IN-06

NASA Contractor Report 189605

P_291

Advanced Transport Operating System (ATOPS) Color Displays Software Description Microprocessor System

Christopher J. Slominski
Valerie E. Plyler
Richard W. Dickson

Computer Sciences Corporation
Hampton, Virginia

Prepared For
Langley Research Center
under Contract NAS1-19038
January 1992

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665-5225

TABLE OF CONTENTS

LIST OF FIGURES

Section 1.0 INTRODUCTION

This document describes the software modules used in the Primary Flight, Navigation, Takeoff, and Engine Display Formats delivered in the spring of 1991. This work was performed under contract NAS1-19038. The display formats will henceforth be referred to by the mnemonics PFD, NAV, TOP, and ENG respectively. Both applications and system level software are described in the document. Hardware description is included only to the extent required to understand the functions of the system software. Those readers whose interests lie in the Sperry hardware involved in the display system should read the document

NASA TSRV COLOR DISPLAY SYSTEM
SYSTEM SPECIFICATION
REVISION A

or contact The Experimental Flight Systems Section (EFSS) of the Advanced transport Operating System (ATOPS) project office.

There is no attempt to teach the reader the use of Sperry's "Core Graphics System". If detailed information about performing graphics commands is desired, then the document

EFIS
CORE GRAPHICS SYSTEM
PROGRAMMER'S GUIDE
FOR
D-SIZE INDICATORS

should be referenced. Detailed interpretation involving the meaning of the display formats and their symbology should be obtained from the ATOPS project office.

The display software receives data from a MicroVAX computer, often referred to as the host computer. The host software is described in another document entitled

ADVANCED TRANSPORT OPERATING SYSTEMS
COLOR DISPLAY SOFTWARE
DESCRIPTION
(MicroVAX System)

The Sperry display system is controlled by a Sperry PC called the System Control Panel (SCP). The use of the SCP for loading formats, displaying memory, etc... is described in the document

THE LOW END PANEL

which is the original name given to the SCP.

Throughout this document, descriptions of software modules are presented in a standardized format. The basic template is shown on the next page. At the top of the form is a header block containing miscellaneous information about the module. Note that no CALLING SEQUENCE entry is shown in the header block. This is because most modules do not use formal calling parameters. For those procedure having a parameter list, a Calling Sequence entry will appear in the header (not shown in the template). Some of the modules in the document do not have a CALL RATE line in the header. These modules are utility modules which can be called a different number of times in any given software cycle.

Next appears a one or two sentence synopsis used as a quick reference stating the purpose of the module. A detailed description follows which may be a small paragraph to several pages in length. Global symbol references are listed next. These are the subroutines and common variables referenced by the particular module. When an asterisk is appended to the name of a data variable listed in the gloabl reference section it denotes a memory location modified by the module.

```
MODULE NAME:  ........
FILE NAME:    ........
CALL RATE:    ........
CALLED BY:    ........

PURPOSE:     ................................................

REMARKS:     ................................................
.............................................................
.............................................................

GLOBAL REFERENCES:

VARIABLES
  ....... ....... .......

PROCEDURES
  ........ ........ ........
```

Section 2.0 SYSTEM OVERVIEW

The Sperry Color Display System communicates with a MicroVAX host computer through a DMA channel. The information from the host is processed by Intel 80186 micro-processors which generate data usable by Sperry display units for the creation and positioning of display symbology. There are twelve individual pieces of hardware involved in the standard Sperry display configuration which the reader must be familiar with to understand the software described in this document. Included are eight color display units (DUs), three display electronic units (DEUs), and one bus interface unit (BIU).

The job of each DU is to create a picture on its "D" size display screen (6.7 x 6.7 inches) from data received on the high-speed bus. A Zilog Z-8000 microprocessor controls the DU by running the "Core Graphics" program fixed in ROM. No user programs can be loaded into DUs.

Each DEU has four Intel 80186 microprocessors. They contain user programs which perform I/O functions and create data, through "Core Graphics" routines, in formats acceptable to the DUs.

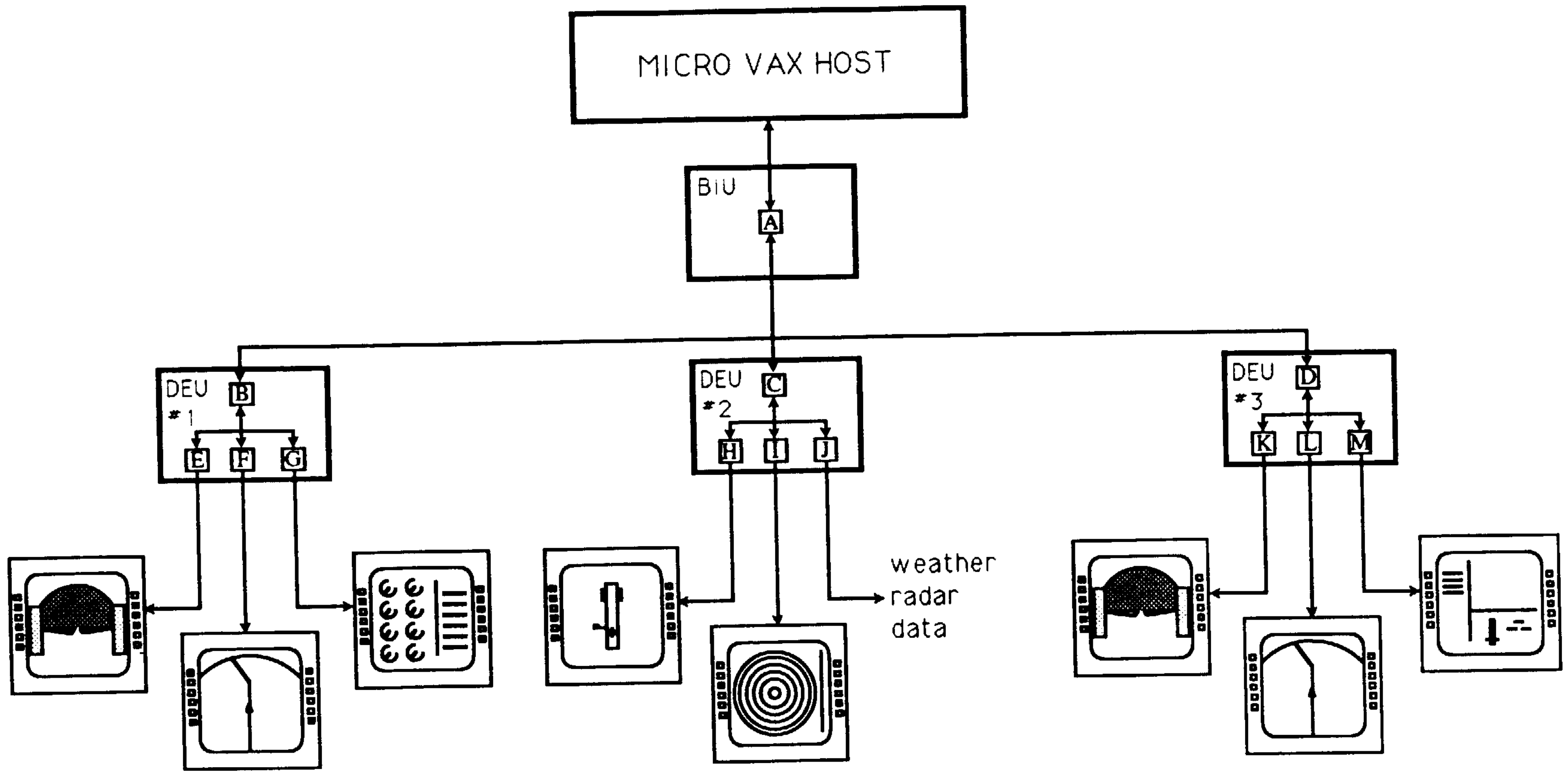
The BIU has a single Intel 80186 microprocessor which is dedicated to performing I/O operations between the host computer and the DEUs.

Section 2.1 THE USER PROGRAMS

This document covers six user programs which can be loaded into one or more of the Intel 80186 display processors (DPs). Four of these programs are the PFD, NAV, TOP, and ENG graphics programs. The others are the I/O interface program for the BIU and the DEU on-board I/O processor program used in each DEU.

Figure 2.1 depicts the 13 DPs with the letters A - M. The table below identifies each DP with the SCP name used for it and a description of its function.

DP	NAME	DESCRIPTION
A	DP01	I/O interface processor. Transmits data between the MicroVAX computer and all DEUs. Loaded with special BIU software.
B	DP14	On-board I/O processor. Performs I/O operations with the BIU and reads DEU discretes such as the bezel button inputs. Loaded with a copy of the "DP4" I/O program.
C	DP24	See description for processor B.
D	DP34	See description for processor B.
E	DP11	Graphics processor. Uses data from its DP4 on-board I/O processor to create graphics commands that drive one DU. Any applications format may be loaded into this processor.
F	DP12	See description for processor E.
G	DP13	See description for processor E.
H	DP21	See description for processor E.
I	DP22	See description for processor E.
J	DP23	Weather radar processor. This DP feeds radar data to each DEU. Special Sperry software not covered in this document is loaded.
K	DP31	See description for processor E.
L	DP32	See description for processor E.
M	DP33	See description for processor E.



DISPLAY PROCESSORS

-figure 2.1-

Section 2.2 BEZEL BUTTONS AND POTENTIOMETERS

Each DU has sixteen buttons and two potentiometers fixed on the outer front edge (bezel area). The state of these devices is made available to the DEU which contains the DP that drives that DU. The DEUs read the discrete inputs and save them in common memory which is available to all DPs within the DEU. The DPs may then fetch the bezel data that is pertinent to their application. Note also that each application format program places copies of the bezel data into the data buffer which is transmitted from the DEU to the MicroVAX via the BIU.

Section 3.0 THE I/O SYSTEM

Data transfer to and from the host MicroVAX and the application display processors (DP) (which drive each display unit) is through the BIU-DP4 system. The BIU is responsible for interfacing the MicroVAX to the DP4 processors resident in each DEU. The BIU and DP4 communicate through a 1 megabit/second serial data link known as the High-Speed Bus (HSB). The display system is said to be synchronized with the host computer (which is in turn synchronized to the DATAC clock). This means that all I/O activity is scheduled relative to the 50 millisecond major frame of the MicroVAX. Once per 50 millisecond frame the MicroVAX will initiate a read-write data transfer cycle with the BIU. This event is used by the BIU as a start of frame time reference. The BIU will then send each DP4 a start frame indicator across the HSB in the form of a data packet containing the information just received from the MicroVAX. After receiving this information, the DP4 will then alert the application DPs, through a common memory area, to start their 50 millisecond frame. This chain of events keeps the application processors in phase with the 50 msec frame rate of the MicroVAX. This system eliminates the windowing effect in autonomously clocked systems as slight variations in clock frequency between processors causes events in each processor to shift in time relative to each other.

Section 3.1 DSPHDL, INTERFACE TO THE BIU

The MicroVAX's data link with the BIU is through a DR11 DMA Controller. Processor Control and communication with this DMA device is through a Command and Status Register (CSR). This CSR is written to and read from by both the MicroVAX (with the I/O handler task DSPHDL) and BIU and serves to coordinate data transfers between them. During its fourth minor frame the host computer performs I/O with the BIU. This I/O involves a read of 320 words from the BIU followed by a write of 704 words to the BIU. Before initiating each of these I/O transfers, the MicroVAX will check the status of the BIU by reading the CSR. The BIU, if ready for an I/O command, will insert a code into three bits of the CSR (these bits are referred to as the DSTAT bits). If the DSTAT bits contain a different value than the host is expecting, no DR11 I/O is initiated and a check is made to see if the invalid code is the result of a BIU power fail or a malfunctioning BIU. This guards against DSPHDL initiating I/O which cannot proceed due to an inoperative BIU. After DSPHDL issues a command an interrupt is sent to the BIU. While fielding this interrupt the BIU will read the DSTAT bits which were preloaded by DSPHDL with a code corresponding to the command requested. This command will be either a DR11 read or write function. The BIU will then interpret this code and service the DR11 command. All possible values for the DSTAT bits and their meanings are covered in Section 3.2.3.

Section 3.2 BIU SOFTWARE

The Bus Interface Unit (BIU) system is used to interface the MicroVAX host computer to the Display Electronic Units (DEUs). The BIU system utilizes an Intel 80186 based computer that communicates with the MicroVAX through a DR11 DMA and communicates with the display system through the Sperry High-Speed Bus. During each 50 millisecond frame, data is transferred to and from the VAX and also to and from each DEU.

The BIU software system consists of the following modules containing the listed procedures:

BIU_RAM.ASM	- (data declarations)
BIU_EQU.INC	- (include file)
BIU_MAIN.ASM	- BIU_MAIN
BIU_INIT.ASM	- BIU_INIT
BIU_ISR.ASM	- TX_FAIL_ISR, TX_DMA_ISR, RTC_ISR, RX_ISR, MONITOR, DR11_ISR, DR11_TRAP_ISR, WAIT_ISR, TRAP
DR11.ASM	- DR11_READ, DR11_WRITE
HSB_SND.ASM	- HSB_SND
HSB_RCV.ASM	- HSB_RCV
HSB_INIT.ASM	- HSB_INIT
TRANSMIT.ASM	- TRANSMIT
RECEIVE.ASM	- RECEIVE

Section 3.2.1 BIU SOFTWARE EXECUTION

The following is a step by step overview of the BIU software system in a very general form followed by a more detailed description of each step. At times throughout these steps the possibility exists that the processor may be servicing an interrupt request. These interrupts are thus listed briefly in this section. For a more detailed description of any interrupt service routine see Section 3.2.2.

1) INITIALIZE BIU
2) SET UP INTERNAL TIMER 2 INTERRUPT VECTOR TO POINT TO DR11_TRAP_ISR
3) WAIT_FOR DR11 READ FUNCTION (DATA TO MicroVAX)
4) PERFORM DR11 READ FUNCTION
5) WAIT FOR DR11 WRITE FUNCTION (DATA FROM MicroVAX)
6) PERFORM DR11 WRITE FUNCTION
7) SEND DATA TO DEUs ON HIGH-SPEED BUS (HSB)
8) SET UP INTERNAL TIMER 2 INTERRUPT VECTOR TO POINT TO WAIT_ISR
9) WAIT_6 MSEC
10) GET DATA FROM EACH DEU
11) GO TO LINE 2

1) INITIALIZE BIU
   (Performed in BIU_INIT)
   A) Set all interrupt vectors to point to "TRAP". This routine is the handler for all unexpected interrupt types.
   B) Set all used interrupt vectors to point to the correct interrupt service routine (ISR). Interrupt type 19 (internal timer 2) is not set here but rather in the main BIU loop.
   C) If this is a cold start then reset variables in RAM.
   D) Set up 80186 chip select registers so the processor can access memory correctly.
   E) Set all interrupt priorities and set trigger modes to level triggered.
   F) Clear all interrupt in service bits to erase any previous pending interrupts.
   G) Set attention bit to reset DR11.
   H) Preset selected variables in RAM to initial values.
   I) Clear attention bit to arm DR11.

PAGE 24 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

2) SET UP INTERNAL TIMER 2 INTERRUPT VECTOR TO POINT TO DR11_TRAP_ISR (Performed in BIU_MAIN)
Because of a lack of available internal timers, timer 2 is shared between two separate functions. Since only one function is required at any given time, this can be accomplished by modifying the interrupt vector for this timer (interrupt type 19) to point to the required ISR. In this case the function to be accomplished is a failsafe timer for a DR11 DMA function. If this DMA takes longer than expected the ISR "DR11_TRAP_ISR" will be executed.

3) WAIT FOR DR11 READ FUNCTION (DATA TO MicroVAX) (Performed in BIU_MAIN)
At this point the BIU must wait for the MicroVAX to signal it is ready to accept data. This is sensed by the reception of the command from the DR11 for a read function via an external interrupt 3 (type 15).

4) PERFORM DR11 READ FUNCTION (Performed in DR11_READ)
The DR11 data transfer from the BIU to the MicroVAX is now performed. The software mechanism by which this takes place is the ASM86 instruction "REP MOVSW" (REPeat MOVe-String-Word). Timer 2 is set to interrupt (DR11_TRAP_ISR) if the transfer is not completed in 2 milliseconds. In this transfer 320 words of data are written.

5) WAIT FOR DR11 WRITE FUNCTION (DATA FROM MicroVAX) (Performed in BIU_MAIN)
At this point the BIU must wait for the MicroVAX to signal it is ready to send data. This is sensed by the reception of the command from the DR11 for a write function via an external interrupt 3 (type 15).

6) PERFORM DR11 WRITE FUNCTION (Performed in DR11_WRITE)
The DR11 data transfer from the MicroVAX to the BIU is now performed. The software mechanism by which this takes place is the ASM86 instruction "REP MOVSW" (REPeat MOVe-String-Word). Timer 2 is set to interrupt (DR11_TRAP_ISR) if the transfer is not completed in 3 milliseconds. In this transfer 704 words of data are read.

7) SEND DATA TO DEUs ON HIGH-SPEED BUS (HSB)
(Performed in HSB_SND, HSB_INIT, and TRANSMIT)
The data received from the previous DR11 write is now transmitted on the HSB to all three DEU's. This is done by setting up an internal DMA channel to pipe data to the Zilog Z8030 Serial Communication Controller (SCC). The SCC will then send the data to all DEU's simultaneously by specifying this data frame as a broadcast frame. This is performed by using the value 0FFh as a destination code for the frame rather than a specific address. The SCC is commanded to transmit the data received from the DMA via a set of command registers. These registers are detailed in the Zilog SCC documentation. Internal timer 0 is used as a fail-safe for this transfer. Should the transfer take twice as long as expected an interrupt type 8 (TX_FAIL_ISR) will be issued.

8) SET UP INTERNAL TIMER 2 INTERRUPT VECTOR TO POINT TO WAIT_ISR (Performed in BIU_MAIN)
As explained in #2 above, timer 2 is shared by two functions. The second function is a general purpose wait function. It uses the ISR "WAIT_ISR". Communication between this ISR and the executing application code is thru the flag "TIMER_FLAG". It is cleared when timer 2 is initialized and set by TIMER_ISR after the terminal count of the timer is reached.

9) WAIT 6 MILLISECONDS
(Performed in BIU_MAIN)
This wait gives the DEU's time to process data just received. The wait is timed by timer 2.

10) GET DATA FROM EACH DEU
(Performed in BIU_MAIN, HSB_RCV, TRANSMIT, and RECEIVE)
Data is now obtained from each DEU so that it may be relayed back to the MicroVAX on the next frame's DR11 read. To receive this data the BIU must first transmit a prompt to the appropriate DEU on the HSB and then wait for the data response. The DEU after receiving this prompt will transmit back to the BIU the requested data. Timer 2 is used to terminate the BIU's waiting if no response is received from the DEU 5 MSEC after issuing the prompt. All three DEUs are handled sequentially in this manner (DEU1, DEU2, DEU3).

11) GO TO LINE 2
Start next 50 MSEC frame.

The following interrupts are used as real time events for the BIU. They are described more fully in the Section covering procedures (3.2.2).

TX_FAIL_ISR	- This interrupt is used as a failsafe timer for the HSB transmitter.
TX_DMA_ISR	- This interrupt is generated by the internal DMA channel upon completion of a HSB transmit.
RTC_ISR	- This interrupt is caused by the external real time clock.
RX_ISR	- This interrupt is generated by the external SCC upon completion of a HSB reception.
MONITOR	- This interrupt comes from the Sperry SCP. This ISR is used as an intermediary procedure between the interrupt and the execution of Sperry monitor code.
DR11_ISR	- This interrupt is provided by the MicroVAX DR11 to initiate a data transfer between it and the BIU.
DR11_TRAP_ISR	- This ISR is used to handle DR11 DMA timeouts.
WAIT_ISR	- This ISR is a general purpose timer interrupt handler.
TRAP	- This ISR handles any interrupt not expected by the system.

