1...	CHPC	On - pulse counter value being recorded
	.... .1..	CHINP	On - stripchart command in progress
1(1)	3	CHADDR	Address of point in data base
4(4)	2	CHSCALEA	A scale factor in hex
6(6)	2	CHSCALEB	B scale factor in hex
8(8)	7	CHNAME	Name of point as it appears in the data base
15(F)	1	CHTYPE	Type code for point

CASPECLM (SPECIAL TRANSDUCER LIMIT ARRAY)

TOTAL SIZE: 4 bytes/special transducer type analog points

CREATED BY: System generation

PURPOSE: The special transducer limits are used for additional offscale checking of certain analog transducer types. A typical transducer supplies an input voltage in the range -5 to +5 volts which is applied to the analog to ditigal converter (ADC). The usual ADC range is -5 to +5 volts. If an input voltage of 6 is applied to the ADC, and ADC offscale error will be detected and bit 15 of the input value turned on. Some transducers provide input voltages in other ranges, for example, -4 to +4 volts. If a +4.5 volt signal comes from this transducer, it is clear the transducer is offscale, but the 4.5 volts will be converted by the terminal ADC. The special transducer limits provide a software method of detecting this exception. In this example, the limits would be 4/5 of the +32767 ADC high value; that is, +26214. Each entry consists of twc halfword values in ADC format. Bit position 16 is the implied binary point. A maximum of 255 special limits is possible.

POINTED TO BY: The start of CASPECLM is pointed to by the ECVTSPLM field of the EMSCVT. The ANXDUCER field of CANALOG contains the relative position in the special transducer limit table for each specified analog point.

Storage Map of CASPECLM

DEC	HEX	
0	0	Low Limit n
2	2	High Limit n

Data Area Layout of CASPECLM

Offset	Bytes and Bit Patterns	Field	Description
0 (0)	2		Special transducer low limit
2 (2)	2		Special transducer high limit

CASTATUS (STATUS DATA ARRAY)

TOTAL SIZE: 198 bytes/group DSECT: STATD

CREATED BY: System generation

PURPOSE: This array contains the latched and unlatched status data items for the local and remote status points. Each group contains 16 data items and a header. The item name for the status point is the seven byte user defined name.

POINTED TO BY: The remote control block (RCB) contains a pointer to the status data for that remote terminal in the RCBSTAT field.

Storage Map of CASTATUS

DEC	HEX				
0	0	STATL		STATS	
4	4	STATIND		STIND1	STIND2
8	8	STIND3	STDTYP	STNAME	
12	C				
16	10		STFCODE		
184	B8			STIND1	STIND2
188	BC	STIND3	STDTYP	STNAME	
192	C0				
196	C4		STFCODE		

Alphabetical List of Fields in CASTATUS

Field	Dec	Hex	
STATIND	0004	0004	
STATL	0000	0000	
STATS	0002	0002	
STDTYP	0008	0008	
STFCODE	0016	0010	Displacements true only for
STIND1	0006	0006	first data array entry.
STIND2	0007	0007	
STIND3	0008	0008	
STNAME	0010	000A	

Data Area Layout of CASTATUS

Offset	Bytes and Bit Patterns	Field	Description
0 (0)	2	STATL	Latched bits for latched data, zero for unlatched data
2 (2)	2	STATS	Status bits
4 (4)	2	STATIND	Status indicator
	U...	STATTYP	Type of status data; on - latched, off - unlatched
	.VVV VVVV	STATADD	Address of status group
	VVVV VVVV		5 bits for remote; 15 bits for local
6 (6)	1	STIND1	Status indicators
	1...	STALRM	Alarm outstanding - on
	.1..	STALRB	Not alarmable - on
	..1.	STARM	Device armed - on
	...1	STTAG	Device tagged - on
	.... 1...	STEXEC	Device executing - on
	.... .1..	STCONT	Not controllable - on
	.... ..1.	STSTAT	Status of device
	1	STCOS	Change of status expected - on
7 (7)	1	STIND2	Indicators
	1111	STTYPE	Type of device
	0000		- generator
	0001		- reserved
	0010		- switch
	0011		Motor-operated switch
	0100		Breaker
	0101		Reserved
	0110		Reserved
	0111		Reserved
	1000		Tap changing transformer type 1
	1001		Tap changing transformer type 2

Offset	Bytes and Bit Patterns	Field	Description
	1010		Tap changing transformer type 3
	1011		Reserved
	1100		Reserved
	1101		Reserved
	1110		Reserved
	1111		Reserved
	.... 1...	STMAN	Device is manual-on
	.... .111	STERROR	Error conditions
8 (8)	1	STIND3	Indicators as follows:
	1111		Type flags as follows:
	1...		On - 3707 DO
	.1..		On - local pulse DO
	..1.		On - local non-pulse DO
	...1		On - 3707 DI
			Off - local DI
	.... 11..		Reserved
	.... ..1.	STALARMA	On - alarm acknowledged
	1	STWALLBD	On - point is on wallboard
9 (9)	1	STDTYP	Device type code
10 (A)	7	STNAME	Device name
17 (11)	1	STFCODE	Function code

CASTLOG (Status log array)

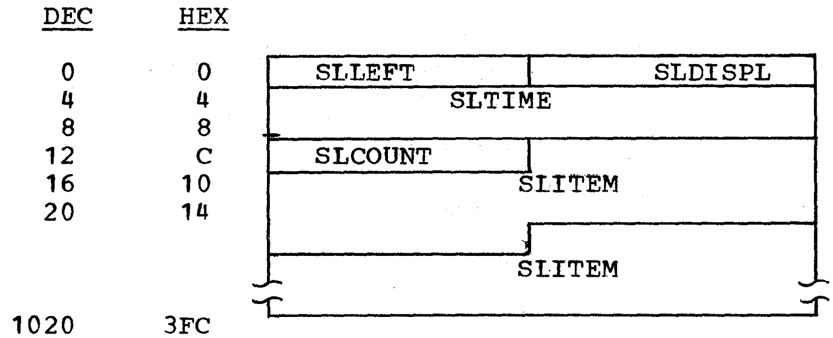
TOTAL SIZE: 1024 bytes

CREATED BY: System generation

PURPOSE: This array contains the status changes which occur during operation of the System/370 Energy Management System. These changes reflect device control operations and alarm conditions. This array is logged to STLOG when full.

POINTED TO BY: EMSCVT - ECVTSLOG field of the TMSCVT

Storage Map of CASTLOG



Alphabetical List of Fields in CASTLOG

Field	DEC	HEX
SLCOUNT	0012	000C
SLDISPL	0002	0002
SLITEM	0014	000E
SLLEFT	0000	0000
SLTIME	0004	0004

Data Area Layout of CASTLOG

Offset	Bytes and Bit Patterns	Field	Description
0 (0)	2	SLLEFT	Number of bytes unused in array
2 (2)	2	SLDISPL	Number of bytes used in array
4 (4)	8	SLTIME	Date and time change of status occurred
12 (C)	2	SLCOUNT	Number of status changes grouped under the SLTIME field
14 (E)	12	SLITEM	The status item which changed

Note: The SLTIME and SLCOUNT fields are repeated in the array every time a new group of status changes are passed. The SLITEM field is repeated with the SLTIME group for the count in SLCOUNT.

CASYS (SYSTEM GENERATED OPTIONS ARRAY)

TOTAL SIZE: 75 bytes DSECT: DOMSYSG

CREATED BY: System generation

PURPOSE: This array contains the system generated options chosen by the user for supervisory control and data acquisition. It is used by all programs which reference the variable user-defined options.

POINTED TO BY: A pointer to the array is found in the EMSCVT in the ECVTOSPT field.

Storage Map of CASYS

DEC	HEX				
0	0	DOMWARN	DOMOFFS	DOMLIMT	DOMMISS
4	4	DOMBCH	DOMDEVA	DOMDEVC	DOMTAGN
8	8	DOMFLG1	DOMFLG2	DOMEXOUT	DOMELOG
12	C	DOMDCTS	DOMGAA	DOMAS7	DOMPULSE
16	10	DOMBSC			
20	14	DOMMAST	DOMTODF	DOMGEA	DOMADL
24	18	DOMWB7	DOM370ID	DOMSIO	
28	1C	DOMXCE		DOMEDEAD	DOMDEVB
32	20	DOMUSCUR			
36	24				
40	28	DOMPCFLT			
44	2C	DOMSC7ID	DOMQLPVC	DOMQLALR	RESERVED
48	30	DOMMDAA			
52	34				
56	38	DOMMDFC			
60	3C				
64	40	DOMSIOCT		DOMRLSLV	
68	44	DOMMODLV		DOMCCLV	
72	48	DOMSYSHW	DOMMOSS	DOMMOSH	

Alphabetical List of Fields in CASYS

DOMADL	0023	0017
DOMAS7	0014	000E
DOMBCH	0004	0004
DOMBSC	0016	0010
DOMCCLV	0070	0046
DOMDCTS	0012	000C
DOMDEVA	0005	0005
DOMDEVB	0031	001F
DOMDEVC	0006	0006
DOMEDEAD	0030	001E
DOMELOG	0011	0008
DOMEXOUT	0010	000A
DOMFLG1	0008	0008
DOMFLG2	0009	0009
DOMGAA	0013	000D
DOMGEA	0022	0016
DOMLIMT	0002	0002
DOMMAST	0020	0014
DOMMDAA	0048	0030
DOMMDFC	0056	0038
DOMMISS	0003	0003
DOMMODLV	0068	0044
DOMMOSH	0074	004A
DOMMOSS	0073	0049
DOMOFFS	0001	0001
DOMPCFLT	0040	0028
DOMPULSE	0015	000F
DOMQLALR	0046	002E
DOMQLPVC	0045	002D
DOMRLSLV	0066	0042
DOMSC7ID	0044	002C
DOMSIO	0026	0020
DOMSIOCT	0064	0040
DOMSYSHW	0072	0048
DOMTAGN	0007	0007
DOMTODF	0021	0015
DOMUSCUR	0032	0020
DOMWARN	0000	0000
DOMWB7	0024	0018
DOMXCE	0028	001C

Field	DEC	HEX
DOM370ID	0025	0019

Data Area Layout of CASYS

Offset	Bytes and Bit Patterns	Field	Description
0 (0)	1	DOMWARN	Entity attribute for warning alarms
1 (1)	1	DOMOFFS	Entity attribute for offscale alarms
2 (2)	1	DOMLIMT	Entity attribute for out of limit alarms
3 (3)	1	DOMMISS	Entity attribute for missing data alarms

Offset	Bytes and Bit Patterns	Field	Description
4(4)	1	DOMBCH	Entity attribute for BCH check alarms
5(5)	1	DOMDEVA	Entity attribute for uncommanded change of status alarms
6(6)	1	DOMDEVC	Entity attribute for device closed (commanded)
7(7)	1	DOMTAGN	Entity attribute for tag/manual mode
8(8)	Byte 0	DOMFLG1	Indicators
	1...	DOMWN	Automatic delete for warning alarms if on
	.1..	DOMOF	Automatic delete for offscale alarms if on
	..1.	DOMOUT	Automatic delete for out of limits alarms if on
	...1	DOMMS	Automatic delete for missing data alarms if on
	.... 1...	DOMBC	Automatic delete for BCH alarms if on
	.... .1..	DOMDAL	Automatic delete for device open/trip alarms if on
	.... ..1.	DOMCVN	If on, conversion required
	1	DOMEXT	If on, external timer present
9(9)	Byte 1	DOMFLG2	Indicators
	1...	DOMFLC	If on, AGC system generated
	.1..	DOMEDC	If on, EDC system generated
	..1.	DOMOR	Automatic delete for PC rollover alarms if on
	...1	DOMRS	Automatic delete for PC reasonableness alarms if on
	 XXXX		Reserved
10(A)	1	DOMEXOUT	Execution time-cut interval for device control
11(B)	1	DOMELOG	Rate of update for events log display
12(C)	1	DOMDCTS	Number of DCT entries in index table
13(D)	1	DOMGAA	Routing code for general alarms typer

Offset	Bytes and Bit Patterns	Field	Description
14 (E)	1	DOMAS7	ID of local System/7 for audible alarms
15 (F)	1	DOMPULSE	Pulse frequency for time synchronization
16 (10)	4	DOMBSC	Basic scan cycle
20 (14)	1	DOMMAST	ID of System/7 which is master time source
21 (15)	1	DOMTODF	Time of day format
22 (16)	1	DOMGEA	Routing code for general events typer
23 (17)	1	DOMADL	Administrative message length
24 (18)	1	DOMWB7	System/7 which drives wallboard
25 (19)	1	DOM370ID	ID of System/370
26 (1A)	2	DOMSIO	Start I/O appendage qualifier
28 (1C)	2	DOMXCE	Abnormal end appendage qualifier
30 (1E)	1	DOMEDEAD	Number of records in event dead zone
31 (1F)	1	DOMDEVB	Entity attribute byte for device open (uncommanded)
32 (20)	8	DOMUSCUR	Name of user supplied load module to be used as a user exit to monitor outgoing transactions to System/7s
40 (28)	4	DOMPCFLT	Filter value for PC extrapolation
44 (2C)	1	DOMSC7ID	System/7 which drives stripchart recorders
45 (2D)	1	DOMQLPVC	Queue length for device control task
46 (2E)	1	DOMQLALR	Queue length for alarm management task
47 (2F)	1	DOMTPNTP	Position in hierarchy at sysgen X'F0' = TOP, X'00' = NOT TOP
48 (30)	8	DOMMDAA	Power system operator console access area name
56 (38)	8	DOMMDFC	Power system operator console function area name
64 (40)	2	DOMSIOCT	Start I/O loop limit count
66 (42)	2	DOMRLSLV	Program product release level

Offset	Bytes and Bit Patterns	Field	Description
68(44)	2	DOMMODLV	Program product mod level
70(46)	2	DOMCCLV	User configuration level
72(48)	1	DOMSYSHW	Analog high limit percent
73(49)	1	DOMMOSS	Entity attribute byte for MOS-stuck error
74(4A)	1	DOMMOSH	Entity attribute byte for MOS-hardware error

CATIMES (TIMES TABLE ARRAY)

TOTAL SIZE: 28 bytes DSECT: CATIMESD

CREATED BY: System generation

PURPOSE: Central time table for reference purposes

POINTED TO BY: ECVTTIME field of the EMSCVT

Storage Map of CATIMES

Item Name	DEC	HEX	
DAYOFYER	0	0	Day of the year
STNDTIME	4	4	Standard time from System/7
SYSTIME	8	8	Power system time (STNDTIME-TIMERROR)
TIMERROR	12	C	Time deviation
SYSFREQ	16	10	System frequency
FASTTCF	20	14	Time correction frequency - Fast
SLOWTCF	24	18	Time correction frequency - Slow

Alphabetical List of Fields in CATIMES

F DEC	HEX	
DAYOFYER	0000	0000
FASTTCF	0020	0014
SLOWTCF	0024	0018
STNDTIME	0004	0004
SYSFREQ	0016	0010
SYSTIME	0008	0008
TIMERROR	0012	000C

Data Area Layout of CATIMES

Offset	Bytes of Bit Patterns	Field	Description
0(0)	4	DAY OF YR	Day of the year
4(4)	4	STNDTIME	Standard time from System/7 or System/370 time if no time from System/7
8(8)	4	SYSTIME	Power system time (Standard Time - time deviation)
12(C)	4	TIMERROR	Time deviation input from operator
16(10)	4	SYSFREQ	System-generated system frequency
20(14)	4	FASTTCF	System-generated fast time correction frequency
24(18)	4	SLOWTCF	System-generated slow time correction frequency

CATYPES (TYPES TABLE ARRAY)

TOTAL SIZE: 4096 bytes (16 bytes/entry) DSECT: TYPESD

CREATED BY: System generation

PURPOSE: This array contains device type names defined by the user. Each device type name is sixteen characters long, and there is a maximum of 256 entries in the table. A one-byte device type number, which serves as an index into this table, is present in all sensor based data points.

POINTED TO BY: ECVTYPES field of the EMSCVT

Data Area Layout of CATYPES

Offset	Bytes and Bit Patterns	Field	Description
0(0)	16		16-character device type name

CAUNITS (UNITS TABLE ARRAY)

TOTAL SIZE 2048 bytes (8 bytes/entry) DSECT: UNITSD

CREATED BY: System generation

PURPOSE: This array contains unit names defined by customer. Each unit name is eight characters long, and there is a maximum of 256 entries in the table. This unit name can be applied to a data point by having a one-byte index into this table associated with this data point.

POINTED TO BY: No pointer to CAUNITS, Special Real Time Operating System GETARRAY is used to return the address of this array.

Data Area Layout of CAUNITS

Offset	Bytes and Bit Patterns	Field	Description
0(0)	8		8-character unit name

CAWBCONF (Wallboard configuration array)

TOTAL SIZE: 12 decimal bytes DSECT: WBCONF

CREATED BY: System generation

PURPOSE: This array contains the configuration states used in System/370 wallboard processing and the manual/auto switch name (converted to address) used to place the wallboard in the manual or automatic mode.

POINTED TO BY: ECVTWBCN field in the EMSCVT table.

STORAGE MAP OF CAWBCONF

DEC	HEX				
0	0	WBCMANAT or WBCMAPTA + WBCSPARE			
4	4				WBCNOST
8	8	WBCSTONEE	WBESTTWO	WBCSTHRE	WBCSTFOR

ALPHABETICAL LIST OF FIELDS IN CAWBCONF

Field	DEC	HEX	Field	DEC	HEX
WBCMANAT	0	0	WBCSTFOR	11	B
WBCMAPTA	0	0	WBCSTHRE	10	A
WBCNOST	7	7	WBCSTONE	8	8
WBCSPARE	4	4	WBCSTTWO	9	9

DATA AREA LAYOUT OF CAWBCONF

offset	Bytes and Bit Patterns	Field	Description
0(0)	7	WBCMANAT	Manual/Auto wallboard switch item name
0(0)	4	WBCMAPTA	Manual/Auto Wallboard Switch item address in data base
4(4)	3	WBCSPARE	Reserved for future use
7(7)	1	WBCNOST	No. of configuration states defined
8(8)	1	WBCSTONE	State one

	1.......	WBCRES1	Reserved for System generation
	.111....	WBCSTAT1	STATE One Command (bits 4, 5, 6, 7 reserved)
9 (9)	1	WBCSTTWO	State two (bit 0 reserved for sysgen)
	.111....	WBCSTAT2	State two command (bits 4, 5, 6, 7 reserved)
10 (A)	1	WBCSTHRE	State three (bit 0 reserved for sysgen)
	.111....	WBCSTAT3	State three command (bits 4, 5, 6, 7 reserved)
11 (B)	1	WBCSTFOR	State Four (bit 0 reserved for sysgen)
	.111....	WBCSTAT4	State four command (bits 4, 5, 6, 7 reserved)

CAWBNAME (WALLBOARD NAME ARRAY)

TOTAL SIZE: 2 byte header Each entry is 10 bytes + 2 bytes for each lamp address DSECT: WBNAME

CREATED BY: System generation

PURPOSE: This array defines each wallboard item in the data base. The item address is maintained, the configuration number for the item, the number of lamps for the item, and the lamp addresses assigned to the wallboard item.

POINTED TO BY: ECVTWBNA field in the EMSCVT table.

STORAGE MAP OF CAWBNAME

DEC	HEX	Header
0	0	
2	2	WBHDRENT

Multiple Entry Portion

DEC	HEX				
0	0	WBINAME (7 bytes) or WBIADR (4 bytes)			
4	4	WBISPAR4 (3 bytes)			WBIFLAG
8	8	WBICONF	WBILAMPN	WBILADD1	
12	c	WBILADD2		WBILADD3	
16	10				

ALPHABETICAL LIST OF FIELDS IN CAWBNAME

field	DEC	HEX	Field	DEC	HEX
WBHDRENT (header)	0	0	WBILADD2	12	C
WBIADR	0	0	WBILADD3	14	E
WBICONF	8	8	WBILAMPN	9	9
WBIFLAG	7	7	WBINAME	0	0
WBILADD1	10	A	WBISPAR4	4	4

DATA AREA LAYOUT OF CAWBNAME

offset	Bytes and Bit Patterns	Field	Description
0 (0)	2	WBHDRENT	(header) No. of items in array
0 (0)	7	WBINAME	Wallboard item name

0 (0)	4	WBIADR	Wallboard item address in data base (resolved during initialization)
4 (4)	3	WBISPAR4	Reserved for future use
7 (7)	1	WBIFLAG	Flag byte
	......1.	WBISTAT	Status item indicator
	1	WBIANAL	Analog item indicator
8 (8)	1	WBICONF	Configuration number
9 (9)	1	WBILAMPN	No. of lamps
10 (A)	2	WBILADD1	Address of first lamp (12 bits right justified)
12 (C)	2	WBILADD2	Address of second lamp (12 bits right justified)
14 (E)	2	WBILADD3	Address of third lamp (12 bits right justified)

CAnnRDAM* (RAW DATA ARRAY MAP)

TOTAL SIZE: 2-byte header plus 6* Number of IBM 3707 Remote Data Acquisition and Control Stations attached DSECT: RDAMAP

CREATED BY: System generation

PURPOSE: The purpose of CAnnRDAM is to provide a m... of the raw data array.

POINTED TO BY: S7CTRDAM field of the S7CT.

*nn is the logical System/7 ID. There is one raw data array map per System/7.

<u>Storage Map of CAnnRDAM</u>

<u>Header</u>

DEC	HEX		
0	0	RDA MAP SIZE	This field is two bytes long

<u>Multiple Entry Portion</u>

0	0	TERMHEDO
2	2	STATHEDO
4	4	PCHEDO

<u>Alphabetical List of Fields in CAnnRDAM</u>

Field	Dec	Hex
PCHEDO	0004	0004
STATHEDO	0002	0002
TERMHEDO	0000	0000

Data Area Layout of CAnnRDAM

Offset	Bytes and Bit Patterns	Field	Description
0(0)	2	TERMHEDO	Displacement to start of control station data in raw data array
2(2)	2	STATHEDO	Dispoacement to start of status data in raw data array
4(4)	2	PCHEDO	Displacement to start of pulse counter data in raw data array

DASTLOG (Temporary log array)

TOTAL SIZE: 1024 bytes

CREATED BY: System generation

PURPOSE: This array is located on a direct access device and is used as a temporary log until the CASTLOG is full and is logged to the STLOG. It is updated on every change of status (or group of changes of status). It is used to update the data base in the case of a warm start.

The DASTLOG is identical to the CASTLOG. See CASTLOG for storage map and data area layout.

TAS7COMM (SYSTEM/7 COMMUNICATION TABLE)

TOTAL SIZE: 6 + 2* number of System/7 unit IDs + number of local logical IDs *(6+ number of System/7 unit IDs)+2* number of remote logical IDs. DSECT: S7COMM

CREATED BY: System generation

PURPOSE: Provide information about the channel attached System/7s.

POINTED TO BY: ECVTCOMM field of the EMSCVT.

Note: This table is divided into four sections, three of which are of variable length. The length of each section is dependent upon the value contained in one of the fields of the fixed section. Each section will be described separately. The displacement of each section relative to the start of the table will be given through a formula. The displacement of each field within each section will be given relative to the start of the section. Each section follows the preceding section immediately.

Sectional Organization of TAS7COMM

Header Section

Unit DD Section

Logical ID Section

Remote ID Section

Storage Map of TAS7COMM - Header Section

Offset from start of table = 0 bytes
Total size this section = 6 bytes

DEC	HEX		
0	0	#LOCLID	#UNITS
4	4	#RMTLID	

Alphabetical List of Fields in Header Section

F	DEC	HEX
#LOCLID	0	0
#RMTLID	4	4
#UNITS	2	2

Data Area Layout - Header Section

Offset	Bytes and Bit Patterns	Field	Description
0(0)	2	#LOCLID	The number of logical ID channel attached
2(2)	2	#UNITS	The number of channel-attached System/7s
4(4)	2	#RMTLID	The number of remote logical IDs addressable by this leg of the hierarchy

Storage Map of TAS7COMM - Unit DD Section

Offset from start of table = 6 bytes
Size per entry - 2 bytes
Number of entries - number of System/7 unit IDs
Total size of section = (2* number of System/7 unit IDs) bytes

DEC	HEX	
0	0	UNITDD

Alphabetical List of Fields in Unit DD Section

Field	Dec	Hex
UNIT#1	0	0

Data Area Layout - Unit DD Section

Offset	Bytes and Bit Patterns	Field	Description
0 (0)	2	UNIT#1	Two character identifier, xx, to complete the DD names on the three JCL cards S7XXI, S7XXW, and S7XXW (one entry each for units defined)

Storage Map of TAS7COMM - Logical ID Section

Offset from start of table = 6 + (2*#UNITS) bytes
Size per entry = 6+#UNITS bytes
Number of entries = #LOCLID
Total size of section = #LOCLID * (6+#UNITS) bytes
one entry:

DEC	HEX				
0	0	COMMLID	COMMROUT	COMMLFLG	COMMBKUP
4	4	COMMRDA		COMMPFLG	

Alphabetical List of Fields - Logical ID Section

F	DEC	HEX
COMMBKUP	3	3
COMMLFLG	2	2
COMMLID	0	0
COMMPFLG	6	6
COMMRDA	4	4
COMMROUT	1	1

Data Area Layout - Logical ID Section

Offset	Bytes and Bit Patterns	Field	Description
0(0)	1	COMMLID	The local logical ID
1(1)	1	COMMROUT	Index (0 to (#UNITS-1) indicating which System/7 is primary
2(2)	1	COMMLFLG	Logical ID flags
	1...	ACTIVE	The logical ID has a primary, is active
	.1..	DISK	LID uses disk
	..1.	SCANNING	The logical ID is scanning
	...0	LIDISKST	The System/370 disk data set supporting the System/7 IPL, core load and disk load is operable
	...1		System/370 disk data set for System/7 support is in error
	.... 1...	BACKUP	The logical ID has a communicating backup
	.... .1..	MODESTOP	On if scan mode change to occur
	.... ..1.	NODEISUP	On if Logical ID has been commanded to enter hierarchy
	x	Reserved	
3(3)	1	COMMBKUP	Index (0 to (#UNITS-1)) indicating which System/7 is backup

Offset	Bytes and Bit Patterns	Field	Description
4 (4)	2	COMMRDA	Size of the raw data array received from this logical ID
6 (6)	1	COMMPFLG	Unit flags (one entry per unit defined)
	1...	ABLE	Unit available as backup or primary for this logical ID
	.1..	READY	Unit is in operational condition
	.0..		Unit not operational (has failed or not wired)
	..1.	IPLDME	Unit initialized program loaded with this logical ID
	...1	IPLDYOU	Unit initialized program loaded with some other logical ID
	.... 1...	IPLNGME	Unit is being IPL'd with this logical ID
	.... .1..	IPLNGYOU	Unit is being IPL'd with other logical ID
	.... ..1.	DISKLDME	Disk load processing complete for this logical ID
	x		Reserved

Storage Map of TAS7COMM - Remote ID Section

Offset from start of table - 6+(2*#UNITS) + #LOCLID*(6*#UNITS)
Size per entry = 2 bytes
Number of entries = #RMTLID
Total size of section = 2*#RMTLID

DEC	HEX		
0	0	COMMRLID	COMMLLID

Data Area Layout - Remote ID Section

Offset	Bytes and Bit Patterns	Field	Description
0 (0)	1	COMMRLID	Remote logical ID
1 (1)	1	COMMLLID	Local logical ID in path to remote logical ID

TAS7HIER (System/7 Hierarchy Array)

TOTAL SIZE: 4 + (12 * number of variable entries) DSECT: DOM7HIER

CREATED BY: System generation

PURPOSE: Provide information as to the structure of the hierarchy from the current level down

Pointed to By: ECVTHIER field of EMSCVT

NOTE: The format of this table is variable based upon the values contained in the first two fixed entries.