Section 3.2.2 BIU PROCEDURES

The following is an alphabetical list of the procedures which comprise the BIU software system along with a functional summary of each.

BIU_INIT - This procedure is called from BIU_MAIN on startup. It is responsible for performing the majority of the BIU system initialization. The following initialization is performed:

a) Initialize all interrupt vectors with the address of their corresponding service routines.
b) If initializing immediately after a power-up, clear memory in DATSEG used for variables.
c) Initialize 80186 configuration control registers. These include peripheral chip selects, memory chip selects, interrupt control, timer control, and DMA control registers.
d) Initialize selective variables.

BIU_MAIN - This procedure serves as the executive routine for the entire BIU system. Processor control is transferred here after executing the PROM boot in high memory. Upon startup BIU_MAIN initiates the system initialization process by calling BIU_INIT. After returning BIU_MAIN settles into its main processing loop. This loop is synchronized with the VAX DR11 I/O and as such shares the same 50 millisecond period. While executing the main loop BIU_MAIN calls the routines DR11_READ, DR11_WRITE, HSB_SND, and HSB_RCV.

DR11_ISR - This interrupt service routine processes interrupts from the VAX DR11. It uses external interrupt 3. The MicroVAX will place a code into a register indicating what what type of I/O function it is requesting, which is read by this routine and saved for use by the main processing loop.

DR11_READ - This routine is called by BIU_MAIN after receiving a command from the MicroVAX for the BIU to send data (data to DR11). The ASM86 instruction "REP MOVSW" is used to transfer 320 words of data to the DR11. This data is primarily composed of returned information from the DEUs (32 words per display processor) along with 32 words of status information from the BIU. This transfer has a 2 millisecond timeout associated with it.

DR11_TRAP_ISR - This ISR processes interrupts caused by the timeout of a DR11 I/O function. If a function commanded by the MicroVAX is not completed within a specified time interval, internal timer 2 will interrupt the CPU. Upon receiving this interrupt this routine will attempt to terminate the DR11 function and increment an error counter. The vector for this service routine is shared with WAIT_ISR.

DR11_WRITE - This routine is called by BIU_MAIN after receiving a command from the MicroVAX for the BIU to receive data (data from DR11). The ASM86 instruction "REP MOVSW" is used to transfer 720 words of data from the DR11. This data is then sent to the DEUs for processing by the display processors so it may be displayed on the display units. This transfer has a 3 millisecond timeout associated with it.

HSB_INIT - This routine is called from HSB_SND and is used to initialize the Zilog SCC to operate using the HSB. This initialization procedure is done by writing to control registers mapped to a special area of memory (HSBSEG). The bit significance of these registers is covered in the Zilog Serial Communication Controller handbook.

HSB_RCV - This routine is called by BIU_MAIN when data is needed from the DEUs. This routine will first set up TRANSMIT to send a prompt message to the DEU of interest requesting data. The BIU will then go into a listening mode with a call to RECEIVE while waiting for the DEU to respond. Reception of a frame of data from the DEU will be made known to HSB_RCV by the interrupt service routine RX_ISR through the word HSB_STATUS. However, should the reception not occur within 5 milliseconds the attempt will be aborted. Data received from the DEUs will be sent to the MicroVAX during the next frame by the routine DR11_READ.

HSB_SND - This routine is called by BIU_MAIN and is used to setup a HSB transfer to the DEUs of the data received from a DR11_WRITE. This routine initializes parameters needed by the subroutine TRANSMIT (which actually performs the HSB I/O). After calling TRANSMIT this routine will poll the word HSB_STATUS waiting for the transfer to complete. Upon completion the interrupt service routine TX_DMA_ISR will clear HSB_STATUS and allow HSB_SND to finish.

MONITOR - This interrupt service routine processes interrupts originating from the Sperry SCP. The vector for this interrupt is initially set by the PROM boot code to 4000:80H. This would send control directly to the SCP upon an interrupt from the SCP. As a debug aid the need arose to construct an intermediary routine between the occurrence of the interrupt and control transfer to the SCP. Such an intermediary allows recording any data not normally available from the SCP; e.g., microproccesor I/O space. This would be done by inserting code into this routine which would store the data of interest into memory for examination later by the SCP. The method used for this was to overwrite the previously initialized interrupt vector with the address of this ISR. This is done in BIU_INIT and causes the interrupt to transfer control to MONITOR. MONITOR upon entry writes to the DSTAT lines to inhibit any VAX I/O. At this point any code needed to record data for debug purposes may be inserted. Control is then transferred to the SCP by generating a software interrupt type 32 to the SCP address (4000:80H). When the SCP relinquishes control, via an IRET, processing again returns to MONITOR. MONITOR then outputs to the DSTAT lines enabling VAX I/O and issues its end of interrupt.

RECEIVE - This routine is called by HSB_RCV. It is used to set up and arm the SCC for reception of data across the HSB. This reception mode is entered via a two step process. First, the DMA 0 is initialized to accept data from the SCC and place it into memory. Secondly, the SCC is set up to receive data from the HSB and send it to DMA 0. Upon receiving a frame of data, the SCC will interrupt the processor through the service routine RX_ISR.

RTC_ISR - This ISR handles interrupts from the Real Time Clock (RTC). This external interrupt is fed to the system thru external interrupt 0. The action taken by this procedure is to pulse the heart-beat monitor and increment timer variables.

RX_ISR - This interrupt service routine processes interrupts resulting from the reception of a HSB frame. This external interrupt originates from the Zilog SCC and is issued to the processor through external interrupt 1. This service routine terminates both DMA 0 and the SCC receiver, which were both used for the received frame. It then clears the word HSB_STATUS so that the main loop processing will be aware of the reception. Data integrity is checked by verifying there were no CRC errors and that the expected number of bytes were received from the DEU. Lastly, several SCC receiver error indicators are reset and an end of interrupt issued.

TRANSMIT - This routine is called by HSB_SND and HSB_RCV, and is used to initiate a data transfer across the HSB. A transmission is initiated basically in a two part operation. First, DMA 1 is set up to transfer data from memory to the SCC via the HSB FIFO (First In First Out). Secondly, the SCC is set up to receive this data and transmit it across the HSB. This routine uses timer 0 as a watchdog for transmission hangups.

TRAP - This interrupt service routine handles unexpected interrupts. All unused interrupt vectors are initialized to send control here. This routine will increment a counter and reset all in-service bits on the internal interrupt controller.

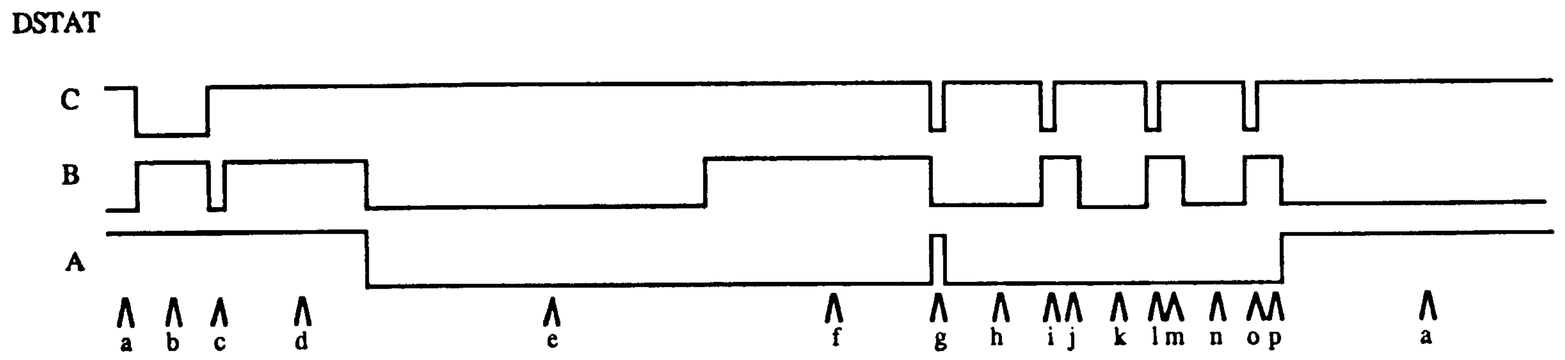
TX_DMA_ISR - This ISR processes interrupts resulting from the completion of a HSB transmission. This interrupt is generated by DMA 1 at the completion of frame transmission to the SCC. The purpose of this procedure is to acknowledge to the system that the transmission has completed.

TX_FAIL_ISR - This ISR responds to interrupts generated when the HSB transmitter does not complete its transfer within a reasonable ammount of time. The interrupt source is timer 0 which is set up at the start of a transmit. The action taken by this service routine is to terminate all DMA activity and flag an error.

WAIT_ISR - This interrupt service routine processes interrupts from internal timer 2. The vector for this ISR is shared with DR11_TRAP_ISR. This routine is used to implement a general_purpose timer function. Normally the main loop processing would set up timer 2 to interrupt the CPU after a given time interval. The main loop would also clear the word TIMER_FLAG. Upon a timer interrupt this routine would set TIMER_FLAG which in turn would be sensed by the main loop. Upon sensing this transition the main loop could then execute whatever action is necessary.

Section 3.2.3 DSTAT BITS

The following page shows an approximation of how a logic analyzer trace of the three DR11 DMA status bits (DSTAT) should appear for a typical 50 msec frame using a 100 micro-second sampling period. These three signals are available through an external connector on the BIU and are modified by the BIU software to signal the occurrence of certain events in the main processing loop.



DSTAT BITS FOR A 50 MSEC FRAME

-figure 3.1-

34

The following describes the states shown on the diagram of the previous page:

a) The BIU is ready to accept an IO function from the DR11 (expecting a read).
b) A DR11 read is in progress (data to MicroVAX).
c) The BIU is ready to accept an IO function from the DR11 (expecting a write).
d) A DR11 write is in progress (data from MicroVAX).
e) The BIU is now sending newly received data to the DP4s across the HSB.
f) The BIU will wait 6 msec before prompting the DEUs for data.
g) Timer interrupt for the 6 msec wait has occurred.
h) The BIU sends a request for data to DEU 1 and waits for a reply.
i) Receiver interrupt indicating data has been received from DEU 1. A code as in g (001) at this point implies no response from DEU 1.
j) The BIU is idle for a short time to make sure the HSB has stabilized.
k) The BIU sends a request for data to DEU 2 and waits for a reply.
l) Receiver interrupt indicating data has been received from DEU 2. A code as in g (001) at this point implies no response from DEU 2.
m) The BIU is idle for a short time to make sure the HSB has stabilized.
n) The BIU sends a request for data to DEU 3 and waits for a reply.
o) Receiver interrupt indicating data has been received from DEU 3. A code as in g (001) at this point implies no response from DEU 3.
p) The BIU is now executing end of frame clean-up code.

The following is a list of the eight possible DSTAT codes and their interpretation:

C	B	A	
0	0	0	Hardware reset state or SCP interrupt.
0	0	1	WAIT_ISR - timer interrupt.
0	1	0	RX_ISR - HSB receiver interrupt.
0	1	1	DR11 read function.
1	0	0	HSB_SND or HSB_RCV - HSB active.
1	0	1	BIU idle, ready for DR11 function.
1	1	0	BIU not ready for DR11 function.
1	1	1	DR11 write function.

Section 3.3 DP4 SOFTWARE

The DP4 system is the interface of each DEU to the BIU. The DP4 software system consists of the following modules containing the listed procedures:

```
DP4_RAM.ASM   - (memory declarations)
DP4_EQU.INC   - (include file with equated constants)
DP4_MAIN.ASM  - DP4_MAIN
DP4_INIT.ASM  - FIRST_PASS, DP4_INIT
DP4_ISR.ASM   - TX_FAIL_ISR, TX_DMA_ISR, RTC_ISR, RX_ISR,
                MONITOR, BZL_ISR, TRAP, WX_ISR
DP4_IO.ASM    - LOAD_IO_RAM
DP4_BZL.ASM   - DP4_BZL, BIT_SET
HSB_INIT.ASM  - HSB_INIT
TRANSMIT.ASM  - TRANSMIT
RECEIVE.ASM   - RECEIVE
WX_EXEC.ASM   - WX_EXEC
WX_INIT.ASM   - WX_INIT
WX_RAD.ASM    - WX_RAD
WX_FRAME.ASM  - WX_FRAME
WX_SINCO.ASM  - (none - SINE, COSINE lookup tables)
```

Section 3.3.1 DP4 SOFTWARE EXECUTION

The following is a general step by step overview of the DP4 software execution. At times throughout these steps the possibility exists that the processor may be servicing an interrupt request. These interrupts are thus listed briefly in this section. For a more detailed description of any interrupt service routine see Section 3.3.2.

1) Perform hardware generated initialization. This is done in the procedure FIRST_PASS. In this procedure variables are initialized which need initialization after a hardware reset only.

2) Perform non-hardware generated initialization. This is done in the procedure DP4_INIT. In this procedure variables are intialized which need initialization after both a hardware and software reset.

3) If application DPs haven't been allowed to run in 100 milliseconds then issue a go signal to all application DPs and go back to item 2 above. This allows the DPs to execute at a reduced rate even if the BIU or MicroVAX computer were to fail. In this event a status flag (DP4_STAT) is set invalid which causes the DPs to display their skeleton display pattern.

4) If DP4 has received a High-Speed Bus (HSB) frame from the BIU then continue below, otherwise go back to item 3 above.

5) If the received frame is a prompt from the BIU that it is ready to accept data then go to item 10 below and transmit, otherwise continue below and process received data. Received frames from the BIU should alternate between received data and prompts. If two of the same type frames are received consecutively, perhaps indicating a missed frame, an error counter is incremented (either UNEXP_IN or UNEXP_OUT).

6) Move data received from BIU to I/O RAM so that it may be referenced by the application DPs. Also move data placed in I/O RAM by the application DPs into DP4 local RAM so that it may be sent back to the BIU. This includes status information on whether the DP is up and running. The DP4, immediately before issuing a go signal to the DPs, will set all bits in three words (DP1_STAT, DP2_STAT, and DP3_STAT) in I/O RAM which will be cleared by the DP after its frame processing is complete. Whether or not these words were cleared by the start of the next 50 millisecond frame gives

information as to the status of each DP. This status information is sent back to the MicroVAX in the words STAT1, STAT2, and STAT3.

7) Issue a go signal to each DP so that it will start its processing of data. This is done through the words SYNC_1, SYNC_2, and SYNC_3 in I/O RAM.

8) Read bezel switches and set up timer so a new set can be read in 25 milliseconds. The bezel switches are read four at a time. The selection of which four is based on which row on the bezel matrix is being driven by the output discrete DSC_OUTPUT. Thus it takes four reads of DSC_INPUT to gather data on all sixteen bezel buttons. These reads are separated by 25 milliseconds to allow the matrix drivers to stabilize. The first read in the 50 millisecond frame is done at the start of a frame. The second is interrupt driven (handled by BZL_ISR) from the internal timer 2 and occurs 25 milliseconds after the start of a frame. The bezel pots are read at the same rate as the bezel switches (every 25 milliseconds).

9) Go to line 11 below.

10) Transmit data back to BIU on HSB. This data includes data returned from the DP, bezel information, and DP status. The HSB transmission is set up in the procedure TRANSMIT.

11) Set up HSB receiver to receive next frame from BIU. This is done in the procedure RECEIVE.

12) Call WX_EXEC weather radar executive.

13) Go to line 3 above.

The following interrupt service routines are used as real time events for the DP4:

BZL_ISR - This interrupt originates from internal timer 2. It is used such that the bezel switches will be read every 25 milliseconds.

MONITOR - This interrupt comes from the Sperry SCP. This ISR is used as an intermediary procedure between the interrupt and the execution of Sperry monitor code.

RTC_ISR - This interrupt is caused by the external real time clock.

RX_ISR - This interrupt is generated by the external SCC upon completion of a HSB reception.

TRAP - This ISR handles any interrupt not expected by the system.

TX_DMA_ISR - This interrupt is generated by the internal DMA channel upon completion of a HSB transmit.

TX_FAIL_ISR - This interrupt is used as a failsafe timer for the HSB transmitter.

WX_ISR - This interrupt signals the arrival of a weather radar radial into the weather radar card FIFO from the external weather radar unit (or DP23, if using the weather radar simulator).

Section 3.3.2 DP4 PROCEDURES

The following is an alphabetical list of the procedures which comprise the DP4 system and an explanation of their functions:

BIT_SET - This procedure is called by DP4_BZL and is used to convert bezel input data into a format that can be used by the host computer.

BZL_ISR - This interrupt service routine processes interrupts from internal timer 2. The main loop will set timer 2 to interrupt 25 milliseconds after the start of the DP4's 50 millisecond frame (at which time the bezels were read). At this mid-frame mark BZL_ISR will instigate another bezel switch reading by calling DP4_BZL.

DP4_BZL - This procedure converts the bezel switch input data into packed words for transmission back to the host computer via the BIU. This routine is called every 25 milliseconds, alternately from DP4_MAIN and BZL_ISR. During each call the bezel matrix driver word (DSC_OUTPUT) is modified so that on the next call a new set of four bezel buttons can be analyzed. The subroutine BIT_SET is called to analyze each new set of bezel inputs.

DP4_INIT - This procedure is called from DP4_MAIN and is responsible for the majority of DP4 initialization. The following initialization is performed:

a) Initialize all interrupt vectors with the address of the corresponding service routines.
b) Initialize 80186 configuration control registers. These include peripheral chip selects, memory chip selects, interrupt control, timer control, and DMA control registers.
c) Preset selected variables to initial values.
d) Determine which DEU this DP4 resides in from hardware discretes.
e) Initialize weather radar data areas.
f) Initialize and arm the HSB to receive data from the BIU.

DP4_MAIN - This procedure serves as the executive routine for the DP4 system. Processor control is transferred here after executing the PROM boot in high memory. Upon starting, DP4_MAIN invokes system initialization by calling FIRST_PASS and DP4_INIT. After returning DP4_MAIN settles into its main loop processing. This loop is synchronized by the reception of data from the BIU and thus shares its 50 millisecond period. While executing its main loop, DP4_MAIN calls the routines HSB_INIT, RECEIVE, LOAD_IO_RAM, DP4_BZL, WX_EXEC, and TRANSMIT.

PAGE 44 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

FIRST_PASS - This procedure is called from DP4_MAIN only after a hardware reset (re-boot). This procedure clears memory in the system variable space if called from a cold start (after power-up). It also resets selected error counters unconditionally.

HSB_INIT - This routine is called from DP4_MAIN and DP4_INIT and is used to initialize the Zilog SCC to operate using the HSB. This initialization procedure is done by writing to control registers mapped to a special area of the memory space (HSBSEG). The bit significance of these registers is covered in the Zilog Serial Communications Controller Handbook.

LOAD_IO_RAM - This procedure is called from DP4_MAIN and is used to move data received by the DP4 from the BIU to the I/O RAM so that it can be referenced by the application processors. This data is then used by the application processors to drive the display format loaded into the DU. This procedure also reads the potentiometer values coming from the front panel of the DUs. Lastly, this routine moves data originating from the application DPs into buffers for transmission back to the BIU for processing by the host computer.

MONITOR - This interrupt service routine processes interrupts originating from the Sperry SCP. The vector for this interrupt is initially set by the PROM boot code to 4000:80H. This would send control directly to the SCP upon an interrupt from the SCP. As a debug aid the need arose to construct an intermediary routine between the occurrence of the interrupt and control transfer to the SCP. Such an intermediary allows recording any data not normally available from the SCP; e.g., microprocessor I/O space. This would be done by inserting code into this routine which would store the data of interest into memory for examination later by the SCP. The method used for this is to overwrite the previously initialized interrupt vector with the address of this ISR. This is done in DP4_INIT and causes the interrupt to transfer control to MONITOR. At this point any code needed to record data for debug purposes may be inserted. Control is then transferred to the SCP by pushing the SCP address (4000:80H) onto the stack and executing a FAR RETURN. When the SCP relinquishes control, via an IRET, processing returns to main loop processing at the point of interruption.

RECEIVE - This routine is called by DP4_MAIN and DP4_INIT. It is used to set up and arm the SCC for reception of data across the HSB. This reception mode is entered via a two step process. First, the DMA 0 is initialized to accept data from the SCC and place it into memory. Secondly, the SCC is set up to receive data from the HSB and send it to DMA 0. Upon receiving a frame of data, the SCC will interrupt the processor through the service routine RX_ISR.

RTC_ISR - This ISR handles interrupts from the Real Time Clock (RTC). This external interrupt is fed to the system through external interrupt 0. The action taken by this procedure is to pulse the heart-beat monitor and increment timer variables.

RX_ISR - This interrupt service routine processes interrupts resulting from the reception of a HSB frame. This external interrupt originates from the Zilog SCC and is issued to the processor through external interrupt 1. This service routine terminates both DMA 0 and the SCC receiver, which were both used for the received frame. It then clears the word HSB_STATUS so that the main loop processing will be aware of the reception. Data integrity is checked by verifying that there were no CRC errors and that the expected number of bytes were received from the BIU (count check done in DP$_MAIN). Lastly, several SCC receiver error indicators are reset and an end of interrupt issued.

TRANSMIT - This routine is called by DP4_MAIN. It is used to initiate a data transfer across the HSB. A transmission is initiated basically in a two part operation. First, DMA 1 is set up to transfer data from memory to the SCC via the HSB FIFO. Secondly, the SCC is set up to receive this data and transmit it across the HSB. This routine uses timer 0 as a watchdog for transmission hangups.

TRAP - This interrupt service routine handles unexpected interrupts. All unused interrupt vectors are initialized to send control here. This routine will increment a counter and reset all in-service bits on the internal interrupt controller.

TX_DMA_ISR - This ISR processes interrupts resulting from the completion of a HSB transmission. This interrupt is generated by DMA 1 at the completion of frame transmission to the SCC. The purpose of this procedure is to acknowledge to the system that the transmission has completed.

TX_FAIL_ISR - This ISR responds to interrupts generated when the HSB transmitter does not complete its transfer within a reasonable amount of time. The interrupt source is timer 0 which is set up at the start of a transmit. The action taken by this service routine is to terminate all DMA activity and flag an error.

WX_EXEC - This routine is called by DP4_MAIN. It is responsible for enabling weather radar radial interrupts if weather radar has been selected on the bezel panel. It will disable weather radar if there has been no radial input for over a second. It also tests if frame processing is busy. If not, it will call WX_FRAME.

WX_FRAME - This routine is called by WX_EXEC. It will move the local frame output block to the weather radar hardware and initiate frame processing.

WX_INIT - This routine is called by DP4_INIT. It is responsible for the initial setup of the six radials used for output to the weather radar hardware. It also initializes the frame output block with initial values.

WX_ISR - This ISR handles weather radar radial input interrupts. These radials originate from the actual weather radar unit (or the weather radar simulator in DP23). This ISR will call WX_RAD.

WX_RAD - This routine is called by WX_ISR upon reception of an incoming set of weather radar radials. Input data is formatted into a form usable by the weather radar card hardware, and upon reception of the next set of input data, is output to the hardware (mapped through RADSEG).

Section 3.3.3 WEATHER RADAR SYTEM OVERVIEW

The path of data from the weather radar device to DU output begins with the reception of an incoming radar radial transmitted from the weather radar or weather radar simulator (DP23). This radial is placed into the weather radar card FIFO. With placement commands from the DP4, the FIFO data is converted and placed into the weather radar card memory. This memory is arranged in a cartesian format representing pixel positions on the display screen. Each radial is stored as a line of data originating from a start pixel position and extending at a specified angle from that position. Each input radial is actually stored as four lines starting at the angle specified for the first, and then with a slightly incremented angle for each of the remaining. This has the effect of smoothing the display and assuring that all pixels are filled. In addition to these four, two erase radials are also generated. These will remove old display information from the previous radar sweep. The weather radar card memory is then sent to the DUs by the weather radar card hardware in the form of a single frame (frame processing). This frame contains all the information needed to display a complete screen.

Section 3.4 WEATHER RADAR DATA GENERATION (DP23)

The weather radar simulator resides in DP23. It is used to send simulated weather radar radials to the weather radar display system in the absence of a genuine weather radar device. Theoretically, if the display system can display the synthesized data output by DP23, it could also display real radar information from a real radar device. The DP23 system will output six different rainbow patterns sequentially to the display system. The simulator consists of the following software modules containing the listed procedures:

1.) RAD_SIM.ASM - RAD_SIM
2.) SIM_INIT.ASM - SIM_INIT
3.) GET_SUM.ASM - GET_SUM
4.) SIM_ISR.ASM - TX_FAIL_ISR, TX_DMA_ISR, RTC_ISR, MONITOR, TRAP
5.) SIM_RAM.ASM - None, data declaration.
6.) HSB_INIT.ASM - HSB_INIT
7.) FIRST.ASM - None, pointer to beginning of code for checksum.
8.) LAST.ASM - None, pointer to end of code for checksum.

These routines are described below:

1) GET_SUM - This routine is called from SIM_INIT to calculate a checksum of bytes between the first and last byte in the code space. This is used for configuration control purposes.

2) HSB_INIT - This routine is called from RAD_SIM to initialize the high-speed bus hardware for transmission of radar radials to the weather radar cards in the DEUs.

3) MONITOR - This ISR is used to intercept SCP interrupts for debugging purposes. It is identical to that used in the BIU and DP4. See Section 3.3.2 for a more detailed description.

4) RAD_SIM - This routine serves as the main loop scheduler for the weather radar simulator system. Before falling into this main loop, it initializes the stack and calls SIM_INIT. The main loop will copy one of six types of radials into an output buffer for transmission to the weather radar cards in each DEU. Along with this radial will be its scan angle. This angle is incremented for each radial output in order to provide an artificial radar sweep. Once a complete sweep is sent, a new radial type is used for the next sweep.

5) RTC_ISR - This interrupt service routine responds to interrupts from the real-time clock. It will read the input discrete word, pulse the heart-beat monitor, and increment the real-time clock count word.

6) SIM_INIT - This routine is called from RAD_SIM upon restart to initialize the weather radar simulator system. This includes loading interrupt vectors, finding the code checksum by calling GET_SUM, and calling HSB_INIT to initialize the High-Speed Bus.

7) TRAP - This interrupt service routine is included to service any interrupts that were not expected by the system. It will increment an error counter.

8) TX_DMA_ISR - This interrupt service routine is activated when the DMA controller #1 completes. The service routine will disable a timeout counter and reset the Tx busy bit of the serial controller.

9) TX_FAIL_ISR - This interrupt service routine responds to the timer interrupt indicating a DMA did not complete in the expected amount of time. It will reset both DMA controllers and increment an error counter.

Section 4.0 EXECUTIVE SOFTWARE

The software described in this section resides in each processor within a DEU that has an applications format loaded into its memory. The Executive consists of several PLM/86 and ASM/86 modules which are built into each format during the LINK process. The following is a brief outline of the functions performed by the executive software:

- Initialize segment registers and stack pointer upon cold start of the processor.
- Initialize data and high-speed bus RAM memory segments.
- Initialize the interrupt vector addresses and set up the registers in the interrupt controller.
- Identify the particular DEU and DP that the software is loaded into. This in turn is used to set up variables that are dependent on DEU/DP identification.
- Process interrupts and pulse the heart-beat monitor.
- Compute EEPROM checksums for verification of format integrity.
- Detect the start of 50 millisecond frames and record diagnostic information associated with each frame.
- Control how often and in what order format subroutines are to be called.
- Interface to the I/O processor (DP4) within the DEU through I/O RAM memory.
- Control the transmission of graphics data to the DU.
- Initiate the software format switching between primary and secondary display formats.

Section 4.1 HARDWARE RESET TIME & THE MODULE "RESET"

The hardware reset can be initiated by power up, System Control Panel (SCP) intervention, or heart-beat monitor failure. After the reset has occurred the processor starts executing instructions in UVPROM memory at address FFFF0H. This set of instructions cannot be changed except by chip replacement in the DEU hardware. The following are the "boot code" instructions:

```
ADDRESS          INSTRUCTION

FFFCA            CLI
                 MOV      AX,0000H
                 MOV      SS,AX
                 MOV      SP,3FFEH
                 MOV      DX,FF3CH
                 MOV      AX,0010H
                 OUT      DX,AX
                 MOV      AX,0080H
                 MOV      BX,0038H
                 MOV      [BX],AX
                 MOV      AX,4000H
                 MOV      BX,003AH
                 MOV      [BX],AX
                 STI
                 JMP      F000H:8000H
FFFF0            MOV      DX,FFA0H
                 MOV      AX,FFFBH
                 OUT      DX,AX
                 JMP      F000H:FFCAH
```

The boot code causes some processor control registers to be set up for processor configuration, the definition of a stack area, and the placement of the Control Panel Interface (CPI) monitor routine address into position #14 of the interrupt vector. The last operation is to jump to the first instruction of the loaded format at F8000H.

The ASM/86 routine RESET, found on the file RESET.ASM, is "LOCated" at address F8000H or FB800H depending on the particular format's role as primary or secondary display (Section 4.6). The first thing it does is disable interrupts. It then clears the data RAM used by the applications code of the format (00400H - 03FFFH). The RAM buffers reserved for use by the high-speed bus are also cleared (10000H - 13FFFH). The two variables that are set to system discrete words are cleared to assure they do not contain previous values (before power up). This is essential since these values will later be checked to verify that the DEU has stabilized after power up (Section 4.2). The addresses of the interrupt routines are set into the interrupt vector memory and the registers of the interrupt controller are set to configure the system. Information about the interrupt controller can be found in Section 5.10 of "iAPX 86/88, 186/188 User's Manual - Programmer's Reference". The final step performed by this module is to enable interrupts and call the PLM/86 format executive routine which performs format specific executive operations (Section 4.2). Whenever a PLM executive returns to the RESET module a format switch has been requested. RESET determines first which type load was made for its display format (primary or secondary). If the current format requesting a switch is a secondary format then an immediate transfer to the primary format at F8000 is performed. However if the initiating format is the primary one, testing is performed to ensure the existence of a secondary format in memory since double display format usage is not always used. The test consists of the RESET module comparing the memory locations starting at FB800 to those at F8000 (itself). When identical, a valid RESET module also exists in the secondary format position since all display formats have the identical RESET module.

Section 4.2 THE FORMAT EXECUTIVES

Each format has its own PLM/86 executive module. These modules perform the same functions, but contain instructions that are relevant only to a particular format. There are four steps that each executive performs, all in the same order.

Step one is the initialization of the high-speed bus and some executive variables. The CGS procedure HSB_INIT is called for the bus initialization process.

The DEU number (1 - 3) and the processor number within the DEU (1 - 4) is ascertained in step two. The processor identification is done for two reasons. The first reason is to verify that the DEU has stabilized after a power up. The identification information is obtained from hardware discrete words which do not contain valid data until the DEU has stabilized. The second reason for determining the ID values is to set variables and address pointers that vary depending on either DEU or DP number.

Step three is the calculation of the EEPROM checksum values. Checksums are computed to verify the validity or version of a particular format. Section 4.4 gives more details involved in the checksum process.

The scheduling and computations involved with each 50 millisecond frame is performed as the fourth step. Every 50 milliseconds the DEU I/O processor (DP4) flags the start of a new frame by using a word in shared I/O memory. When this occurs each format executive module performs calls to core graphics and applications subroutines to create the data used to drive the display units. The functions performed in each frame can be classified as follows:

- Operations which need to be done only once, immediately after a processor reset (see Section 4.1). Initializing core graphics data structures is included here.

- Operations performed every frame. The data enqueued for transmission during the previous frame is sent to the DU at the start of each frame. Then the latest I/O data from the host computer is gotten (see Section 4.5). If the data from the host was obtained properly the format is considered valid (FMT_VLD) and calls to subroutines for rapidly changing graphics objects are made. Note that the amount of idle time at the end of each frame and the number of frame overflows are computed for each frame also.

. Operations performed one out of N frames, where N is one of the numbers; 2, 4, 8, 16 or 32. Computations of data for objects that change slowly or in discrete increments are done here. Also the enqueuing of static graphic objects for transmission is performed at the slowest rate used by the format. Note that the functions performed at faster rates are inhibited until the slowest rate loop has been completed once.

The MAP_EXC module contains some unique instructions that initiate the processing of the map background data buffer. To attain synchronization of the background transform and graphics objects, updates of these objects must be performed in several steps using ENABLE/DISABLE objects. See the CGS Programmer's Guide for information on these special objects.

Section 4.3 INTERRUPTS (INT_40HZ INT_NMI INT_NULL INT_0 P_RTC)

The 80186 microprocessor supports 256 interrupts. Of these only three have unique interrupt routines defined in this executive. One extra is defined by the SCP monitor routine and the rest are all considered null interrupts.

The first 1024 bytes of memory, addresses 00000H through 003FF, are reserved for the interrupt vector entries. Four bytes, two for segment and two for offset, are used for each of the 256 interrupts. The following table shows which routines service the various interrupts.

Number	Routine	Function
0	INT_0	Divide instruction error. No action besides end-of-interrupt has been defined.
2	INT_NMI	Non-maskable interrupt (power failure). No action besides end-of-interrupt has been defined.
12	INT_40HZ	Real-time clock. Every 25 milliseconds this interrupt occurs. The subroutine P_RTC is called and the end-of-interrupt is issued.
14		The CPI monitor of the SCP. All SCP functions on a processor are performed through this interrupt.
REST	INT_NULL	Since these should not occur the action is zero the interrupt "In Service" register.

The subroutine P_RTC is called to perform 3 operations. First the system interrupt input discrete is read for storage in local RAM. The processor heart-beat monitor is then pulsed to keep the hardware from initiating an automatic reset (when no pulse detected within 1 second). The last function is the acknowledgement of the real-time clock interrupt.

All these modules are built into the shared memory regions (Section 4.6.1).

Section 4.4 CHECKSUMMING EEPROM SEGMENTS (FIRST_FILE.ASM & LAST_FILE.ASM)

The purpose of performing checksums is the validation of EEPROM memory. If EEPROM memory has been modified, due to memory failure, loading a different version, or modification through the SCP, the checksum values will change. The checksum process is actually performed by format executive modules described in Section 4.2.

There are two contiguous regions of EEPROM memory in each processor within the DEU. Their address ranges are; 0C000H through 0FFFFH and F8000H through FFC00H. The first range contains the Segment CONST. The second range contains the CODE Segment. See the Intel LINK and LOC manuals for information about segmentation. Each segment has its own checksum value computed.

Since each format uses different amounts of memory for the two segments, the beginning and ending addresses of each segment must be known. This is achieved with the files FIRST_FILE.ASM and LAST_FILE.ASM. These files must be the first and last files in the list of object files used by the LINK86 program. Since LINK86 adds to segments in the order that it receives additions from object files, the address associated with the first and last files will bound the segments. The two aforementioned files contain segment specifications for both segments residing in EEPROM memory. Note the labels defined in LAST_FILE will have the address of the word following the segment end. The labels are used as parameters to the utility function GET_SUM which is described in Section 5.1.

As a segment is being built by the LINK86 software, each new entry obtained from an object file will begin on an even address (word alignment). Refer to the "align-type" parameter of the segment directive in chapter two of the ASM86 manual. The word alignment may cause undefined gaps of one byte within a segment. These undefined bytes cause the computation of an invalid checksum. Therefore a process was developed to modify the load file of the formats to fill in the gaps with zero bytes. This process is described in appendix A.

Section 4.5 DATA FROM/TO THE HOST COMPUTER (INPUT F3RINPUT NEW_DATA)

Each DEU receives the entire 704 word host computer buffer via the BIU and each processor within the DEU has access to all the data. Also 96 words of data, 32 from each applications processor, is sent from each DEU to the host computer via the BIU. See Section 3.0 "The I/O System" for detailed information.

The job of the executive software is to detect when there is valid input data, set local memory variables that are dependent on input data, and fill in the output buffer with display format status.

Each format has a subroutine INPUT which performs very similar functions. The items associated with each format's INPUT will be described first, followed by an explanation of differences. Note that INPUT is called every 50 milli-seconds by the format executive modules (Section 4.2).

Three conditions must be met for a format to consider the data available to it current or valid.

. The DP4 status flag must be good (FFFFH).

. The DP4/DPn toggle word must be in the proper state. This word is set to FFFFH by DP4 after each reception of data from the BIU and set to the format ID (1-6) by the DP processor after the current frame completes.

. The host computer frame count, sent within the 704 word buffer, must be one greater than it was in the previous iteration.

If these conditions are met the INPUT routine proceeds to process the data in I/O RAM. Otherwise the format is marked as invalid.

The PFD and TOP formats, when I/O RAM data is valid, unpacks the bits of discrete words and stores off the boolean results as the byte values TRUE or FALSE (FFH or 00H). All other variables used by this format are accessed directly from I/O RAM. A copy of the data in local RAM is not needed with the synchronized system (see Section 3.1).

The NAV format, in addition to unpacking discretes, calls the subroutine NEW_DATA to set up other NAV variables computed from the input data and determine if the format should be invalidated due to discrepancies found in related input variables. Note also that the format is invalidated if a new NAV "background buffer" has not been received in 15 seconds.

PAGE 62 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

Special processing of the NAV background buffer must be performed by the INPUT subroutine associated with the NAV format. The background buffer consists of a 512 word area of I/O RAM containing path and waypoint type data from the MicroVAX computer. Appendix C contains the VAX I/O buffer layout. Since map data is not updated every 50 millisecond frame and background data for all NAV formats loaded in the system may differ, the NAV format must be able to identify when the background buffer contains data designated for it. This is done through the Map Control word in the I/O data buffer. Bits 0 - 3 of the word contain the ID of the DP that the background buffer is designated for. Bit 4 is set when the buffer is part two of a 2-stage background update. The upper byte contains a counter that changes with each new map background. The counter is not incremented for the second part of a double length background. When INPUT finds background data designated for it, a local copy is made so the buffer can be filled for another NAV format or with part two of the same buffer.

The ENG format calls F3RINPUT when the format has been declared valid. This routine also unpacks discrete words and checks the Engine parameters fetched from I/O RAM to verify they are in the proper ranges for display. If not, the corresponding valid bytes are set to FALSE, even if the valid was unpacked as TRUE.

Section 4.6 SOFTWARE FORMAT SWITCHING

One desirable feature for the display system is having more than one display format appear on the same DU. This can be done with the hardware switches that control which high-speed bus connects to which display units. In effect the new format is a repeat copy of another format in the system. The disadvantages of the hardware technique are the required manual intervention by someone with access to the bus switches, the limit to formats that already are being shown somewhere in the display system, and the loss of bezel button usage since the switched format is just a repeat copy.

Software format switching is a technique where two separate display formats are loaded into the same DP memory. The decision as to which format will be active is made by software executing on either the display microprocessor or the display host computer.

Section 4.6.1 SHARED MEMORY SECTIONS

There is a large amount of common software which is used by all display formats. This includes the core graphics library routines and application utility procedures. The software format switching technique conserves memory by sharing common subroutines and data between both formats loaded into a single display processor. One object module is linked containing all the common software (SHARE.LNK). This object module in turn is linked into all the display formats. SHARE.LNK contains the definition of three unique memory segments shown below.

Segment name	usage	load address
@DAT	common data variables	03FA0
@CONS	common data constants	0FEC0
@CGS	common code	FDD00

Since the segments are loaded at the same addresses for each format, each time a format is loaded into a DP the identical binary data overlays itself. Refer to Appendix B for memory layout diagrams.

Much of the core graphics software has PLM as source code. The PLM language provides no ability to create unique segment names. Since application software also uses PLM, a way of renaming PLM memory segments is necessary to allow separate location of shared and non-shared memory. The object file "SHARE.LNK" is patched to change the names of the default PLM segments (DATA->@DAT CONST->@CONS CODE-> @CGS).