Storage Map of TAS7HIER Array
fixed header section

DEC	HEX	
0	0	HIER#ENT
2	2	HIER#LID

Alphabetical List of Fields

Field	DEC	HEX
HIER#ENT	0	0
HIER#LID	2	2

Data Area layout - fixed header section

Offset	Bytes	Field	Description
0	2	HIER#ENT	Number of variable entries following the fixed header section
2	2	HIER#LID	Number of local logical ID entries

Storage Map of TAS7HIER - Variable entry section

Offset from start of table = 4 bytes

Size per entry = 12 bytes

Number of entries = HIER#ENT

Total size of section = 12 * HIER#ENT

NOTE: The entries are ordered. All entries for local logical ID (see HIFR#LID) are listed first. All entries which have the same parent must be listed consecutively.

DEC	HEX				
0	0	HIERLID	HIERFLG1	HIERFLG2	HIERHOST
4	4	HIER#KID	HIERKID1		
8	8	HIERMOTH			

Alphabetical List of Fields - Variable entry section

Field	DEC	HEX
HIER#KID	4	4
HIERFLG1	1	1
HIERFLG2	2	2
HIERHOST	3	3
HIERKID1	5	5
HIERLID	0	0
HIERMOTH	8	8

Data Area Layout - variable entry section

Offset	Bytes and Bit patterns	Field	Description
0	1	HIERLID	Logical ID of System/7
1	1	HIERFLG1	Reserved
2	1	HIERFLG2	Reserved
3	1	HIERHOST	Logical ID of host System/370 or zero
4	1	HIER#KID	Number of children for this logical id or zero
5	3	HIERKID1	Address of the list of children Entries for this logical ID or zero
8	4	HIERMOTH	Address of the entry of the parent System/7 logical ID or zero

TABLES

This section describes the major tables used by the System/370 Energy Management System.

ALARM CONDITION TABLE

TOTAL SIZE: 448 bytes

CREATED BY: EMS initialization

PURPOSE: The alarm condition table contains the 14 character phrases which describe the conditions which caused the alarm. The table is created by reading messages DPP328I and DPP329I into an area. The table will contain a maximum of 32 entries. The user may add entries to describe additional alarm conditions.

POINTED TO BY: EMSCVT (ECVTCOND)

This table consists of thirty-two 14-byte entries. The alarm condition code in the alarm record, multiplied by 14, gives the displacement into the table. This table is used by the alarm processor and the alarm display processor. The entries in this table are found in messages DPP328I and DPP329I in the System/370 Energy Management System message data set (S370EMS.MSGFILE).

ALARM RECORD

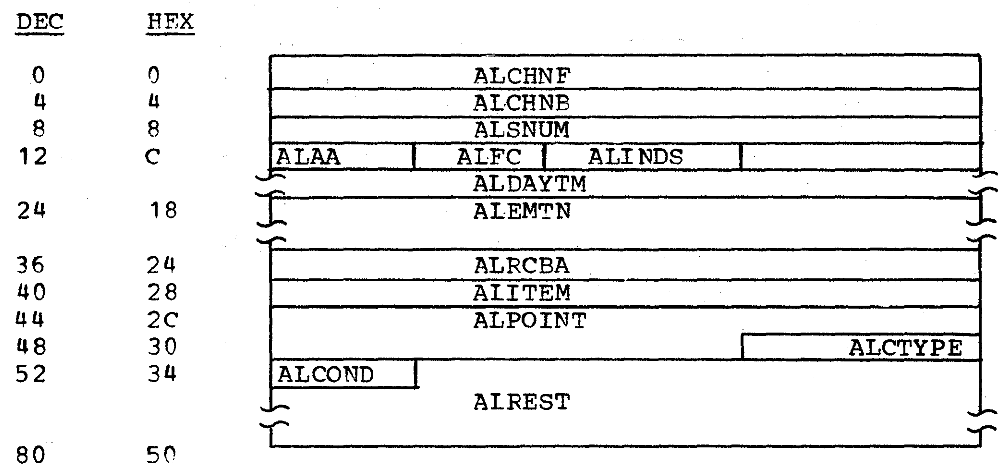
TOTAL SIZE: 84 bytes/active alarm DSECT: ALARMD

CREATED BY: Alarm management processing

PURPOSE: The alarm record contains information about the alarm which has occurred, such as the alarm condition, the access area, and the remote station to which it belongs.

POINTED TO BY: Two fields in the Access Area Table point to the alarm record chain - AAFIRST and AALAST point to the first and last alarm records in the chain for the access area. Each alarm record contains pointers to the previous and succeeding alarms: ALCHNB and ALCHNF, respectively.

Storage Map of The Alarm Record



Alphabetical List of Fields in Alarm Record

Field	Dec	Hex
ALAA	0012	000C
ALCHNB	0004	0004
ALCHNF	0000	0000
ALCOND	0052	0034
ALCTYPE	0051	0033
ALDAYTM	0015	000F
ALEMTN	0024	0018
ALFC	0013	000D
ALINDS	0014	000E
ALITEM	0040	0028
ALPOINT	0044	002C
ALRCBA	0036	0024
ALREST	0053	0035
ALSNUM	0008	0008

Data Area Layout of Alarm Record

Offset	Bytes and Bit Patterns	Field	Description
0 (0)	4	ALCHNF	Address of next alarm record in chain or zero
4 (4)	4	ALCHNB	Address of previous alarm record in chain or zero
8 (8)	4	ALSNUM	Alarm sequence number
12 (C)	1	ALAA	Access area code
13 (D)	1	ALFC	Function code
	Byte	ALINDS	Indicators
14 (E)	1...	ALTYPE	Type of alarm; on - message off - regular
	.1..	ALACKN	Acknowledged if on
	..1.	ALSTAT	Status item alarmed

Offset	Bytes and Bit Patterns	Field	Description
	...1	ALPC	Pulse counter alarmed
	.... 1...	ALANA	Analog alarmed
15(F)	9	ALDAYTM	Day and time alarm occurred (DDDHHMMSS)
24(18)	12	ALEMTN	Name of Energy Management System
36(24)	4	ALRCBA	Address of RCB
40(28)	4	ALITEM	Address of item in data base
44(2C)	7	ALPOINT	Name of point alarmed
51(33)	1	ALCTYPE	Type code
52(34)	1	ALCOND	Code for condition which caused alarm
53(35)	31	ALREST	

CEATAB (Control Element Address Table)

TOTAL SIZE: 36 bytes DSECT: CEATAB

CREATED BY: DOMCSOMD (5985 display sign on module)

PURPOSE: This area is for the use of tasks which require and use IBM 5985 Display Units. Each functional area is assigned a four-byte area which is initially zero. The general use of this table is to contain control element address pointers. There is one CEATAB generated for each 5985 display screen. There is a four-byte area reserved for the user; it is the first word of the CEATAB table.

POINTED TO BY: Display control element (DCE) DSECT DCEUSER field.

Storage Map of CEATAB

DEC	HEX	
0	0	CEAUSER
4	4	CEALARM
8	8	CEAEREQ
12	C	CEASCAN
16	10	CEACONF
20	14	CEADCT
24	18	CEASBDT
28	1C	CEAEDCM
32	20	CEALFCM

Alphabetic List of Fields in CEATAB

Field	Dec	Hex
CEACONF	0016	0010
CEADCT	0020	0014
CEAEDCM	0028	001C
CEAEREQ	0008	0008
CEALARM	0004	0004
CEALFCM	0032	0020
CEASBDT	0024	0018
CEASCAN	0012	000C
CEAUSER	0000	0000

Data Array Layout of CEATAB

Offset	Bytes and Bit Patterns	Field	Description
0(0)	4	CEAUSER	User area
4(4)	4	CEALARM	Detail alarm
8(8)	4	CEAEREQ	Event log
12(C)	4	CEASCAN	Scan control
16(10)	4	CEACONF	Power configuration
20(14)	4	CEADCT	Device control table
24(18)	4	CEASBDT	Sensor based data
28(1C)	4	CEAEDCM	Economic disPATCH control
32(20)	4	CEALFCM	Automatic Generation Control

DEVICE CONTROL TABLE (DCT) INDEX TABLE

TOTAL SIZE: 12* number of DCTs generated DSECT: DCTINDEX

CREATED BY: Supervisory control initialization

PURPOSE: This table is used to control the number of device conttrol actions taking place at one time and to ensure that more than one control action is not attempted on a particular device at the same time.

POINTED TO BY: A pointer to the table is found in the EMSCVT in the ECVTDCIX field.

Storage Map of DCT Index Table

DEC	HEX	
0	0	DCTIDEV
4	4	DCTISPR
8	8	DCTIADD

Alphabetical List of DCT Index Table

Field	Dec	Hex
DCTIADD	0008	0008
DCTIDEV	0000	0000
DCTISPR	0007	0007

Data Area Layout of DCT Index Table

Offset	Bytes and Bit Patterns	Field	Description
0(0)	7	DCTIDEV	Device item name
7(7)	1	DCTISPR	Reserved
8(8)	4	DCTIADD	Address of DCT entry

DEVICE CONTROL OPTIONS TABLE

TOTAL SIZE: 340 bytes

CREATED BY: EMS initialization

PURPOSE: This table is used by the device control processor to display the control action options which are open to the power system operator at the time he selects a device to be controlled.

POINTED TO BY: EMSCVT (ECVTOTBL)

This table contains thirty-four 10-byte entries which describe the possible control actions for each type of device. The first two entries in the table are TAG and UNTAG which apply to all devices. The rest of the table is divided into groups of two entries, each one pertaining to a type of device. The table is created by reading messages 331 and 332 into an area. The type of device is determined from the indicators in the status item (STTYPE). The value of these indicators, multiplied by 20, is the displacement into the table for the type of device after bypassing the tag/untag entries (+20). These values are as follows:

VALUE	TYPE
0	Generator
1	Reserved
2	Switch
3	Motor-operated switch
4	Breaker
5-7	Reserved
8	Tap changing transformer - type 1
9	Tap changing transformer - type 2
10	Tap changing transformer - type 3
11-15	Reserved

Storage Map Off Device Control Option Table

DEC	HEX				
0	0	GENERATOR	RESERVED	SWITCH	MOS
4	4	BREAKER	RESERVED		
8	8	TCT-I	TCT-II	TCT-III	
12	C	RESERVED			

DOMTEAD (EVENT ALARMS BUFFER)

TOTAL SIZE: 560 bytes; 6 bytes(HEADER)+10*56(EAENTRY)

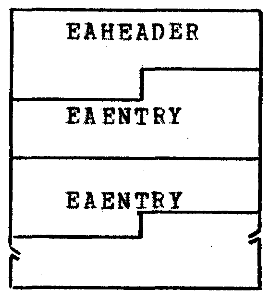
OPERATED BY: DOMCALR6

PURPOSE: Pass large numbers of events from alarms processor to events processor.

PROVIDED TO BY: PATCH PROBL

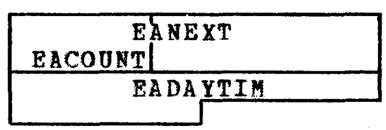
<u>Storage Map of DOMTEAD</u>

DEC	HEX
0	0
4	4
8	8
12	C
16	10
20	14
24	18
28	1C
558	140



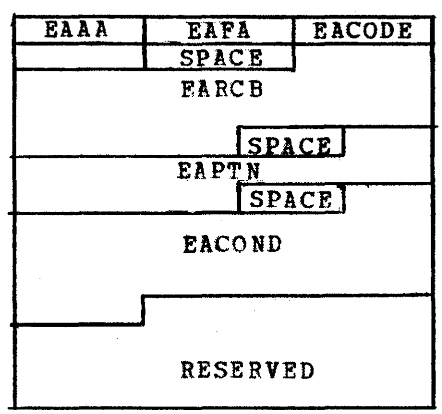
<u>Storage Map of EAHEADER</u>

DEC	HEX
0	0
4	4
8	8



<u>Storage Map of EAENTRY</u>

DEC	HEX
0	0
4	4
8	8
12	C
16	10
20	14
24	18
28	1C
32	20
36	24
40	28
44	2C
48	30
52	34



Alphabetical List of Fields - EAHEADED

Field	DEC	HEX
EACOUNT	0	0
EANEXT	0	0
EADAYTIM	4	4
EAHEADER	0	0

Alphabetical List of Fields - EAENTRY

Field	DEC	HEX
EAAA	0	0
EACODE	2	2
EACOND	27	1B
EAENTRY	0	0
EAFA	1	1
EAMSG	19	13
CAPTN	19	13
EARCB	6	6
EATEXT	2	2

Offset	Bytes and Bit Patterns	Field	Description
0 (0)	56	EAENTRY	Event entry
0 (0)	1	EAAA	Event access area
1 (1)	1	EAFA	Event function area
2 (2)	54	EATEXT	Event text
2 (2)	3	EACODE	Action code
6 (6)	12	EARCB	RCB name
19 (13)	37	EAMSG	Message text
19 (13)	7	EAPTN	Point name
27 (1B)	14	EACOND	Alarm condition
0 (0)	6	EAHEADER	Buffer header
0 (0)	1	EACOUNT	Number entries this buffer
0 (0)	4	EANEXT	Address next buffer or zero
4 (4)	6	EADAYTIM	Day and time in packed decimal

DOMTSRDA (RAW DATA ARRAY FORMAT)

TOTAL SIZE: 28 byte header pulse variable length, depending on control stations

CREATED BY: System generation DSECT: DOMTSRDA

PURPOSE: The purpose of DOMTSRDA is to provide the format of the raw data array that is passed to scan processing.

POINTED TO BY: The pointer is passed with the queue to scan processing. The raw data array map CAnnRDAM points to the control station portions of the raw data array.

Storage Map of DOMTSRDA

Header

DEC	HEX		
0	0	TRANCODE	TRANFLAG
2	2	MSGLENG	
4		TRANDEST	TRANORIG
6	6	SCANDAY	
8	8	SCANHOUR	
10	A	SCANMIN	
12	C	SCANSEC	
14	E	RESERVED	
16	10	ETSHOUR	
18	12	ETSMIN	
20	14	ETSSEC	
22	16	PSTDMIN	
24	18	PSTDSEC	
26	1A	RESERVED	SCANID

Control Station Portion

DEC	HEX		
0	0	ORIGS7ID	TERMID
2	2	S7FLAG	IDTSIZE

Analog DATA Portion

DEC	HEX	
0	0	AICOUNT
2	2	AIDATA

Status Data Portion

DEC	HEX	
0	0	STATCCUN
2	2	STATDATA

PC Data Portion

DEC	HEX	
0	0	PCCOUNT
2	2	PCIDTBTS

Alphabetical List of Fields in DOMTSRDA

Field	Dec	Hex
ETSHOUR	0016	0010
ETSMIN	0018	0012
ETSSEC	0020	0014
MSGLENG	0002	0002
PSTDMIN	0022	0016
PSTDSEC	0024	0018
SCANDAY	0006	0006
SCANHOUR	0008	0008
SCANID	0027	001B
SCANMIN	0010	000A
SCANSEC	0012	000C
TRANCODE	0000	0000
TRANDEST	0004	0004
TRANFLAG	0001	0001
TRANORIG	0005	0005

Data Area Layout of DOMTSRDA

Header

Offset	Bytes and Bit Patterns	Field	Description
0 (0)	1	TRANCODE	Transaction code = X'8A'
1 (1)	1	TRANFLAG	Transaction code flag
2 (2)	2	MSGLENG	Message length
4 (4)	1	TRANDEST	Logical ID of receiving System/370
5 (5)	1	TRANORIG	Logical ID of originating System/7
6 (6)	2	SCANDAY	Day of scan in Julian days
8 (8)	2	SCANHOUR	Hour of scan time
10 (A)	2	SCANMIN	Minutes of scan time
12 (C)	2	SCANSEC	Seconds of scan time
14 (E)	2		Reserved
16 (10)	2	ETSHOUR	External time standard hours
18 (12)	2	ETSMIN	External time standard minutes
20 (14)	2	ETSSEC	External time standard seconds
22 (16)	2	PSTDMIN	Power system time deviation minutes
24 (18)	2	PSTDSEC	Power system time deviation seconds
26 (1A)	1		Reserved
27 (1B)	1	SCANID	Scan ID

Control ~~Station Portion~~

Offset	Bytes and Bit Patterns	Field	Description
Offset	Bytes and Bit Patterns	Field	Description
0 (0)	1	ORIGS7ID	ID of originating System/7 -- may be remote System/7
1 (1)	1	TERMID	Logical ID of this control station
2 (2)	1	S7FLAG	Flag byte as follows:
	1...	INSERVIC	Inservice flag; 0 - active, 1 - not active
	.1..	NOIDTERR	Reinitialize terminal flag; 0 - has not been reinitialized 1 - has been reinitialized
	..1.		Reserved
	...1 1111	LOWDIGRP	Lowest Digital Input (DI) card address
3 (3)	1	IDTSIZE	Size of invalid data table in System/7 words (used for pulse counter data)

Analog Data Portion

Offset	Bytes and Bit Patterns	Field	Description
0 (0)	2	AICOUNT	Number of analog values read under current scan
2 (2)	2	AIDATA	Positional analog data ordered in sequence acquired Bit 15 of each analog data value field is used as a validity indicator. If bit 15 is off, the data is good. If bit 15 is on, the data is bad, as follows: Bit 8 on - missing data Bit 9 on - BCH error Bit 10 on - bad ADC Bit 11 on - overload

Status Data Portion

Offset	Bytes and Bit Patterns	Field	Description
0(0)	2	STATCCUN	Number of status values read under current scan
2(2)	1	STATCARD	Status group card address
	...x xxxx		Card address
	xxx.		Reserved
3(3)	1	STATCRD2	Type of status group
	.... ..x.		Status group latched if on. If off, then status group is unlatched.
	xxxx xx.x		Reserved
4(4)	2		If latched group, latch bits. If unlatched goup, unlatched bits.
6(6)	2		Status group, status bits

Pulse Counter (PC) Data Portion

Offset	Bytes and Bit Patterns	Field	Description
0(0)	2	PCCOUNT	Number of pulse counter values read under current scan
2(2)	2	PCIDTBTS	Pulse counter invalid data table (IDT) bits. One bit for each pulse counter value.
n(M)	2	PCDATA	Positional pulse counter data received under current scan If corresponding IDT bit is off, then if bit 15 is off - good data bit 15 on - counter overrun If corresponding IDT bit is on, then if bit 8 is on - missing data, or if bit 9 is on - BCH error

DOMTWRKA (DATA ACQUISITION WORK AREA)

TOTAL SIZE: 816 bytes DSECT: DOMTWRKA

CREATED BY: Data acquisition initialization

PURPOSE: To map the work area and common data pointers for data acquisition.

POINTED TO BY: ECVTDQWA field of EMSCVT

Storage Map of DOMTWRKA

DEC	HEX				
0	0	SAVECONV			
72	48	SAVE1			
76	4C	HSA1			
80	50	LSA1			
84	54	REGS1			
144	90	SAVE2			
148	94	HSA2			
152	98	LSA2			
156	9C	REGS2			
216	D8	TCBXALR1			
220	DC	TCBXDC01			
224	EO	PATCHPBL			
240	FO	SETHEADP			
244	F4	CURRENT			
248	F8	EMTP			
252	FC	BUFRENDP			
256	100	FPWORK			
260	104	STUNLT			
264	108	IADDRESS			
268	10C	IZERO			
272	110	SCANSCHD			
276	114	ECVTLCO			
280	118	CLOCK1			
284	11C				
288	120	CLOCK2			
292	124				
296	128	ADRSPA			
300	12C	LENGSPA			
304	130	ADRIND			
308	134	ADRLFCGO			
312	138	RTLOCKA			
316	13C	CDLOCKA			
320	140	SAFEPRBL			
324	144				
328	148	RELTIVS7	LFCOUNT	ANYBUFCT	CONSISCT
332	14C	FAILIND	Flags	DAQSPARE	Flags
336	150	PCFILTER			
340	154	PCBLKDAT			
344	158	PCARRAY			
348	15C	PCITEMNO			
352	160	CSESDC			
396	18C	CSESLOG			
440	1B8	CSESPTCH			
444	1BC				
448	100	STORE1			
452	1C4	STOREC			
456	1C8	STORETD			
460	1CC				
464	1DO	LSABSET			
536	218	SAVE432			
548	224	SAVE14			
552	228	PDCID	PDCXCVT		
556	22C	PDCRDA			

560	230	PDCITEM	
564	234	PDCRCB	
568	238	RESERVED	
572	23C	MSG1	
628	274	WBPROB	
636	27C	WBSUP	
680	2A8	MES1	
732	2DC	LTOHUNTS	
736	2E0	WRNLIMIT	DAQHSPAR
740	2E4	ASTAELST	
744	2E8	FPWRNLMT	
748	2EC	FPLTOHU	
752	2F0	VPROBL	
756	2F4		
760	2F8	VSUPL	
804	324	WBLIST	
808	328	STAEBSET	
812	32C	BSETADDR	
816	330	STAECSES	
832	340	STAEND	RESERVED
836	344	WBCT	WBNEXT
840	348	WBADDR	

Alphabetical List of Fields in DOMTWRKA

Field	Dec	Hex
ADRIND	0304	0130
ADRLFCGO	0308	0134
ADRSPA	0296	0128
ANYBUFCT	0330	014A
ASTAELST	0740	02E4
BSETADDR	0812	032C
BUFRENDP	0252	00FC
CDLOCKA	0316	013C
CLOCK1	0280	0118
CLOCK2	0288	0120
CONSISCT	0331	014B
CSESDC	0352	0160
CSESLOG	0396	018C
CSESPTCH	0440	01B8
CURRENT	0244	00F4
DAQHSPAR	0738	02E2
DAQSPARE	0334	014E
ECVTLOC	0276	0114
EMTP	0248	00F8
FAILIND	0332	014C
FPLTOHU	0748	02EC
FPWORK	0256	0100
FPWRNLMT	0744	02E8
HSA1	0076	004C
HSA2	0148	0094
IADDRESS	0264	0108
IZERO	0268	010C
LENGSPA	0300	012C
LFCOUNT	0329	0149
LSABSET	0464	01D0
LSA1	0080	0050
LSA2	0152	0098
LTOHUNTS	0732	02DC
MES1	0680	02A8

Field	Dec	Hex
MSG1	0575	023D
PATCHPBL	0224	00E0
PCARRAY	0344	0158
PCBLKDAT	0340	0154
PCFILTER	0336	0150
PCITEMNO	0348	015C
PDCID	0552	0228
PDCITEM	0560	0230
PDCRCB	0564	0234
PDCRDA	0556	022C
PDCXCVT	0553	0229
REGS1	0084	0054
REGS2	0156	009C
RELTIVS7	0328	0148
RTLOCKA	0312	0138
SAFEPPBL	0320	0140
SAVECONV	0000	0000
SAVE1	0072	0048
SAVE14	0548	0224
SAVE2	0144	0090
SAVE432	0536	0218
SCANSCHD	0272	0110
SETHEADP	0240	00F0
STAEBSET	0808	0328
STAECSES	0816	0330
STAEND	0832	0340
STOREC	0452	01C4
STORETD	0456	01C8
STORE1	0448	01C0
STUNLT	0260	0104
TCBXALR1	0216	00D8
TCBXF401	0220	00DC
VPROBL	0752	02FO
VSUPL	0760	02F8
WBADDR	0840	0348
WBCT	0836	0344
WBLIST	0804	0324
WBNEXT	0837	0345
WBPROP	0628	0274
WBSUP	0636	027C
WRNLIMIT	0736	02E0

Data Area Layout of DOMTWRKA

Offset	Bytes and Bit Patterns	Field	Description
0 (0)	72	SAVECONV	Data acquisition save area
72 (48)	4	SAVE1	Start of save area for data conversion routine
76 (4C)	4	HSA1	Pointer to higher save area in program calling data conversion routine
80 (50)	4	LSA1	Address of SAVE2
84 (54)	60	REGS1	Save area for data conversion calling program
144 (90)	4	SAVE2	Start of save area for individual processor routines

Offset	Bytes and Bit Patterns	Field	Description
148(94)	4	HSA2	Address of SAVE1
152(98)	4	LSA2	Pointer to low save area in data conversion routine
156(9C)	60	REGS2	Save area for individual processor routines
216(D8)	4	TCBXALR1	Address of TCBX for DOMCALR1
220(DC)	4	TCBXDC01	Address of TCBX for DOMCDC01
224(E0)	8	PATCHPBL	List form of PATCH parameter list for PATCH to DOMCALR1
240(F0)	4	SETHEADP	Address of the start of SET buffers
244(F4)	4	CURRENT	Address where next SET entry is placed
248(F8)	4	EMTP	Address of current control station header in SET buffer
252(FC)	4	BUFRENDP	Address of the end of the current 128 byte GETWA SET buffer
256(100)	4	FPWORK	Floating point work area
260(104)	4	STUNLT	Saved work area for DOMTCSES (Status data processor)
264(108)	4	IADDRESS	Indicator set by any individual processor and used by DOMTBSET
268(10C)	4	IZERO	Indicator set by any individual processor and used by DOMTBSET
272(110)	4	SCANSCHD	Special Real Time Operating System clock time of last scan scheduled
276(114)	4	ECVTLOC	Address of EMSCVT
280(118)	8	CLOCK1	Save area for clock time used in calculating scan performance
288(120)	8	CLOCK2	Save area for clock time used in calculating scan performance
296(128)	4	ADRSPA	Address of scan performance array CADSPA
300(12C)	4	LENGSPA	Length and number of items in CADSPA
304(130)	4	ADRIND	Address of data base indicator array CADBIND
308(134)	4	ADRLFCGO	Address of AILFCGO item-set to 1 at the end of each complete data scan

Offset	Bytes and Bit Patterns	Field	Description
312(138)	4	RTLOCKA	Address of control block defined for realtime data lock
316(13C)	4	CDLOCKA	Address of control block defined for consistency data lock
320(140)	8	SAFEPRBL	List form of PATCH parameter list for PATCH to DOMTFAIL
328(148)	1	RELTIVS7	One bit for each active System/7 up to a maximum of eight to indicate which System/7 has shipped data this scan
329(149)	1	LFCOUNT	One bit for each System/7 to determine when all data has been shipped from System/7 for AGC processing
330(14A)	1	ANYBUFCT	One bit for each System/7 to indicate if any buffer has arrived for this System/7 in current scan period
331(14B)	1	CONSISCT	One bit for each System/7 to indicate that data was consistent
332(14C)	1	FAILIND	One bit for each System/7 to indicate if this System/7 failed in last scan cycle
333(14D)	1	(FLAGBYTE)	Flag byte as follows:
	1...	S7TIMEIN	Set on when System/7 has input a valid system time
	.1..	PCDATADN	Set on indicates PC data processing has been down
	..1.	PCDATAOT	Set on indicates PC data has been output
	...1	FIRSTIME	Set on after first complete data scan
	.... 1...	INITDONE	Set on indicates the initial scan has been processed
	.... .1..	NOACTS7	Set on indicates an inactive System/7 has sent data
	.... ..1.	PCMINDAT	Set on indicates PC data has been received within the last minute cycle
	1	NSCANING	Set on indicates active System/7 is not scanning
334(14E)	1	DAQSPARE	Reserved

Offset	Bytes and Bit Patterns	Field	Description
335(14F)	1	(FLAGBYTE)	Flag byte as follows:
	1...	CLOKDONE	Set on when the CATIMES array, has been updated
	.1..	NOPCDATA	Set on indicates no pulse counter data in the system
336(150)	..1.	PCFILTER	Pulse counter data sysgened filter value
340(154)	4	PCBLKDAT	Address of block data for array CACOUNT
344(158)	4	PCARRAY	Address of pulse counter array CACOUNT
348(15C)	4	PCITEMNO	Number of pulse counter points in array CACOUNT
352(160)	44	CSESDC	Macro list form for PATCH to DOMCDC01 (SUPL)
396(18C)	44	CSESLOG	Macro list form for PATCH to DOMCSLOG (SUPL)
440(1B8)	8	CSESPTCH	PROBL macro list form for PATCHes
448(1C0)	4	STORE1	Address of first buffer for parameter to DOMCSLOG
452(1C4)	4	STOREC	Address of current buffer being used for parameters for DOMCSLOG
456(1C8)	8	STORETD	Save area for scan time and date
464(1D0)	72	LSABSET	Save area for DOMTBSET
536(218)	12	SAVE432	Save area for registers 2-4
548(224)	4	SAVE14	Save area for register 14
552(228)	1	PDCID	ID for call to DOMCPDC1
553(229)	3	PDCXCVT	XCVT address
556(22C)	4	PDCRDA	RDA address for DOMCPDC1
560(230)	4	PDCITEM	ITEM address for DOMCPDC1
564(234)	4	PDCRCB	RCB address for DOMCPDC1
568(238)	5	(RESERVED)	
573(23D)	54	MSG1	Message retrieval area
628(274)	8	WBPROB	List form of wallboard PROBL
636(27C)	44	WBSUP	List form of wallboard SUPL
680(2A8)	52	MES1	List form of MESSAGE macro with routing code

Offset	Bytes and Bit Patterns	Field	Description
732(2DC)	4	LTOHUNTS	Number of units between low and high warning limit per cent values (for example, 80-20 or 90-10)
736(2E0)	2	WRNLIMIT	Analog data low warning limit per cent value
738(2E2)	2	DAQHSPAR	Reserved
740(2E4)	4	ASTAELST	Address of STAE list
776(308)	4	STAEADDR	Address of STAE flag to zero
780(30C)	4	STAELIST	Indicator for STAE processor to release RDA buffer
784(300)	4	STAERDA	Address of RDA buffer to release
808(328)	4	STAEBSET	Indicator for STAE processor to free S.E.T. buffers
812(32C)	4	BSETADDR	Address of the S.E.T. buffer to free
816(330)	16	STAECSES	STAE list for DOMTCSES
832(340)	1	STAEND	End of STAE list indicator
833(341)	3	(RESERVED)	
836(344)	1	WBCT	Count of items in wallboard list
837(345)	3	WBNEXT	Address of next wallboard buffer
840(348)	4	WBADDR	Address of wallbcard item in data base

ECTABLE (EVENT CONTROL BLOCK TABLE)

TOTAL SIZE: 192 bytes DSECT: ECTABLE, ECXIP

CREATED BY: Supervisory control initialization

PURPOSE: To contain information about the physical and logical organization of the event file and to keep track of the position to write the next record.