PRECEDING PAGE BLANK NOT FILMED

PAGE 66 INTENTIONALLY BLANK

Section 4.6.2 PRIMARY AND SECONDARY FORMATS

The two formats loaded into a single display processor are known as the primary and secondary formats. The format whose CODE segment is loaded at F8000 is the primary format. Each display processor must have a valid format loaded at this position. The primary format's CONST segment is loaded at 0C000. Optionally a secondary format, for the purpose of software format switching, is loaded into DP memory with its CODE and CONST segments at FB800 and 0E000 respectively. CODE and CONST are the only unique segments for a format so all other defined memory segments overlay each other. Each format has its own data variable layout, but the same DATA segment is used since all data variables for a format may be overwritten while the other format is running.

Display programs are created as primary formats which can be loaded and executed by themselves. A simple conversion process exists to convert from a primary to a secondary format. First the addresses assigned to the CONST and CODE segments must be changed in the LOC.CSD file which exists for each format. Then both formats must have a logical expression added to the "infinite" DO WHILE TRUE loop which exists in the PLM executive module (see Section 4.2). The added logic can be a test of data from the host computer or the state of a bezel button for example.

Section 4.6.3 TRANSFER BETWEEN FORMATS

When the display processor hardware is reset, execution of the primary format's RESET module automatically begins (see Section 4.1). A call to the format's individual PLM executive routine (see Section 4.2) is made and under normal circumstances a return to the RESET module never occurs because of the "infinite" DO WHILE TRUE loop. When the infinite loop is broken by a "switch format" request the PLM execute returns and RESET transfers control to the secondary format at FB800. Similarly the secondary format branches to the primary format at F8000.

Since the identical reset module is used in primary and secondary formats a format must determine at run time which type of format it is. This is performed by comparing the start label "RESET" with the fixed start addresses defined for primary and secondary formats. Once the format type has been determined the RESET module will be able to make the correct jump when a format switch is commanded.

Section 5.0 UTILITIES

The following pages of this section describe eight ASM86 utility routines used by modules in all formats to perform specific functions. Each utility description gives an example of how it is used. The ASM86 source code for all the utility routines is on the file UTILITY.ASM which is built into the shared memory sections (Section 4.6.1).

MODULE NAME: GET_SUM

PURPOSE: To compute the sum of all words in a contiguous memory region for the validation of EEPROM memory.

REMARKS: The checksum of a region is computed by the following function call:

```
CHECKSUM = GET_SUM (@START_LABEL, @END_LABEL, SEGMENT_FLAG);
```

The first two parameters are the variable names or code labels that delimit the segment to be checksummed. Note that END_LABEL must be the address of the word following the last word to be included in the sum. The SEGMENT_FLAG parameter is used to define which segment register, CS or DS, is the base for the offsets associated with START_LABEL and END_LABEL. When a checksum is performed on the EEPROM used for constant data, zero is used for SEGMENT_FLAG. A one is used otherwise.

MODULE NAME: U_BNRBCD

PURPOSE: To convert numeric values to their 8 bit BCD representations for display as numeric text on DU screen.

REMARKS: When text information is placed on the display, each byte of the character string data is mapped to the character PROM in the DU through core graphics translation tables. The bytes containing numeric text do not use the ASCII format for characters as do the letters A - Z. Each byte contains the actual binary number that the ASCII code would represent. For example the string used for placing the text "254" on the screen is the following:

```
    03H, 02H, 05H, 04H  (first byte is count)
```

This differs from the standard ASCII string which would be

```
    03H, 32H, 35H, 34H  (first byte is count)
```

An integer value is converted to this format through the call to the function as follows:

```
  CALL U_BNRBCD (NUMBER, DIGIT_COUNT, @STRING, MIN_COUNT);

    NUMBER       integer value
    DIGIT_COUNT  number of character positions available
    STRING       byte array
    MIN_COUNT    minimum number of non-blank digits to show
```

Note the character count byte is filled in by this function. The parameters (9, 3, @STRING, 2) produce the following in the byte array STRING:

```
        03H, 20H, 00H, 09H -> (3, ' ', 0, 9)
```

These bytes can then be sent to the core graphics text mapping and display routines such as IPVGSUB_CSTRING.

The first parameter may be interpreted in two ways. The normal way, as described above, is as a 16-bit signed integer having a value from -32786 to 32767. When the desired interpretation of the input value is a 16-bit unsigned number between 0 and 65535 the utility is notified through special usage of the minimum digit count (fourth parameter). The low bit of the upper byte in the word is set to designate the alternate interpretation. Therefore adding 256 to the minimum digit count enables this usage.

MODULE NAME: U_COSINE

PURPOSE: To compute the cosine function of an angle.

REMARKS: The angular value used as input to this function must be scaled in Standard Angle Format (SAF) format. This means the full angular range is represented as +1 to -1 with a scale factor of 2 ** 15 to maintain fractional precision. The angle 27.419 degrees is represented by the integer value 4991. This number was obtained by dividing 27.419 by 180 and multiplying by 2 ** 15.

The result of the function is the cosine of the input angle scaled with 14 bits of fraction (COS(N) * 2 ** 14). This scale factor gives ample precision for further calculations with the result.

To find the cosine of 27.419 degrees, the call

```
        COS_VAL = U_COSINE (4991);
```

would produce the value 14543 as COS_VAL. This result can be divided by 16384 (2 ** 14) to get the result .8876627 which is the cosine of 27.419.

MODULE NAME: U_DIV

PURPOSE: To perform double precision division on scaled integer values.

REMARKS: This function is the counterpart of the U_MULT function (see U_MULT description). It is used for division of scaled integer numbers. The call is made as follows:

```
X = U_DIV (A, S, B);
```

The value of X after this function call is

```
(A * 2 ** (16 - S)) / B
```

The following example demonstrates the operation 32.53125 divided by 4. In this example the dividend will be scaled for 5 bits of fraction (N * 2 ** 5) and the result will be scaled with 10 bits of fraction (N * 2 ** 10). The operation is performed as follows:

```
SCALED_RESULT = U_DIV (1041, 11, 4);
```

After the function call SCALED_RESULT is equal to 8328 which is 8.132815 * 2 ** 10 (32.53125 / 4 = 8.132815).

MODULE NAME: U_INSERT_DEC_PT

PURPOSE: To insert a decimal point into a character string containing numeric characters.

REMARKS: When an integer value has been converted to its ASCII character string representation, a decimal character can be inserted into the string of bytes with this function. As an example, the following illustrates placement of an ASCII '.' in the character string already containing the characters '2992'. Note the first byte of a character string contains the string length.

```
DECLARE STRING(6) BYTE INITIAL(4,'2992',0);

CALL U_INSERT_DEC_PT (4, 2, @STRING);
```

The first parameter is the current character count, the next is the number of characters to be placed to the right of the decimal point. The last parameter is a pointer to the character string. After the call the six bytes of STRING will contain the following:

```
5, '2', '9', '.', '9', '2'
```

The first byte, string length, is automatically incremented by the function. See the description of the U_BNRBCD utility for information about formatting integer data in ASCII character representation.

MODULE NAME: U_MULT

PURPOSE: To perform double precision multiplication on scaled integer values.

REMARKS: Since there is no floating point capability in the processors used by the displays, scaled arithmetic must be used. For example the number 17.125 can be represented by the integer value 137 (17.125 * 2**3) with no loss of fractional precision.

When multiplication of scaled numbers, which will overflow a 16-bit integer, needs to occur this function is used. It performs the multiplication in a 32-bit register and uses a shift factor to keep the desired 16 bits of the product. The call is made as follows:

```
    X = U_MULT (A, B, S);    /* S = shift factor */
```

The integer X is set to A * B * (2 **(S - 16)). For example if a display symbol needs to be moved 17.125 screen units for each unit of an input value, the following call would be made:

```
   SCREEN_UNITS = U_MULT(137, IN_VALUE, 13);
```

If IN_VALUE was equal to 520, SCREEN_UNITS would be set to 8905 units (17.125 * 520 = 137 * 520 * (2 ** -3)).

MODULE NAME: U_SINE

PURPOSE: To compute the sine function of an angle.

REMARKS: Since the description of the sine function is totally analogous to the cosine function, please refer to the description for the module U_COSINE.

MODULE NAME: U_SQRT

PURPOSE: To compute the square root of a number.

REMARKS: This function performs the square root of an integer and allows for the result to be scaled to retain fractional bits of precision. The function call has the form:

```
X = U_SQRT (N, S);
```

After this function call X has the value SQRT(N) * 2 ** S. For example the square root of 453 can be computed by the following call:

```
SCALED_RESULT = U_SQRT (453, 10);
```

The result of this operation, the scaled value 21794, has ten fractional bits of accuracy.

[21794 / (2 ** 10) = 21.2832; SQRT(453) = 21.2834]

See the descriptions of the U_MULT and U_DIV utilities for more information on scaled arithmetic.

Section 6.0 THE NAVIGATION DISPLAY FORMAT

The navigation display shows the position of the airplane relative to ground positions and terrain features. The diagrams at the beginning of Section 6.1 depict typical NAV display configurations for the two available map background orientations (Map and Plan modes). The map orientation is a selectable feature controlled by a bezel button. Note that Map mode includes some symbology not found in Plan mode. A description of NAV format symbols is included in the following sections.

The airplane chevron in Map mode is fixed 1.25 inches below the screen center and the symbology is displayed in a "track-up" orientation. This means that the airplane's current track is always the center of the compass arc at the top of the display.

In Plan mode some other reference point, typically a waypoint, is used for the fixed position 1.25 inches below screen center. All other map background symbology is displayed in a "north-up" orientation. In this orientation the vertical axis of the display screen represents "true" north. The airplane chevron symbol moves about the display in this mode.

The NAV format screen is divided into three distinct areas. The major one is the airplane and map background features area. The second is the flight information area at the top of the display above the compass arc. The vertical deviation scale area is the third. The three areas exist in both Map and Plan modes. The compass arc, which forms a partition between background and flight information areas in Map mode, is not shown in Plan mode however.

The flight information area contains wind speed and direction information in the upper left corner in Map mode. Just a north pointer is displayed there in Plan mode. The right hand side of this area has four lines of flight information text. The following describes the information present on each line.

#1 'TO' waypoint information. Includes destination waypoint name and distance from present airplane position. Also the current Greenwich Mean Time is shown at the end of the line.

#2 Bezel selection indicators. Three letter mnemonics appear when bezel buttons have been used to select the map options; airports, navaids, time box, or altitude range arc.

#3 Like line #2, mnemonics are shown when terrain features, ground reference points, or the boundaries of restricted regions have been selected.

#4 Guidance and navigation modes. Aircraft modes include 2D/3D/4D guidance, track select, altitude hold, flight path angle select, and air speed select. The aircraft navigation modes are shown at the end of the line.

The aircraft and map background area shows the aircraft's present position along with selected reference points. The orientation of the symbology depends on which of the two modes is selected from the bezel panel. Flight plans are displayed as a series of straight and curved line segments with four point STAR symbols designating specific waypoints. The aircraft chevron has the current ground speed and altitude appended to the bottom when in Map mode. A list of the symbols that can appear in the Map background area follows in the next sections. The symbology appearing in this area is masked from the other two NAV format areas. However the masking is only performed for the vertical deviation area when the vertical deviation scale is present on the display.

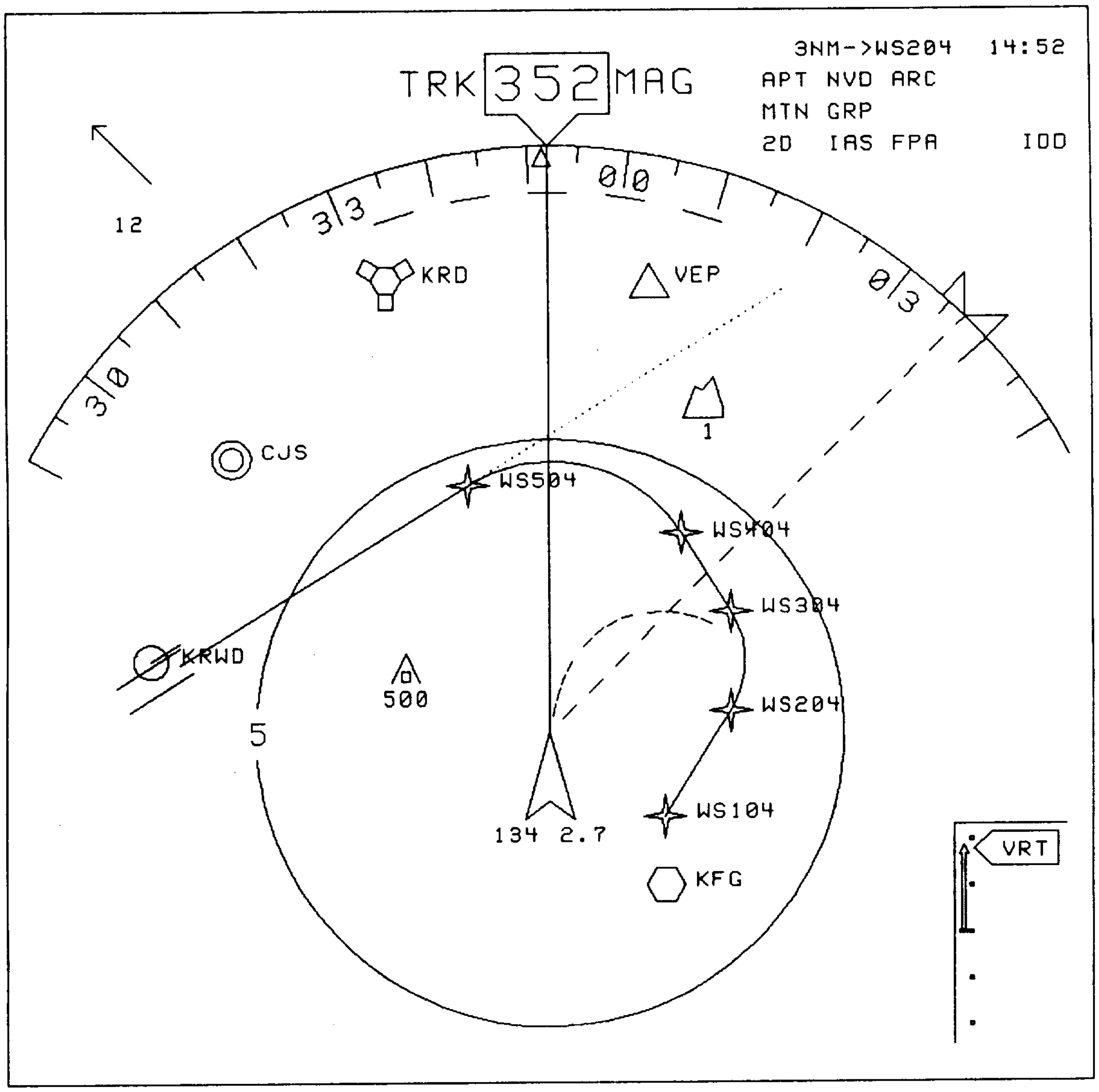
The altitude profile of the aircraft is shown via the vertical deviation scale. This scale appears in the lower right hand corner of the display screen when appropriate conditions arise. A rectangular mask around the scale keeps background features from interfering when the vertical deviation scale is present.

The NAV format uses twelve of the sixteen buttons provided on the bezel panel of a DU. Neither of the two potentiometer dials are used. The following table describes the function of each NAV format bezel button. The naming conventions are: L - left hand set, R - right hand set, (1 through 8) - button number with one being the top.

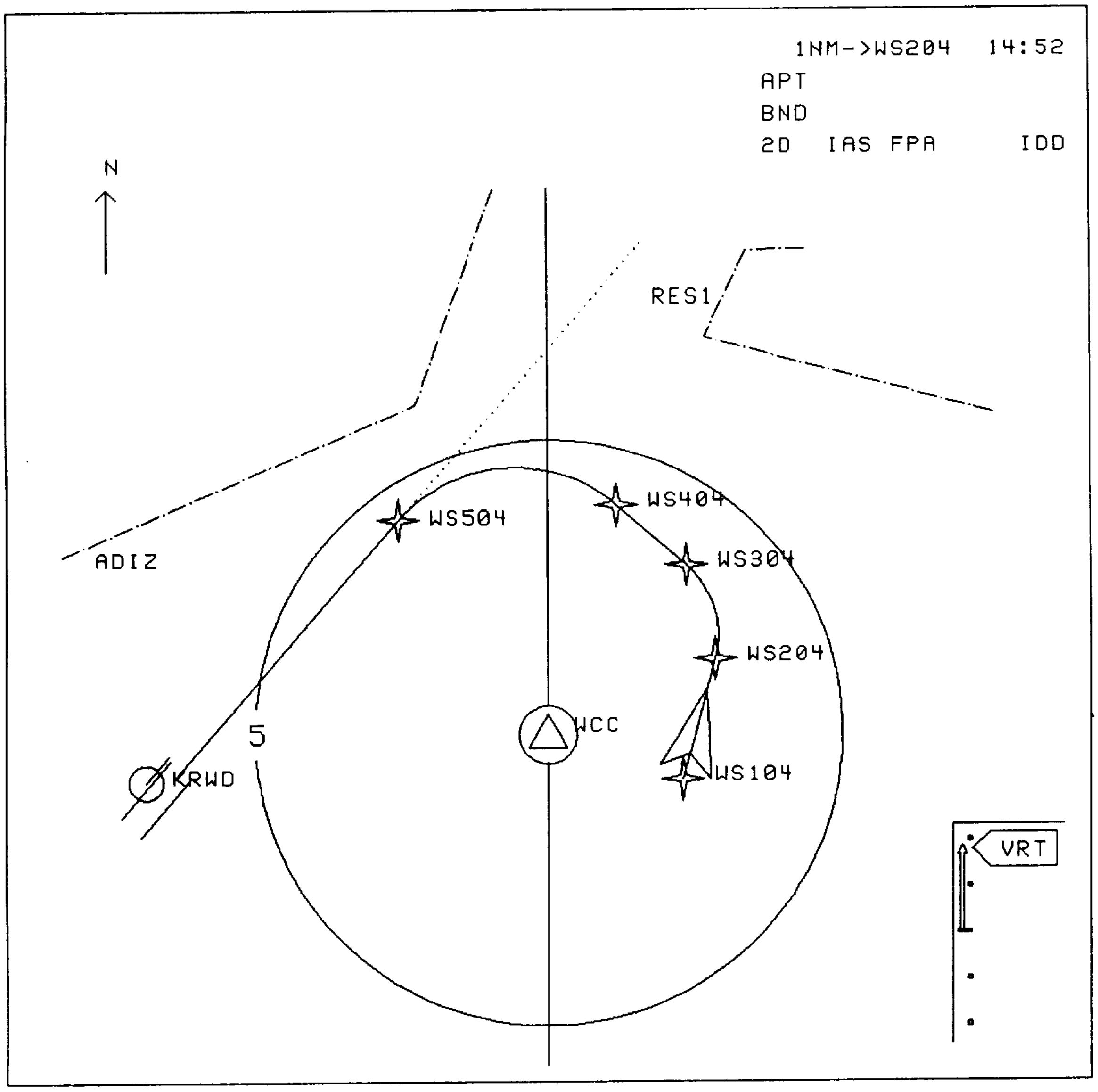
L1	not used
L2	not used
L3	not used
L4	Terrain features option
L5	GRPs option
L6	not used
L7	ADIZ boundaries option
L8	Map/Plan mode toggle
R1	MLS select/deselect
R2	Airports option
R3	Navigational aids option
R4	Time box option
R5	Altitude range arc option
R6	Path waypoint information option
R7	Zoom out (scale change)
R8	Zoom in (scale change)

Section 6.1 INDIVIDUAL NAV SYMBOLS

This section has drawings of the NAV format followed by brief descriptions of individual symbology. The first two drawings represent typical NAV formats for both Map and Plan mode. The remaining drawings break down the format into individual sections of the display with numbered items for identification.

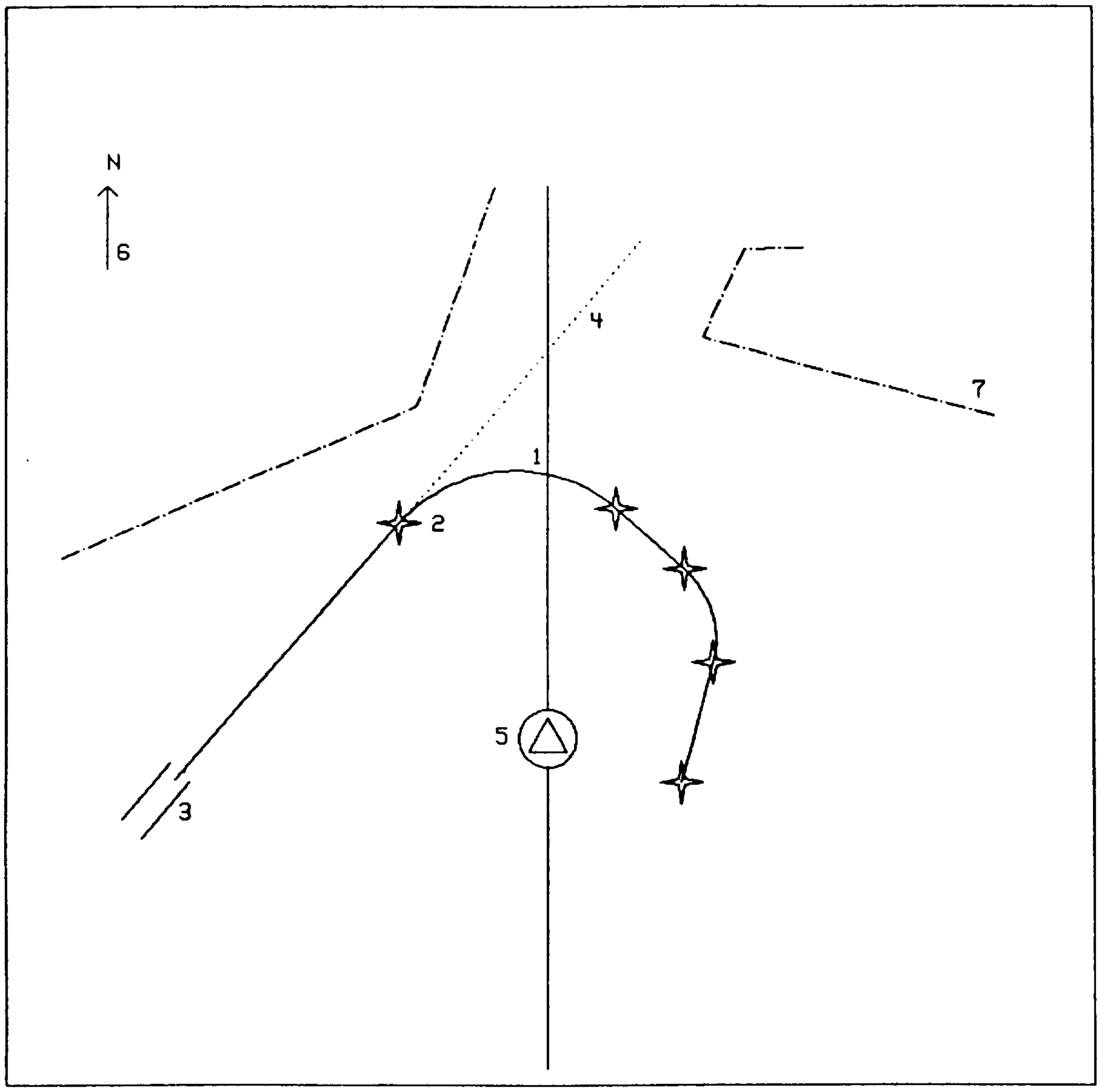


NAVIGATION DISPLAY FORMAT
(MAP MODE)
-figure 6.1-



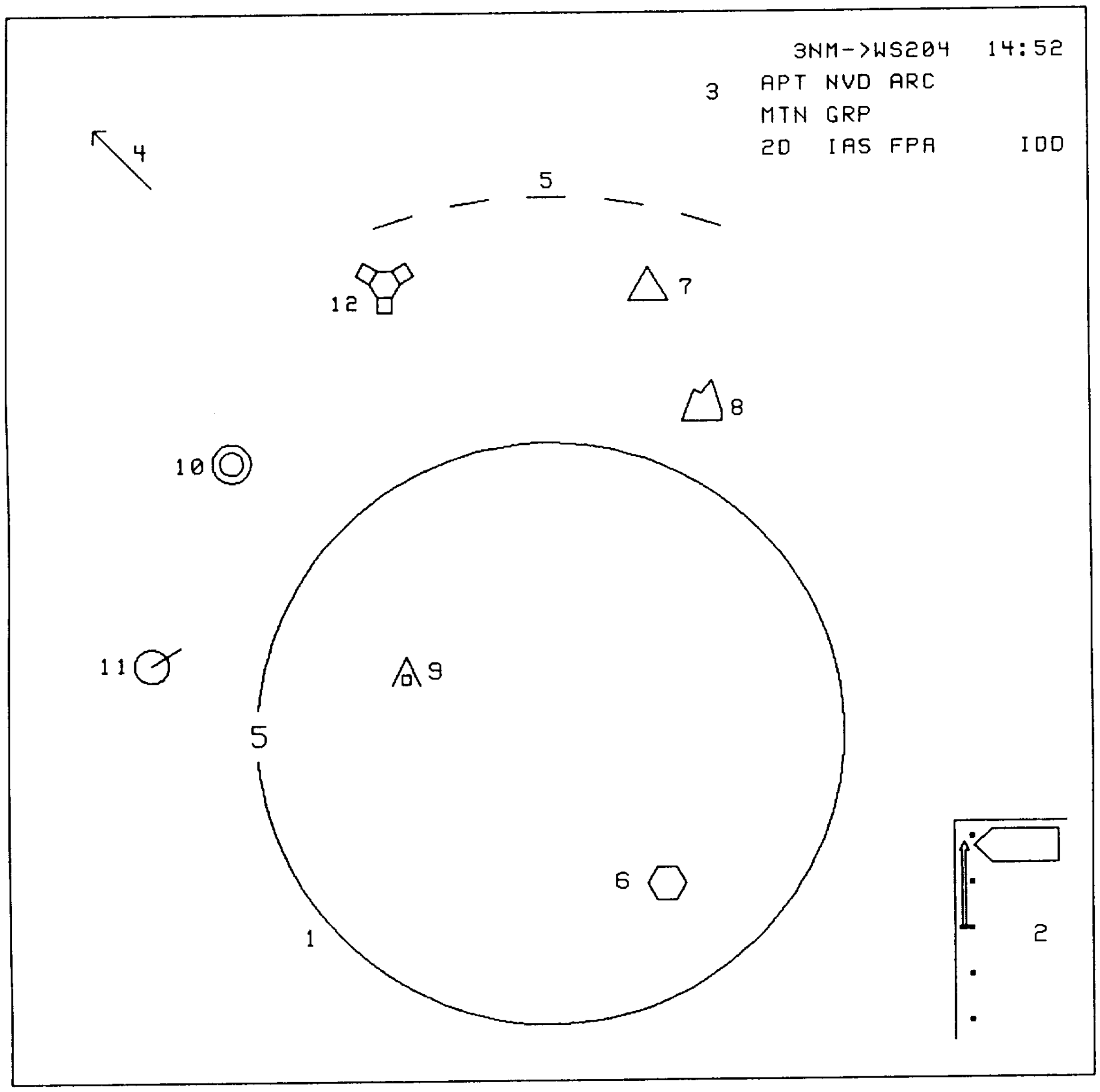
NAVIGATION DISPLAY FORMAT
(PLAN MODE)

-figure 6.2-



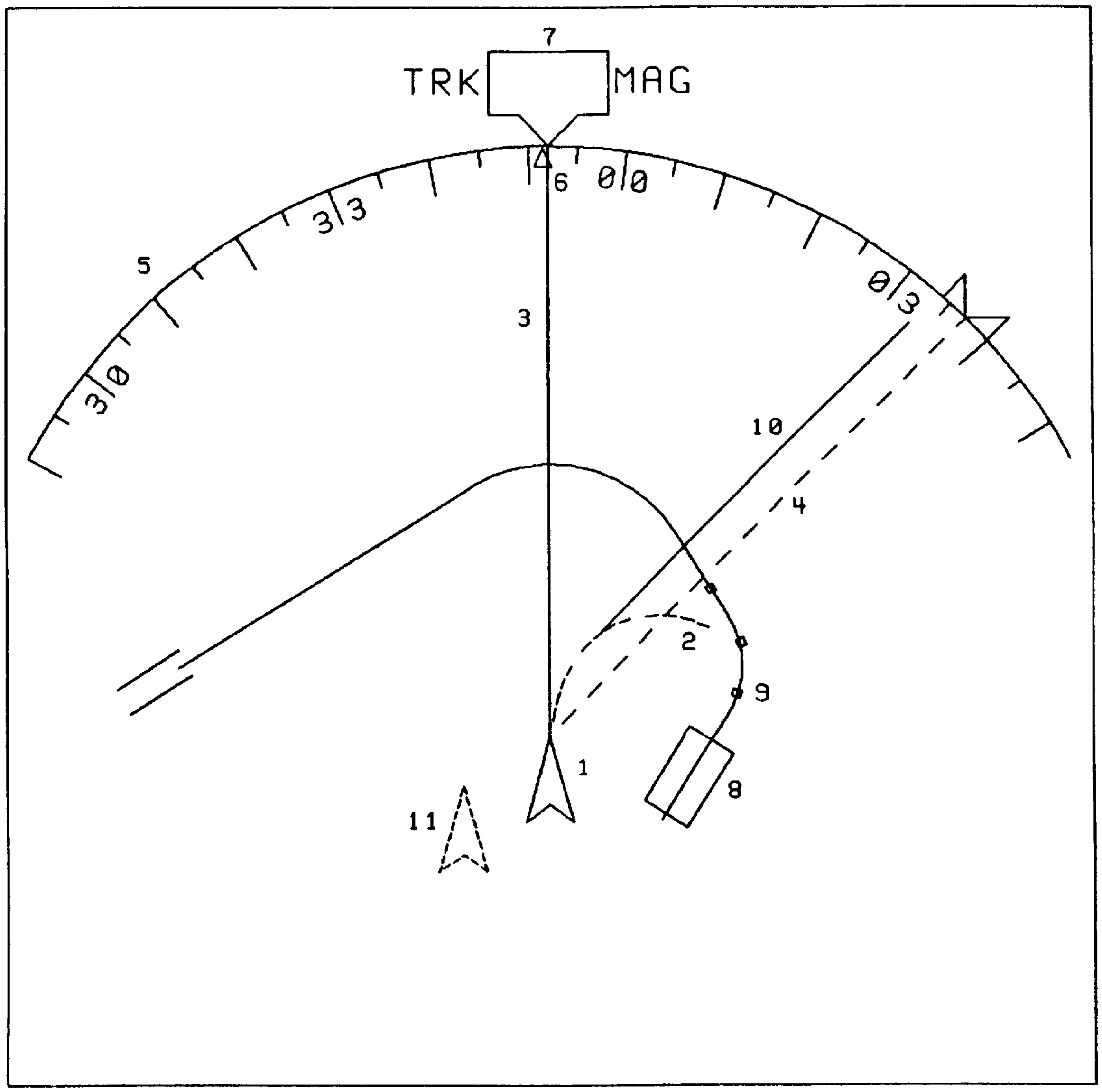
NAVIGATION DISPLAY FORMAT
(SYMBOLOGY PART 1)

-figure 6.3-



NAVIGATION DISPLAY FORMAT
(SYMBOLOGY PART 2)

-figure 6.4-



NAVIGATION DISPLAY FORMAT
(SYMBOLOGY PART 3)

-FIGURE 6.5-

NAVIGATION DISPLAY SYMBOLOGY PART 1

1. FLIGHT PATH

The flight path is drawn based on information contained in the guidance buffer. It is oriented in a track-up position in map mode, and north-up position in plan mode. A track-up orientation indicates that the airplane's track is straight up on the display. In map mode, the airplane remains fixed near the center of the display and the path rotates as the track changes. A north-up orientation means straight up is North. In plan mode, the flight path remains fixed (excluding any map scale changes) while the airplane moves around the screen as its lat/lon position changes.

The two basic flight paths are the active and the provisional. An active path is a path that has been executed on the CDU, and it is displayed as a solid white line. A provisional path is a flight path that has been entered, or is being entered, but has not been executed. It is shown as a series of dashed green lines.

2. WAYPOINT

Each flight path has a set of waypoints associated with it. The waypoint symbols are shown in the same color as the associated path. A four phase bezel switch controls the amount of information that can be displayed next to waypoints on the active flight plan. The four phases are: "to" waypoint name only, "to" waypoint plus altitude and ground speed data, names for all waypoints on the path, names and altitude and ground speed data for all waypoints. The altitude and ground speed data come from the guidance buffer and are shown below the waypoint name. Note that all this information is shown with provisional path waypoints.

3. RUNWAY

The runway symbol is displayed in white. It is only shown on map scales of 40 nautical miles or less, otherwise an airport symbol is displayed. The orientation of the runway is based on the runway heading. The airport name is output with the symbol, unless the runway has been selected via the Reference Nav Data page on the CDU in which case the runway number is output instead.

4. RUNWAY CENTERLINE EXTENSION

The runway centerline is a green dotted line that extends from the runway threshold latitude and longitude to a point 10 nautical miles away. The line is oriented according to the runway heading. The runway centerline is only displayed when the runway is shown.

5. RADIAL

The radial symbol appears when a position fix is chosen on the FIX page of the CDU. The inputs to the FIX page are the name of the reference point and the bearing of the radial. Up to four radials can be drawn through a single fix. There is also the capability of having two fixes. The yellow radial symbol consists of a circle drawn around the referenced position, with two solid lines extending from opposite ends of the circle oriented according to the selected bearing.

Another option available on the FIX page but not depicted in the diagram is the circle around a fixed point. The inputs to the CDU are the name of the reference point and the radius of the circle. It is also displayed in yellow.

6. NORTH POINTER

The north pointer is displayed in white and is only shown in plan mode. This symbol indicates the orientation of the map in this mode is such that straight up is north (true). The "N" displayed above the pointer emphasizes the north direction.

7. BOUNDARIES

The boundary areas consist of three different types; CADIZ (coastal air defense identification zones), ADIZ (air defense identification zones), and restricted areas. Whenever any of these zones lie within the viewing area they are bounded by dot/dash lines. The color of the line is determined by the type. ADIZ lines are displayed in red, CADIZ in yellow, and restricted areas in amber. A tag indicating the type of boundary, or the name of the restricted area, will be displayed somewhere along the line. Boundary lines are selectable via a bezel switch.

NAVIGATION DISPLAY SYMBOLOGY
PART 2

1. RANGE CIRCLE

The range circle is displayed in dim cyan. On the left center edge is a gap in the circle which contains a small number that tells how many nautical miles the radius of the circle represents. The group of selectable circle radius values are: 2, 5, 10, 20, 40, and 80 nautical miles. The range circle is shown in both plan and map modes.

2. VRT DISPLAY

The VRT display represents vertical deviation from the source of vertical guidance (which may be, for example, the flight path, glideslope beam, or MLS beam). It repeats some of the information already shown on the PFD format's altitude tape. The VRT display consists of a green pointer box, a yellow vertical speed arrow and a white vertical scale.

The display represents one of three vertical guidance modes being engaged. The pointer contains either the reference altitude in feet when in altitude select mode, the text 'G/S' in glideslope mode, or 'VRT' for a valid 3D path. When no vertical guidance is engaged the scale is removed from the display. The pointer moves along the scale which indicates a +/- .7 degree range in 'G/S' mode, and a +/- 1000 foot range in the other two modes.

The vertical speed arrow indicates the rate of change of aircraft altitude. The arrow does not grow at a linear rate. It represents 800 feet per minute of vertical speed when it is about one half dot from the center. At one dot of deviation it is approximately 2,300 feet per minute.

The VRT display and some of its surrounding area form a rectangular patch that masks everything that passes through it. The mask area is outlined by a green colored border. This mask disappears when the VRT display is not shown.

3. MESSAGES

Four text lines are output in small white characters in the upper right hand corner of the NAV display. These lines contain information pertinent to the NAV format. The following chart shows the information presented on these lines.

LINE #1

ITEM	CONDITIONS
distance to go	when active flight plan exists
'TO' wpt name	when active flight plan exists
time of day	always

LINE #2

ITEM	CONDITIONS
airport indicator	'ARPT' bezel selected
navaid indicator	'NVD' bezel selected
time box indicator	'BOX' bezel selected
range arc indicator	'ARC' bezel selected

LINE #3

ITEM	CONDITIONS
terrain indicator	'MTN/OBSTR' bezel selected
GRP indicator	'GRP' bezel selected
boundary indicator	'BNDS' bezel selected

LINE #4

ITEM	CONDITIONS
2D/3D/4D	when auto guidance engaged
TRK indicator	auto-track select mode engaged
ALT indicator	auto-altitude hold mode engaged
IAS indicator	auto throttle engaged
FPA indicator	auto-flight path angle engaged
navigation mode	always

Below the fourth line, but not depicted in this diagram, the text 'MLS VLD' or 'MLS ON' will be displayed in medium sized blue characters if the conditions for either situation are met.

4. WIND DIRECTION POINTER

The wind direction pointer is displayed in yellow. It is only shown when the wind speed is greater than four knots and the format is in map mode. The wind direction pointer is oriented so that it correctly shows the way the wind is blowing with respect to the airplane. It points in the direction that the wind is blowing to. The wind speed is displayed beneath the wind pointer in small yellow characters.

5. ALTITUDE RANGE ARC

The altitude range arc is displayed when an altitude has been selected on the AGCS mode control panel and its bezel switch has been pressed on. The range symbol is yellow and can either be a solid or dashed arc. The altitude range arc is displaced from the tip of the airplane by the distance it will take to reach the selected altitude given current conditions such as ground speed and flight path angle. If that distance is too great to be displayed, a dashed arc is shown at a fixed location directly below the compass, otherwise a solid arc is shown. When the selected altitude and current altitude are within five feet of each other the arc disappears. The altitude range arc is only shown in map mode.

6 - 12. REFERENCE POINTS

The set of reference points consists of the following symbols displayed at fixed lat/lon locations: VOR (6), GRP (7), mountain (8), obstruction (9), non-directional beacon (10), airport (11), and VORTAC (12). Available to be shown but not currently used or depicted in this diagram is the DME/TACAN symbol.

The subset of navigation aids include the VOR, VORTAC, TACAN, and non-directional beacon. All these navaids have a three character name that is output to the right of the symbol. The VOR and VORTAC are white unless tuned in which case they are green. The TACAN symbol can also be tuned. The color of the non-directional beacon is cyan.

Another subset group is terrain data which consists of the mountain and obstruction symbols. Both are shown in white and each has a numeric value displayed below the symbol. For the obstruction symbol the tag represents the reference point height in feet, and for the mountain in thousands of feet.

The GRP and airport reference point symbols are displayed in white and cyan respectively. The GRP has a five character name that is shown to the right of the symbol, and the airport symbol has a four character name.

All symbols listed above are selectable through bezel switches for navaids, terrain data, airports, and GRP's, except for navigational aids that have been tuned.

NAVIGATION DISPLAY SYMBOLOGY
PART 3

1. AIRPLANE

The airplane symbol is displayed in yellow. It is oriented in a track-up position in map mode, and north-up position in plan mode. In map mode the symbol is fixed at the location (0",-1.25"), with the center of the map being (0", 0"). Since straight up in map mode represents the airplane's track, other symbols based on lat/lon positions will rotate as the track changes. Also in map mode, two numbers are displayed in small yellow text below the airplane. On the left side is the ground speed in knots, and on the right side is the altitude in thousands of feet.