POINTED TO BY: ECVTEVNT field of EMSCVT

Storage Map of ECTABLE

DEC	HEX		
0	0	ECTCBX	
4	4	ECDCB	
8	8	ECLOCN	
12	C	ECLOCB	
16	10	ECATIMED	
20	14	ECLOG	
24	18	ECXNUM	
28	1C	ECESIZE	
32	20	ECXSIZE	
36	24	ECTYPE	
50	32	ECDEAD	ECXREC
		ECACODE	
192	C0	ECPT	
196	C4	RESERVED	

Alphabetical List of Fields in ECTABLE

Field	DEC	HEX
ECACODE	0054	0036
ECDCB	0004	0004
ECDEAD	0052	0034
ECESIZE	0024	0018
ECLOCB	0012	000C
ECLOCN	0008	0008
ECLOG	0016	0010
ECPT	0188	018D
ECTCBX	0000	0000
ECTYPE	0032	0020
ECXNUM	0020	0014
ECXREC	0053	0035
ECXSIZE	0028	001C
RESERVED	0190	00BE

Offset	Bytes and Bit Patterns	Field	Description
0 (0)	4	ECTCBX	Address of TCBX for DOMCEVT1
4 (4)	4	ECDCB	Address of DCB of Event Log File
8 (8)	4	ECLOCN	Lock Resource name address
12 (c)	4	ECLOCB	Lock Control Block address
16 (10)	4	ECATIMED	Address of Special Real Time array
20 (14)	4	ECLOG	Next logical logging position
24 (18)	4	ECXNUM	Current index record number
28 (1C)	4	ECESIZE	Number of event records
32 (20)	4	ECXSIZE	Number of index records
36 (24)	14	ECTYPE	User-defined type codes
50 (32)	1	ECDEAD	Size of the dead zone
51 (33)	1	ECXREC	Flag byte, X'FF' = UNUSED RECORD
52 (34)	141	ECACODE	Access area codes
	1	ECIXAA	Access area ID
	1	ECIXTY	Event type code
	1	ECIXFA	Function area
189 (BD)	1	ECPT	PATCHID of last PTIME issued
190 (BE)	2	RESERVE	Reserved field

EMSCVT (ENERGY MANAGEMENT SYSTEM COMMUNICATION VECTOR TABLE)

TOTAL SIZE: 76 bytes DSECT: EMSCVT

CREATED BY: Data acquisition initialization

PURPOSE: A list of pointers to tables, arrays, and work areas used by System/370 Energy Management System programs.

POINTED TO BY: DPPXCVT

Storage Map of EMSCVT

Dec	Hex				
0	0	ECVTXCVT			
4	4	ECVTLOAD			
8	8	ECVTCOMM			
12	C	ECVTECBI			
16	10	ECVTECBW			
20	14	ECVTLIDC	ECVTS7CT		
24	18	ECVTDCIX			
28	1C	ECVTEVNT			
32	20	ECVTPTSM			
36	24	ECVTDQWA			
40	28	ECVTIOCT			
44	2C	ECVTFCVT			
48	30	ECVTSPLM			
52	34	ECVTACT7	ECVTINT3	ECVTINT2	ECVTINT1
56	38	ECVTYPES			
60	3C	ECVTSOPT			
64	40	ECVTPNTP	ECVTCOND		
68	44	ECVTSLOG			
72	48	ECVTSTCH			
76	4C	ECVTTIME			
80	50	ECVTSTAE			
84	54	ECVTALOK			
88	58	ECVTSCAN			
92	5C	ECVTLPDO			
96	60	ECVTAPIL			
100	64	ECVTAPLN			
104	68	ECVTAGCF	ECVTAGCA		
108	6C	ECVTAGCC			
112	70	ECVTAPIT			
116	74	ECVTOTBL			
120	78	ECVTUNIT			
124	7C	ECVTAATB			
128	80	ECVTMDAA	ECVTMDAC	ECVTMDFC	ECVTMDFU
132	84	ECVTASAD			
136	88	ECVTAEAD			
140	8C	ECTBWBNA			
144	90	ECVTWBCN			
148	94	ECVTWBCF	ECVTWBCL		
152	98	ECVTS7IN			
156	9C	ECVTS7LB			
160	A0	ECVTSCIN			
164	A4	ECVTSCLB			
168	A8	ECVTHIER			
172	AC	RESERVED			

Alphabetical Listing of Fields in EMSCVT

Field	Dec	Hex
ECVTAATB	0124	007C
ECVTACT7	0052	0034
ECVTAEAD	136	88
ECVTAGCA	0105	0069
ECVTAGCC	0108	006C

Field	Dec	Hex
ECVTAGCF	0104	0068
ECVTALOK	0084	0054
ECVTAPLL	0096	0060
ECVTAPLN	0100	0064
ECVTAPLT	0112	0070
ECUTASAD	0132	0084
ECVTCOMM	0008	0008
ECVTCOND	0065	0041
ECVTDCIX	0024	0018
ECVTDQWA	0036	0024
ECVTECBI	0012	000C
ECVTECBW	0016	0010
ECVTEVNT	0028	001C
ECVTFCVT	0044	002C
ECVTHIER	0168	00A8
ECVTINT1	0055	0037
ECVTINT2	0054	0036
ECVTINT3	0053	0035
ECVTIOCT	0040	0028
ECVTLIDC	0020	0014
ECVTLOAD	0004	0004
ECVTLPDO	0092	0056
ECVTMDAA	0128	0080
ECVTMDFC	0130	0082
ECVTMDFU	0131	0083
ECVTOTBL	0116	0074
ECVTPNTP	0064	0040
ECVTPTSM	0032	0020
ECVTSCAN	0088	0058
ECVTSCLB	0164	00A4
ECVTSCLN	0160	00A0
ECVTSLOG	0068	0044
ECVTSPLM	0048	0030
ECVTSOPT	0060	003C
ECVTSTAE	0080	0050
ECVTSTCH	0072	0048
ECVTS7CT	0021	0015
ECVTS7LB	0156	009C
ECVTS7LN	0152	0098
ECVTTIME	0076	004C
ECVTTYPES	0056	0038
ECVTUNIT	0120	0078
ECVTWBCF	0148	0094
ECVTWBCL	0149	0095
ECVTWBCN	0144	0090
ECVTWBNA	0140	008C
ECVTXCVT	0000	0000

Data Area Layout of EMSCVT

Offset	Bytes and Bit Patterns	Field	Description
0 (0)	4	ECVTXCVT	Address of XCVT
4 (4)	4	ECVTLOAD	Address of macro services branch table
8 (8)	4	ECVTCOMM	Address of System/7 communication table
12 (C)	4	ECVTECBI	Address of System/7 initialized program load request ECB list

Offset	Bytes and Bit Patterns	Field	Description
16(10)	4	ECVTECBW	Address of System/7 write request control blocks
20(14)	1	ECVTLIDC	Number of entries in S7CT
21(15)	3	ECVTS7CT	Address of System/7 control table (S7CT)
24(18)	4	ECVTDCIX	Address of device control index table
28(1C)	4	ECVTEVNT	Address of event control block table (ECTABLE)
32(20)	4	ECVTPTSM	Address of point summation table
36(24)	4	ECVTDQWA	Address of data acquisition work area
40(28)	4	ECVTIOCT	Address of System/7 communication control table
44(2C)	4	ECVTFCVT	Address of System/7 failover control table
48(30)	4	ECVTSPLM	Address of special transducer limit array
52(34)	1	ECVTACT7	Data acquisition flag byte
53(35)	1	ECVTINT3	Data acquisition flag byte
54(36)	1	ECVTINT2	Data acquisition flag byte
55(37)	1	ECVTINT1	Data acquisition flag byte
56(38)	4	ECVTYPES	Address of types table array
60(3C)	4	ECVTSOPT	Address of sysgen options array
64(40)	1	ECVTPNTP	Top/not top indicator
65(41)	3	ECVTCOND	Address of condition code table
68(44)	4	ECVTSLOG	Address of CASTLOG array
72(48)	4	ECVTSTCH	Address of stripchart table
76(4C)	4	ECVTTIME	Address of times array CATIMES
80(50)	4	ECVTSTAE	Address of DOMCSTAE
84(54)	4	ECVTALOK	Alarms CBLOC address
88(58)	4	ECVTSCAN	Address of CASCAN array
92(5C)	4	ECVTLPDO	Address of APDO lock for for AGC output
96(60)	4	ECVTAPLL	Address of CAPTLOG array

Offset	Bytes and Bit Patterns	Field	Description
100(64)	4	ECVTAPLN	Address of CAPLNAME array
104(68)	1	ECVTAGCF	Flag indicator for AGC output
105(69)	3	ECVTAGCA	Address of AGC array AALFCPDO
108(6C)	4	ECVTAGCC	Number of generators in array AALFCPDO
112(70)	4	ECVTAPLT	Address of CALOGTIM array
116(74)	4	ECVTOTBL	Address of PDC options table
120(78)	4	ECVTUNIT	Address of CAUNITS array
124(7C)	4	ECVTAATB	Address of CAAATBL array
128(80)	1	ECVTMDAA	Power system operator access area ID
129(81)	1	ECVTMDAC	Power system operator console access area ID
130(82)	1	ECVTMDFC	Power system operator console function code ID
131(83)	1	ECVTMDFU	Power system operator console function code ID
132(84)	4	ECVTASAD	Start address of CANALOG array
136(88)	4	ECVTAEAD	END address of CANALOG array
140(8C)	4	ECVTWBNA	Address of CAWBNAME array
144(90)	4	ECVTWBCN	Address of CAWBCONF TABLE
148(94)	1	ECVTWBCF	Flag byte for Wallboard command list
148(94)	.... ..1.	ECVTWBIO	On to indicate command list buffer to be released
	1	ECVTWBPI	On if command list is initial
149(95)	3	ECVTWBCL	Wallboard command list buffer address
152(98)	4	ECVTS7LN	Lock name for TAS7COMM array
156(9C)	4	ECVTS7LB	Address of lock control block for TAS7COMM array
160(A0)	4	ECVTSCLN	Lock name for CASCAN array
164(A4)	4	ECVTSCLB	Address of lock control block for CASCAN array
168(A8)	4	ECVTHIER	Address of TAS7HIER array
172(AC)	4	Reserved	

Former text has been moved or deleted.

RETFDBCK (RETRIEVED FEEDBACK)

TOTAL SIZE: 32 bytes/entry DSECT: RETFDBCK

CREATED BY: DOMCLGET macro

PURPOSE: This area contains information which is required to update logged history data base arrays with modified retrieved data. The feedback block is generated by the DOMCLGET macro and is used by the DOMCLPUT macro.

POINTED TO BY: Keyword parameter (FEEDBCK) of the DOMCLGET and DOMCLPUT macros

Storage Map of RETFDBCK

DEC	HEX		
0	0	RETFAHDR	
4	4		
8	8		
12	C		
16	10		
20	14		
24	18	RETFDPL1	RETFLNG1

Alphabetical List of Fields in RETFDBCK

Field	Dec	Hex
RETFAHDR	0000	0000
RETFDPL1	0024	0018
RETFLNG1	0026	001A

Data Array Layout of RETFDBCK

Offset	Bytes and Bit Patterns	Field	Description
0(0)	24	RETFAHDR	Array header
24(18)	2	RETFDPL1	Displacement to start of data to be replaced for first array
26(1A)	2	RETFLNG1	Length of data to be replaced for first array

RETPARMS (RETRIEVAL PARAMETERS)

TOTAL SIZE: 16 bytes/entry DSECT: RETPARMS

CREATED BY: DOMCLPUT or DOMCLGET macros

PURPOSE: This DSECT contains a parameter list which is passed to a control program (DCMCLRMT) via DOMCLPUT or DOMCLGET macros for retrieving or updating the incore or logged history data base.

POINTED TO BY: Register 1 contains the address of the parameter list when DOMCLRMT gains control.

Storage Map of RETPARMS

DEC	HEX				
0	0	RETTMDT			
4	4	RETFDBK			
8	8	RETMACID	RETS7ID	RETRDAS	RETSP1
12	C	RETSTEP			

Alphabetical List of Fields in RETPARMS

Field	Dec	Hex
RETFDBK	0004	0004
RETMACID	0008	0008
RETRDAS	0010	000A
RETSP1	0011	000B
RETSTEP	0012	000C
RETS7ID	0009	0009
RETTMDT	0000	0000

Data Area Layout of RETPARMS

Offset	Bytes and Bit Patterns	Field	Description
0(0)	4	RETTMDT	Time or date address
4(4)	4	RETFDBK	Feedback block address
8(8)	1	RETMACID	Macro ID
	XXXX 0000	N/A	Remote control/block (RCB)
	XXXX 0010	N/A	Analog data
	XXXX 0100	N/A	Pulse counter data
	XXXX 0110	N/A	Status data
	0000 XXXX	N/A	Retrieve incore data base data
	0010 XXXX	N/A	Retrieve logged history data base data
	0100 XXXX	N/A	Update incore data base data
	0110 XXXX	N/A	Update logged history data base data
9(9)	1	RETS7ID	System/7 ID - two-digit number 1-99
10(A)	1	RETRDAS	3707 RDAS ID - three-digit number 0-255
11(B)	1	RETSP1	Spare (not used)
12(C)	4	RETSTEP	Step value used to step forward or backward through logged history data base arrays

SET (SCAN EXCEPTION TABLE)

TOTAL SIZE: 136 bytes/buffer DSECT: SETD

CREATED BY: GETWA issued for 136 bytes by anyone issuing a scan exception.

PURPOSE: Used by scan processing and supervisory control initialization during a valid warm start to pass alarms to DOMCALR1.

Storage Map of SET

Buffer Header

DEC	HEX				
0	0	SETNUM	SETNEXT		

Terminal Header

DEC	HEX				
4	4	SETHFLG	(Reserved)	SETS7	SETEMT

Data Entry

DEC	HEX		
8	8	SETIFLG	SETILOC
12	C	SETVAL	

Alphabetical List of Fields in SET

Field	Dec	Hex
SETEMT	0007	0007
SETHFLG	0004	0004
SETIFLG	0008	0008
SETILOC	0009	0009
SETNEXT	0001	0001
SETNUM	0000	0000
SETS7	0006	0006
SETVAL	0012	000C

Data Area Layout of SET

Offset	Bytes and Bit Patterns	Field	Description
0(0)	1	SETNUM	Number of data entries in buffer
1(1)	3	SETNEXT	Address of next buffer in a chain or zero if this is the end of the chain
4(4)	1	SETHFLG	Flag byte as follows:
	11..	SETTYPE	Type of entry - 00 = control station header
	..11 1111		Reserved
5(5)	1		Reserved
6(6)	1	SETS7	System/7 identifier
7(7)	1	SETEMT	Terminal identifier
8(8)	1	SETIFLG	Flag byte as follows:

Offset	Bytes and Bit Patterns	Field	Description
	11..	SETITYPE	Type of entry - 01 = analog alarm entry, 10 = status alarm entry, 11 = pulse counter alarm entry
	..1.	SETCLR	Clear alarm condition indicator
	...1 1111		Condition which casused the alarm
9(9)	3	SETIIOC	Address of alarmed point in data base
12(C)	4	SETVAL	Value as received

SYSTEM/7 CHECKPOINT ADDRESS TABLE

ENTRY SIZE: 8 bytes per entry

TOTAL SIZE: 8 * number System/7 checkpoint data sets

CREATED BY: Initialization (DOMTINFO) DSECT: CKPTDSCT

PURPOSE: Provide pointers for System/7 checkpoint processing

POINTED TO BY: CKPTLIST field of FCVT

Storage Map

DEC	HEX	Field	
0	0	CKPTLID	CKPTDCBA
4	4	CKPTARAY	

Alphabetical List of Fields

Field	DEC	HEX
CKPTARAY	4	4
CKPTDCBA	1	1
CKPTLID	0	0

Data Area Map

Offset	Bytes and Bit positions	Field	Description
0(0)	1	CKPTLID	Logical id of this entry
1(1)	3	CKPTDCBA	Address of checkpoint DCB
4(4)	4	CKPTARAY	Address of checkpoint array ID list

S7CT (SYSTEM/7 CONTROL TABLE)

TOTAL SIZE: 16 bytes/entry DSECT: S7CT

CREATED BY: Data acquisition initialization

PURPOSE: This table contains pointers to all data base arrays containing information about the remote control stations.

POINTED TO BY: ECVTS7CT field of the EMSCVT

Storage Map of an S7CT

DEC	HEX				
0	0	S7CTLID	RESERVED	S7CT#RCB	
4	4	S7CTRCBA			
8	8	S7CTRDAM			
12	C	S7CTFLG1	S7CTFLG2	S7CTFLG3	S7CTFLG4

Alphabetical List of Fields in S7CT

Field	Dec	Hex
S7CT#RCB	0002	0002
S7CTFLG1	0012	000C
S7CTFLG2	0013	000D
S7CTFLG3	0014	000E
S7CTFLG4	0015	000F
S7CTLID	0000	0000
S7CTRCBA	0004	0004
S7CTRDAM	0008	0008

Data Area Layout of S7CT

Offset	Bytes and Bit Patterns	Field	Description
0 (0)	1	S7CTLID	System/7 logical ID
1 (1)	1		Reserved
2 (2)	2	S7CT#RCB	Number of RCB addresses in list
4 (4)	4	S7CTRCBA	Address of RCB address list
8 (8)	4	S7CTRDAM	Address of Raw data array map
12 (C)	1	S7CTFLG1	Emergency scan flag byte
13 (D)	1	S7CTFLG2	Reserved
14 (E)	1	S7CTFLG3	Minutes of last RDA input
15 (F)	1	S7CTFLG4	Seconds of last RDA input

SYSTEM/7 DISK CONTROL TABLE

ENTRY SIZE: 4 bytes per entry

TOTAL SIZE: 4 * number of physical System/7s

CREATED BY: Initialization (DOMTINFO)

PURPOSE: Provide information about which System/7 disk load PDS member and relative sector is to be processed next.

Pointed to by: DISKCTRL field of FCVT

Storage Map

DEC	HEX	Field	
0	0	RELMEMBR	RELSECTR

Alphabetical List of Fields

Field	DEC	HEX
RELMEMBR	0	0
RELSECTR	2	2

Data Area Map

Offset	Bytes and Bit positions	Field	Description
0 (0)	2	RELMEMBR	Relative displacement into DOTHCORR member to next data set member
2 (2)	2	RELSECTR	Relative sector with member to be processed next

SYSTEM/7 I/O: Data extent control block load macros I/O

TOTAL SIZE: 12 bytes DSECT: USERDECB

CREATED BY: Program issuing System/7 initialized program load

PURPOSE: To pass parameters to the System/7 I/O preprocessor

POINTED TO BY: Register 1

Storage Map

DEC	HEX			
0	0	USERECB		
4	4	USERTYPE	USERLID	USERLNGH
8	8	USERID	USERDATA	

Alphabetical Listing of Fields

Field	Dec	Hex
USERDATA	0009	0009
USERECB	0000	0000
USERID	0008	0008
USERLID	0005	0005
USERLNGH	0006	0006
USERTYPE	0004	0004

Offset	Bytes and Bit Patterns	Field	Description
(0)	4	USERECB	ECB to be POSTed when I/O completed
4 (4)	1	USERTYPE	Type of I/O
	1...	USERIPL	Initialized program load transaction, issued through initialized program load address
	.1..	USERTXT	Transaction has no header, write through write address

Offset	Bytes and Bit Patterns	Field	Description
	..1.	USEREND	Issue cyclic READ
	.... 1...	USERMSG	Transaction has standard header
	.... .1..	USERUNIT	Transaction to specified unit
5(5)	1	USERLID	System/7 logical ID
6(6)	2	USERLNGH	Length of transaction
8(8)	1	USERID	System/7 unit ID
9(9)	3	USERDATA	Address of transaction

SYSTEM/7 SUPPORT CONTROL TABLE (FCVT)

TOTAL SIZE: 40 bytes

CREATED BY: Initialization (DOMTINFO) DSECT: FAILCVT (FCVT)

PURPOSE: Provide pointers to other data areas necessary to support the System/7s.

POINTED TO BY: ECVTFCVT field of EMSCVT

Storage Map

DEC	HEX	Field
0	0	AXCVT
4	4	TAS7COMM
8	8	STATUNIT
12	C	DISKCTRL
16	10	DECBLIST
20	14	RESERVED
24	18	VARYLOCK
28	1C	CKPTLIST
32	20	FCVTPURG
36	24	RESERVED

Alphabetical List of Fields

Field	DEC	HEX
AXCVT	0	0
CKPTLIST	28	1C
DECBLIST	16	10
DISKCTRL	12	C
FCVTPURG	32	20
STATUNIT	8	8
TAS7COMM	4	4
VARYLOCK	24	18

Data Area Map

Offset	Bytes and Bit positions	Field	Description
0 (0)	4	AXCVT	Address of DPPXCVT
4 (4)	4	TAS7COMM	Address of TAS7COMM array
8 (8)	4	STATUNIT	Address of unit status table
12 (C)	4	DISKCTRL	Address of disk control fields
16 (10)	4	DECBLIST	Address of System/7 disk support DECB address list
20 (14)	4		Reserved
24 (18)	4	VARYLOCK	Address of lock control block for the lock name defined by this address
28 (1C)	4	CKPTLIST	Address of System/7 checkpoint address table
32 (20)	4	FCVTPURG	Address of the purge request ECB in the System/7 I/O EDB list
36 (24)	4		Reserved

SYSTEM/7 UNIT STATUS TABLE

ENTRY SIZE: One byte

TOTAL SIZE: Number of physical System/7s (rounded up to a fullword)

CREATED BY: Initialization (DOMTINFO) DSECT: none

PURPOSE: Provide information on the control of a System/7.

POINTED TO BY: STATUNIT field of FCVT

Storage Map - By bit number

DEC	HEX	Field
0	0	NOACTION
1-3	1-3	Reserved
4	4	MDWAIT
5	5	PRIMES7
6	6	BKUPS7
7	7	OFFLNE7

Alphabetical List of Fields

Field	Bit Number
BKUPS7	6
MDWAIT	4
NOACTION	0
OFFLNE7	7
PRIMES7	5

Data Area Map

Offset	Bytes and Bit positions	Field	Description
0	1...	NOACTION	No action required to enter the System/7 into the hierarchy
0	.xxx	reserved	
0	.... 1...	MDWAIT	Wait for action
0	.... .1..	PRIMES7	System/7 to be assigned as primary
0	.... ..1.	BKUPS7	System/7 to be assigned as backup
0	1	OFFLNE7	System/7 to be taken offline

TRANSACTION ROUTING TABLE

ENTRY SIZE: 48 bytes

TOTAL SIZE: Defined in module DOMTROUT

CREATED BY: System generation

PURPOSE: Provide information as to what transaction codes are valid and where they should be routed

POINTED TO BY: ROUTABLE FIELD OF IOCVT

Storage Map

DEC	HEX	Field		
0	0	TCODE	PATCHID	(RESERVED)
4	4	SUPL		

Alphabetical List of Fields

Field	DEC	HEX
PATCHID	1	1
SUPL	4	4
TCODE	0	0

Data Area Map

Offset	Bytes and Bit positions	Field	Description
0 (0)	1	TCODE	Transaction code
1 (1)	1	PATCH ID	PATCH ID
2 (2)	2		Reserved
4 (4)	44	SUPL	PATCH SUPL

VTABLE (VERIFICATION TABLE)

TOTAL SIZE: 12 bytes DSECT: VERTAL

CREATED BY: VARYSCAN or VARYCONF

PURPOSE: This table contains the hardware address of the IBM 5985, the status, the System/7 logical identifier, scan identifier, line number, control station identifier, point name, item count, point address, and card address.

POINTED TO BY: Register 1 in DOMTVARY

Storage Map of Verification Table

DEC	HEX				
0	0	VTTPST	VTSYS	VTSCAN	VTMODE
4	4	VTTERM	CARDADOR	POINTADD	ITEMCOUN
8	8	UNUSED			

DEC	HEX		
4	4	VTTERM	VTPOINT
8	8		

Alphabetical List of Fields in Verification Table

Field	Dec	Hex
CARDADDR	0005	0005
ITEMCOUN	0007	0007
POINTADD	0008	0008
VTMODE	0003	0003
VTPOINT	0005	0005
VTSCAN	0002	0002
VTSYS	0001	0001
VTTERM	0004	0004
VTTPST	0000	0000

Data Area Layout of Verification Table

Offset	Bytes and Bit Patterns	Field	Description
0 (0)	1	VTTPST	Type 1 status
	XXXX	VTRES	Reserved
	.... 1...	VTSCSTAT	Change scan status
	.... .1..	VTNCSTAT	Change network configuration
	.... ..1.	VTACT	Change to active or online
	1	VTINACT	Change to inactive or offline
1 (1)	1	VTSYS	System/7 identifier
2 (2)	1	VTSCAN	Scan identifier
3 (3)	1	VTMODE	Scan mode indicator
4 (4)	1	VTTERM	Control station identifier

Offset	Bytes and Bit Patterns	Field	Description
5(5)	1	CARDADDR	Card address
5(5)	7	VTPOINT	Point name
6(6)	1	POINTADD	Point address
7(7)	1	ITEMCOUN	Number of points to be processed
8(8)	4		Reserved

CONTROL BLOCKS

This section describes the major control blocks used by the System/370 Energy Management System. The format of each description may vary.

Alarm Control Element

TOTAL SIZE: 40 bytes DSECT: ACE

CREATED BY: Alarm Display Processing

PURPOSE: The ACE is used by the alarm display processor to keep track of paging, the alarms being viewed, and access area and function code information.

POINTED TO BY: The address of the ACE is found in the CEALARM field of the CEATAB for the unit when the display is active. It is created by DOMCALD1 when the detail alarm display is called up and deleted when the display is changed.

Storage Map of the ACE

DEC	HEX			
0	0	ACEAA		
4	4	ACEFUNC		
8	8	ACESVAL		
12	C	ACES#		
16	10	ACEFIRST		
20	14	ACEPG#	ACETPGS	ACEAAID
24	18	ACEAAN		
28	1C			ACEFCD
32	20	ACEFN		

Data Area Layout of ACE

Offset	Bytes and Bit Patterns	Field	Description
0(0)	4	ACEAA	Address of access area table entry
4(4)	4	ACEFUNC	Address of function information for access area
8(8)	4	ACESVAL	Address of save area for alarms chosen
12(C)	4	ACES#	Size of save area (ACESVAL)

Offset	Bytes and Bit Patterns	Field	Description
16(10)	4	ACEFIRST	Sequence number of first alarm on page
20(14)	1	ACEPG#	Page number currently being viewed
21(15)	1	ACETPGS	Total number of pages
22(16)	1	ACEAAID	Access area ID
23(17)	8	ACEAAN	Access area name
31(1F)	1	ACEFCD	Function code
32(20)	8	ACEFN	Function name

CONFIGURATION CONTROL ELEMENT (CCE)

TOTAL SIZE: 25 bytes   DSECT: CCE

CREATED BY: Power Configuration Control

PURPOSE: The CCE control block is used in the power configuration control display processor to keep track of paging, which display is being viewed. It is created when DOMCFGD1 is initially PATCHed from Display Management System and is deleted when another display is requested.

POINTED TO BY: Control element address table (CEATAB)

Storage Map of CCE Table

DEC	HEX				
0	0	CCEDCE			
4	4	CCEFEMT			
8	8	CCES7CTA			
12	C	CCE#PT		CCEBPTS	
16	10	CCES7	CCEEMT	CCEPG#	CCETPGS
20	14	CCEINDS	CCET1	CCEUID	
24	18	CCEATTR			

Data Area Layout of CCE

Offset	Bytes and Bit Patterns	Field	Description
0(0)	4	CCEDCE	Address of display unit's CEATAB
4(4)	4	CCEFEMT	Address of RCB for first terminal displayed
8(8)	4	CCES7CTA	Address of S7CT entry for System/7
12(C)	2	CCE#PT	Total number of points in type

Offset	Bytes and Bit Patterns	Field	Description
14(E)	2	CCEBPTS	Number of points to bypass in type
16(10)	1	CCES7	System/7 ID
17(11)	1	CCEEMT	IBM 3707 ID
18(12)	1	CCEPG#	Number of current page being displayed
19(13)	1	CCETPGS	Total number of pages for current display
20(14)	Byte	CCEINDS	Indicators
	1...	CCEVI	PCC I being viewed
	.1..	CCEVII	PCC II being viewed
	..1.	CCEVIII	PCC III being viewed
	...1	CCERLS	On - do not release areas
	....1...	CCEEOD	On - point is not wired (status)
	.... .1..	CCESDY	On - same display - new page
	.... ..1.	CCESPG	On - refresh same page
	1	CCEBACK	On - page backward
21(15)	1	CCET1	Type of point displayed
22(16)	2	CCEUID	Unit ID of System/7 for PCC II
24(18)	1	CCEATTR	Attribute of System/7 ID for PCC II

DEVICE CONTROL TABLE (DCT)

TOTAL SIZE: 39 bytes

CREATED BY: Supervisory control

PURPOSE: A DCT control block is created for each device control action which is initiated and is deleted when the action is complete or is cancelled. The address of the DCT is stored in the DCT Index Table for reference and control by the device control processor (DOMCDC01).

Storage Map of DCT

DEC	HEX				
0	0	DCTADDR			
4	4	DCTRCB			
8	8	DCTDCTI			
12	C	DCTDCE or DCTECB			
16	10	DCTCEAT			
20	14	DCTS7	DCTACT	DCTTYPE	DCTI#
24	18	DCTAADDR			
28	1C	DCTINDS	DCTACTN		
32	20	DCTACTN			
36	24				

Data Area Layout For DCT

Offset	Bytes and Bit Patterns	Field	Description
0(0)	4	DCTADDR	Address of item in data base
4(4)	4	DCTRCB	Address of RCB
8(8)	4	DCTDCTI	Address of DCT index table item
12(C)	4	DCTDCE	Address of CEATAB (if display initiated)
or 12(C)	4	DCTECB	Address of ECB to be posted (if software initiated)
16(10)	4	DCTCEAT	Address of CEATAB entry
20(14)	1	DCTS7	System/7 ID
21(15)	1	DCTACT	Action to be taken (code)
22(16)	1	DCTTYPE	Type of device
23(17)	1	DCTI#	PATCH ID of PTIME for this device
24(18)	4	DCTAADDR	Address of section of options table which applies to device type
28(1C)	1	DCTINDS	Device Control flags
	1...	DCTDS	On - software initiated; Off - display initiated
	.1..	DCTETO	On - execute time-out
	..1.	DCTVTO	On - verify time-out

Offset	Bytes and Bit Patterns	Field	Description
	...1	DCTEXEC	On - being executed
	.... 1...	DCTEXED	On - device has executed
	.... .1..	DCTARMD	On - device is armed
	.... ..1.	DCTCANC	On - control cancelled by power system operator
	X		Reserved
29(1D)	10	DCTACTN	Action to be taken

INPUT/OUTPUT BLOCK, SYSTEM/7 EXTENSION

TOTAL SIZE: 64 bytes EXTENSION: 16 bytes DSECT: IOBDSECT

CREATED BY: Data acquisition initialization

PURPOSE: Contains necessary information for System/370 Energy Management System to control System/7 I/O.

Storage Map

DEC	HEX	Field			
-12	-C	CCW 8 BYTES			
-4	-4	ECB			
0	0	IOB			
32	20	IOBLID	IOBFLAG71	IOBFLAG72	IOBFLA73
36	24	IOBSIOAC		IOBSIOAW	
40	28	IOBTIME			
44	2C	IOBUSER			
48	30	IOBREQST			
52	34	Reserved			

Alphabetical Listing of Fields

Field	DEC	HEX
CCW	-12	-C
ECB	-4	-4
IOB	0	0
IOBFLAG71	33	21
IOBFLAG72	34	22
IOBFLAG73	35	23
IOBLID	32	20
IOBREQST	48	30
IOBSIOAC	36	24
IOBSIOAW	38	26
IOBTIME	40	28
IOBUSER	44	2C

Data Area Layout

Offset	Bytes and Bit Patterns	Field	Description
-12(-C)	8	CCW	Channel program

Offset	Bytes and Bit Positions	Field	Description
-4(-4)	4	ECB	The ECB
0(0)	32	IOB	Standard OS EXCP type IOB
32(20)	1	IOBLID	The logical ID associated with this System/7
33(21)	3	IOBFLAG7	
	1	IOBFLAG71	Byte one of flags
	1...	IOBCYCLE	IOB in cyclic read mode
	.1..	IOBBUFF	BUFFER = scan buffer
	.0..		BUFFER = utility buffer
	..1.	IOBUFFST	No buffer available
	...1	IOBLINE	System/7 offline
		IOBFAIL	
	.... 1...	IOBEXCP	EXCP outstanding
	xxx	RESERVED	
34(22)	1	IOBFLAG72	Byte two of flags
	1...	IOBTIMES	Timeout interval set
	.1..	IOBTOUT1	First timeout interval expired
	..1.	IOBSCAN	System/7 cyclic scanning
	...1	IOBISCAN	Initial scan complete but not cyclic scanning
	.... 1...	IOBTMODE	Timeout mode = scan
	xxx	RESERVED	
35(23)	1	IOBFLAG73	Byte three of flags
	xxxx xxxx	RESERVED	
36(24)	2	IOBSIOAC	Start I/O appendage constant
38(26)	2	IOBSIOAW	Start I/O appendage work area
40(28)	4	IOBTIME	Time I/O to be complete
44(2C)	4	IOBUSER	Address of DECB associated with this I/O operation
48(30)	4	IOBREQST	Address of ECB requesting I/O operation
52(34)	4	RESERVED	

SYSTEM/370 POINT SUMMATION MODULE DOMTPTSM

DOMTPTSM, which is built during system generation, will be loaded at initialization time.

SCAN ID(1)	LENGTH(3)	SUMMATION DURATION(2) FLAG(1) # OF POINTS(1)
	Name of sum point	(8)
	Name of point 1	(8)
	Name of point N ($0<N\le 8$)	(8)

- One scan ID block per scan with sum points
- Length - total number of bytes in point summation block for a scan
- Summation duration - the number of minutes elapsed before the sum total is replaced and not updated.
- Flag - ....1 type of sum point indicator

  0 = pulse counter
  1 = analog

POINT SUMMATION TABLE DOMTSMTB

This table is built at system initialization time and contains the information necessary for DOMTPSUM to do point summation.

```
SCAN ID A - Group 1 - No. of points to sum in this group (1-8)
                      Sum Total data point
                      Point No. 1 to be summed in this group

                      Point No. (2-8) to be summed in this group
                  .
                  .
                  .
            Group n


SCAN ID X - Group 1 - (same format as above)
```

The scan ID input as part of the summation data in module DOMTPTSM will be placed directly in the table. A group identifier will be input followed by a set of point names (summation point name and the names of the points to be summed) from which the addresses used in the above table are resolved for this group.

<u>SCAN BUFFER POOL ADDRESS LIST</u>

ENTRY SIZE:	4 bytes per entry
TOTAL SIZE:	4 * # scan buffers
CREATED BY:	Initialization (DOMTP1IN)
PURPOSE:	List of addresses of all scan buffers for this logical ID.

POINTED TO BY: RDAPADDR field of scan buffer pool locator list

Storage Map

DEC	HEX	Field
0	0	BUFFLAGS | BUFFADDR

Alphabetical List of Fields

Field	DEC	HEX
BUFFADDR	1	1
BUFFLAGS		

Storage Map

Offset	Bytes and Bit positions	Field	Description
0(0)	1	BUFFLAGS	Flags
	1...		Last address in list
	.1..		Last buffer allocated
1(1)	3	BUFFADDR	Address of buffer prefix

SCAN BUFFER POOL LOCATOR LIST

ENTRY SIZE: 4 bytes per entry

TOTAL SIZE: 4 * # of logical IDs

CREATED BY: Initialization (DOMTP1IN)

PURPOSE: Point to the buffer pool for each logical System/7

POINTED TO BY: RDABUFFP entry of the IOCVT

Storage Map

DEC	HEX	Field
0	0	RDALID | RDAPADDR

Alphabetical List of Fields

Field	DEC	HEX
RDALID	0	0
RDAPADDR	1	1

Data Area Map

Offset	Bytes and Bit positions	Field	Description
0(0)	1	RDALID	Logical ID of the pool
1(1)	3	RDAPADDR	Address of pool address list

System/7 CHECKPOINT ARRAY ID LIST

ENTRY SIZE: 4 bytes

TOTAL SIZE: Determined during system generation

CREATED BY: System/7 system generation DSECT: CKPTALST

PURPOSE: Define supported array IDs and sectors.

POINTED TO BY: CKPTARAY field of CKPTDSCT

Storage Map

DEC	HEX	Field	
0	0	ARAYID	ARAYBLK

Alphabetical List of Fields

Field	DEC	HEX
ARAYID	0	0
ARAYBLK	2	2

Data Area Map

Offset	Bytes and Bit positions	Field	Description
0 (0)	2	ARAYID	Array ID or x'FFFF' which indicates the end of the list
2 (2)	2	ARAYBLK	Relative block number in the checkpoint data set to the first sector for this array ID NOTE: on last entry, ARAYID=x'FFFF', this value is the number of records in the file. Subtract two consecutive ARAYBLK fields and obtain the number of sectors in the first array ID.

SYSTEM/7 DECB

TOTAL SIZE: 24 bytes

CREATED BY: Initialization (DOMTINFO) DSECT: DOMTDECB

PURPOSE: The DECB plus extension for I/O.

POINTED TO BY: DECBADDR field of the DECB address list

Storage Map

DEC	HEX	Field	
0	0	PDSECB	
4	4	PDSTYPE	PDSLNGTH
8	8	PDSDCB	
12	C	PDSAREA	
16	10	PDSIOB	
20	14	PDSBLOCK	PDSRECPT

Alphabetical List of Fields

Field	DEC	HEX
PDSAREA	12	C
PDSBLOCK	20	14
PDSDCB	8	8
PDSECB	0	0
PDSIOB	16	10
PDSLNGTH	6	6
PDSRECPT	22	16
PDSTYPE	4	4

Data Area Map

Offset	Bytes and Bit positions	Field	Description
0 (0)	4	PDSECB	event control block
4 (4)	2	PDSTYPE	type of I/O request
6 (6)	2	PDSLNGTH	length of data
8 (8)	4	PDSDCB	address of DCB
12 (C)	4	PDSAREA	address of data
16 (10)	4	PDSIOB	address of IOB
20 (14)	2	PDSBLOCK	length of current block
22 (16)	2	PDSRECPT	displacement into current block of current logical record

System/7 I/O BUFFER

TOTAL SIZE: 4 + BUFFLENG

CREATED BY: Initialization (DOMTP1IN) DSECT: BUFFMAP

PURPOSE: Define System/7 input buffers

POINTED TO BY: BUFFADDR field of scan buffer pool address list or SMLBUFFP in IOCVT

Storage Map

DEC	HEX	Field		
0	0	BUFFLENG	BUFFLAST	BUFFALOC
4	4	BUFFERP		

Alphabetical List of Fields

Field	DEC	HEX
BUFFALOC	3	3
BUFFERP	4	4
BUFFLAST	2	2
BUFFLENG	0	0
BUFFPRE	0	0

Data Area Map

Offset	Bytes and Bit positions	Field	Description
0(0)	4	BUFFPRE	Buffer prefix
0(0)	2	BUFFLENG	Length of BUFFERP field
	1...	BUFFDYN	Dynamic GETWA BUFFER (this bit does not figure in length)
2(2)	1	BUFFLAST	X'FF' indicates last buffer in chain
3(3)	1	BUFFALOC	X'FF' indicates buffer allocated
4(4)	N	BUFFERP	Buffer (length obtained from BUFFLENG field)

Note: These buffers are chained such that the address of the next buffer is calculated as the address of BUFFLENG plus the length of the prefix plus the length of the buffer proper

SYSTEM/7 I/O CONTROL TABLE (IOCVT)

TOTAL SIZE: 328 bytes

CREATED BY: Initialization (DOMTP1IN) DSECT: IOCVT

PURPOSE: Provide a work area for the System/7 I/O load module and provide pointers to other I/O control blocks.

Storage Map

DEC	HEX	Field
0	0	SAVEA1
72	48	SAVEA2
144	90	ICVTXCVT
148	94	ECBLIST
152	98	READLIST
156	9C	WRITLIST
160	A0	EIPLIST
160	A0	PURGECB
164	A4	STIMER
168	A8	INTERVAL
172	AC	STCKAREA
180	B8	PINTRVAL
184	BC	BINTRVAL
188	C0	OINTRVAL
192	C4	SMLBUFFP
196	C8	RDABUFFP
200	CC	ROUTABLE
204	D0	WORK1
208	D4	WORK2
212	D8	PURGLIST
232	E8	PROBLIST
240	F0	FOSUPL
240	F0	ESUPL
284	11C	QQ
284	11C	IOMSG
284	11C	WORK3
288	120	WORK4
324	144	Reserved

Alphabetical List of Fields

Field	DEC	HEX
BINTRVAL	184	BC
ECBLIST	148	94
EIPLIST	160	A0
ESUPL	240	F0
FOSUPL	240	F0
ICVTXCVT	144	90
INTERVAL	168	A8
IOMSG	284	11C
OINTRVAL	188	C0
PINTRVAL	180	B8
PROBLIST	232	E8
PURGECB	160	A0
PURGLIST	216	D8
QQ	284	11C
RDABUFFP	192	C4
READLIST	152	98
ROUTABLE	200	CC
SAVEA1	0	0
SAVEA2	72	48
SMLBUFFP	192	C4
STCKAREA	172	AC
STIMER	164	A4
WORK1	204	D0
WORK2	208	D4
WORK3	284	11C
WORK4	288	120
WRITLIST	156	9C

Data Area Map

Offset	Bytes and Bit positions	Field	Description
0(0)	72	SAVEA1	Save area used by CSECT DOMTS7IO
72(48)	72	SAVEA2	Save area used by CSECTS DOMTPUNT and DOMTIOER
144(90)	4	ICVTXCVT	Address of DPPXCVT
148(94)	4	ECBLIST	Address of the ECB list
152(98)	4	READLIST	Address of read segment of ECB list
156(9C)	4	WRITLIST	Address of write segment of ECB list
160(A0)	4	EIPLIST	Address of IPL segment of ECB list
160(A0)	4	PURGECB	Address of PURGE request ECB
164(A4)	4	STIMER	Address of STIMER exit routine ECB
168(A8)	4	INTERVAL	Interval for STIMER macro
172(AC)	8	STCKAREA	Work area to store system clock
180(B8)	4	PINTRVAL	Primary time-out interval divided by 2
184(BC)	4	BINTRVAL	Backup time-out interval divided by 2
188(C0)	4	OINTRVAL	Output time-out interval
192(C4)	4	SMLBUFFP	Address of first buffer prefix in utility buffer pool
196(C8)	4	RDABUFFP	Address of scan buffer pool locator list
200(CC)	4	ROUTABLE	Address of transaction routing table
204(D0)	4	WORK1	Work area
208(D4)	4	WORK2	Work area
212(D8)	20	PURGLIST	PURGE SVC parameter list
232(E8)	8	PROBLIST	PATCH PROBL
240(F0)	44	FOSUPL	PATCH SUPL
240(F0)	44	ESUPL	PATCH SUPL
284(11C)		QQ	MESSAGE list form with space for 8 routing codes and 4 variables
284(11C)		IOMSG	Redefine QQ
284(11C)	4	WORK3	Work area
288(120)	4	WORK4	Work area
324(144)	36		Reserved

DATA SETS

This section describes the data sets used by the System/370 Energy Management System.

DECB ADDRESS LIST FOR SYSTEM/7 SUPPORT DATA SET

ENTRY SIZE: 4 bytes per entry

TOTAL SIZE: 4 * the number of locally attached logical System/7s

CREATED BY: Initialization (DOMTINFO)

PURPOSE: Provide address list to the DECBs for each System/7 support data set defined.

POINTED TO BY: DECBLIST field of the FCVT

Storage Map per entry

DEC	HEX	Field	
0	0	DECBLID	DECBADDR

Alphabetical List of Fields

Field	DEC	HEX
DECBADDR	1	1
DECBLID	0	0

Data Area Map

Offset	Bytes and Bit patterns	Field	Description
0 (0)	1	DECBLID	Logical ID of this DECB
1 (1)	3	DECBADDR	Address of the DECB

DOTHCKPT MEMBER FORMAT

ENTRY SIZE: 4 bytes

CREATED BY: System/7 system generation

PURPOSE: Define those array ids and number of sectors per ID that are subject to checkpoint processing.

POINTED TO BY: This is a member of the System/7 support data set

Note: The data is in FORMAT/7 output form. This description is the format of this data if it were in machine-readable form.

Storage Map

DEC	HEX	Field	
0	0	CKPTAID	CKPTSECT

Alphabetical List of Fields

Field	DEC	HEX
CKPTAID	0	0
CKPTSECT	2	2

Data Area Map

Offset	Bytes and Bit positions	Field	Description
0 (0)	2	CKPTAID	Array ID
0 (2)	2	CKPTSECT	Number of sectors this array ID

DOTHCORR MEMBER FORMAT

ENTRY SIZE: 12 bytes

CREATED BY: System/7 System generation

PURPOSE: Provide a list of PDS member names for System/7 disk load.

POINTED TO BY: This is a member of the System/7 support data set

Note: This data set does not exist if the System/7 is not generated with a local disk. The data is in FORMAT/7 output form. This description is the format of this data if it were in machine loadable form.

Storage Map

DEC	HEX	Field	
0	0	DOTHNAME	
8	8	DOTHSECT	DOTHCNT

Alphabetical List of Fields

Field	DEC	HEX
DOTHCNT	10	A
DOTHNAME	0	0
DOTHSECT	8	8

Data Area Map

Offset	Bytes and Bit positions	Field	Description
0 (0)	8	DOTHNAME	Name of the member of the partitioned data set
8 (8)	2	DOTHSECT	Relative sector on System/7 this member resides
10 (A)	2	DOTHCNT	Number of sectors this member

EVENTD: EVENT LOG FILE

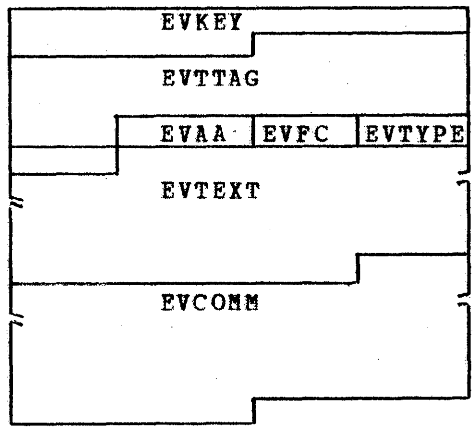
TOTAL SIZE: 142 bytes per second DSECT: EVENTD

CREATED BY: DOMCEVT1

PURPOSE: Inside this area is constructed the event record to be written to the event file.

Storage Map of EVENTD

DEC	HEX
0	0
4	4
8	8
12	C
16	10
20	14
72	48
140	8C



Alphabetical List of Fields in EVENTD

Field	DEC	HEX
EVAA	17	11
EVCOMM	75	4B
EVDATA	6	6
EVDATE	6	6
EVDAY	8	8
EVFC	18	12
EVHOUR	11	B
EVKEY	0	0
EVMIN	13	D
EVSEC	15	F
EVTIME	11	B
EVTTAG	6	6
EVTYPE	19	13
EVYEAR	6	6

Data Area Layout of EVENTD

Offset	Bytes and Bit Patterns	Field	Description
0(0)	6	EVKEY	Key in packed decimal, YYDDDNNMMSSF
6(6)	11	EVTTAG	Time TAG
6(6)	5	EVDATE	Date
6(6)	2	EVYEAR	Year
8(8)	3	EVDAY	Julian day
11(B)	6	EVTIME	Time event occurred
11(B)	2	EVHOUR	Hour event occurred
13(D)	2	EVMIN	Minute event occurred
15(F)	2	EVSEC	Second event occurred
17(11)	1	EVAA	Event access area code
18(12)	1	EVFC	Event function area code
19(13)	2	EVTYPE	Event type
21(15)	54	EVTEXT	Event text
75(4B)	67	EVCOMM	Event comment

SYSTEM/7 CHECKPOINT RECORD FORMAT

TOTAL SIZE: 260 bytes

CREATED BY: Initialization (DOMTINFO) DSECT: CKPTREC

PURPOSE: Contain System/7 checkpoint data

Storage Map

DEC	HEX	Field	
0	0	CKPTAID	CKPTSECT
4	4	CKPTDATA	

Alphabetical List of Fields

Field	DEC	HEX
CKPTAID	0	0
CKPDATA	4	4
CKPTSECT	2	2

Data Area Map

Offset	Bytes and Bit positions	Field	Description
0(0)	2	CKPTAID	Array ID
2(2)	2	CKPTSECT	Relative sector number
4(4)	256	CKPTDATA	Sector data

MACRO PARAMETER DSECTS

This section shows the contents of the System/370 Energy Management System macros. In addition, information such as the total size of the macro, how the macro is created, the purpose of the macro, and how the area for the macro is found in main core, is provided for each macro.

SCVEND (SCEVENT MACRO)

TOTAL SIZE: 76 bytes DSECT: SCVEND

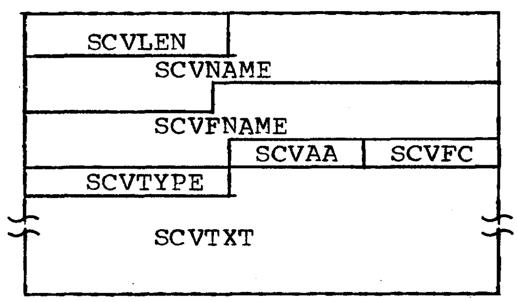
CREATED BY: SCVEND macro expansion

PURPOSE: Pass the event macro parameters to DOMCEVT1 which builds an event record and adds it to the event file.

POINTED TO BY: Parameter list address from the SCVEND macro

Storage Map of SCVEND

DEC	HEX
0	0
4	0
8	4
12	C
16	10
20	14
24	18
72	48



Alphabetical List of Fields in SCVENTD

Field	Dec	Hex
SCVAA	0018	0012
SCVFC	0019	0013
SCVFNAME	0010	000A
SCVLEN	0000	0000
SCVNAME	0002	0002
SCVTXT	0022	0016
SCVTYPE	0020	0014

Data Area Layout of SCVEND

Offset	Bytes and Bit Patterns	Field	Description
0 (0)	2	SCVLEN	Length of SCVTXT
2 (2)	8	SCVNAME	Access area name
10 (A)	8	SCVFNAME	Function code name
18 (12)	1	SCVAA	Access area code
19 (13)	1	SCVFC	Function code ID
20 (14)	2	SCVTYPE	Event type
22 (16)	54	SCVTXT	Event text

STAE Processor Parameter List

TOTAL SIZE: 8 bytes per entry if the re-initialize area flag is off (FLAG1)
12 bytes per entry if re-initialize flag (FLAG1) is on

CREATED BY: The maximum area per task is reserved at initialization. The user program fills in the necessary information.

PURPOSE: Pass parameters to the STAE processor (DOMCSTAE)

Storage Map

DEC	HEX			
0	0	FLAG1	FLAG2	LENGTH
4	4	ADDRESS OF AREA		
8	8	RE-INITIALIZATION DATA		
		X'FF'		

Data Area Layout

Offset	Bytes and Bit Patterns	Field	Description
0(0)	1	FLAG1	Flag byte
	1...		On- ABEND to occur
	.1..		On- FREEMAIN area
	..1.		On- FREEWA area
	...1		On- zero area
	.... 1...		On- re-initialize area
	.... .1..		On- 'AND' bits
	.... ..1.		On- 'OR'bits
	1		On- release message input buffer
1(1)	1	FLAG2	Flag byte
	1...		On- chain of areas to FREEMAIN or FEEEWA
	.xxx xxxx		Reserved
2(2)	2	LENGTH	Length of area or if 'AND' or 'OR' flag is on, the mask to be used (in the second byte of the field)
4(4)	4		The address of the area.
8(8)	8		The data with which the field in address is to be re-initialized. (This field is nct present in the parameter list, if the re-initialization flag is not on.)
End of List	1	X'FF'	End of list indicator for the STAE processor.

The areas are chained so that each entry is either 8 or 12 bytes depending upon the contents of the first byte of each entry (re-inititialize flag). The end of the chain is indicated by one byte of X'FF'.

0	FLAG1	FLAG2	LENGTH	re-init flag (FLAG1) On
4	ADDRESS OF AREA			
8	RE-INITIALIZATION DATA			

0	FLAG1	FLAG2	LENGTH	re-init flag (FLAG1) Off
4	ADDRESS CF AREA			
8	X'FF'			

S7WRITE AND S7IPL PARAMETER LIST

TOTAL SIZE: 12 bytes

CREATED BY: S7WRITE or S7IPL user's DSECT:USERDECB

PURPOSE: Pass parameters to System/7 output interface program DOMTWRIT.