In plan mode, straight up on the display is North. When this mode is selected the airplane, instead of being fixed, moves around the screen as its position changes. The symbol is output based on its lat/lon coordinates relative to the screen center lat/lon coordinates.

2. CURVED TREND VECTOR

The curved trend vector is an indicator of where the airplane will be in 30, 60, and 90 seconds given the current ground speed and cross track acceleration. The trend vector consists of three segments, each containing five dashes. The first dash of each segment is blank. Each dash represents a six second time period. The curved trend vector is colored yellow and is only displayed in map mode.

3. TRACK LINE

The track line is a straight line drawn from the tip of the airplane to the track box whose bearing is the current track angle of the airplane. It is displayed in cyan and is only shown in map mode.

4. SELECTED TRACK

The selected magnetic track represents the track dialed in from the mode control panel. If the selected track differs from the current track by more than one degree, or if the track has been preselected, then a dashed line is drawn from the tip of the airplane in the direction of the selected track (with respect to current track which is straight ahead on the display). The dashed line is 3.8" long which is the distance to the compass arc. A symbol resembling a large "M" moves along the top of the compass and points to the selected track. The "M" disappears when it moves off the compass arc but the dashed line is displayed even if it does not point to a location along the arc. The selected track line and pointer are displayed in amber and are only shown in map mode.

5. COMPASS ARC AND DIGITS

The compass arc and digits are displayed in white and are only shown in map mode. The arc displays a range of angular values of approximately 120 degrees, or about 60 degrees in either direction of the current track. The large ticks on the arc represent ten degree intervals and the smaller ticks five degree intervals. The numbers on the compass arc are degree values divided by ten, such that "03" actually means 30 degrees.

6. PRESENT HEADING

The present magnetic heading symbol is a small yellow triangle that moves along the underside of the compass arc. It points to some angular value along the arc. The difference between the present heading and the current track is the drift angle. The present heading symbol is only shown in map mode.

7. TRACK READOUT

The track display consists of the current magnetic track value output in large yellow letters inside a cyan box. The box sits on top of the compass arc with one side pointing to the place along the arc where the track line intersects it. The text "TRK" and "MAG" is displayed in cyan to the left and right of the box respectively. The track readout is only shown in map mode.

8. TIMEBOX

The timebox is colored magenta and is displayed only in map mode. The timebox follows the flight path. It represents the position the airplane should be along the path, given the times the guidance buffer says the airplane should reach each waypoint. If the airplane is flying without a time error it will be positioned inside the box. The timebox symbol is only displayed when the airplane is in a 4-D navigation mode and the timebox is selected via a bezel switch.

9. BUBBLES

There are three small dots called "bubbles" that are displayed along with the timebox. Like the timebox they are colored magenta and are only shown in map mode. They indicate where the timebox will be on the path in 30, 60, and 90 seconds.

10. TRACK SELECT LINES

Two track select lines can optionally be shown to display the projected ground track at rollout. The two lines have been termed the "extended" trend vector, and the "offset" vector. The variable TKBITS must be manually set to the appropriate values to display either one of these lines.

The airplane of course does not automatically change its track to the selected track. The "extended" trend vector uses the rate of change of track to show the time along the trend vector before the desired track is reached. The "offset" vector uses the speed of the airplane and a fixed bank angle of 25 degrees to show the distance ahead the airplane will have to fly before desired track is reached. The "extended" trend vector is a solid yellow line extending off the curved trend vector, while the "offset" vector is a dashed cyan line extending off the straight trend vector. The "offset" vector is not pictured in this diagram.

11. MLS AIRPLANE

A dashed airplane symbol is displayed at the MLS computed position when MLS is valid (but MLS mode is not engaged), and the MLS position is within 10 nautical miles of the current airplane position. The symbol will be white if MLS altitude differs by less than 50 feet, and MLS cross track differs by less than 500 feet. The symbol is amber if it is outside these limits. When MLS mode is engaged the real airplane symbol goes to the MLS position.

Section 6.2 THE NAV BACKGROUND BUFFER

The background data buffer is the first 400 words of the 704 word buffer sent to the Sperry display system from the host computer. Appendix A gives the layouts of the I/O buffers. The background data occurs in varying amounts depending on the position of the aircraft on the chosen route, and contains information on the reference points and flight plan. Unlike the rest of the data sent to the microprocessor system, the background data is a variable length stream, up to 800 words, of information which cannot be identified by its I/O buffer position. Interpretation of the data must be performed with a sequential parsing, starting at the first location. Special code words are used within the data stream to identify what type of background information follows.

Each NAV format loaded into a display processor may have unique background data. However the same 400 word data area is used by all NAV formats, therefore they must take turns using the background data buffer. Actually many of the 704 word data buffers read from the host computer contain no new data in the background area since background updates occur infrequently (every five seconds, after a NAV bezel button selection, or following a flight plan modification).

A word of data, "OUTDAT(599)" in Appendix A, is used to control the use of the background buffer. The meaning of the bits in the Map Control word are as follows:

BITS	DESCRIPTION
0-3	Display processor identifier code.
4	Part two update flag.
5-7	Unused.
8-15	Update sequence number.

The display processor identifier code is a number that designates the processor which should use this data. The next chart gives the code values used for the various processors.

CODE	DEU #	DP #
1	1	1
2	1	2
3	1	3
4	2	1
5	2	2
7	3	1
8	3	2
9	3	3

The part two flag informs the NAV format that the background buffer is to be considered a continuation of the last one received. This gives the ability to send 800 words of background data to a NAV format in two consecutive updates. The sequence number informs the NAV format that the data in the background buffer is new data. This number increments every time a new background buffer is created. This allows the NAV format to distinguish between new data designated for it and old data that has been in the buffer for a while.

All data within the background buffer is stored in groups of 2 or more words. The first word of the group is always a "group identifier word" having the group ID label, class, and word count. The diagrams at the end of this section show the various formats of background data words. Bits zero through two of the group ID word are used for the label. Presently only the following five of the possible eight group types are used.

LABEL	USAGE
0	Control group
1	Text group
2	Symbol group
3	Rotatable symbol group
4	Line segment group

The class code, bits three through seven, is used to differentiate between variants of the given group. The upper eight bits of the group ID word are the word count. The value in this byte is the total words associated with the particular data group following the ID word.

The control group is used to set up map display software prior to the processing of the remaining background data. Two classes exist in this group; start of transmission (SOT) data and mode controls. The SOT class is always the first data in the background buffer. The total number of words used in the buffer for the entire background update is stored along with the map sequence index. The index is used to pick the correct map center displacement out of the array of four sent from the host computer. Four map center values exist because up to four independent NAV formats may run simultaneously. The second class, mode controls, sets map scale, enables selectable symbology (weather radar, time box, range arc), and chooses map orientation (track-up/north-up).

The text group controls the placement of ASCII data on the four available text lines appearing at the upper right corner of the NAV format. The class value selects which line data is written to.

Reference position symbols are placed on the display by symbol group entries. The data for this group contains the north and east displacements from the map center and optional descriptive text to be shown with the symbol. The type of symbol (DME, GRP, AIRFIELD ...) is selected by the class code.

Symbols which require a particular orientation on the display screen are handled by the rotatable symbol group. The bearing and length (for runway only) is stored in the data group in addition to the position and text data.

The final group currently used is the line segment group. The class field of this group's ID word sets the line type and color for the following sequence of data. The two words after the identifier word are the north/east position values from map center for the line segment initial point. The remaining data in this group occurs in blocks of two or more words and is very similar to the group technique described above. Each block contains one element of the connected line/path segments being created. The first word of these blocks is the path element ID, containing the element type and word count. The element types are line, arc, on path waypoint, and off path waypoint. The line element commands a line to be drawn from the last path position to the position designated by the north/east coordinates provided. The arc type gives the subtended angle, initial inbound angle, and the arc radius. The on path waypoint element places a waypoint symbol at the last defined path position. Off path waypoints have their own north/east coordinates for placement.

The symbology created from the background data is repositioned on the display screen every 50 milliseconds by the airplane position and track data received from the I/O buffer. This allows the airplane to move smoothly over the map data. The only requirement is that the background be updated at a fast enough rate so that the display screen always covers the area defined by the background.

A sample background data buffer is shown below. The values shown are the binary words presented in hexadecimal format. The data produces a simple flight plan consisting of two straight segments, one arc segment, and a waypoint symbol entitled "COLIN". Text data for lines #1 and #4 also appear in the data. Figure 6.6 shows the path drawn from this data. Note the overlap of the physical viewing window which allows displacing the background from the physical screen center as the airplane flies.

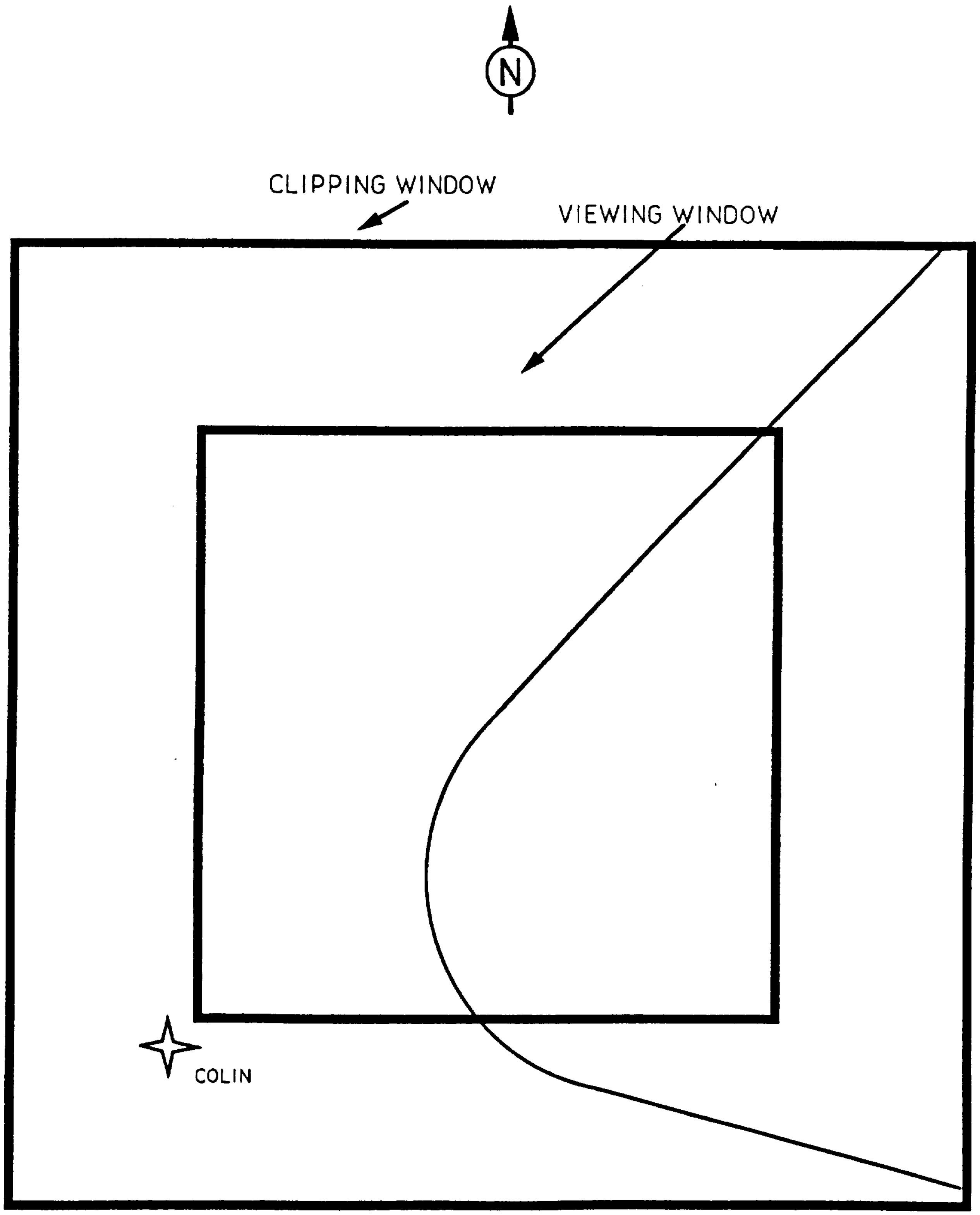
```
0100        ; SOT.
0039        ; 57 words background update.

0108        ; controls.
0001        ; scale is 5NM radius.

1204        ; line segments - 18 words.
11FE        ; initial position: 4.606 inches east
1307        ;                   4.871 inches north.
0200        ; draw line to:
FFF2        ;               .014 inches west
FFE3        ;               .029 inches south.
0301        ; draw arc with:
5977        ;                125.8 degree turn
F61D        ;                2.531 inch radius (left turn)
9F11        ;                -136.3 degree inbound bearing.
0503        ; place off path wpt at:
F298        ;                        3.432 inches west
F1E9        ;                        3.607 inches south
4F43        ;    text: "COLIN "
494C        ;
204E        ;
0200        ; draw line to:
1307        ;               4.871 inches east
ED81        ;               4.735 inches south.

0A01        ; text line #1.
2020        ; "  7NM->COLIN  12:32 ".
4E37        ;
2D4D        ;
433E        ;
4C4F        ;
4E49        ;
2020        ;
3231        ;
333A        ;
2032        ;

0A19        ; text line #4.
4433        ; "3D  IAS         IXX ".
2020        ;
4149        ;
2053        ;
2020        ;
2020        ;
2020        ;
2020        ;
5849        ;
2058        ;
```



SAMPLE PATH DATA

-figure 6.6-

BACKGROUND BUFFER LAYOUT

Buffer consists of 5 data groups:

Label	Title
0	Control Group
1	Text Group
2	Symbol Group
3	Rotatable Symbol Group
4	Line Segment Group

Group identifier words always precede data belonging to a specific group.

Group Identifier Word:

WORD COUNT	CLASS	LABEL
8 bits	5 bits	3 bits

Where:

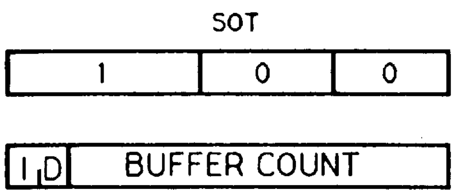
Word count	= # of words in data group following identifier.
Class	= identifies subgroup items, line types, colors, etc.
Label	= group number 0 - 4 listed above.

Notes on units for following pages:

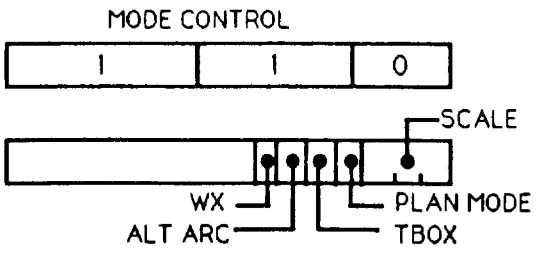
-- all distance values : 1/1000th inches.
-- all angular values: (degrees/180) * 2**15.
-- all text: standard ASCII codes in consecutive byte order.

-FIGURE 6.7-

LABEL 0 CONTROL GROUP:



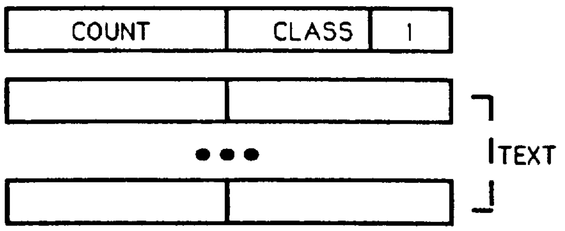
Scale	Range (nm)
0	2
1	5
2	10
3	20
4	40
5	80



LABEL 1 TEXT GROUP

CLASS

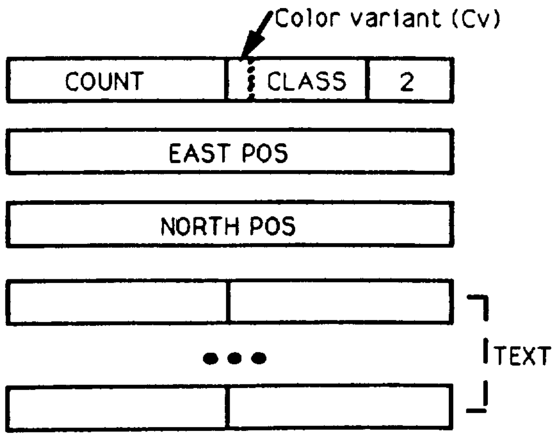
0: Text line 1
1: Text line 2
2: Text line 3
3: Text line 4



LABEL 2 SYMBOL GROUP

CLASS

0: Mountain (Cv ignored)
1: Obstruction (Cv ignored)
2: Non-directional Beacon (Cv ignored)
3: VORTAC (Cv ON = tuned)
4: VOR (Cv ON = tuned)
5: DME/TACAN (Cv ON = tuned)
6: Airfield (Cv ignored)
7: GRP (Cv ignored)
8: Marker Beacon (Cv ignored)
9: Waypoint (Cv ON = provisional)



PRECEDING PAGE BLANK NOT FILMED

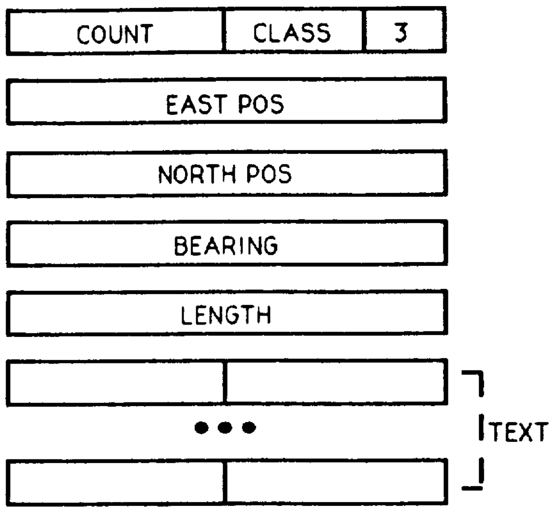
-FIGURE 6.7 (continued)-

112 [illegible]

LABEL 3 ROTATABLE SYMBOLS GROUP

CLASS

0: AIRFIELD (Length ignored)
1: Runway
2: SRP (Length ignored)



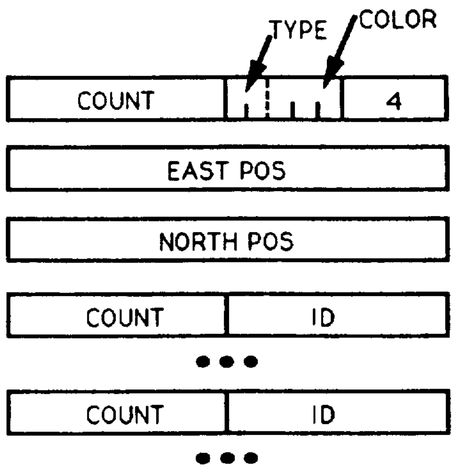
LABEL 4 LINE SEGMENT GROUP

LINE COLOR

0: White	4: Red
1: Green	5: Cyan
2: Blue	6: Magenta
3: Amber	7: Yellow

LINE TYPE
0: Solid
1: dot
2: Dash
3: Dot/Dash

PATH SUBGROUP ID:
0: Position
1: Arc
2: Wpt (on path)
3: Wpt (off path)

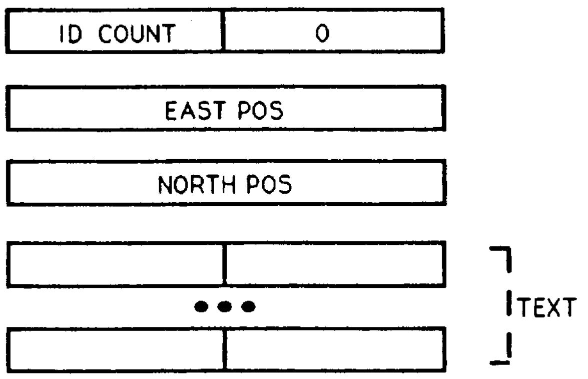


PRECEDING PAGE BLANK NOT FILMED

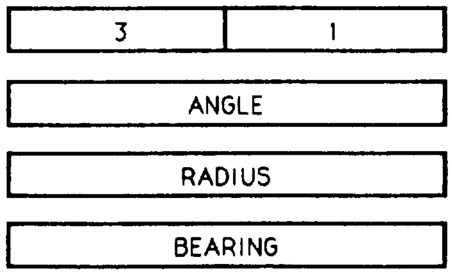
114

-FIGURE 6.7 (continued)-

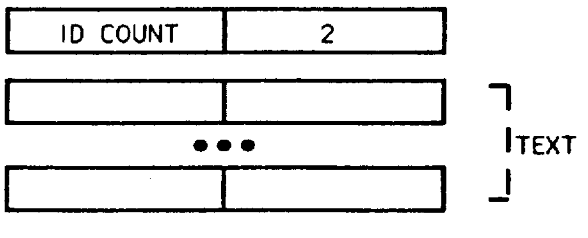
ID 0: Position



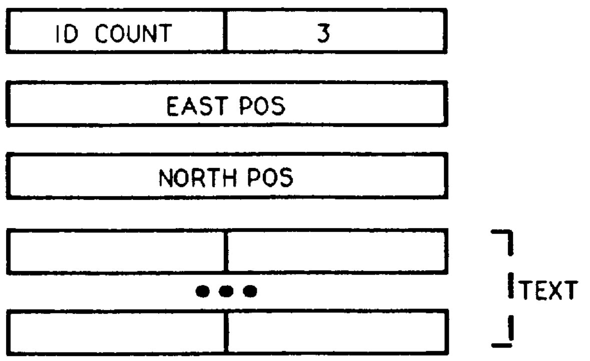
ID 1: Arc



ID 2: WPT (ON)

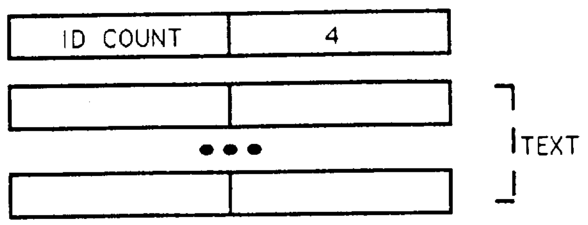


ID 3: WPT (OFF)



-FIGURE 6.7 (continued)-

ID 4: TEXT



-figure 6.7 (continued)-

Section 6.3 BACKGROUND PROCESSING ROUTINES

The following pages contain descriptions of those software modules involved in the processing of the background buffer, which was described in Section 6.2.