Storage Map

DEC	HEX			
0	0	ECB		
4	4	TYPE	LID	Length of data
8	8	Unit Index	Address of data	

Data Area Layout by Section

Offset	Bytes and Bit Patterns	Field	Description
0(0)	4	USERECB	ECB used by DOMTWRIT
4(4)	1	USERTYPE	Type of output
	1...	USERIPL	IPL request
	.1..	USERTXT	TX record - no read
	..1.	USEREND	EN record - start cyclic read
	.... 1...	USERMSG	Message has transaction header
	.... .1..	USERUNIT	Unit index explicitly given
	....X..XX		Reserved
5(5)	1	USERLID	System/7 logical ID
6(6)	2	USERLNGH	Length of data
8(8)	1	USERID	Target System/7 unit index
9(9)	3	USERDATA	Address of data to be written

Alphabetical List of Fields

Field	Dec	Hex
USERDATA	0009	0009
USERECB	0000	0000
USERID	0008	0008
USERLID	0005	0005
USERLNGH	0006	0006
USERTYPE	0004	0004

VARYS7 PARAMETER LIST

TOTAL SIZE: 12 bytes

CREATED BY: User of VARYS7 macro

PURPOSE: Pass parameters to VARYS7 processor, DOMTVS7.

Storage Map

DEC	HEX	Field		Description
0	0	RESERVED	LID	UNIT DD
4	4	STATUS		
8	8			

Data Area Layout by Section

Offset	Bytes and Bit Patterns	Field	Description
0 (0)	1		Reserved
1 (1)	1	LID	Reserved
2 (2)	2	UNITDD	System/7 unit DD qualifier
4 (4)	8	STATUS	Desired status ('PRIMARYb̸', 'OFFLINEb̸' or 'BACKUPb̸b̸')

APPENDIX A TABLE OF CONTENTS

DOMCLGET

This macro provides the user with the capability of retrieving sensor based data from the logged history data base or the incore data base. The address of the buffer containing the retrieved data is returned via general purpose register one. This buffer must be freed when the user has completed using it by issuing a DOMCFREE macro.

The first byte of the buffer contains control flags, and the next three bytes contain the length of the data which begins in byte number four.

Note: A 72-byte user save area pointed to by general purpose register 13 must be provided by the calling program.

[symbol]	DOMCLGET	DBLOC = { current history } TYPE = { RCB analog counter status } [,RMT =({ (r) System/7 ID } , { (r) RMT ID })] [, FEEDBCK = { (r) address } , TIME = { (r) address } , STEP = { (r) value }] [, MF = { L (E, { (r) address }) }] [, { DCVTR= (r) DCVTLOC = { (r) address } }]

DOMCLGET RETURN CODES

0	Successful completion
8	Data requested not found in data base
12	Not enough core available for return data buffer.
16	Invalid macro ID

Note: It is the user's responsibility to check the return code.

DBLOC

Specifies the retrieval location and method. 'CURRENT' indicates the incore data base is to be used and 'HISTORY' indicates retrieval will be from the logged history data base.

TYPE

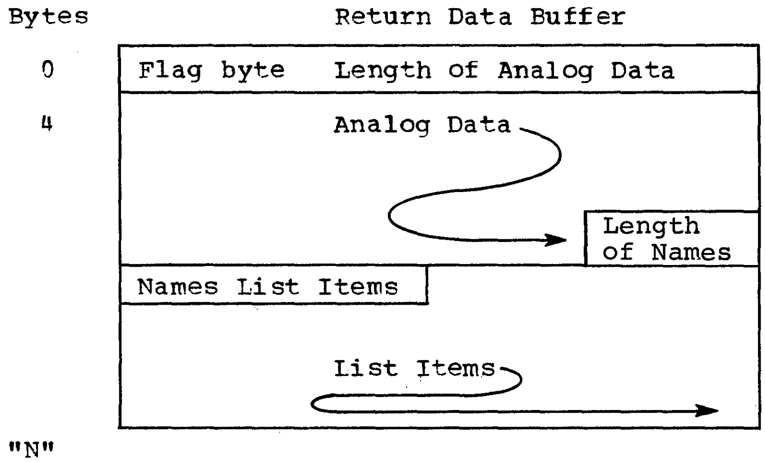
Specifies the type of remote data desired.

RCB

This causes the remote control block to be retrieved.

ANALOG

This will cause retrieval of the analog data and names list items associated with that data. The returned buffer will look as follows:



COUNTER

This causes retrieval of pulse counter data.

STATUS

This causes retrieval of status data.

RMT

The RMT parameter identifies the remote for which data is to be retrieved.

System/7 ID

The System/7 ID is a value or a general purpose register, enclosed in parentheses, which contains a value, indicating a valid System/7 ID number from 1-99.

RMT ID

This is a value or a general purpose register, enclosed in parentheses, containing a value which indicates a valid IBM 3707 ID from 0-251.

FEEDBCK

address

This parameter is a label of a 32-byte core location to be used by the DOMCLGET macro to store feedback locator information to be used in updating retrieved history data base items via the DOMCLPUT macro.

(r)

This parameter is a general purpose register contained in parentheses containing the address of a 32-byte core location to be used by the DOMCLGET macro to store feedback locator information to be used in updating retrieved items via the DOMCLPUT macro.

The FEEDBCK parameter is ignored if DBLOC=CURRENT is specified.

TIME

address

'address' specifies an address of a 6-byte core location or

(r)

'r' specifies an address of a 6-byte core location in a general purpose register enclosed in parentheses ... which contains a time and day field beginning on a fullword boundary. The first four (4) bytes will contain a time in ten millisecond units. The last two bytes will contain a binary value fron 1-366 representing the day of the year. This time and day will be used as a comparison value to establish a relative starting pcint to determine which copy of the array will be retrieved from the logged history data set.

The TIME parameter is ignored if DBLOC=CURRENT is specified.

STEP

(r)

'r' is a general purpose register, enclosed in parentheses, containing a positive or negative value.

value

'value' is a signed integer ... which is used to determine which copy of a logged array, relative to the TIME prameter, will be retrieved from the logged history data base. The value is a signed number which may be either positive, negative, or zero. If a register is specified, the number will be binary, and in two's complement notation if negative. If no sign is specified, the number is assumed to be positive. The STEP parameter is ignored if DBLOC=CURRENT is specified.

MF

L

The 'L' parameter indicates that the list form of the macro is used to create a parameter list that can be referenced by an execute form of the DOMCLGET macro instruction. Register notation is nct allowed when using the list form of the macro.

E

The 'E' parameter indicates that the execute form of the macro is used and an existing parameter list exists.

E, (r)

The 'r' parameter is a general purpose register containing the address of the parameter list to be used.

E, address

The 'address' parameter is the symbolic label associated with the parameter list to be used.

DCVTR

The 'r' parameter is a general purpose register containing the address of the DPPXCVT.

DCVTLOC

(r)

The 'r' parameter is a general purpose register containing the address of a four-byte core location which contains the address of the DPPXCVT.

address

This parameter is the location of a four byte core location which contains the address of the DPPXCVT.

DOMCLGET Examples: These examples will not describe the Assembler Language statement used to call the DOMCLGET macro, but will describe the response of the DOMCLGET routine to the different combinations of the TIME parameter with the STEP parameter.

1. TIME is specified and STEP = 0 -- An attempt will be made to retrieve a copy of the array logged at the exact time specified. If the array is not logged at that exact time, the first copy of the array logged after that time is retrieved.

2. TIME is specified and STEP = -2 -- The second copy of the array logged prior to the time specified is retrieved.

3. TIME is specified and STEP = +5 -- The fifth copy of the array logged after the time specified is retrieved. If the STEP parameter is positive, it will not retrieve past the last logged copy of the array.

DOMCLPUT

This macro provides the user the ability to update previously retrieved sensor based data.

[symbol]	DOMCLPUT	DBLOC = {CURRENT / HISTORY} , TYPE = {ANALOG / COUNTER}, RMT = ({(r) / System/7}, {(r) / RMT ID}), DATA = {(r) / address} [,FEEDBCK = {(r) / address}] [, MF = {L / (E, {(r) / address})}] [, {DCVTR=(r) / DCVTLOC=} {(r) / address}]

Note: The user must provide a 72 byte save area pointed to by general purpose register 13.

DOMCLPUT RETURN CODES

0	Successful completion
4	Remote not in an offline status for current data - data not updated
8	Data item to be updated not found in data base - data not updated
12	Not enough core available for DOMCLPUT buffer areas.
16	Invalid macro ID

Note: It is the user's responsibility to check the return code.

DBLOC

Specifies the update location and method. 'CURRENT' indicates the incore data base is to be updated and 'HISTORY' indicates the logged history data base is to be updated. This parameter is required in the standard form of the macro.

Note: 'CURRENT' data is updated only if the remote is in an offline status.

TYPE

This parameter specifies the type of remote data to be updated. 'ANALOG' causes the updating of analog data items. Names List items will not be updated when 'ANALOG' is specified. 'COUNTER' causes updating of counter data items to take place. This parameter is required in the standard form of the macro.

RMT

The RMT parameter identifies the remote for which data is to be retrieved. This parameter is required in the standard form of the macro.

SYSTEM/7 ID

The System/7 ID is a value or a general purpose register, enclosed in parentheses, which contains a value, indicating a valid System/7 ID number from 1-99.

RMT ID

This is a value or a general purpose register containing a value which indicates a valid IBM 3707 from 0-251.

DATA

This parameter is required in the standard form of the macro.

address

If 'address' is specified, it is a symbolic label of a core location containing the data to be used in updating the data base.

(r)

If 'r' is specified, it is a general purpose register and contains the address of a core location containing the data to be used in updating the data base.

Note: The data pointed to by the DATA parameter must be an exact duplicate in size and type of the data it is replacing. The address must point to the first byte of data to be updated.

FEEDBCK

address

If 'address' is specified, it is a label of a core location of locator data associated with a particular 'RMT ID' which has been previously filled in via the DOMCLGET macro. This data will be used to locate and update the remote data identified in the RMT parameter.

(r)

If 'r' is specified, it is a general purpose register which contains the address of a core location containing a remote locator data filled in previously by a DOMCLGET macro execution.

MF

L

The 'L' parameter indicates that the list form of the macro is used to create a parameter list that can be referenced by an execute form of the DOMCLPUT macro instruction. Register notation is not allowed when using the list form of the macro.

E

The 'E' parameter indicates that the execute form of the macro is used and an existing parameter list exists.

E, (r)

The 'r' parameter is a general purpose register containing the address of the parameter list.

E, address

The 'address' parameter is the symbolic label associated with the parameter list to be used.

DCVTR

The 'r' parameter is a general purpose containing the address of the DPPXCVT.

DCVTLOC

(r)

The 'r' parameter is a general purpose register, enclosed in parentheses, containing the address of a four-byte core location which contains the address of the DPPXCVT.

address

This parameter is the location of a four byte core location which contains the address of the DPPXCVT.

DOMCFREE

This macro provides the user the capability to release control of a returned buffer obtained via the DOMCLGET macro. This macro, DOMCFREE, should be issued when the user has completed use of the buffer obtained through the DOMCLGET macro.

Note: A 72-byte user save area pointed to by general register 13 must be provided by the calling program.

[symbol]	DOMCFREE	{ (r) | address } [,{ DCVTR=(r) | DCVTLOC= { (r) | address } }]

DOMCFREE RETURN CODES

0 Successful completion

4 Invalid DOMCFREE

- Core already free
- Invalid address

Note: It is the user's responsibility to check the return code.

(r)

If 'r' is specified, the general purpose register contains the address of the buffer returned after a DOMCLGET macro execution.

address

If 'address' is specified, it is a label of a fullword that contains the address of the buffer as returned to the caller after a DOMCLGET macro execution.

DCVTR

The 'r' parameter is a general purpose register containing the address of the DPPXCVT.

DCVTLOC

(r)

The 'r' parameter is a general purpose register containing the address of a four-byte core location which contains the address of the DPPXCVT.

address

This parameter is the address of a four byte core location which contains the address of the DPPXCVT.

SCEVENT

The SCEVENT macro provides the user the capability to enter an event into the system. Supervisory Control adds the event to the Event Log File and routes it to the general event typer. If the user specifies an access area, the event is also routed to the typers assigned to that access area.

[symbol]	SCEVENT	ETEXT = { 'text' | text addr | (r) } , { TYPE = { SCAN | SECURITY | DATA | SYSTEM | ALARM | DEVICE | CONFIG } | TYPELOC = { type addr | (r) } } [, { AA = name | AAID = { (r) | n } }] [, { DCVTR = r | DCVTLOC = addr }] [, { FUNC = name | FUNCID = { n | (r) } }] , AALOC= { aa addr | (r) } [, FUNCLOC = { fc addr | (r) }] [, MF= { L | (E, { (r) | addr }) }]

ETEXT

The text of the message up to 54 characters in length, enclosed in single quotation marks.

ETEXT = text addr
(r)

The address of the message in which the first fullword contains the length of the message in the first halfword and zeros in the second halfword.

TYPE

The type of event

TYPELOC = type addr
(r)

Address of the type of event

AA = name

The customer-defined name of the access area to which the event is to be routed

```
        (r)
AAID =   n
```

The identifying code for the access area (r) indicates that the user has stored the access area ID in the low-order byte of general register r.

```
AALOC = aa addr
           (r)
```

The address of the access area name

DCVTR = r

Register R contains DPPXCVT address

DCVTLOC = addr

Address of variable containing the addr of the DPPXCVT

MF =

(L)

Indicates list form

```
(E, addr)
    (r)
```

Indicates execute, and addr contains the address of the list

FUNC = name

The customer-defined name of the function code of the event.

```
        (r)
FUNCID = n
```

The function code ID itself (n) or contained in the low-order byte of register (r).

```
FUNCLOC = fcaddr
           (r)
```

Address of the function code name

RETURN CODES

00	EVENT WAS ADDED
04	Specified length GT max. allowed (default tc max.)
08	NO GETWA AREA AVAILABLE
12	INVALID TYPE SPECIFIED
16	PATCH DID NOT WORK - PATCH to DOMCEVT1
20	INVALID LENGTH
32	AA and/or FC specified wrong

DOMCALRM

The DOMCALRM macro enables the user to cause an alarm to be generated in the system. The alarm becomes part of the incore detail alarm list, and a general alarm is issued on an access area basis. The detail alarm is routed to the appropriate units (displays, typers, etc.) as defined by the customer at system generation time. The customer defines the units by means of access area and function codes or by routing codes. The former are used for display units and the latter for other devices which would use the Message Handler facility of Special Real Time Operating System.

The detail alarm generated remains part of the incore alarm list until it is deleted by user action through the detail alarm display or through the macro DOMCALRM specifying TYPE=DEL.

The user may code a non-sensor based alarm message or code the sensor based point name and alarm condition allowing the alarm processor to build the alarm message as it would from the scan exception table (SET).

The alarm control block refers to an entry in the CAAPPL array and is used for the non-sensor based alarms. The APPOINT parameter refers to an entry in the CAAPLPT array and is also used for non-sensor based alarms. Refer to the Data Areas (Arrays) section for further information on these two arrays.

[symbol]	DOMCALRM	see operands below

```
{ { {SYS7= {(r) | id} | SYS7LOC= {(r) | addr}}        {EMT= {(r) | id} | EMTLOC= {(r) | addr}} }
    {APPOINT= name | APPLOC= {(r) | addr}}            {APACB= name | ACBLOC= {(r) | addr}} }

  { {SBPOINT= name | SBPLOC= {(r) | addr}} {COND= {(r) | code} | CONDLOC= {(r) | addr}}
        {PTYPE= {STATUS | PC | ANALOG} | PTLOC= {(r) | addr}}
    {ATEXT= ' ' | ATXTLOC= {(r) | addr}} } }

  [,TYPE= {ADD | DEL}] [,MF= {L | (E, {(r) | addr})}]

  [,{DCVTR= r | DCVTLOC= {(r) | addr}}]
```

SYS7 — The two-digit System/7 logical ID which may be stored in the low-order byte of the register specified by (r).

SYS7LOC — The address of a one-byte area containing the System/7 logical ID in hexadecimal.

EMT — The three-digit IBM 3707 identifier which may be stored in the low-order byte of the register specified by (r).

EMTLOC — The address of a one byte area containing the IBM 3707 identifier in hexadecimal.

APPOINT — The 7-character item name for the non-sensor based alarms.

APPLOC — The address of the 7-character item name for non-sensor based alarms.

APACB — The 8-character name of the alarm control block for non-sensor based alarms.

ACBLOC — The address of the 8-character name of the alarm control block for non-sensor based alarms.

ATEXT — The text of the alarm message which is up to 40 bytes in length and enclosed in single parentheses.

ATXTLOC — The address of the alarm message text which is 40 bytes in length.

SBPOINT — The seven-byte name of the sensor based point which caused the alarm (as system generated through the point definition macro).

SBPLOC — The address of an area containing the seven-byte name of the sensor based point which caused the alarm (as system generated through the point definition macro).

COND — The two digit code of the condition which caused the alarm. The code is stored in the low order byte of the register specified by (r). (See Note 1.)

CONDLOC — The address of an area containing the one-byte hexadecimal code representing the condition which caused the alarm. (See Note 1.)

PTYPE — The type of sensor-based point which is being alarmed. The user codes STATUS, PC, or ANALOG.

PTLOC — The address of the type of sensor based point which is being alarmed. The word STATUS, PC, or ANALOG must be located at the address coded.

TYPE — The type of alarm processing to be done. ADD indicates to add the detail alarm to the incore alarm list and is the default value. DEL indicates to delete the alarm from the incore alarm list.

DCVTR — The address of the DPPXCVT contained in register (r). This operand enables the routine to bypass generating code to find the DPPXCVT by way of the OS CVT and TCB chain in order to retrieve the address of the DPPXCVT from the TCB extension.

DCVTLOC — The address of an area containing the address of the DPPXCVT.

Notes:

1. The condition codes and their text descriptions are as follows:

Code	Text
00	(BLANK)
01	WARNING LIMITS
02	CTR ROLLOVER
03	OUT OF LIMITS
04	TRNSFR OVERRUN
05	MISSING DATA
06	OFFSCALE
07	PARITY ERROR
08	BCH CHECK
09	EXEC FAILED
0A	REASON CHECK
0B	ADC FAILURE
0C	LOSS ACCUM SIG
0D	XMSN MISSING
0E	OPENED
0F	TRIPPED
10	CLOSED
11	TCT-AUTOMATIC
12	GEN LIMIT
13	TCT-MANUAL
14	MOS-STUCK
15	MOS-HARDWARE
16	TCT-LIMIT
17	LOW WARN LIMIT
18	LOW OPER LIMIT
19	OFFSCALE-LOW
20	HIGH WARN LIMIT
21	HIGH OPER LIMIT
22	OFFSCALE-HIGH

Condition code descriptions are 14 bytes in length and appear on the detail alarm display and as part of the alarm messages sent to the typers.

The texts are contained in message numbers 328 and 329 of the message data set. Codes and descriptions may be added to cover alarm conditions on sensor based points which are not incorporated in the above list. Up to 32 conditions may be used (Codes 00-31). The texts are defined offline using the Special Real Time Operating System message definitions facility.

2. The user may specify the alarm text or the sensor based point and condition code operands. If the user specifies the latter operands, the alarm processor generates the detail alarm text in the same manner as alarms passed via the scan exception table.

If both methods are used to code an operand, for example,SYS7 and SYS7LOC, the macro generates a warning and uses the -LOC operand. However, in the case of the PTYPE and PTLOC operands, the PTYPE operand is used.

RETURN CODES

The macro returns the following codes in register 15:

00	Successful completion
04	Invalid types found at location pointed to by PTLOC operand
08	PATCH of alarm processor failed
12	No GETWA core available at this time
16	Conflicting parameters
20	Missing parameter

DOMCALRM generates a parameter list which is passed to DOMCALM2. The format of the fixed portion of the parameter list is as follows:

DEC	HEX		
0	0	FLAGS	ADDRESS OF System/7 ID/APPOINT
4	4	ADDRESS OF IBM 3707 ID/ACB	
8	8	ADDRESS OF POINT NAME/TEXT	
12	C	ADDRESS OF COND. CODE or zero	
16	10	ADDRESS OF POINT TYPE or zero	

The variable portion of the parameter list contains the information coded with the SYS7, EMT, APPOINT, APACB, ATEXT, SBPOINT, and COND codes.

The macro flags are as follows:

Bit 0 and 1	01 - Analog data 10 - Status data 11 - Pulse counter data if PTYPE= is coded; otherwise, 00
Bit 2	On - if delete
Bit 3	On - if condition code present
Bit 4	On - if point name present
Bit 5	On - if APPOINT present
Bit 6	On - if ACB present
Bit 7	On - if text present

The displacement, length and description of the list entries are as follows:

Displacement	Length	Description
0	1	Macro flags - as above
1	3	Address of the System/7 ID or the APPOINT name
4	4	Address of the IBM 3707 ID or the ACB name
8	4	Address of the point name or the address of the alarm message text
12	4	Address of the condition code
16	4	Address of the point type if PTLOC= is coded, otherwise zero
20		The variable portion of the parameter list as follows:
	1	The System/7 ID in hexadecimal if SYS7= is coded
	1	The IBM 3707 ID in hexadecimal if EMT= is coded
	7	The sensor based point name if SBPOINT= is coded
	1	The condition code in hexadecimal if COND= is coded
	40	The alarm message text if ATEXT= is coded

Displacement	Length	Description
	7	The item name if APPOINT is coded
	8	The ACB name if APACB is coded

Register usage by the macro is as follows:

Reg. 0	Address of the EMSCVT
Reg. 1	Address of the parameter list
Reg. 2-12	Transparent
Reg. 13	Address of a valid save area
Reg. 14	Return address
Reg. 15	Linkage and return codes

SCDEVICE

The SCDEVICE macro provides the user with a method of controlling power devices and units within the power network through program control. control. The user may specify a single device or unit or a list of devices or units. The user's ECB is posted at the end of the device control action(s) with a return code indicating the success or failure of the action(s).

[symbol]	SCDEVICE	LIST= {(r) | addr} [,DEP= {YES | NO}] ,{SYS7= {(r) | id} | SYS7LOC= {(r) | addr}} ,{EMT= {(r) | id} | EMTLOC= {(r) | addr}} ,{DEVICE= name | DEVLOC= {(r) | addr}} ,{ACTION= action* | ACTLOC= {(r) | addr}} ,ECB= {(r) | a} [,{DCVTR=r | DCVTLOC= {(r) | addr}}] [,AREA= {(r) | addr}] [,MF= {L | (E,{(r) | addr})}]

*Valid control actions are:

OPEN
TRIP
CLOSE
RAISE
LOWER
TAG
UNTAG
MANUAL
AUTOMATIC

LIST — The address of an area containing a list of one or more devices or units to be controlled. (See Note 1.)

DEP — Dependency of entries in the list upon the successful completion of the preceding ones. If YES is coded, the processing of the list is terminated when an individual entry fails to execute. If NO is coded, the entire list is processed regardless of the sucess or failure of the preceding entries. NO is the default if this operand is omitted.

SYS7 — The two digit identifier of the System/7 to which the IBM 3707 and device or unit are attached. The identifier may be stored in the low-order byte of the register specified by (r).

SYS7LOC — The address of an area containing the one byte identifier of the System/7 to which the IBM 3707 and device or unit are attached.

EMT — The three-digit identifier of the IBM 3707 to which the device or unit is attached. The identifier may be stored in the low-order byte of the register specified by **(r)**.

EMTLOC — The address of an area containing the one-byte identifier of the IBM 3707 to which the device or unit is attached.

DEVICE — The seven byte name of the device or unit to be controlled as defined by the customer during system generation through the point definition macro.

DEVLOC — The address of an area containing the seven-byte name of the device or unit to be controlled as defined by the customer during system generation through the point definition macro.

ACTION — The controlling action which is to take place.

ACTLOC — The address of an area containing the action which is to take place.

ECB — The address of a user-defined fullword which is posted with a return code indicating the success or failure of the device control action(s). (See Note 2.)

AREA — The address of an area where the device control processing will place an error message, the device name and the type if the control action was not successful. The area is 80 bytes in length.

DCVTR — The address of the DPPXCVT contained in register **(r)**. This operand enables the routine to bypass generating code to find the DPPXCVT from the TCB extension.

DCVTLOC — The address of an area containing the address of the DPPXCVT.

Notes:

1. The format of the list is as follows:

DEC	HEX				
0	0	COUNT			0
4	4	RC	System/7	EMT	
8	8	DEVICE			NAME
12	C				
16	10	ACTION			UNUSED

The list is headed by two halfwords, the first containing the number of entries and the second containing zeros. These are followed by 15-byte entries which contain:

RC	The return code which applies to the individual entry. (See Note 2.)
System/7	The one-byte System/7 identifier.
EMT	The one-byte IBM 3707 identifier.
DEVICE NAME	The seven-byte device name.
ACTION	The selected action (OPEN, TRIP, TAG, UNTAG,

MANUA, AUTOM, CLOSE, RAISE, LOWER) left justified in the field.

2. The user EDB is posted with one of the following return codes which are generated by device ccntrol processing and are as follows:

Code Hex	Dec	Explanation
00	00	Successful device control processing
04	04	Device item name not in data base
0C	12	Execution time-out
10	16	Device failed to execute
18	24	Device already being controlled or PDC getitem failed
1C	28	Device is in TAG/Manual mode - no control allowed except UNTAG/Automatic
20	32	Device is already in requested state
28	40	Unable to communicate with System/7
30	48	Device is out of service - no control allowed
34	52	Device is nct controllable
40	64	Device failed in independent list
44	68	Device failed in dependent list
48	72	Unable to PATCH main device control processor (DOMCDC01)
4C	76	Unable to log status change because no GETWA core available. Otherwise control action completed.
50	80	No GETWA core available - control action cancelled.
54	84	Unable to get the RCB item with the GETITEM macro - control cancelled.

In the case of a list of devices, codes 00 through 34 or 4C through 54 appear in the RC bytes of the list and the user ECB contains 00, 40, 44, or 48. When there is no list, the user ECB is posed with one of the codes 00-54.

The MF=L or standard macro forms generate a parameter list containing the information coded in the macro. The parameter list is passed to DOMCDC10 and has the following format:

DEC	HEX		
0	0	FLAGS	ECB Address
4	4	MESSAGE Area Address	
8	8	LIST or System/7 Address	
12	C	Zero or EMT Address	
16	10	Zero or DEVICE NAME Address	
20	14	Zero or ACTION Address	

The macro flags are as follows:

Bit 0-3	Zero
Bit 4	If on - message area is given
Bit 5	If on - list is independent
Bit 6	If on - list of devices
Bit 7	If on - single device

Displacement	Length	Description
0	1	Flag byte - as described above
1	3	Address of the ECB to be posted upon completion of the device control processing
4	4	Address of the user's message area if AREA= operand is coded, otherwise, zero
8	4	Address of the list of devices if LIST= is coded, or the address of the System/7 ID
12	4	Address of the remote if a single device, otherwise, zero
16	4	Address of the device name if a single device, otherwise, zero
20	4	Address of the action to be performed if a single device; otherwise, zero

The 24 byte fixed portion of the parameter list may be followed by a variable portion containing the System/7 ID, the EMTID, the device name, and the action. These wculd appear if the SYS7=, EMT=, DEVICE=, or ACTION= operands are used.

The macro uses the following registers:

0	Address of the EMSCVT
1	Address of the parameter list
2-12	Transparent
13	Address of the valid save area
14	Return register
15	Return codes

The SCDEVICE macro returns the following codes in register 15:

00	Successful PATCH to the device control processor
04	PATCH to the device contrcl interface processor failed
08	No work area (GETWA) core available at this time

If the user codes both LIST= and SYS7= or SYS7LOC= operands an MNOTE with a severity code of 12 is generated. If the user codes both operands, such as EMT= and EMTLOC= or SYS7= and SYS7LOC=, a warning with a severity code of 4 is generated and the -LOC operand is used.

RLSEBUFF

This macro is used to release an input I/O buffer back to the System/7 Communication Task.

[name]	RLSEBUFF	{BUFFR = register | BUFFLOC = address} [, {DCVTR = register | DCVTLOC = address}]

BUFFR — Specifies the address of the buffer is in the specified register, (1-12)

BUFFLOC — Specifies the address of the buffer is at the location specified

DCVTR — Specifies the address of the DPPXCVT is in the specified register (2-12)

DCVTLOC — Specifies the address of the DPPXCVT is at the specified address

Note: There is no list or execute form nor is there a parameter list.

Return code:

0 - successful

INITSCAN

This macro is utilized to initiate scanning. An initial scan is performed and repetitive scan processing initiated as defined in the System/7 Scan Definition Table. This macro is used during system initialization or after the System/7 has been varied online.

[name]	INITSCAN	[SYS={ALL | N | (r)}] [,MODE={INIT | CYCLIC | EMER}] [,{DCVTR=(r) | DCVTLOC=addr}]

SYS — System/7 identifier. Scan initiation in all System/7s is assumed as the default. If in register form, place ID in low-order byte.

MODE — Determines the new scanning mode as Initial, normal (Cyclic) or Emergency.

DCVTR[1] — The address of DPPXCVT is provided in a register.

DCVTLOC[1] — The location of DPPXCVT is provided at this symbolic address.

----

[1] If DCVTR and DCVTLOC are not specified, the macro will expand the code to locate the DPPXCVT.

VARYCONF

This macro is used to set parts of the data acquisition network to online or offline status. If a point is offline, then the missing data condition is not raised and limit checking is bypassed. Points and terminals can be varied offline or online. DOMTVDB handles the System/7 response. The register form requires the information in the low order byte.

[name]	VARYCONF	SYS= {N | (r)} , STAT= {ONLINE | OFFLINE}, [,{TERM= {(r) | ID} | POINT= {(r) | addr | NAME}}] [,{DCVTR= (r) | DCVTLOC= address}] [,MF={L | (E,{addr | (r)})}]

SYS = N - System/7 logical identifier (1-99)
(r) - The System/7 logical identifier is inserted in register (r)

STAT = The word online or offline

TERM = ID - The terminal ID
(r) - The terminal ID is inserted in register (r)

POINT = addr - The address of an area that contains the 7 character point name - to be specified in execute or standard forms

(r) - This address is in register (r) - to be specified in execute or standard forms

NAME - The 7 character point name - to be specified only in the list form

DCVTR* Enable the user to provide the address of DPPXCVT in a register

DCVTLOC* Enable the user to provide the address of DPPXCVT.

*If DCVTR and DCVTLOC are not specified, the macro will expand the code to locate the DPPXCVT.

RETURN CODES

00	Successful table update and message sent to System/7
04	No GETWA core available or CBGET core
08	SYS/TERM combination is invalid
12	Terminal is manual - No control is possible
16	Point name given is invalid
20	Point is manual - No control is possible
24	The device is already in the requested state

VARYS7

This macro changes the status of a System/7 The three states of a System/7 are Primary, Backup, and Offline. A System/7 may go from its current state to any of the other two states.

[name]	VARYS7	LID = {(r) | address}, UNIT = {(r) | address}, {STAT = {OFFLINE | BACKUP | PRIMARY} | STATLOC = {(r) | addr}} [,MF= {L | (E, {addr | (r)})}] [,{DCVTR = r | DCVTLOC={(r) | address}}]

LID - Defines the location of a one byte field containing the logical ID for the indicated System/7

UNIT - Defines the location of a two byte field containing the System/7 DD qualifier for the desired System/7

STAT - Specifies the mode to which the indicated unit is to be modified

STATLOC - Specifies an address at which the eight byte field containing the desired mode is located

OFFLINE - Specifies the System/7 is to cease communication

BACKUP - Specifies the System/7 is to assume the role of a backup System/7

PRIMARY - Specifies the System/7 is to assume the role of a primary System/7

DCVTR - Address of DPPXCVT in register

DCVTLOC - Address of fullword containing DPPXCVT address

RETURN CODES

00	Request accepted
04	System/7 already in requested state
08	System/7 is not controllable
12	System/370 support error or System/7 error
16	Invalid parameters
20	PATCH failed to IPL task, retry

Note: The following parameters are invalid in the list form: LID, UNIT, STATLOC, DCVTR and DCVTLOC.

VARYSCAN

This macro alters a scan definition during realtime processing by changing a scan's status to active or inactive, or to change scanning mode or to change the scan range. All register notations require the value in the low-order bytes.

To Change a Scan's Status

[name]	VARYSCAN	STAT= {INACT | ACT} ,SCAN= {ID | (r)} [,MF={L | (E, {(r) | addr})}] [, {DCVTR= (r) | DCVTLOC={(r) | addr}}]

To Change Scanning Mode

[name]	VARYSCAN	STAT=MODE, MODE= {N | Z | E} [,MF= {L | (E, {(r) | addr})}] [, {DCVTR= (r) | DCVTLOC={(r) | addr}}]

To Change the Scan Range

[name]	VARYSCAN	STAT=RANGE, SCAN= {ID | (r)} , SYS= {ALL | ID}, TERM= {(r) | n} , ITEM= {(r) | n} [,MF={L | (E, {(r) | addr})}] [, {DCVTR= (r) | DCVTLOC={(r) | addr}}]

STAT — Desires scan status change. This is a required keyword.
ACT -Scan is to be activated, Keyword SCAN

		is required.
	INACT	-Scan is to be deactivated. Keyword SCAN is required.
	MODE	-Scan mode is to be changed.

<u>SCAN</u> Scan ID (0-255).
For register notation the ID must be in the low order byte.

<u>SYS</u> System/7 logical identifier.

<u>TERM</u> Terminal ID number.
For register notation the item number just be in the low order byte.

<u>ITEM</u> Item number. For register notation the item number must be in the low order byte.

<u>MODE</u> Desired scan mode. N = normal, Z = Initial, E = Emergency.

<u>MF</u> (L) Indicates list form.
(E,addr) Indicates execute form and addr contains the address of the list.

<u>DCVTR</u>* Register r contains DPPXCVT address.

<u>DCVTLOC</u>* Specifies the address or a register containing the main storage location which contains the address of the DPPXCVT.

<u>RETURN CODES</u>

00	Successful completion.
04-20	Bad S7WRITE macro issued. See S7WRITE macro.
24	Scan ID invalid.

---

*If the DPPXCVT is not provided by the user, the macro will expand the code to find the DPPXCVT.

S7WRITE

This macro allows the writing of transaction messages to a System/7 (see Appendix B, Communications Formats).

[name]	S7WRITE	decb name, AREA = {(r) | address} [,UNIT = {(r) | address}] [,{DCVTLOC = {(r) | address} | DCVTR = r}]

decb name Specifies the name assigned to the data event control block created as part of the macro expansion

AREA= A-type address or (2-12)
Specifies the location of the first byte of the transaction (header) to be written

UNIT=(r)
=addr RX-type address or (2-12)
Specifies the relative unit index to which the transaction is to be written. The index is specified in the low order byte of the specified register or at the byte specified by the address. Note: This parameter is for use only by internal Energy Management System programs. User must specify the unit in the DECB.

DCVTR= Address of DPPXCVT in register r

DCVTLOC= Address of fullword containing address of DPPXCVT

RETURN CODES

00	Successful write
04	Parameter list invalid
08	Logical ID invalid
12	Invalid UNIT/logical ID combination
16	Requested logical System/7 inactive
20	System/7 failure

S7WRITE - LIST FORM

[name]	S7WRITE	decb name,AREA=address,MF=L

decb name - symbol

AREA= A-type address

MF=L Coded as shown
MF=L operand specifies that the S7WRITE macro is used to create a data event control block to be referenced by the execute form.

S7WRITE - EXECUTE FORM

[name]	S7WRITE	symbol, AREA = address [,UNIT = address] ,MF=E [,{DCVTR=r | DCVTLOC=address}]

decb symbol — RX-type address (2-12) or (1) of decb

AREA=address — RX-type address or (2-12)

UNIT= — RX-type address or (2-12) (See standard form for explanation of this operand.)

MF=E — Coded as shown
MF=E specifies an existing data event control block (specifies in the decb address operand) to be used

DCVTR= — Address of DPPXCVT in register R

DCVTLOC= — Address of fullword containing address of DPPXCVT, register or RX-type address

DOMCSCHT

This macro is used to request stripchart changes.

[label]	DOMCSCHT	RID = {(r) | ID} , {RACT = {ON | OFF} | RALOC = {(r) | address}} , {PNAME = name | PLOC = {(r) | address}} [, {SFA = {(r) | member} | SFAL = {(r) | address}}] [, {SFB = {(r) | member} | SFBL = {(r) | address}}] [, {T = {YES | NO} | TLOC = {(r) | address}}] [, {DCVTLOC = {(r) | address} | DCVTR = (r)}]

DROP IN (BLC LM2 A)

RID — The recorder ID (1 through 16 inclusive). ID may be contained in the low order byte of the register specified by r. RACT The action requested - ON or OFF.

RALOC — The address of an area containing the action requested - ON or OFF.

PNAME — The seven-character name of the analog or pulse counter point to record or to stop recording.

PLOC — The address of the seven-character name of the point.

SFA — The A scale factor by which the current value is to be multiplied. The range for this factor is 1 to 32767. The scale factor may be contained in the low-order halfword of register r. The default is 1.

<u>SFAL</u> The address of a halfword containing the A scale factor.

<u>SFB</u> The B scale factor which is added to the result of the value multiplied by the A scale factor. The range for this factor is zero to 32767. The default is zero. The scale factor may be contained in the low-order halfword of register r.

<u>SFBL</u> The address of a halfword containing the B scale factor.

<u>T</u> Indicates whether or not the stripchart data is to be time marked. The default is NO.

<u>TLOC</u> The address of time mark option - YES or NO.

<u>DCVTLOC</u> The address of an area containing the address of the DPPXCVT.

<u>DCVTR</u> The address of the DPPXCUT contained in register r.

If both methods of coding an operand are used, the macro generates a warning and uses the following operands:

RACT and RALOC	,	RACT used
PNAME and PLOC	,	PLOC used
SFA and SFAL	,	SFA used
SFB and SFBL	,	SFB used
T and TLOC	,	T used

<u>RETURN CODES</u>:

The macro returns the following codes in Register 15:

04 -	Invalid ID
08 -	Invalid action
12 -	Invalid time mark option
16 -	Invalid A scale factor
20 -	Invalid B scale factor
24 -	Invalid name
28 -	GETARRAY for RCB failed
32 -	Requested recorder already on
36 -	Requested recorder already off
40 -	S7WRITE failed
44 -	Recorder processing busy - retry
48 -	Stripchart command failed
52 -	Time-out
56 -	no GETWA available
60 -	PATCH failed

DOMCSTCH generates a parameter list which is passed to DOMTCHRT as follows:

Dec.	Hex.			
0	0	FLAGS	ID	RESERVED
4	4	ADDR OF POINT NAME		
8	8	SCALE FACTOR A		SCALE FACTOR B
12	C	ADDR OF TIME MARK OPTION or zero		
16	10	ADDR OF ACTION		
20	14	POINT NAME		
24	18			

The point name is part of the parameter list when PNAME= is coded.

The macro flags are as follows:

BIT 0	-	On	- Action is on
BIT 1	-	On	- Action is off
BIT 2	-	On	- Time mark
BIT 3	-	On	- Scale factor A is coded
BIT 4	-	On	- Scale factor B is coded
BIT 5	-	On	- Address of time mark is coded
BIT 6	-	On	- Request is from stripchart display processor
BIT 7	-	On	- Reserved

The displacement, length and description of the macro list entries are as follows:

Displacement	Length	Description
0	1	Macro flags - as above
1	1	Recorder ID in hexadecimal
2	2	Reserved
4	4	Address of the point name
8	2	The A scale factor in hexadecimal
10	2	The B scale factor in hexadecimal
12	4	The address of the time mark option when TLOC= is coded
16	4	The address of the action choice when RALOC= is coded
20	7	The point name if PNAME= is coded

Register usage by the macro is as follows:

Reg. 0	Address of the EMSCVT
Reg. 1	Address of the parameter list
Reg. 2-12	Transparent
Reg. 13	Address of a valid save area
Reg. 14	Return address
Reg. 15	Linkage and return codes

S7IPL

This macro is used to initial program load a System/7 write the bootstrap routine through the System/7 initial program load address and the subsequent TX and EN records through the System/7 write address.

[name]	S7IPL	decb name, type, LENGTH = {R | (r) | address} ,AREA={(r) | address} ,UNIT={(r) | address} ,ID=address [{DCVTR=r | (r) | DCVTLOC=address}]

decb name — Specifies the name assigned to the data event control block created as part of the macro expansion

type

- IPL — Data to be written is the bootstrap to be sent through the initial program load unit address
- TXT — Data to be written is a FORMAT/7 TX record sent through the write unit address
- END — Data to be written is the FORMAT/7 EN record sent through the write unit address; signals the System/7 I/O supervisor to start cyclic READs.

LENGTH= — Length of the data - absolute value, halfword in register (2-12) or halfword at address (RX-type)

AREA= — The address of the first byte of the data, A-type address or register (2-12)

UNIT= — Specifies the System/7 to which the data is written, specification is a one byte index in low order byte of register (2-12) or at the byte specified by the type address

ID= — Specifies the logical ID this unit is being initial program loaded for, ID in low order byte of register (2-12) or at the byte specified by the RX-type address

DCVTR= — Address of DPPXCVT in register R

DCVTLOC= — Address of fullword containing address of DPPXCVT

RETURN CODES

00	Data written successfully
04	Invalid parameter list
08	Invalid ID
12	Invalid UNIT/ID combination
20	System/7 failure

S7IPL - LIST FORM

[name]	S7IPL	decb name,type,MF=L

decb name - symbol

type IPL, TXT, END

MF=L Coded as shown; specifies that the S7IPL macro is used to create a data event control block to be referenced by the execute form of the macro.

S7IPL - EXECUTE FORM

[name]	S7IPL	decb address, type, LENGTH={(r) | address} ,AREA={(r) | address} ,UNIT={address | (r)} ,ID={(r) | address} ,MF=E [DCVTR=r | ,DCVTLOC={(r) | address}]

decb address RX-type address, (2-12) or (1)

type IPL, TXT, END

LENGTH Length of data - absolute value - RX-type address addressing the length (2-12) containing the length

AREA= RX-type address or (2-12)

UNIT= RX-type address addressing the one byte unit or (2-12) with the unit in the low order byte

ID= RX-type address addressing the logical ID or (2-12) with the logical ID in the low order byte

MF=E Coded as shown; specifies an existing data extent control block to be used

DCVTR= Address of DPPXCT is register R

DCVTLOC= Address of fullword containing address of DPPXCVT

ASCICONV

This macro is used to translate data from EBCDIC to ASCII code or from ASCII to EBCDIC code.

[name]	ASCICONV	TYPE = {ASCICONV | EBCDIC} LENGTH = {number | (r)}, AREA = {address | (r)} [, {DCVTR = r | DCVTLOC = {(address) | (r)}}]

TYPE — Specifies the code being translated to

AREA — Specifies the first byte of the data to be translated

LENGTH — Specifies how many bytes are to be translated, the maximum value allowed is 256 bytes

DCVTR — Address of DPPXCVT is provided in register R

DCVTLOC — Address of an area containing the address of DPPXCVT

UPD7COMM

This macro is used to lock, update, log, unlock and event actions to the TAS&COMM array.

```
[name]   UPD7COMM   TYPE = ( PRIMARY   )
                           ( BACKUP    )
                           ( INACTIVE  )   ,LID = {(r)     }
                           ( FAILED    )          {address }
                           ( SCANNING  )
                           ( IPLING    )   ,UNIT = {(r)     }
                           ( IPLED     )           {address }
                           ( DISKLDED  )
                           ( STOPSCAN  )   [{DCVTR = r              }]
                           ( DISKFAIL  )   [{                       }]
                           ( NODEUP    )   [{DCVTLOC = {address}    }]
                                           [{          {  (r)  }    }]
```

TYPE: Specifies what update is to be made.

PRIMARY - Assign the unit as the primary for the logical ID; log the array; event the action.

INACTIVE - Flag the unit as ready but not IPLed; remove unit from back-up or primary; log the array.

FAILED - System/7 has failed; flag unit a not ready and remove unit from backup or primary; log array.

BACKUP - Assign the unit as the backup for the logical ID; log the array; event the action.

SCANNING - Flag the logical ID as scanning; event the action.

IPLING - Flag the unit as IPLing for the logical ID.

IPLED - Flag the unit as IPLed for the logical ID.

DISKLDED - Flag the unit as having its disk loaded this logical ID.

STOPSCAN - Flag the logical ID as having stopped scanning; event the action.

DISKFAIL - Flag the logical ID with a System/370 support disk failure; flag the unit ready and not IPLed.

NODEUP - Flag the logical ID as having been commanded to enter the hierarchy.

LID Specifies the address of the one byte logical ID. RX-type address or in the specified register (2-12).

UNIT Specifies the address of the one-byte System/7 unit index. RX type address or in the specified register (2-12).

DCVTR Address of DPPXCVT in specified register.

DCVTLOC Address of DPPXCVT is at the specified address.

Return Codes:

0 - Successful update
4 - Invalid logical ID

 8 - Invalid unit index
12 - Unit not available this logical ID
16 - Invalid type

S7CHECK

This macro is used to wait on the completion of the output request and unlock the resource.

[name]	S7CHECK	DECB = {(r) / address}	[,{DCVTR = r / DCVTLOC = {(r) / address}}]

DECB Specifies the address of the request DECB. RX-type address or address in the specified address (2-12).

DCVTR Address of the DPPXCVT is in the specified register.

DCVTLOC Address of DPPXCVT is at the specified address.

Note: Generates a WAIT and LOCK TYPE = UNLOCK

DOMTPURG

This macro is used by the System/7 I/O program to generate the parameter list to the PURGE SVC.

[name	DOMTPURG	DEB = {ONE | ALL}, POST = {YES | NO}, HOW = {HALTIO | QUIESCE} ,REQ = {ALL | REL}, RB = {NO | YES}, BY = {DEB | TCB} ,MF = {L | (E,address)} ,PRGDEB = address

DEB	ONE	Purge only the request queue element for the DEB specified (default)
	ALL	Purge request queue elements for all entries in the DEB chain
POST	Post the event control blocks (ECB) for the request queue elements YES is the default.	
HOW	HALTIO	Halt the I/O activity (default)
	QUIESCE	Allow the I/O to quiesce
REQ	ALL	Purge all requests (default)
	REL	Purge only related requests
RB	YES	Purge the asynchronous exit queue, the request block queue, the logical channel queue, and the DDR wait queue
	NO	Purge the asynchronous exit queue, the logical channel queue, and the DDR wait queue, but not the request block queue (default)
BY	DEB	Purge by DEB (default)
	TCB	Purge by TCB
PRGDEB	Specifies the location of the DEB address RX-type address or in registers (2-12)	
MF	L	List form generates the parameter list
	E	The options byte generated by all parameters except MF and PRGDEB will be stored into the parameter list.
	Address	RX-type or register (1-72)

GETCVT

This macro is used to obtain the address of either the DPPXCVT or EMSCVT.

[name]	GETCVT	{XCVT | ECVT}

XCVT obtain DPPXCVT address

ECVT obtain EMSCVT address

Register 15 contains the address of the desired CVT.

DOMDISP

This macro is used to generate the DOMTABLE address list and to obtain the displacement into the list of a given program name. All program names are of the form DOMXXXXX. A global symbol must be defined in the issuing module as follows: GBLA &OFFSET.

[name]	DOMDISP	MOD=name, [T= {O/C}]

MOD name is the five character suffix used to identify the desired program.

T = O Specifies that the offset (displacement) is to be returned in the global &OFFSET.

T = C Specifies that the address list is to be expanded.

EMSLINK

This macro is used to generate the linkage to a subroutine which is linkedited as part of DOMTABLE.

[name]	EMSLINK	offset,dcvtr,dcvtloc

offset — The displacement into the DOMTABLE address list to the address of the desired program (the displacement may be obtained using the DOMDISP macro)

dcvtr — The register containing the address of the DPPXCVT

dcvtloc — The location containing the address of the DPPXCVT

Notes: All operands are positional. The dcvtr and dcvtloc operands are mutually exclusive. The macro will generate code to load the address of the EMSCVT into register 0 and branch and link to the subroutine.

COMMUNICATION FORMATS

All data sent between the System/370 and System/7 is prefaced by a standard header.

Transaction Code	Flags	LENGTH	Destination	Origin
0	1	2	4	5

Transaction Code - indicates the nature of the data

Flags (bit)

X'80'	(Reserved)	
X'40'	(Reserved)	
X'20'	(Reserved)	
X'10'	Raw data array changes	0 = no status changes 1 = status changes
X'08'	Invalid transaction returned	
X'04'	Communication indicator (System/7 to IBM 3707) (0 = success; 1 = failure)	
X'02'	Transaction results (0 = success, 1 = failure)	
X'01'	Text format (0 = ASCII; 1 = binary)	

LENGTH - count in bytes of the length of the data field

Destination - Logical ID of the processor to which the transaction is directed

Origin - Logical ID of the processor from which the transaction originated

TRANSACTION CODE SUMMARY

Host to System/7

00-03	Reserved
04	System/7 Primary initialization
05	AGC Pulse duration output
06	PDC Command
07	Reserved
08	General Commands
09	Reserved
0A	Initiate scanning
0B,0C	Reserved
0D	Stop scanning
0E	Host date/time synchronization
0F	Reserved

10	Audible alarms
11	Reserved
12	System/7 Disk load
13	Stripchart data
14	Wallboard data
15	Vary a point online or offline
16	PDC Reply
17	Set System/7 offline
18	Reserved
19	Requested System/7 checkpoint data
1A-6F	Reserved
70-7F	Available for use by customer

<u>System/7</u> <u>to</u> <u>Host</u>

08	Transfer of control
0F	Time/date synchronization
80-82	Reserved
83	Backup initialization complete
84	Top/not top notification
85	Reserved
86	PDC Reply
87	Reserved
88	General Command response
89	Administrative message
8A	Scan Data
8B	Initial scan timeout
8C	Reserved
8D	Scanning has stopped
8E	Reserved
8F	Security message
90	Poll overload
91	Reserved
92	Request for disk load
93	Reserved
94	Wallboard
95	Point online/offline
96	PDC Reply
97	System/7 now offline
98	Log System/7 checkpoint data
99	Request retrieval of System/7 checkpoint data
9A-EF	Reserved
F0-FF	Available for customer use

HOST TO SYSTEM/7

04 <u>System/7 Primary Initialization</u> - this transaction informs the System/7 that it is now the primary System/7 and directs it to perform any initialization required and to return a top/not top reply text:

0000	0000
0 1	2 3

05 <u>Pulse duration output from AGC</u> - passes the generator names and signed halfword pulse durations to the System/7 controlling these generators

# items	Generator name	Pulse duration	Generator name	Pulse duration
0	2	10	12	20

06 PDC Command - this transaction requests a PDC action from the System/7 or informs it of a execute or verify time-out. The 06 transaction message is variable in length and is as follows:

name prefix	blank	name suffix in binary	PT type	ACTION	zero

The fixed portion of the message is 10 bytes long and is as above. The fields in it are as follows:

Displacement	Length	Description
0(0)	3	Prefix of point name (first 3 alphabetic characters).
3(3)	1	Blank to pad the prefix.
4(4)	2	Suffix of name in binary (last 4 numeric characters of name converted).
6(6)	2	Type and action flags as follows:
	1111	Type as follows:
	1...	On - 3707 DO Off - not 3707 DO
	.1..	On - local pulse DO Off - not local pulse DO
	..1.	On - local non-pulse DO Off - not local non-pulse DO
	...1	on - 3707 DI Off - local DI
	 1111	Action bits as follows:
	1111 1111	
	 1111	Reserved
	1...	On - tag/untag dummy DI, raise/lower, or execute time-out. Off - tag/untag real DI trip/close, or verify time-out.
	.111	Reserved
	.... 1...	On - close/lower/manual
	.... .1..	On - trip/raise/automatic
	.... ..1.	On - untag
	1	On - tag
	 1111	Off - execute or verify time-out
8(8)	2	Zero.

The variable portion of the message is as follows:

1. For tag/untag, verify time-out or execute time-out:

```
 DI       DI        PT        PT        DI        DI
DATA     DATA                         SEL DO    SEL DO
 GP       MA     (DATA)     (SEL)      GP        MA    X'8D'   local data

 DI       DI        DI
3707     CARD
 ID       ADDR      PT        --       X'8D'                   3707 data
```

2. For execute without verify (no change of status expected). ie. raise or lower

```
 DO    DO    PT     PT    DO    DO   LOCAL PULSE
DATA  DATA               SEL   SEL
 GP    MA  (DATA)  (SEL)  GP    MA   DURATION or zero   X'8D' local data

 DO       DO       PULSE      P       P
3707     CARD                  T       T
 ID       ADDR     DURAT        1       2  X'8D'   3707 data
```

3. For execute with verify (change of status expected). ie. trip or close

```
   DO    DO    PT     PT    DO    DO     LOCAL PULSE      DI   DI
  DATA  DATA               SEL   SEL                     DATA DATA
   GP    MA  (DATA) (SEL)   GP    MA     DURA or zero     GP   MA
   PT    PT  DI SEL   DI
                SEL DO
(DATA) (SEL) DO GP    MA   X'8D'                          local data

   DO    DO    PULSE P P     --            --             DI   DI
  3707  CARD         T T                                 3707 CARD
   ID   ADDR DURATION 1 2                                 ID   ADDR
   DI
   PT    --      X'8D'                                    3707 data
```

The variable portion of the message is contained in the CAPDC array. The X'8D' is the end of message indicator for the System/7.

08 General Commands - This transaction is used for several functions.

1. VARY SCAN

   Transaction Code '08'

   a. VARY SCAN MODE
      SCMƀM,M=N(ASCII)8D
      SCMƀM,M=E(ASCII)8D

      SCM is the vary scan command on the System/7
      ƀ is a blank in ASCII
      N is for normal mode
      E is for emergency mode
      8D is EOM character for the System/7

   b. VARY SCAN ACTIVE OR DEACTIVE

      SCMƀA,S=XXX(ASCII)8D
      SCMƀD,S=XXX(ASCII)8D

      SCM is the vary scan command on the System/7
      ƀ is a blank in ASCII
      A is to activate a scan
      D is to deactivate a scan

XXX is the scan id
8D is EOM character for the System/7

2. VARY STRIPCHART = command length = 80 bytes

Transaction Code X'08'

SCCbR=XX,C=ON/OFF,O=370,S=XXXXX,B=XXXXX,T=NO/YES8D

SCC is the stripchart command code on the System/7
b is a blank in ASCII
R is the stripchart recorder id (xx)
C is the command to turn the recorder 'on' or 'off'
S is the A scale factor
B is the B scale factor
T is the timing control mark option 'no' or 'yes'
8D is EOM character for the System/7

3. VARY TERMINAL

Transaction Code X'08'

VRYbS,3707,XXX8D-------------

------------------------------ID

Displacement	Length	Use
0	3	VRY - identifies type of message
4	1	S = N - IN SERVICE S = F - OUT OF SERVICE Identifies the type of the vary
5	6	,3707, - Identifies to vary a terminal (IBM 3707 Remote Data Acquisition and Control Station)
11	3	XXX - RCB ID - Terminal identification
14	1	8D - End of message
15	65	ZEROS
80	1	ID - Identification of the System/370 sending the message

4. TIME CORRECTION REQUEST

Transaction Code X'08'

TCRbI,nn(ASCII)8D
TCRbD,nn(ASCII)8D

TCR is the time correction request code on the System/7
b is a blank in ASCII
I is for increase time
D is for decrease time
nn is the number of seconds to increase or decrease

system time
8D is EOM character for the System/7

5. TRANSFER OF CONTROL

Transaction Code X'08'

CTLȸnnnnnnn,XX,YY8D

CTL is the transfer of control code on
the System/7 in ASCII
ȸ is a blank in ASCII
nnnnnnn is the seven character device
name to receive control in ASCII
, is a comma in ASCII
XX is the CPUID to receive control in ASCII
, is a comma in ASCII
YY is the CPU ID where the device is attached
in ASCII
8D is the EOM character for System/7

0A Initiate Scanning - informs the System/7 of the desired scanning mode

Type TYPE - 'EEEE' start cyclic scanning

0 1 'FFFF' perform initial scan, return raw data array and schedule cyclic scanning

0D Stop Scanning - informs System/7 to quiesce all scans. Upon reply to this command the System/7 enters backup mode.