MODULE NAME: BACK_MGR
FILE NAME: BACK_MGR.PLM
CALLED BY: MAP_EXC

PURPOSE: To serve as the executive for the creation of the background buffer graphic objects.

REMARKS: This procedure controls the processing of the background buffer. The structure of the background buffer was described in Section 6.2. All background data is built into one of three graphic objects: GR_PATH, GR_SYMBOL, or GR_MESSAGE. For data to be used it must first be parsed to determine what kind of data it is. This identification process is done through the use of group identifier words. Group identifier words contain three important fields of information: bits 0-2 hold the group label, bits 3-7 store the "class" field which allows for varieties within a group, and bits 8-15 contain the number of words following that are associated with the group. BACK_MGR parses each group ID word and separates the fields into global variables. Using the group label it calls the appropriate procedure to process that group's data. The five available data groups and corresponding labels are:

0 - Control group
1 - Text group
2 - Symbol group
3 - Rotatable symbol group
4 - Path group

BACK_MGR continues processing group ID words until the buffer is exhausted or an error in format has been detected.

BACK_MGR does some minor error checking itself. It makes sure the very first data word in the buffer is the start-of-transmission word (label 0, class 0). Also it checks to see if the extracted group label falls in the allowable range of 0-4. If an error is detected, processing of the buffer discontinues immediately. No effort is made to recover from bad data.

GLOBAL REFERENCES:

VARIABLES
BACKGROUND BPTR* BUF_ERROR* FCB GE_MESSAGE GE_PATH
GE_SYMBOL GR_MESSAGE GR_PATH GR_SYMBOL

PROCEDURES
CONTROL_GROUP ENQ_OBJECT_BLOCK GET_BUFWRD
NEW_OBJECT_BLOCK PATH_GROUP ROTANG_GL ROTATE_GROUP
SYMBOL_GROUP TEXT_GROUP TRIM_OBJECT_BLOCK

MODULE NAME: CONTROL_GROUP
FILE NAME: BACK_MGR.PLM
CALLED BY: BACK_MGR

PURPOSE: To process the start-of-transmission (SOT) and mode control words.

REMARKS: This procedure is called by the background executive whenever a "group identifier word" with label 0 is encountered in the buffer data stream. There are two types of data in this group: the start-of-transmission word (class 0), and the mode control word (class 1).

The start-of-transmission word must be the first item in the buffer. The lower bits of the SOT word tell the number of words contained in the background buffer. This value is saved in a global variable and will be used by the background executive to determine when the buffer is exhausted. The upper two bits serve as an index into the map displacement array in I/O RAM.

The mode control word contains several items of information. The lower three bits hold an index (0-5) that corresponds to one of the available Map scales (2,5,10,20,40, 80 nautical miles). Bits 3 is set if PLAN mode is selected and zeroed for MAP mode. Bits 4-6 are for bezel selectable features: timebox and bubbles, altitude range arc, and weather radar respectively. The mode control word information is extracted and stored in global variables.

GLOBAL REFERENCES:

VARIABLES
  ALT_ARC_SEL* BUF_ERROR* NAV_ID* ND_CSERNG* ND_MODE*
  TIME_BOX_SEL* WX_SEL*

PROCEDURES
  GET_BUFWRD

```
MODULE NAME:  GET_ARC_ENDPT
FILE NAME:    BACK_MGR.PLM
CALLED BY:    PATH_GROUP
PARAMETERS:   HDG, ANGLE, RADIUS
```

PURPOSE: To compute the endpoint of an arc segment.

REMARKS: GET_ARC_ENDPT is called by the global procedure PATH_GROUP after a path arc has been drawn. It computes the X,Y position of the endpoint of the arc based on turn angle, heading, and turn radius. These three values are all passed to the procedure through the parameter list. This point represents the current beam position on the path from which the next path segment will be drawn.

GLOBAL REFERENCES:

PROCEDURES
  U_COSINE U_MULT U_SINE

MODULE NAME: GET_BUFWRD
FILE NAME: BACK_MGR.PLM
CALLED BY: BACK_MGR, CONTROL_GROUP, SYMBOL_GROUP, GET_TEXT, ROTATE_GROUP, PATH_GROUP

PURPOSE: To retrieve the next word from the background buffer.

REMARKS: GET_BUFWRD is a typed procedure that returns as output the next unprocessed word from the background buffer. It increments the buffer pointer BPTR upon entering the procedure to the correct location. It also decrements the count variable ND_COUNT to reflect the number of words left to be processed for the current label type.

GLOBAL REFERENCES:

VARIABLES
  BACKGROUND BPTR*

MODULE NAME: GET_TEXT
FILE NAME: BACK_MGR.PLM
CALLED BY: TEXT_GROUP,SYMBOL_GROUP,ROTATE_GROUP,PATH_GROUP
PARAMETERS: COUNT

PURPOSE: To fetch a character string from the background buffer.

REMARKS: GET_TEXT fetches the character string and stores it in the global byte array TEXT. The input parameter tells how many consecutive words in the buffer the string is contained in. Each word holds two characters.

Once the string is stored in TEXT, the text buffer is scanned for any carriage return, line feed codes. If one is found character codes are inserted in the text buffer to relocate the beam before the remainder of the text is displayed. This is done through a set of user defined characters created to simulate small character backspaces from one to six characters, and a small character line feed. These user developed characters have been mapped into the small character set so that these codes can be imbedded in the same text arrays with small sized text. An example of imbedded line feed and carriage returns would be waypoint information text that includes name, altitude, and ground speed data, each displayed on separate lines.

GLOBAL REFERENCES:

VARIABLES
  ND_BKS_1 ND_BKS_6 ND_LF_1

PROCEDURES
  GET_BUFWRD

MODULE NAME: MAP_XFRM
FILE NAME: BACK_MGR.PLM
CALL RATE: .05 SECONDS
CALLED BY: MAP_EXC

PURPOSE: To generate the runtime transform for the path and symbol objects.

REMARKS: The transform for the symbol and path objects differ in MAP and PLAN modes. In MAP mode the airplane stays fixed and the background objects move with respect to the current airplane position. Therefore a pre-rotation translation using the airplane position for the current map must be performed to shift the map. Also in MAP mode, the airplane's bearing is used to rotate the map appropriately.

In PLAN mode the airplane moves around a fixed path. Since the background objects are created with a north-up orientation, and that is what is needed in PLAN mode, no rotation or pre-rotation translation is needed.

A post-rotation translation is performed for either mode. It will shift the map reference point (in MAP mode the airplane's position, and in PLAN mode some position on the path) down 1.25 inches.

GLOBAL REFERENCES:

VARIABLES
AP_E AP_N FCB ND_BEARING ND_MODE

PROCEDURES
ENQ_OBJECT_BLOCK MAKE_XFORM_ZERO

MODULE NAME: PATH_GROUP
FILE NAME: BACK_MGR.PLM
CALLED BY: BACK_MGR

PURPOSE: To output the graphics necessary to show the flight path.

REMARKS: PATH_GROUP outputs the graphics used to draw the flight path. This includes path lines, turns, and waypoints on and off the path. Different line types and colors can be selected. The set of colors available are: white, green, blue, amber, red, cyan, magenta, and yellow. The line type specifications include: solid, dot, dash, or dot/dash.

Path data is identified by a label type of 4 in bits 0-2 of the "group identifier word". Bits 3-5 contain an index for line color, and bits 6-7 hold the line type code. The upper byte tells how many consecutive buffer words following are associated with the path data. This total includes the path subgroups described below. Immediately after the "group identifier word" are two words specifying the X and Y coordinates for the path segment's initial offset from map center, in one-thousandths inches.

Path data, as mentioned, is divided into subgroups for lines, turns, and waypoints. They are identified by ID words structured much like the "group identifier words". The ID word precedes the path subgroup data. Its lower byte contains an index 0-4 which corresponds to a particular subgroup type, and the upper byte tells the number of words following that are associated with the subgroup.

Subgroup ID type 0 indicates data follows for the endpoint position of a line. The two words that immediately follow contain the X,Y values for that point. A line will be drawn from the previous beam position to the new one. This subgroup is used primarily for drawing path lines. However, one special application is for the drawing of boundary lines. A boundary line can have text written next to it, so this subgroup may additionally have a variable amount of text following the X,Y position.

Subgroup ID type 1 identifies path turn data. The word count field of the subgroup ID word has a fixed value of 3 corresponding to the three background buffer words immediately following which contain the angle, radius, and bearing of the turn. These values are used to call the CGS routine that will draw the arc. The beam must be moved to the new position, so GET_ARC_ENDPT is called to figure the X,Y value of the arc endpoint.

Subgroup ID type 2 indicates data follows for a waypoint on the path. The waypoint symbol is displayed at the current beam position, so no new X,Y point is provided by the background buffer. A variable amount of text may follow the subgroup ID word for the waypoint name.

Subgroup ID type 3 is used for a waypoint off the path. Since the symbol won't be shown at the current beam position, two words following the ID word specify the X,Y offsets from map center for the waypoint symbol. Once again a set of text words may follow containing the waypoint name.

The last subgroup, ID type 4, is used to place text at the current path position. The count contains the number of words of text supplied.

GLOBAL REFERENCES:

VARIABLES
  GR_PATH

PROCEDURES
  DRAW_ARC_REL_1 DRAW_REL_XY_VIS GET_ARC_ENDPT
  GET_BUFWRD GET_TEXT IPVGSUB_CSTRING MOVE_ABS_XY_BLANK
  SET_DASH U_DIV U_MULT U_SQRT VGSUB_CSTRING VID_PRI

MODULE NAME: ROTATE_GROUP
FILE NAME: BACK_MGR.PLM
CALLED BY: BACK_MGR

PURPOSE: To output the rotatable navigation symbols to the display.

REMARKS: There are three classes of rotatable characters currently provided for: origin/destination airports, runways, and selected reference points. Rotatable symbol data is recognized in the background buffer by a label type of 3 in the "group identifier word". The data for this group type consists of the following: an X,Y screen position for the symbol, a bearing by which to rotate, a length value that is only used by the runway, and a variable amount of text depending on whether a name is associated with the symbol.

The origin and destination airport symbol consists of a circle with a line drawn from the center out past the edge of the circle. The line represents the runway and the symbol is rotated according to the input bearing such that the runway line is appropriately positioned. The airport name is output in small characters next to the symbol. Both the text and symbol are displayed in white.

The runway symbol is only shown on map scales 2, 5, 10, and 20 nautical miles. It is represented as two white parallel lines. The runway width is a fixed arbitrary value of 1400 feet, but the runway length as mentioned is an input value. Using runway length, width, and bearing the four coordinates are transformed to a north-up orientation for proper display on the map. A runway ID consisting of the assigned runway number is output next to the symbol in small text.

The third rotatable symbol is the selected reference point, otherwise known as a "fix". This symbol is shown when a fix is selected via the Fix page of the CDU. The symbol is shown in yellow and consists of a small circle with two lines drawn outward from opposite edges of the circle extending to the edges of the screen (or mask area). The circle is drawn around a fixed reference point (navaid, airport, or GRP). The input X,Y position is the same as the coordinates of the fixed reference point. The circle and lines are rotated according to the input bearing selected on the CDU. The purpose of the "fix", simply put, is to draw a line through a reference point on the screen at a desired bearing.

```
GLOBAL REFERENCES:

VARIABLES
  BUF_ERROR* GR_SYMBOL ND_CSERNG TRU_TRK

PROCEDURES
  DRAW_ABS_XY_VIS GET_BUFWRD GET_TEXT IPVGSUB_CSTRING
  MOVE_ABS_XY_BLANK ROTANG ROTANG_GL U_COSINE U_MULT
  U_SINE VGSUB_CSTRING VID_PRI
```

MODULE NAME: SYMBOL_GROUP
FILE NAME: BACK_MGR.PLM
CALLED BY: BACK_MGR

PURPOSE: To output the non-rotatable navigation symbols to the display.

REMARKS: There are currently ten non-rotatable NAV symbols that can be output to the display. They constitute the following classes of data (0-9): mountains, obstructions, non-directional beacons, VORTACs, VORs, DME/TACANs, airports, ground reference points, marker beacons, and waypoints.

Non-rotatable symbol data in the background buffer is recognized by a label type of 2 in the "group identifier word". Bits 3-6 of that word contain the class number mentioned above. Bit 7 is used as a color variant bit such that any of the non-rotatable symbols can have two color options. In the current displays application the color variant bit is set on if a symbol has been tuned. Tunable symbols are displayed in green and include VORTACs, VORs, and DME/TACANs. All other non-rotatable symbols are shown in white except for the non-directional beacon and obstruction symbols which are displayed in cyan.

The buffer data for this label type includes an X,Y screen position for the symbol and a variable amount of text for its name, or in the case of the mountain and obstruction symbols - an altitude. The symbols come from one of two character sets: the PROM characters grafted into the medium sized character set (CS_MED), or the set of user defined characters (CS_ROT).

GLOBAL REFERENCES:

VARIABLES
BUF_ERROR* GR_SYMBOL

PROCEDURES
GET_BUFWRD GET_TEXT IPVGSUB_CSTRING VGSUB_CSTRING
VID_PRI

MODULE NAME: TEXT_GROUP
FILE NAME: BACK_MGR.PLM
CALLED BY: BACK_MGR

PURPOSE: To output the navigation information text lines.

REMARKS: The NAV information area is located at the top right side of the display format. It consists of four text lines. Each text line has certain types of information that can appear on it. The NAV information relates such things as distance to the next waypoint, time of day, bezel selections, and navigation modes. For more information on what can appear on each line refer to Section 6.1.

The procedure TEXT_GROUP processes four classes of data (0-3), corresponding to the four text lines. The class field determines the vertical positioning of the text line. GET_TEXT is called to retrieve the character string from the background buffer. It stores the string in the global byte array TEXT, the first byte of which contains the character count. Text line data is output in small white characters.

GLOBAL REFERENCES:

VARIABLES
BUF_ERROR* GR_MESSAGE

PROCEDURES
GET_TEXT IPVGSUB_CSTRING VID_PRI

Section 6.4 COMPILE TIME GRAPHICS FILES

The files covered in this section define core graphics data constants. They create system level data base tables, which define the format to the display units, and constant CGS objects for static graphic items. The blocks of data defined in these files will be loaded into the processor's EEPROM memory for data integrity.

FILE: COMPILE.ASM

COMPILE.ASM contains the hardcoded data for the navigation display's compile time graphic objects. One of the first things contained in this data file is the NAV format user defined character set. This consists of the binary data that defines the moves and draws of certain characters unique to the NAV display. They will be used to build the compile time and run time graphic objects. This set includes such things as the waypoint, altitude range arc, GRP, and airport symbols.

Nil variants of some of the graphic objects are also found here. When enqueued they will take away the appropriate graphics from the screen. The nil objects in this file are for the text, aircraft, path, reference points, compass and digits, wind display, north pointer, vertical deviation scale, bearing digits, weather radar text, and raster graphics.

Several other compile time objects used by the NAV display are defined here. They are the compass arc and track box, the range circle, north pointer, and the weather radar text. Also the weather radar compile time transform, and the raster graphics for the scan lines and raster patch.

FILE: SYS_TABS.ASM

This data file contains six system level compile time objects. They are: the event list, new format setup, top object, object table definition (OTD), translation tables, and the screen center transformation.

The event list is used to describe objects that are affected by dynamically changing transforms. Each entry in the event list represents a transform. Within the entry is a list of objects that will be repositioned based on that particular transform. For the NAV format the graphic objects that are moving are the path in map mode, the set of navigation symbols, and the aircraft in plan mode. The object table definition will associate these graphic objects with a transform. The transforms also have entries in the OTD which will map them to the proper entry in the event list. For more information on event lists consult Section 3.5 in the CGS manual.

The new format setup compile time object starts the initialization of a new format. It is always the first object block processed by CGS and contains resource information necessary to display the format, such as the total number of objects, the unique format ID, the number of OTD and event list tables, and the object number of the top object. It also specifies the colors that are used for stroke and raster symbology. For further details on the new format setup and its parameters refer to Section 3.6 in the CGS manual.

The object table definition defines the NAV format to CGS, and the resources needed to support it. It consists of a series of object table entry macros, one for each object in the format. Entries are listed in ascending order with the new format setup always object number 1. Special macros are used to define the new format setup, OTD table, event list, translation tables, and the top object. Also found is an entry for the NAV user defined character set CS_ROT, which contains symbols unique to the NAV display.

Following the system level OTD entries, the transformation and graphic objects for the NAV format are listed. Through OTD entries graphic objects are associated with transforms and transforms with event list entries (except for compile time transforms which use a special code indicating no action is taken). Transforms must be entered in the OTD table preceding the graphic objects they affect.

The buffering method required for the objects, static or pingpong, is also set up here. The pingpong method is used for objects that are constantly changing, however it requires more VG memory than the static method. To conserve memory, switching between static and pingpong is done whenever possible. For further information on building an OTD, and the individual OTE macros consult Section 3.4 of the CGS manual.

The top object identifies all the graphic picture objects for the NAV format. Each entry is essentially a call to display that object on the DU. Entries are made up of both run time and compile time objects. Despite the fact that an object may be enqueued, if it is not listed in the top object, it will not be displayed. Section 4.1 in the CGS manual describes the top object in more detail.