0000 Stop in synchronization
FFFF Stop immediately

0E Host Date/Time Synchronization - Send to the System/7 the date and time prior to initiating scans.

year	day	hour	minute	second	hundredths second
0	2	4	6	8	10

All fields are binary

year - full year, for example 1974
day - julian day

10 Audible Alarms - activates an audible alarm attached to the System/7

This data is sysgened into the
CAAATBL field AAAUDIB field.
0 3

12 System/7 Disk Load - this transaction sends the next requested disk sector to the System/7

Module name	Relative sector	Sector count	Sector data	
0	7 8 9	A B C		10B

or

```
'EEEEEE...EEEE' 0000    0000
0             7 8  9    A  B
```

Module name - The member name from the partitioned data set the date was read or

'EE...EE' indicating all sectors have been sent

Rel. sector - The relative sector on the System/7 disk where this data resides

Sector count - The number of sectors in this member

Sector date - 256 bytes of text

13 Stripchart Data - This transaction sends 16 raw data values to be used to drive the 16 stripchart recorders local to a System/7. Each value is contained in two bytes consisting of 15 lower-order bits of data and one high-order bit for the sign of the data.

14 Wallboard System/370 to System/7 Command List

Storage Map of transactions code 14 text:

Offset HEX	DEC		
0	0	Flags	Command(1)
4	4	Comand(2)	Command(3)
		:	:
F8(max)	248	Command (n)	End Indicator

F8 (max) 248 (max) Command(N) End Indicator

Layout of transaction code 14 text

Offset	Bytes and Bit Patterns	Field	Description
0	2	Flags	All zeros indicates regular message. hex "0001" indicates these commands are for and initialization mode. (Return transaction code of 94 expected)
2	2	Command(1)	
		x...	strobe bit
		.xxx	command
		.... xxxx xxxx xxxx	lamp address (hardware address for lamp)
4-F4 (increment by 2)	2	Command(2-N)	optional (if present - same as

Command(1))

4-F8 2 End indicator All F-F's indicates end of list.

15 <u>Vary a Point ONLINE or OFFLINE</u>

Transaction Code X'15'

COMMAND CODE	POINT NUMBER

Displacement	Length	Use
0	2	Identifies the command 3DCC - VARY ONLINE 3CCC - VARY OFFLINE (packed ASCII)
2	2	Relative sequence number of the point on that System/7

16 <u>PDC reply</u> - message length is 12 bytes.
The x'16' PDC reply is received when the System/7 is not scanning and only applies to tag/untag commands. The format of the message is as follows:

name prefix	action	binary number	DEST System/7	TERM ID	DI GP or CARD	DI PT	CONTROL CPU ID	RETURN CODE

0(0) name prefix: The first three characters of the data base name of the status point.

3(3) action: bit settings as follows:

1...	On - tag/untag - dummy DI Off - tag/untag - real DI
.111 11..	Reserved
.... ..1.	Untag
....1	Tag

4(4) binary number - The last four numberic characters of the data base name of the status point converted to the binary value.

6(6) Dest System/7: Destination System/7

7(7) Term ID: Terminal ID

8(8) DI GP or CARD: DI group or card number

9(9) DI PT: DI point

10(A) Control CPU ID: The ID of the CPU which is controlling the point

11(B) Return Code: as follows:
00 - successful execution
non-zero - execute failed

17 <u>Set System/7 offline</u> - commands the System/7 to cease

channel communications and go offline

00 00
0 1

19 Requested System/7 checkpoint data:

Array id		relative sector		requested sector data
0	1	2	3	4

System/7 to Host

08 Transfer of Control

CTLtonnnnnnn,xx,yy8D CTL is the transfer of control code on the System/n AGCII
is blank in ASCII nnnnnnn is the seven character device name to receive control in ASCII , is a comma in AGCII xx is the CPU ID to receive control in ASCII , is a comma in ASCII YY is the CPU ID where device is attached in ASCII 8D is EOM character for System/7

0F Time/date synchronization - Sent to the System/370 for time synchronization prior to starting scans, if the System/370 is not at the top of the hierarchy.

year	day	hour	min	sec	.01 sec
0	2	4	6	8	10

All fields are binary.

year = full year (such as 1974)
day = julian day

83 Backup initialization Complete - informs the System/370 that the System/7 has completed its backup initialization

00 00

84 Top/Not Top notification - sent to the System/370 by System/7 to notify System/370 of its position in the hierarchy. This transaction when sent at primary initialization time also indicates to the System/370 that the System/7 is ready to assume the primary function.

Top/Not Top	Init	line up	line down
0	1	2	3

Top/not top - 00 - System/370 at top of hierarchy
F0 - System/370 not at top of hierarchy

Init - 00 - not initialization phase
01 - initialization successful
FF - initialization failed

line up - A System/7 appear in the hierarchy
F0 - A higher System/7 came up
CPU ID -A lower System/7 came up

line down - F0 - A higher System/7 went down
CPU ID -A lower System/7 went down

86 PDC Reply - message length is 8 bytes. The x'86' transaction code message contains the results of a PDC command for which no change of status is expected. The message is as follows:

ACTION RC name prefix blank binary number

0(0) ACTION: zero

1(1) (RC) return code:

00 - successful execute
01 - 3707 is offline
02 - 3707-BCH error in command transmission
03 - 3707 - feature is offline
04 - 3707 - DO point is busy timing
05 - 3707 - DO is not armed
06 - 3707 - execute doesn't compare with arm
07 - 3707 - invalid command
08 - 3707 - invalid feature address
09 - local - execute could not be completed
0A - local - parity error
0B - local - invalid device address
FF - unable to log RDA data to highest level known due to overflow in mini-RDA.

2(2) name prefix: the first three characters of data base name of the status point

5(5) blank: padding

6(6) binary number: the last four numeric characters of the data base name converted to the binary value.

88 General Command Response

1. Stripchart Reply

Routing Code	Error Code	Recorder ID	Control Flag
0	1	2	4

Routing Code - X'01'

Error Code - X'00' - successful
- X'01' - System/7 error
- X'02' - Invalid command format

Recorder ID - Recorder ID number (1-16)

Control flag - (-1) - OFF
1 - ON
0 - Not specified

2. Scan Control reply

Routing Code	Error Code	Function Code	Scan ID or Mode
0	1	2	4

Routing Code - X'02'

Error Code - X'00' - Successful

X'01' - System/7 error
X'02' - Invalid command format

Function Code	-	1 - mode 2 - range 3 - activate 4 - deactivate
Scan ID	-	Scan ID number
Mode	-	0 = normal 1 = initial 2 = emergency

89 Administrative message - typer or display message

Text in ASCII

8A Scan Data - the scan data from the hierarchy leg

8B Initial scan timeout - an error was encountered during initial scan, cyclic scans will not commence upon completion of initial scan.

00 00 0 1

8D Scanning has stopped - the System/7 has ceased scanning and lapses into backup mode

00	00	00	00
0	1	2	3

8F Security message - message sent to the System/370 every 10 seconds while the System/7 is in backup mode

00 CPU CPU - logical id of the System/7 reporting

90 Poll overload - This transaction informs the host that there has been a scan overrun on this basic scan cycle and the data base transfer (TC 8A) will not be sent for this scan cycle.

00 CPU CPU - System/7 logical ID

92 Request for disk load - requests from the System/370 the next disk load model

Version	mod level	config level
0	2	4

94 Wallboard System/7 to Ssytem/370 - Request for Next Command List

Four bytes of text not currently used.

97 System/7 now offline - sent to the System/370 in response to a TC 17 informing the System/370 that the System/7 is now offline and the System/7 will no longer communicate.

System/7 ID	Array ID	Rel. Sector #	sectoror count
0 1	2	3 4	5

98 <u>Log System/7 checkpoint data</u> - to log information from System/7 on System/370 external storage.

Array ID	displacement	bit # or #of words	words to checkpoint
0 1	2 3	4 5	6

Bit 0 of byte 2 =
    0 = replace System/7 word(s)
    1 = replace bit
Bit 0 of byte 4 is the bit replaced if replace bit option selected.
Displacement is in System/7 words from start of array.

99 <u>Request retrieved of System/7 checkpoint Data</u>

Array ID	relative sector #	#sectors
0 1	2 3	4 5

APPENDIX C: SPECIAL REAL TIME OPERATING SYSTEM MACRO DEFINITIONS

The System/370 Energy Management System uses the realtime support services, provided by Special Real Time Operating System macro calls,to invoke either SVCs of branches to the service routines. The macros are functionally described in alphabetic order. The IBM Special Real Time Operating System PRPQ Description and Operations Manual is referenced for detailed expansions and parameter definitions of each macro.

ARRAY MACRO

The ARRAY macro specifies an array name to the offline utility program.

BEGIN MACRO

The BEGIN macro provides standard OS linkage conventions for re-entrant or non-re-entrant routines. In general, BEGIN statement is designed to:

1. Identify and label the main control section or entry point address
2. Save the calling program's general purpose registers
3. Establish main control section addressability
4. Prepare a save area
5. Define register usage through the EQUATE macro

CHAIN MACRO

The CHAIN macro provides the facility for allowing multiple tasks to modify the same control block chains without the necessity of ENQ/DEQ or being disabled.

DBLDL MACRO

The DBLDL macro retrieves one or more PDS directory entries. Operation is similar to that of the OS/VS BLDL macro with the addition that the directories are gotten from the prime copy of duplicate data sets.

DEFLOCK MACRO

Each resource to be reserved is defined to Special Real Time Operating System by a DEFLOCK macro. The DEFLOCK macro causes a control block to be built describing the resource. The name of the resource is returned in register 0, and the address of the control block is returned in register 1. This control block address is used whenever reserving a resource with the LOCK macro. After all processing for a particular resource is completed,the control block is released by another DEFLOCK macro. Once the control block is released, it must be redefined by a DEFLOCK macro before that resource is reserved again. In the case of two-partition operation, separate lock controls are maintained for each partition. Thus, a program cannot use a lock control block created in the other partition.

DEFMSG MACRO

The DEFMSG macro defines messages used by the message writer function.

DOPEN/DCLOSE MACRO

The DOPEN and DCLOSE macros open or close DCBs for data sets which have duplicate data set support. Operation is similar to an OS/VS OPEN/CLOSE macro.

DPATCH MACRO

An independent task is created to execute continuously over an indefinite time period. When an independent task is no longer required, it can be deleted by the DPATCH macro. Since the task may have several entries in its work queue, an unconditional DPATCH does not allow these work queues to be executed. Any ECBs associated with the work queues are posted with a DPATCH completion code and FREEMAINs are issued for those areas specified. The DPATCH is specified as conditional, which prevents losing any work queues.

DPPXBLKS MACRO

The DPPXBLKS macro generates DSECTs for various Special Real Time Operating System and OS/VS control blocks. The keyword parameters define requested control blocks. When a keyword is omitted, the control block associated with the keyword is not expanded. With the exception of the "TYPE=" parameter, any non-blank character is acceptable as the keyword operand.

DSTOW MACRO

The DSTOW macro adds to or replaces a PDS directory entry in a data set which has duplicate data set support. The operation is similar to an OS/VS STOW macro.

DUMPLOG MACRO

The DUMPLOG macro unloads (dumps) tape logged array data within a given time-frame.

EXIT MACRO

The EXIT macro is used in conjunction with the BEGIN macro and performs the exit linkage convention requirements; that is, register 13 is restored to point to the caller's save area, the other general purpose registers that were saved are restored, and the GETMAINed save area, if one exists, is released.

FREEWA MACRO

The FREEWA macro releases control of a work area obtained through the GETWA macro. If the GETWA was not TYPE=PC, the FREEWA must be issued under the same task as the corresponding GETWA.

GETARRAY MACRO

The GETARRAY macro retrieves the data in one or more arrays of the data base, the address of those arrays in the data base, or the names and specifications of all of the items in the array(s).

GETITEM MACRO

The GETITEM macro retrieves the data in one or more items of the data base or, alternately, the address or definition specification of those items in the data base.

GETLOG MACRO

The GETLOG macro retrieves logged arrays by time or by using array locator information gotten from a previous retrieval.

GETWA MACRO

The GETWA macro obtains small, short-term work areas without adversely increasing paging rates. The work areas can be explicity freed with the FREEWA macro or automatically freed at the end of the current PATCH queue or at the end of the current task processing. The address of the work area is returned in register 1.

ITEM MACRO

The ITEM macro defines the data items contained in an array.

LOCK MACRO

Every resource previously defined by a DEFLOCK macro is exclusively reserved by a LOCK macro. The address of the control block (which is returned by the DEFLOCK macro) is specified in the LOCK macro. If the resource is unavailable at the time the LOCK macro is issued, the requesting task is placed in a wait state until that resource becomes available. Another LOCK macro releases the resource.

MESSAGE MACRO

The MESSAGE macro is used by the online or offline programs to print or display a predefined message. The message is defined through the offline utility system using the DEFMSG macro.

PATCH MACRO

The PATCH macro creates a dependent task, an independent task, and queues (request a program to run under) an existing independent task. If no task by that name exists, one is created and the PATCH parameters are passed to it. A dependent task has no name; therefore, a dependent task is created if no name is specified in the PATCH parameters.

Task attributes or characteristics are specified when creating a new task In most cases, a default value is assumed if no value is specified. Several operands of the PATCH macro are used only for independent task creation and are ignored for dependent task creation. The various task attributes available affect overall system overhead, main storage usage, task synchronization, and execution times. They should be considered carefully so they correspond to the requirements of the task they affect.

Each time a program is "called" or executed as a result of a PATCH, it is passed a parameter list. These parameters identify the PATCHing program and the reason for the PATCH, pass data or an address of data arrays, or, in general, provide the PATCHed program any information it might need to execute a given function. This parameter list is always headed by a one-character ID and three bytes containing the length of the parameter area. The remainder of the list can be more IDs or any combination of values and/or addresses needed by the PATCHed program.

PROGRAM CONTROL CARD

The Special Real Time Operating System offline utility program accepts and processes data of several types from any PDS member or sequential data set specified by the user.

PTIME MACRO

The PTIME macro provides Special Real Time Operating System time management services to the user. The macro PATCHes a task at a specific or relative time. Optionally, this PATCH is repeated at specified cycle intervals, continuously or for a certain number of PATCHes. The PTIME macro also allows previous PTIME calls to be modified or deleted. An additional function of the PTIME macro allows access to the correct Special Real Time Operating System time and date.

PUTARRAY MACRO

The PUTARRAY macro moves data into one or more arrays of the data base. The data in the entire array, based on the length defined through the offline utility, is replaced.

PUTITEM MACRO

The PUTITEM macro stores data into one or more items of the data base. If another user of the data base is executing a GETITEM macro with PROTECT=YES, the operation of the PUTITEM macro is delayed until the GETITEM completes.

PUTLOG MACRO

The PUTLOG macro logs demand arrays.

REPLOG MACRO

The REPLOG macro replaces or updates logged arrays using the array locator information buffer gotten from a previous retrieval of the array.

APPENDIX D: DISPLAY MANAGEMENT MACRO DEFINITIONS

The System/370 Energy Management System uses the Display Management System to provide all display oriented functions required to support the IBM 5985 Color Display Control Unit. Display Management provides macros for both display information generation and online display management. The macros are functionally defined in this appendix. The Display Management System PRPQ Description and Operations Manual is referenced for detailed expansions and parameter definitions of each macro.

ONLINE DISPLAY MANAGEMENT MACRO DEFINITIONS

This section describes the macro instructions to request display services. Each macro instruction generates executable code that ends in a branch and link instruction to access the desired supporting function. The processing routine executes as a subroutine under the requestor's task.

The following macros, and their functions, are described:

DALARM	General alarm processing
DAUPDAT	Array address update
DHCOPY	Hardcopy request
DINFO	Dynamic information update
DISPCONF	Display reconfiguration
DISPENT	Entity change request
DISPRD	Read display screen
DISPREQ	Display request
DISPSR	Partial screen read request
DISPUP	Cyclic display update
DISPWT	Write display screen
DLITES	Keyboard backlight control
DOBTAIN	Obtain function key or access/functional area names or identifiers
DAUDIBL	Sound audible alarm
DWZONE	Write to zones

DALARM MACRO

The general alarm display macro adds, modifies, or deletes a general alarm notification. Each general alarm must have a unique identifier and it is passed as an operand when the alarm is added.

DAUPDAT MACRO

During display controller generation, source items that are not included in the array data base are identified by an array name and displacement value. This requires that the main storage residence address of those source items be supplied during online processing before display updating can occur. This is accomplished by the DAUPDAT macro and its operands.

DHCOPY MACRO

The DHCOPY macro writes the contents of the IBM 5985 display buffers to a printer. The hardcopy output is printed just as it is being viewed. There is no distinction between background or foreground items.

DINFO MACRO

Source item locations and data, not identified or described when a display is generated, are defined during online operation by the DINFO macro. These source items consist of alphanumeric, hexadecimal, or binary text and numerical values that are converted before displaying.

DISPCONF MACRO

The basic display configuration is defined prior to the online operations through the data base utility facilities. Each display unit is assigned a unit identifier that remains constant. The access and functional areas are changed by the DISPCONF macro, to reassign the basic functions assigned to each unit. The individual units are also removed from service and, conversely, returned to service.

DISPENT MACRO

The DISPENT macro adds, modifies, or deletes entries from the entity change table. The entity name assigned during display generations identifies the altered entity. The applied attributes are supplied as operands.

DISPRD MACRO

This macro allows all, or a portion, of the display buffer to read. No formatting or translation of the input occurs. The data is as read from the buffer at the time the read was issued. This macro is normally used by Display Management processors when certain Display Management supporting functions are initiated by console operator interaction, but is also issued by programs external to Display Management.

DISPREQ MACRO

The DISPREQ macro activates a named display. This request changes a display currently being viewed by a console operator without his prior knowledge or concurrence. It is used only by programs actually initiated by interactive requests from the console operator. It is a useful tool if one of several displays is selected, the desired display being determined by online information that was not available when the manual input action table was generated.

DISPSR MACRO

The DISPSR macro reads the selected contents of the display buffer defined during display generaticn as partial screen read areas.

DISPUP MACRO

The DISPUP macro executes selected portions of a format control list to update the associated display items on a display. This macro is normally used by programs external to Display Management when source item locations are not defined during display generation or when the dynamic information facilities are used.

DISPWT MACRO

The DISPWT macro allows all, or a portion, of the display buffer to be written. No data conversion is done. The output stream consists of character, position, control, and attributes that are acceptable to the device being accessed.

DLITES MACRO

The DLITES macro turns on or off the backlight associated with each function key. When a keyboard is switched from one display to another, all backlighted keys are set to the off state. Display Management maintains the status of the lights associated with each display unit and refreshes the lights when the switch of control occurs.

DOBTAIN MACRO

Access/functional areas and function keys are assigned names when Display Management is incorporated into OS/VS. These names are then used during display generations. During online operations, Display Management uses a single byte value, an access/functional area identifier or a hardware control key identifier, for internal processing. The DOBTAIN macro is used when it is necessary to get the name or the single character identifier during online processing.

DWZONE MACRO

The DWZONE macro initiates a write request to a specific zone.

DAUDIBL MACRO

The DAUDIBL macro activates the audible alarm on the specified display unit. The display unit must have the audible alarm feature installed or the request is ignored

DISPLAY INFORMATION GENERATION MACRO DEFINITIONS

This section describes the macro instructions to generate display formats, character generator fonts, and manual input actions for online Display Management System processing.

The following macros and their functions are described more fully below.

BGNDGEN	Background generation control macro
DIF, DELSE, DENDIF	Conditional format macros
DISPEND	Generation end macro
DISPROG	Display program action macro
DITEM	Define cyclic items macro
DMIAT	Define manual input action entries
DRAW	Define static items macro
DSYM	Define symbolic operands
DTIMES	Display time and date definition macro
DYNAMIC	Dynamic update definition macro
FONT	Define character generators macro
FONTGEN	Character font generation control macro
MIATGEN	Manual input action table control macro
PSREAD	Partial screen read definition macro
DISPGEN	Display control generation control macro

CONTROL MACROS

A control macro selects and defines the generation phase. It also determines which post-processor is used. There are four control macros and generation phases.

BGNDGEN Macro

This macro specifies the generation of the static, or background, items associated with a display control generation.

DISPGEN Macro

This macro specifies the generation of a display control member. Operands and the supporting definition/action macros that follow supply all the information needed to construct a display information controller member.

FONTGEN Macro

This macro specifies the generation of the characters loaded into the IBM 5985 character generator.

MIATGEN Macro

This macro specifies the generation of a manual input action table that describes the desired activity for operator interactions.

DEFINITION/ACTION MACROS

Definition/action macros supply the supporting information and describe various actions associated with the control statements. Some macros in this group are restricted to use in a single phase, others are used in more than one.

DISPEND Macro

This macro signals the end of the generation phase.

DISPROG Macro

This macro specifies the program(s) that becomes active when a display becomes active. The program is on a specific time queue or accessed once. Parameters are passed for specific processing.

DITEM Macro

This macro describes the display output items and their formats that are updated on time cycles or by demand. The core resident data areas (arrays) and the display items are identified during the generation phase, and display updating during online processing is done without additional programming.

DMIAT Macro

This macro defines entries in the manual input action table. A program or display controller is identified by name to become active because of operator interaction. Screen coordinates are specified to further qualify the activity.

DRAW Macro

This macro defines non-changeable or static graphic and character display items usually associated with background generation. However, they can be included in the display control generation phase.

DSYM Macro

This macro assigns symbolic names to certain operand fields.

DTIMES Macro

This macro defines an area in the common zone where the time and/or date is displayed.

DYNAMIC Macro

This macro describes dynamic portions of a display that have applicable information supplied during online processing. It is used when data and its incore locations are not resolved effectively through the normal item name or array/displacement facilities of the cyclic definition macro, DITEM.

FONT Macro

This macro defines a bit pattern loaded into the IBM 5985 character generator. It generates the character font when it is desirable to deviate from the standard supplied font. In addition, a mnemonic name is assigned to each pattern, and the name is used in other definition statements and phases to invoke the character.

PSREAD Macro

This macro defines partial areas of a display screen to be read. The resultant definitions are based on the type of unit. The IBM 5985 defines one or more areas. Each area is assigned a unique identifier used during online processing to reference the area. Only the contents of the specified location are passed to the requestor during online processing. IBM 3277 defines the unprotected formatted area used for data entry. All areas defined are considered as a single read area and do not require an identifier; the unprotected fields should be defined as background by the DRAW macrc.

CONDITIONAL FORMAT MACRO

In addition, the following three macros permit selective options during cyclic updating based on dynamically changing data.

DELSE Macro

This is an optional macro which, if present, deletes the "true" processing and signifies the start of the "false". Absence of this macro specifies there is no "false" processing.

DENDIF Macro

This macro is required and signals the end of the selective definitions.

DIF Macro

This macro describes a series of bits or values compared against a mask or value and the test to apply to it. If the test is true, the definition statements that follow are acted upon during online processing. If the test results in a false condition, the defined statements are bypassed.

APPENDIX E: PROGRAMMING MACROS

CONCEPTS

Macros generate all frequently used segments of code. These include those common to all applications and those that fulfill individual requirements of each application. All macros are coded such that:

1. They are self-documenting

2. They are written to process higher level language type statements

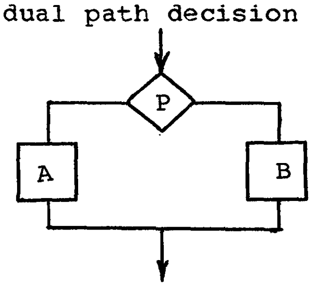
3. The code generated to perform a given function is optimized and debugged when the macro is originally written, such that coding errors are reduced, resultant code is more efficient, and the function does not have to be redesigned and rewritten each time it is used.

```
IF                  BIT,X,IS,ON,THEN
ELSE
ENDIF
```

The common set of macros defines the beginning and ending block segments used for programming in the structured form.

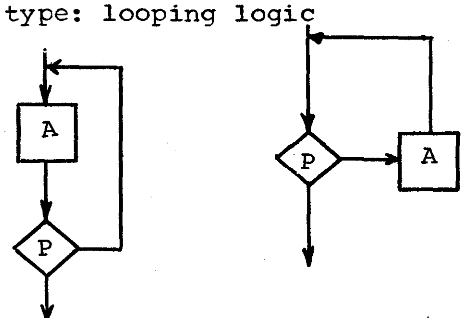
```
IF      P       THEN
        A
ELSE
        B
ENDIF
```

type: dual path decision logic



```
{UNTIL}
{WHILE}     P       DO
 A
ENDDO
```

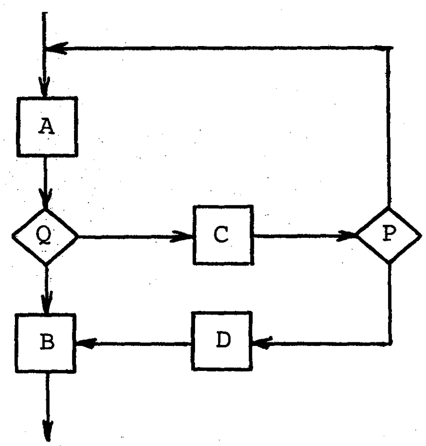
type: looping logic



(UNTIL) (WHILE)

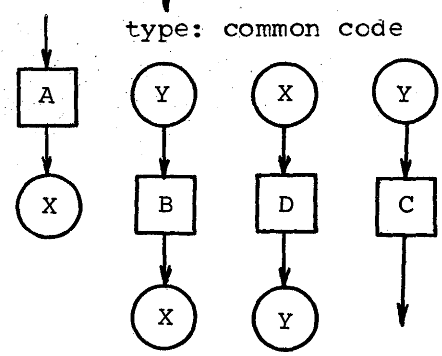
```
STRTSRCH   { UNTIL }        P     DO
           { WHILE }

   A

EXITIF q



   B

ORELSE

   C

ENDLOOP

   D

ENDSRCH
```

type: table search logic




```
A
DO   X
B
DO   X
C


BGNSEG      X
D
ENDSEG
```

type: common code



MACRO FORMATS, DEFINITIONS

The common macros are subdivided into nine function groups and are ordered as follows:

1. Data base definition:

   BIT
   BYTE

2. Bit manipulation:

   NIBIT
   TMBIT
   XIBIT
   OIBIT

3. Dual-path decision logic:

   IF
   ELSE
   ENDIF

4. Looping logic:

   UNTIL
   WHILE
   BGNWHILE
   ENDDO

5. Table search logic:

   STRTSRCH
   EXITIF
   ORELSE
   ENDLOOP
   ENDSRCH

6. Common code logic:

   DO
   BGNSEG
   ENDSEG

7. Multi-path decision logic:

   CASE

8. Error checking logic:

   ERREXIT
   ERRENTER
   ERRMSG
   ERRETURN

9. Entry, exit logic:

   HEADC
   ENTER
   EQUATE
   GRETURN

BIT MACRO

The BIT macro generates a data base definition whose length is used as a key to test or manipulate a specific bit in a byte.

DEFINITION

symbol	BIT	{ Bit number, or list of bit numbers, or binary 8-bit configuration ON }

where

- Symbol -- any valid non-blank label. If omitted, an error condition is raised with a condition code of 12.
- Bit number -- an unsigned decimal integer, 0 through 7, representing standard bit notation.
- List of bit numbers -- a list of bit numbers separated by commas. The entire list is enclosed by parentheses.
- Binary 8-bit configuration -- notation of the form B'XXXXXXXX', where X is 1 if the corresponding bit is represented by this label and X is 0 if the corresponding bit is not represented by this label.
- ON -- indicates the bit or bits indicated in the first operand are set to 1 in a global variable which is passed to the BYTE macro.

FUNCTION

The BIT macro:

- Checks to see if there is a valid non-blank label attached to the macro
- Processes the information passed by the first operand, checking each time for an invalid bit number or binary character
- Generates a EQU statement to establish a length and a scaling factor used to test or manipulate bit(s), and reset the location counter setting. (However, if the name of the CSECT currently being processed starts with SCDB, the DS and ORG statements are not generated.)

BYTE MACRO

The BYTE macro generates a data base definition using either information passed from previous calls of the BIT macro or a parameter on the BYTE macro.

DEFINITION

symbol	BYTE	one byte hex value

where

- The operand may be blank, or
- The operand is a value hexadecimal number (range is from 0 to 255) i.e., X'FF'.

FUNCTION

The BYTE macro:

- Examines the operand to determine whether it is null
- If the operand is null, the BYTE macro builds a DC using information passed from previous calls of the BIT macro
- If the operand is present, BYTE generates a DC statement using this parameter

NIBIT MACRO

The NIBIT macro generates an immediate instruction which uses the length code of the symbol specified to "turn off" a desired bit in a byte.

DEFINITION

symbol	NIBIT	Symbol

where Symbol is the label of a data base definition which has an associated length code.

TMBIT MACRO

The TMBIT macro generates a test under mask instruction which uses the length code of the symbol to be tested as the mask byte.

DEFINITION

symbol	TMBIT	Symbol

where Symbol is the label of a data base definition which has an associated length code.

XIBIT-OIBIT MACROS

The XIBIT and OIBIT macros generate an exclusive or immediate instruction to invert a specified bit, and an inclusive or immediate instruction to "turn on" a specified bit, respectively. Both macros use the length code of the symbol to be operated upon.

DEFINITION

symbol	XIBIT	Symbol

symbol	OIBIT	Symbol

where Symbol is the label of a data base definition having an associated length code.

IF MACRO

The IF macro generates the labels and instructions that branch to these labels to accomplish the IF-THEN, IF-AND-THEN, IF-OR-THEN, IF-THEN-ELSE, IF-AND-THEN-ELSE, and IF-OR-THEN-ELSE programming functions.

DEFINITION

There are six different IF statements. They are: IF-THEN, IF-AND-THEN, IF-OR-THEN, IF-THEN-ELSE, IF-AND-THEN-ELSE, and IF-OR-THEN-ELSE.

The format for the IF-THEN is:

```
IF condition

 code - body

ENDIF
```

which reads, "IF the tested condition is true, then execute the code-body".

The format for the IF-AND-THEN is:

```
IF condition,AND
IF condition,THEN

 code - body

ENDIF
```

which reads, "IF both conditions are satisfied, then execute the code-body".

The format for the IF-OR-THEN is:

```
IF condition,OR
IF condition,THEN

 code - body

ENDIF
```

which reads, "IF either condition is satisfied, then execute the code-body".

The format for the IF-THEN-ELSE is:

```
IF condition,THEN

 code - body1

ELSE
 code - body2

ENDIF
```

which reads, "IF the condition is true, THEN execute code-body1, ELSE execute code-body2".

The format for the IF-AND-THEN-ELSE is:

```
IF condition,AND
IF condition,THEN

 code - body1

ELSE
 code - body2

ENDIF
```

which reads, "IF both conditions are satisfied, THEN execute code-body1, ELSE execute code-body2".

The format for the IF-OR-THEN-ELSE is:

```
IF condition,OR
IF condition,THEN

 code-body1

ELSE

 code-body2

ENDIF
```

which reads, "IF either condition is satisfied, THEN execute code-body1, ELSE execute code-body2".

THE IF macro FLOWCHARTS

```
IF A, THEN

X

ELSE
Y
ENDIF

IF A, AND

IF B, THEN

X
```

ELSE

Y
ENDIF

IF A, OR

IF B, THEN

X

ELSE

Y

ENDIF

ELSE MACRO

The ELSE macro generates the branch and labels that correspond with the branch instructions generated by the IF macro and the labels generated by the ENDIF macro. See the IF macro.

ENDIF MACRO

The ENDIF macro generates the labels that correspond with the branch instructions generated by the IF macro. See the IF macro.

UNTIL MACRO

The UNTIL macro generates the labels and instructions that branch to these labels to do the programming function of iteration. The UNTIL macro supports both instruction for incrementing/decrementing indexes and instructions for terminating the loop based upon a change in a logical condition. The UNTIL statements support loops in which the indexing/condition-testing instructions are executed after the first pass through the code-body.

DEFINITION

There are three different UNTIL statements, the UNTIL-DO, UNTIL-OR-DO, and the UNTIL-AND-DO. For the flowcharts of the UNTIL statements, reference the ENDDO macro writeup.

The general format for the UNTIL-DO is:

1. Indexed - UNTIL-DO:

   UNTIL (index-instructions), DO

       code-body

   ENDDO
   which reads "UNTIL the following index-instructions fail to branch, continue to execute the code-body".

2. Logical - UNTIL-DO:

```
UNTIL (condition), DO

    code-body

ENDDO
```

which reads "UNTIL the following conditions are true, continue to execute the code-body".

The general format for the UNTIL-OR-DO is:

```
UNTIL (index-instruction), OR
UNTIL (index-instruction), DO

    code-body

ENDDO

UNTIL (condition), OR
UNTIL (condition), DO
    code-body

ENDDO
UNTIL (index-instruction), OR
UNTIL (condition), DO

    code-body

ENDDO
```

The general format for the UNTIL-AND-DO is:

```
UNTIL (index-instruction),AND
UNTIL (index-instructions),DO

    code-body

ENDDO

UNTIL (condition),AND
UNTIL (condition),DO

    code-body

ENDDO
UNTIL (index-instruction),AND
UNTIL (condition),DO

    code-body

ENDDO
```

The following shows the format of UNTIL:

$$\text{UNTIL (TYPE ,LABEL,OPERATICN,CONDITION),} \left\{ \begin{array}{c} \text{OR} \\ \text{AND} \\ \text{DO} \end{array} \right\} \left[\text{,REG=}\right]$$

WHILE MACRO

The WHILE macro generates the labels and instructions that branch to these labels to do the programming function of iteration. The WHILE macro supports both instructions for incrementing/decrementing indexes and instructions for terminating the loop based upon a change in a logical condition. The WHILE statements support loops in which the indexing/condition-testing instructions are executed before the first pass through the code-body.

DEFINITION

There are three different WHILE statements, the WHILE-DO, WHILE-OR-DO, and the WHILE-AND-DO. For the flowcharts of the WHILE statements, see the ENDDO macro writeup.

The general format for the WHILE-DO is:

1. Indexed WHILE-DO:

```
WHILE (index-instruction), DO

     code-body

ENDDO
```

   which reads, "WHILE the index-instruction branches, continue to execute the code-body".

2. Logical WHILE-DO:

```
WHILE (condition), DO

     code-body

ENDDO
```

   which reads, "WHILE the indicated condition is true, continue to execute the code-body."

The general format for the WHILE-OR-DO is:

```
WHILE (index-instruction), OR
WHILE (index-instruction), DO

     code-body

ENDDO

WHILE (condition), OR
WHILE (condition), DO

     code-body

ENDDO
WHILE (index-instruction), OR WHILE (condition), DO

     code-body

ENDDO
```

The general format for the WHILE-AND-DO is:

```
WHILE (index-instruction), AND
WHILE (index-instruction), DO

     code-body
```

```
ENDDO

WHILE (condition),AND
WHILE (condition),DO

     code-body

ENDDO

WHILE (index-instruction),AND
WHILE (condition),DO

     code-body

ENDDO
```

The following shows the format of WHILE:

$$\text{WHILE (}\left[\text{TYPE}\right]\text{,LABEL,OPERATION,CONDITION),}\left\{\begin{array}{c}\text{OR}\\ \text{AND}\\ \text{DO}\end{array}\right\}\left[\text{,REG=}\right]$$

BGNWHILE MACRO

The BGNWHILE macro causes execution of a WHILE loop to begin at the instruction immediately following the BGNWHILE macro. This macro is preceded by a WHILE macro and succeeded by an ENDDO macro. Normally, a WHILE loop begins at the ENDDO macro by checking the condition specified in the WHILE macro.

DEFINITION

The following example illustrates how a BGNWHILE starts execution of a loop between the WHILE and ENDDO macros.

Without BGNWHILE	With BGNWHILE
Instruction Sequence	
A	
WHILE (B), DO	WHILE (B), DO
C	C
A	BGNWHILE
ENDDO	A
	ENDDO

ENDDO MACRO

The ENDDO macro generates the labels that correspond to the labels and instructions generated by the WHILE/UNTIL macros. See the WHILE or UNTIL macros.

DEFINITION

UNTIL A,DO	WHILE A,DO
X	X
ENDDO	ENDDO

UNTIL A,OR	WHILE A,AND
UNTIL B,DO	WHILE B,DO
X	X
ENDDO	ENDDO

```
UNTIL A,AND                     WHILE A,AND
WHILE B,DO                      UNTIL B,DO
  X                                 X
ENDDO                           ENDDO
```

```
UNTIL A,OR                      UNTIL A,AND
WHILE B,DO                      UNTIL B,DO
  X                                 X
ENDDO                           ENDDO
```

```
WHILE A,OR                      WHILE A,OR
WHILE B,DO                      UNTIL B,DO
  X                                 X
ENDDO                           ENDDO
```

Notes: In an UNTIL a BCT = yes when the register = 0 after execution of BCT. In a WHILE a BCT = no when the register = 0 after execution of BCT.

STRTSRCH MACRO

The search macros generate the logic typical to what a programmer does when he sets up a loop to search through a table. The programmer's intent is to exit when he finds what he is searching for and to do process B. If he does not find what he is looking for, he executes process D before joining the alternate path. The ORELSE is optional and if it is omitted, box C does not appear in the flowchart.

DEFINITION

```
STRTSRCH  { WHILE }  , (condition) , { OR  }      [ ,REG= ]
          { UNTIL }                  { AND }
                                     { DO  }
```

The STRTSRCH macro uses the WHILE/UNTIL field to generate a WHILE or UNTIL macro statement. The condition format is the same as the WHILE and UNTIL macro.

EXAMPLE

```
STRTSRCH condition p
    Process A
EXITIF condition q
    Process B
ORELSE
    Process C
ENDLOOP
    Process D
ENDSRCH
```

Note: When using these macros, care should be taken not to confuse the ENDLOOP and ENDSRCH macros. The ENDLOOP defines the end of the loop and the ENDSRCH indicates the end of the completed macro set.

When nesting these macros, each macro set is completely embedded within the process boxes of the higher level ones.

EXITIF MACRO

The EXITIF macro tests a condition to verify whether to continue the loop or exit out of the loop. See the STRTSRCH macro. The following shows the format of the EXITIF macro.

DEFINITION

```
EXITIF   [ condition ]   ,  { OR   }  [ ,REG= ]
                            { AND  }
                            { THEN }
```

The condition format is the same as the IF macro except that the label IF is not specified.

ORELSE MACRO

The ORELSE macro generates the branch and labels that correspond with the branch instructions generated by the EXITIF macro and the labels generated by the ENDLOOP macro. See the STRTSRCH macro.

ENDLOOP MACRO

The ENDLOOP macro defines the end of the loop. See the STRTSRCH macro.

ENDSRCH MACRO

The ENDSRCH macro indicates the end of the complete macro set. See the STRTSRCH macro.

DO MACRO

The DO MACRO generates a branch-and-link to a segment of code, defined by the BGNSEG and ENDSEG macros.

DEFINITION

	DO	SEGMENT,REG

where

SEGMENT is the label of the section of code to be branched to
REG is the register to be used. If the register is not specified, register 14 will be used.

If the register used in branching to and from a segment has been defined by a previous DO or BGNSEG macro, it uses a different register to print an error message. Registers need to be expressed in notation $1, $2, etc. A maximum of 50 segments may appear in an assembly.

BGNSEG MACRO

The BGNSEG macro generates a label for a section of code branched to by the DO macro.

DEFINITION

	BGNSEG	SEGMENT,REG

where

SEGMENT is the name of the label to be generated
REG is the register used in returning from this segment of code.

When the BGNSEG macro follows a DO macro which references it, BGNSEG uses the register specified in the DO macro. If the register is specified in the BGNSEG macro and it does not agree with the register specified in the previous DO macro, an error message is written.

When the BGNSEG macro precedes any DO macro reference to it, the register defaults to $14 unless a register is specified.

A maximum of 50 segments may appear in an assembly. Registers must be expressed in notation $1, $2, etc.

ENDSEG MACRO

The ENDSEG macro generates a BR instruction. It returns from a segment of code that is branch-and-linked to by the DO macro.

DEFINITION

```
            ENDSEG      &SEGMENT
      +     BR          &REG
```

where

&SEGMENT is the name of the segment to be terminated.
&REG is the register to be used, and is determined by either a previous DO or BGNSEG macro.

CASE MACRO

This macro generates the code necessary for certain, frequently encountered, decision table type processing logic. In this type of processing, one usually has a case (index) number in some GPR and desires to execute one of a list of options (cases) based upon the value of the case number in the GPR. The following block diagram shows the basic flow of this type of logic:

In this macro it is assumed that the increase in the case numbers is a power of two (that is, 1, 2, 4, 8, . . .) and that the cases are numbered starting with zero. It should be noted that CASE loads the specified RETREG with the address of the instruction following the macro before branching to the determined case; and, it is the responsibility of each case to return to the address specified in the RETREG (if the requirements of structured coding are to be fulfilled).

DEFINITION

[symbol]	CASE	case register, { AT = (address list) | BT = (address list) | LAT = {(R) | addr.} | LBT = {(R) | addr.} } [,INDX=number] [,RETREG=register]

where

Case register - is the register number (or symbol equated to the register number) of the general purpose register (GPR) that contains the desired case number. This must not be the same register that is used as the RETREG.

AT = (address list) - is a list of up to 255 case labels. This list of case labels generates a corresponding list of address constants. When this form of the CASE macro expands, the case register indexes into this list of address constants (ADCONS) to determine which case is branched to. There is a one-to-one correspondence between a label's position in the list and its associated case number (that is, the first label in the list is the name of the case to receive control when the case register has a zero. If a label is null, an address of zero is generated for the associated case number. (This is used for any embedded case numbers which are not expected to create a desired program check if that case number does occur.) An * is coded in place of any of the labels to signify that processing is to continue at the instruction following the macro when the associated case(s) occurs. It should be noted that by specifying one or more of the labels (used in an AT type expansion) in an EXTRN statement, the CASE macro becomes effectively an indexed CALL macro.

BT = (address list) - is a list of up to 255 case labels, as defined for the AT type expansion. The only difference between the AT and the BT type expansions is that BT generates a branch table instead of an address table for the labels specified. This permits the use of case labels that are not in the same CSECT nor callable, but for which a base register is set up.

(R)
LAT = addr. - is the address of a remote list address table used by CASE in determining where to branch for each value placed in the case register. This address is specified in a register as (R) where R is some register number (not being used as a case register or a RETREG).

(R)
LBT = addr. - is the address of a remote list of branch instructions used by the CASE in branching to the case designated by the value in the case register. As in LAT this address is also specified in a register form.

INDX = number - specifies the increment used in counting the cases. This must be some power of 2 (that is, 1, 2, 4, 8, 16, 32, . . .). The default for INDX is 4. (This says that the cases are numbered 0, 4, 8, 12, 16, . . .).

RETREG = register - specifies the register to be set up as the linkage register on the branch. This is specified as any register number or symbol equated to a register number. The default for RETREG is 14.

ERREXIT MACRO

This macro provides an exit for error checking logic.

DEFINITION

```
          ERREXIT &A,&B,&C,&D,&E,&F,&G
                        'IF'
                  &A =  SYMBOL
```

If a symbol is coded for &A, it is not greater than 4 characters long and it is the operand of an ERRENTER or ERRMSG macro elsewhere in the CSECT. &B - &G are ignored, and the macro generates a BC 15, ERXT (symbol).

If &A = IF, then the operands &B - &G are coded exactly as though they were operands of an IF macro with the exception of &F. &F is normally 'THEN', 'AND', or 'OR' in the IF macro, but it is a symbol not greater than four characters long in the ERREXIT macro and the same symbol is the operand of an ERRENTER or ERRMSG macro elsewhere in the program.

Using ERREXIT in case 2 expands into the same code as the IF macro, except for the BRANCH instruction generated by IF. Instead it generates a BRANCH to the symbol ERXT(symbol) on the condition specified by the operands &B - &E. (No ENDIF is associated with an ERREXIT macro).

ERRENTER MACRO

This macro provides an entry for error checking logic.

DEFINITION

	ERRENTER	&A

&A = a symbol not greater than four characters in length.

The ERRENTER macro begins a segment of special error processing for a particular error designated by &A, specified in an ERREXIT macro. The segment ends with (1) an ERRMSG macro for an error message if one is required, (2) another ERRENTER macro for a different error condition, or (3) the ERRETURN macro.

If the ERRENTER macro is preceded by another ERRENTER macro (with no ERRMSG macro between the two), it expands to a branch to ERRETURN prior to defining the error symbol. Otherwise, it expands to a definition of the error symbol.

The following example shows how ERRENTER processes special error conditions.

Suppose there are three error conditions (ER1,ER2,ER3): one which requires an error message only, one which requires special processing only, and one which requires special processing and an error message. The following code demonstrates the use of ERRENTER in conjunction with the other ERROR macro to accomplish these results:

```
                    body of csect with ERREXIT macros
                        to ER1,ER2,ER3)

        GRETURN
        ERRMSG ER1
        DC   C'(error message for er1)'
        ERRENTER ER2
                    (special processing for er2)
        ERRENTER ER3
                    (special processing for er3)
        ERRMSG
        DC   C'(error message for er3)'
        ERRETURN
                    (common error processing)
        GRETURN
```

ERRMSG MACRO

The ERRMSG macro defines an error message for the error condition designated by &A. &A should be left blank if the error condition is designated by an ERRENTER macro (with the associated special error processing) immediately preceding the ERRMSG macro.

DEFINITION

	ERRMSG	&A ,&B

where

&A = a symbol not greater than 4 characters in length
&B = a register number (defaults to 0)

The error link register is specified by &B and is specified only by the first ERRMSG macro in the CSECT. &B then defaults to that of the first ERRMSG macro for subsequent ERRMSG macros and defaults to 0 on the first ERRMSG macro, if not specified.

The ERRMSG macro expands to a BAL off the error link register to ERRETURN, defining the BAL instruction with the error symbol, if one is specified.

See the ERRENTER macro for an example.

ERRETURN MACRO

The ERRETURN macro expands to a definition of the symbol ERREXIT$. The ERRETURN macro is used to begin common error processing. See the ERRENTER macro for an example of its use.

HEADC MACRO

The HEADC macro generates the CSECT card, save area, entry coding, and return coding for a single entry point assembly language program. It also invokes the EQUATE macro.

DEFINITION

CSECT name	HEADC	INTP = YES ,RET = YES

where

CSECT name is the name on the generated CSECT card and the entry point for the program. If INTP = YES is coded, a 22-word save area is generated in place of the standard 18-word save area. A 22-word save area is needed if the program INTP is used.

If RET = YES is coded, register 15 is restored as part of the return logic. This allows the programmer to store a return code in that register. INTP = YES and RET = YES are not positional parameters; they are keyword parameters.

HEADC points GPR 13 to the save area and does a 'USING' on GPR 13 so it serves as the base register for the program. The return logic is reached by branching to the label RCSECT name. If this label is more than eight characters, the right-most character is truncated in the generated macro label, and an assembly error is flagged in the B RCSECT statement. To avoid the error message, the programmer truncates RCSECT to eight characters in coding the B RCSECT statement.

ENTER MACRO

The ENTER macro generates multiple - entry point code. The macro generates:

1. One CSECT card (CSECT name = 1st subparameter of the first operand.)

2. An ENTRY card for each entry point

3. One save area and SAVE code which establishes R13 as a base register.

4. A label to branch to RETURN (label = an R concatenated with the CSECT name)

5. '$0 EQU 0',...., '$15 EQU 15' so that an XREF is given of register usage if the $XX symbols specify registers. (The EQU's are generated only once per assembly even though more than one ENTER is coded.)

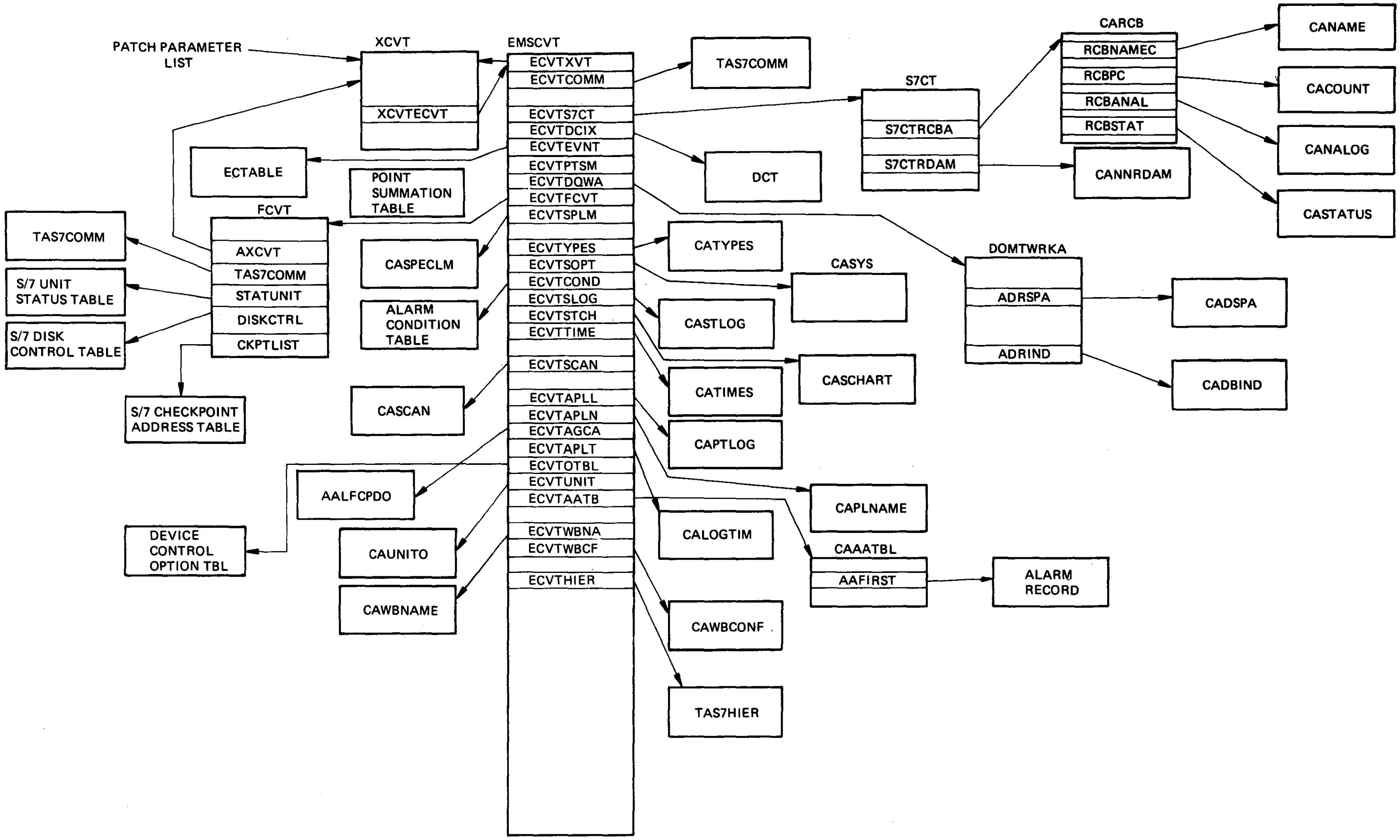
6. Register 15 is loaded with the address of the code associated with the respective entry point (that is, the respective name specified in the second operand sublist - if left blank, '$' is concatenated with the respective entry point specified in the first operand sublist) so that one executes 'BR $15' after executing code which is common for all entry points.

GRETURN MACRO

The GRETURN macro expands to a B R&SYSECT. It is used in conjunction with the HEADC and ENTER macros.

EQUATE MACRO

The EQUATE macro generates $0 EQU 0 . . . $15 EQU 15 and FPRO EQU 0 . . . FPR6 EQU 6 statements by both the HEADC and ENTER macros. A CSECT does not require a save area (and hence would not use HEADC or ENTER). EQUATE is used to generate EQU statements.



Control Block Overview

Page of LY20-2226-0
Added August 31, 1976
By TNL LN20-3620

INDEX

READER'S COMMENT FORM

IBM System/370 LY20-2226-0

Energy Management System

Logic Manual

Please comment on the usefulness and readability of this publication, suggest additions and deletions, and list specific errors and omissions (give page numbers). All comments and suggestions become the property of IBM. If you wish a reply, be sure to include your name and address.

COMMENTS

Fold Fold

Fold Fold

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.
  FOLD ON TWO LINES, STAPLE AND MAIL.

Your comments, please . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. Your comments on the other side of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Fold

LICENSED MATERIAL – PROPERTY OF IBM

Fold

First Class
Permit 40
Armonk
New York

Business Reply Mail

No postage stamp necessay if mailed in the U.S.A.

Postage will be paid by:

International Business Machines Corporation
1133 Westchester Avenue
White Plains, New York 10604

Att: Technical Publications/Industry – Dept. 825

Fold

Fold

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
360 Hamilton Avenue, White Plains, New York 10601
(International)

IBM Technical Newsletter

This Newsletter No. LN20-3620
Date August 31, 1976

Base Publication No. LY20-2226-0
File No.

Previous Newsletters None

IBM System/370
Energy Management System
Logic Manual

Program Number: 5740-U11

This Technical Newsletter, a part of Version 1, Modification Level 1, of IBM System/370 Energy Management System, provides replacement pages for the subject manual. These replacement pages remain in effect for subsequent versions and modifications unless specifically altered. Pages to be inserted and/or removed are listed below.

Front Cover, Inside Front Cover
Contents iii, iv
2-3 - 2-6
2-39, 2-40
2-49, 2-50
2-98.3 - 2-98.6
2-105 - 2-108
2-111 - 2-114
2-183, 2-184
2-195, 2-196
3-5, 3-6
3-15 - 3-24
3-35, 3-36
3-39 - 3-42
3-53 - 3-56
3-91, 3-92 (added)
4-17, 4-18
4-29 - 4-44
4-55, 4-56
4-61 - 4-64
4-79 - 4-82
4-87 - 4-90
4-99 - 4-106 (4-102.1, 4-102.2 added)
4-151 - 4-156
4-167 - 4-178
4-183, 4-184
4-184.1, 4-184.2 (added)
4-201, 4-202
4-204.1 (added)
4-213 - 4-220
4-223 - 4-226
4-229 - 4-238
4-241 - 4-250 (4-242.1 - 4-242.4 added)
4-255 - 4-258
5-1 - 5-6
6-29, 6-30
6-51, 6-52
6-59, 6-60
6-71 - 6-74
6-74.1, 6-74.2 (added)
6-83 - 6-86
6-86.1, 6-86.2 (added)
6-89, 6-90
6-107, 6-108
6-115 - 6-118
A-1, A-2
F-1 (added)
X-1, X-2
Back Cover
Readers' Comment Form

A vertical rule in the left margin indicates a change. Absence of a vertical rule on a page bearing a 'revised' notice means only that existing copy has been moved or that a minor typographical error has been corrected.

Please file this cover letter at the back of this manual to provide a record of changes.

IBM Corporation, Technical Publications/Industry, Dept. 825, 1133 Westchester Ave., White Plains, N.Y. 10604

Printed in U.S.A.

LY20-2226-0

IBM System/370 Energy Management System Logic Manual Printed in U.S.A. LY20-2226-0

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
360 Hamilton Avenue, White Plains, New York 10601
(International)