The translation tables are used to define character sets. They can combine characters from several sources to form a new set. For the NAV format three sets are defined: small, medium, and large. In PROM, several different sizes of standard alphanumeric characters are stored, from which the NAV character sets just mentioned map their entries. The first eight entries of the user defined set have been grafted into the small character set. These correspond to the line feed, six characters that backspace from one to six spaces, and the compass ticks character. Also residing in PROM is another special set of navigation symbols, some of which are used in the current NAV format - such as the mountain, obstruction, aircraft, and timebox symbols. This special set has been grafted into the medium character set by the translation tables. For further information on translation tables consult Section 4.4 in the CGS manual.

One final object created in this data file is the screen center transformation (TR_STAT_CMPSS). All objects which do not move will use this transform. This object in map mode will cause the tip of the aircraft to be repositioned at the location (0,-1.25"), and any other non-moving objects to be reoriented based on this new screen center position.

Section 6.5 RUN TIME GRAPHICS ROUTINES

This section describes the PLM86 modules that perform graphic operations for dynamic display objects on the NAV format.

```
MODULE NAME:  COMP_ARC
FILE NAME:    COMPASS.PLM
CALL RATE:    .2 SECONDS
CALLED BY:    XFRM_COMPASS
```

PURPOSE: To build the compass arc graphic object.

REMARKS: This procedure creates a 180 degree portion of a compass, known also as the track tape, by piecing together six 30 degree segments. Since the view area of the screen allows only a 120 degree portion to be displayed, a rotation of 30 degrees either direction can be handled before a new 180 degree compass portion is necessary. COMP_ARC is called by XFRM_COMPASS once every four frames, except for the situation where the turn rate is so great that a new compass section is needed, and then COMP_ARC is called immediately.

The compass arc is composed of a set of "tick blocks". A "tick block" is a user defined character mapped into the small character set that consists of a series of 6 ticks. These ticks divide the compass into 5 degree segments, and one "tick block" represents a 30 degree section of the compass.

To display the track tape COMP_ARC utilizes two arrays of data from TABLES.ASM, and creates the compass in 6 groups of 30 degree segments. The ND_CMPSS_TICK_PIE array contains the ASCII character codes for the set of bearing values (0, 30, 60, ..., 330 degrees). The bearings are displayed without the rightmost zero such that 330 degrees is shown on the track tape as "33". This array also contains the character code for the user defined "tick block". The other array, ND_CMPSS_IP, gives the x,y position and the rotation angle at which the characters in ND_CMPSS_TICK_PIE should be displayed. The compass arc is displayed in white. The bearing values are output in small characters.

COMP_ARC is also responsible for calling the procedure MOVE_BUGS. MOVE_BUGS contains the graphics necessary to generate the selected track and heading symbols that run along the compass arc.

GLOBAL REFERENCES:

VARIABLES
CMPSRES* FCB GE_CMPSS_DGTS I* K* ND_CMPSBRNG
ND_CMPSS_IP ND_CMPSS_TICK_PIE OLDBRNG*

PROCEDURES
END_OBJECT_BLOCK ENQ_OBJECT_BLOCK IPVGSUB_CSTRING
MOVE_BUGS NEW_OBJECT_BLOCK ROTANG ROTANG_GL VID_PRI

MODULE NAME: DYNAMIC_MASK
FILE NAME: MAP_MASK.PLM
CALL RATE: 0.8 SECONDS
CALLED BY: MAP_EXC

PURPOSE: To define bounded regions in the high resolution bit-plane for the masking of stroke drawn graphics objects.

REMARKS: When performing masking with the CGS, two bit-planes are used to define regions of the display screen. A bit-plane can be thought of as a rectangular grid filled with evenly spaced bits, known as pixels. Regions can be denoted by having some bits "ON" and some "OFF" in a bit-plane. When two bit-planes are involved more areas can be bounded by the intersection of regions defined in each bit-plane. See the "Core Graphics System" manual for information on stroke masking.

This subroutine sets and clears bits in the "High-res" bit plane by making calls to CGS subroutines. The areas defined by this bit-plane do not change position or shape but are not truely static since they will be created or removed depending on certain airplane and NAV format modes. Calls to the subroutine are made infrequently, every .8 seconds, since only discrete events affect the masking regions.

The following figure shows a rectangular area representing the display screen. The shaded region denotes where the pixels (bits) are set to "ON". The coordinates relative to screen center are given for boundary points. Note this configuration of the bit-plane represents the case where the NAV format is in "map" mode and the vertical deviation scale is present.

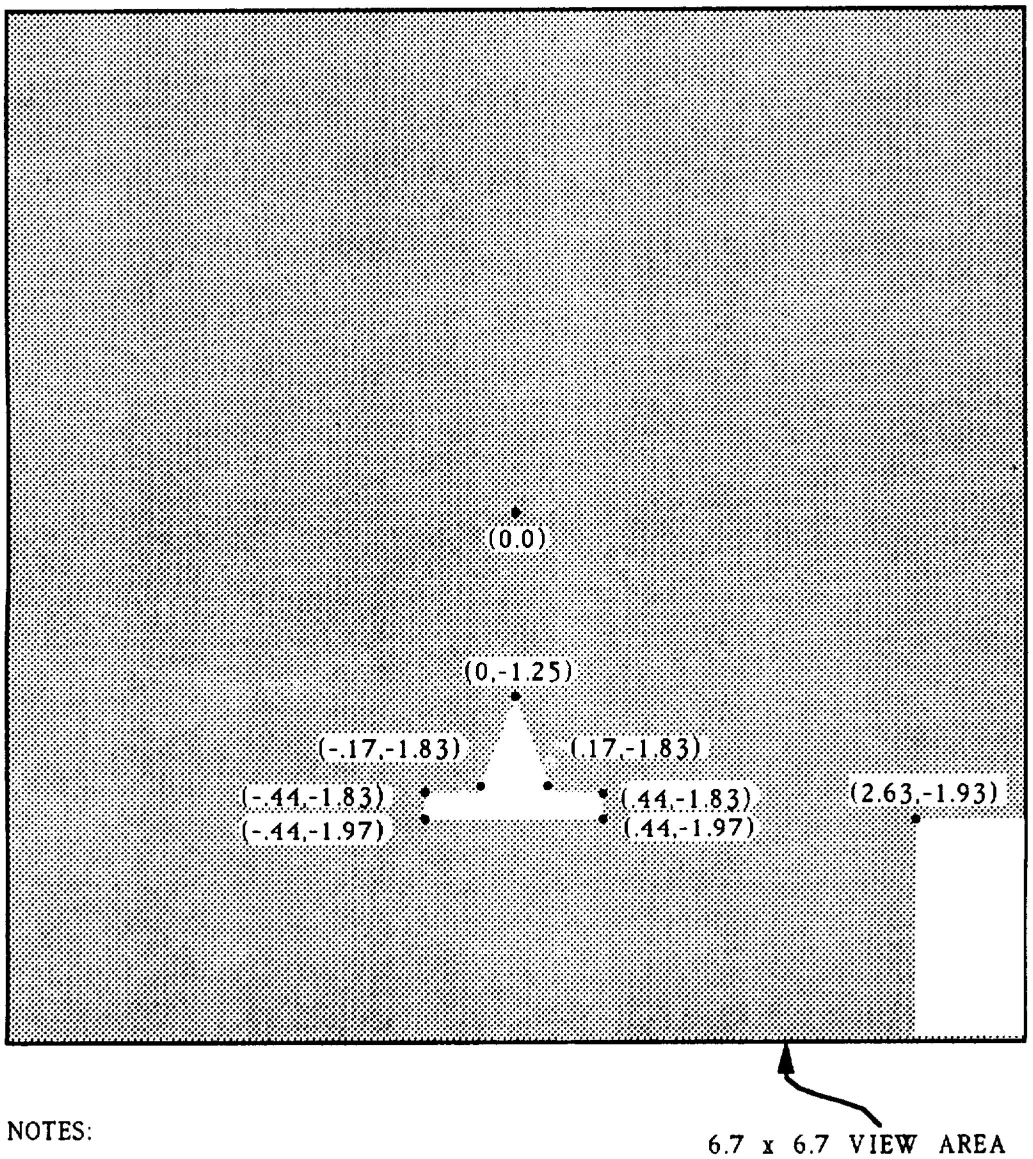
When the NAV display is in "map" mode the airplane and airspeed/altitude tags symbology appears just below screen center. A "witches hat" region is created in the bit-plane to allow the masking out of other map symbology from the airplane symbol region. Likewise a rectangular region in the lower right hand corner of the bit-plane is created when the vertical deviation scale is displayed on the NAV display. When either of the aforementioned conditions is false the corresponding region in the bit map will contain "ON" bits.

GLOBAL REFERENCES:

VARIABLES
FCB GE_AHRM ND_MODE VRT_ENG

PROCEDURES
DRAW_ABS_XY END_MASK_BLOCK ENQ_OBJECT_BLOCK
MOVE_ABS_XY NEW_MASK_BLOCK SET_MASK_CHANNEL
SET_MASK_PRIORITY

HIGH RESOLUTION BIT PLANE
(SHADED REGION - "ON" BITS)



NOTES:

- ALL COORDINATES ARE IN INCHES
- THE "WITCHES HAT" AREA HAS "ON" BITS WHEN MAP IS IN PLAN MODE
- THE RECTANGLE AREA HAS "ON" BITS WHEN VERTICAL DEVIATION SCALE OFF

-figure 6.8-

```
MODULE NAME:  MAP_AC_POS
FILE NAME:    MAP_MGR.PLM
CALL RATE:    0.05 SECONDS
CALLED BY:    MAP_EXC
```

PURPOSE: To create and enqueue the graphic object that in map mode outputs the aircraft, curved trend vector, altitude range arc, timebox, and MLS aircraft.

REMARKS: The first thing this routine does is display the aircraft and the track line that extends from its tip to the compass arc. The aircraft is colored yellow, and the track line cyan. In map mode the aircraft is fixed at the point (0,-1.25"). This is done by a compile time transformation. Immediately below the aircraft the ground speed and altitude are output side by side. The ground speed is displayed to the left, while the altitude is shown to the right in the format XX.X which represents thousands of feet.

The yellow curved trend vector also extends from the tip of the aircraft. It is only displayed if ground speed is greater than 100 feet per second, and the scale is less than 80 nautical miles (ground speed has been scaled by a factor of 32). The trend vector is a predictor of where the aircraft will be in 30, 60, and 90 seconds given the current ground speed and cross track acceleration. Each 30, 60, and 90 second segment consists of five dashes, the first one of which is blank. The second endpoint of each dash ends a six second interval. The host computer determines if the cross track acceleration is great enough to draw a curved trend vector. The host will send to this routine a variable indicating whether a straight or curved trend vector should be drawn.

Since the trend vector is a predictor of position, changes in scale have to be accounted for. As the scale is increased from a low range to a higher one, the trend vector becomes smaller. The trend vector is masked by the compass arc, although the aircraft would probably have to be travelling at a high-speed on a low map scale for it to extend that far.

Related to the curved trend vector are two track lines, termed the "extended" trend vector and the "offset" vector, that can be displayed to show projected ground track at rollout. They are selected by manually setting the variable TKBITS in the MicroVax. Bit zero controls the offset vector and bit one is used for the extended trend vector. The "extended" trend vector is a solid yellow line that literally extends off the curved trend vector to show the time along the trend vector before desired track is reached. The "offset" vector is a dashed cyan line drawn from a point ahead of the aircraft to show the distance the airplane will have to fly to reach desired track.

The altitude range arc represents the distance the aircraft will have to travel to capture the selected altitude given the current flight path angle and ground speed. It is colored yellow also. The arc is only displayed in map mode, and only if a value for its range has been supplied by the host computer. The input value sent from the host is in nautical miles (scaled by a factor of 128). It is converted to CGS units depending upon the map scale. Like the trend vector, as the scale increases from low to high the distance on the screen from the aircraft to the arc will get smaller. The arc will also naturally approach the tip of the aircraft as the plane slowly captures the selected altitude.

If the distance is too great to be represented on the screen, a dashed arc (instead of the normal solid arc) appears directly below the compass. This could be the case, for example, if the aircraft would take a long time to reach the selected altitude - or if it never would reach it. Once the altitude range arc reaches the tip of the aircraft it disappears.

The timebox and bubbles are selected via a bezel switch on the DU. Their symbols are colored magenta. The timebox indicates where the aircraft should be positioned based on time information found in the guidance buffer. The timebox character is a rectangular box whose dimensions are approximately the width and length of the airplane. In 4D guidance mode the aircraft should ideally fly within the box. Displayed along with the box are three small square characters (bubbles) which represent where the timebox will be in 30, 60, and 90 seconds given current guidance buffer data. The host computer determines the north and east delta positions of the timebox and three bubbles based on the guidance buffer, and inputs them to the DEU software. The host also computes a heading for each of the four symbols which will be the same as the heading of the path at the point on which they lie.

Even if the timebox bezel is selected, the timebox position variables must be non-zero for the symbol to be displayed. The host computer will zero these two values when it realizes 4D conditions do not exist. The position inputs are scaled feet values and must be converted into screen coordinates based on the current map scale. The timebox and bubbles are masked by the compass arc.

This procedure also outputs an aircraft symbol at the MLS computed position if MLS is valid, but not in MLS mode. The north and east positions are input as delta values from the real airplane position in scaled nautical mile units. These inputs must be converted to screen coordinates. The MLS aircraft will not be shown if it is outside a 10 nautical mile range. The symbol, if shown, is dashed and will be white if its position is within 50 feet altitude and 500 feet cross track of where the real airplane should be, and yellow otherwise.

```
GLOBAL REFERENCES:

VARIABLES
  AC_NORM ALT_ACTUAL ALT_ARC_SEL ALT_RANGE* BOX_HDG
  BOX_X BOX_Y BUB_COORDS FCB GE_MAP_AC GND_SPD GS_FPS
  LINE_SELECT MAG_TRK MLS_AP_COLOR MLS_MODE MLS_VALID
  MLS_X MLS_Y ND_CSERNG ND_MODE OFF_VECTOR* SEL_TRK
  TANG_TIME TIME_BOX_SEL TK_LINES* XTK_GS_RATIO

PROCEDURES
  DRAW_ABS_XY_BLANK DRAW_ABS_XY_VIS DRAW_REL_RA_VIS
  DRAW_REL_XY_BLANK END_OBJECT_BLOCK ENQ_OBJECT_BLOCK
  IPVGSUB_CSTRING MOVE_ABS_XY_BLANK NEW_OBJECT_BLOCK
  ROTANG SET_DASH U_BNRBCD U_DIV U_INSERT_DEC_PT U_MULT
  VID_PRI
```

MODULE NAME: MAP_KILL
FILE NAME: MAPKILL.PLM
CALL RATE: 0.8 SECONDS
CALLED BY: MAP_EXC

PURPOSE: To clear the screen of invalid objects and display a "MAPFAIL" message when the format becomes invalid.

REMARKS: The "MAPFAIL" message is displayed and all graphic objects except the basic outline of the map are blanked out if FMT_VLD is false. The FMT_VLD flag is cleared when the display processor is not receiving data from the host or an update to the background buffer has not been received within fifteen seconds. "MAPFAIL" should not be associated with the condition of the DU not receiving data, in which case a red "X" would be displayed.

The FMT_VLD flag must remain false for two consecutive frames in order for MAP_KILL to be called. If FMT_VLD was false instantaneously, i.e. for only one frame, and MAP_KILL was called, the "MAPFAIL" message would just flash momentarily on the display, too quickly to be observed.

Because the "MAPFAIL" display is constant, it is sent at the slow rate of once every 16 frames or 0.8 seconds.

GLOBAL REFERENCES:

VARIABLES
FCB GE_RANGE_DGTS GR_BRNG_DGTS_NIL GR_CMPSS_DGTS_NIL GR_MAP_AC_NIL GR_MESSAGE_NIL GR_NORTH_PTR_NIL GR_PATH_NIL GR_PLAN_AC_NIL GR_RANGE_DGTS GR_SYMBOL_NIL GR_VERT_DEV_NIL ND_BCKVALID* ND_MAP_FAIL TR_RASTER_NIL

PROCEDURES
ENQ_OBJECT_BLOCK IPVGSUB_CSTRING NEW_OBJECT_BLOCK TRIM_OBJECT_BLOCK VID_PRI

MODULE NAME: MOVE_BUGS
FILE NAME: COMPASS.PLM
CALL RATE: 0.2 SECONDS
CALLED BY: XFRM_COMPASS

PURPOSE: To create the graphics objects which represent selected track and present heading and add them to the compass digits object.

REMARKS: The selected track graphics object is an "M" shaped object which is amber in color and is displayed along the outside of the track tape whenever selected track and true track are valid. Also, when track angle is preselected or selected track differs from actual track by more than one degree, the selected track bug will be connected to the apex of the airplane chevron with an amber dashed line. If either the "extended" trend vector or the "offset" vector has been selected, as indicated by the global variable TK_LINES, the amber dashed line is suppressed.

The present heading bug consists of a small yellow triangle which is displayed on the inside of the track tape whenever heading and track are valid and differ by less than 45 degrees.

GLOBAL REFERENCES:

VARIABLES
LINE_SELECT ND_BEARING ND_VAL_BUGS SEL_TRK TK_LINES
TRU_HDG

PROCEDURES
IPVGSUB_CSTRING ROTANG ROTANG_GL SET_DASH VID_PRI

MODULE NAME: PLAN_AC_POS
FILE NAME: MAP_MGR.PLM
CALL RATE: .05 SECONDS
CALLED BY: MAP_EXC

PURPOSE: To display the airplane chevron when in PLAN mode.

REMARKS: PLAN_AC_POS is called when PLAN mode is selected via the bezels. In plan mode the airplane's position is not at the center of the screen but at some point relative to the center. It should be noted that screen center is actually the modified position (0", -1.25"). The variables AP_E, AP_N provide the delta north and east positions. The symbol is also rotated according to the bearing. The airplane chevron is displayed in yellow..

GLOBAL REFERENCES:

VARIABLES
AP_E AP_N FCB GE_PLAN_AC ND_BEARING

PROCEDURES
END_OBJECT_BLOCK ENQ_OBJECT_BLOCK MOVE_ABS_XY_BLANK
NEW_OBJECT_BLOCK ROTANG ROTANG_GL VGSUB_CSTRING
VID_PRI

MODULE NAME:	RANGE_EXEC
FILE NAME:	MAP_MGR.PLM
CALL RATE:	0.2 SECONDS
CALLED BY:	MAP_EXC

PURPOSE: To display the current map scale on the range circle.

REMARKS: The set of map ranges that are available to be represented on the display are 2, 5, 10, 20, 40, and 80 nautical miles. The numeric value represents the radius of the range circle. The current range value that is being used will be displayed on the left side of the circle in small cyan characters.

GLOBAL REFERENCES:

VARIABLES
```
  FCB GE_RANGE_DGTS GR_RANGE_DGTS ND_CSERNG
```

PROCEDURES
```
  END_OBJECT_BLOCK ENQ_OBJECT_BLOCK IPVGSUB_CSTRING
  NEW_OBJECT_BLOCK VID_PRI
```

MODULE NAME: STATIC_MASK
FILE NAME: MAP_MASK.PLM
CALL RATE: COLD START ONLY
CALLED BY: MAP_EXC

PURPOSE: To define bounded regions in the low resolution bit-plane for the masking of stroke drawn graphics objects.

REMARKS: When performing masking with the Core Graphics System, two bit-planes are used to define regions of the display screen. A bit-plane can be thought of as a rectangular grid filled with evenly spaced bits, known as pixels. Regions can be denoted by having some bits "ON" and some "OFF" in a bit-plane. When two bit-planes are involved more areas can be bounded by the intersection of regions defined in each bit-plane. See the "Core Graphics System" manual for information on stroke masking.

This subroutine sets and clears bits in the "Low-res" bit-plane by making calls to CGS subroutines. Since the areas defined in this bit-plane are static this subroutine needs only to be called at system reset time.

The following figure shows a rectangular area representing the display screen. The shaded region denotes where the pixels (bits) are set to "ON". The coordinates relative to screen center are given for boundary points. The regions defined by this bit-plane are used to differentiate between the area below and above the compass ticks on the display.

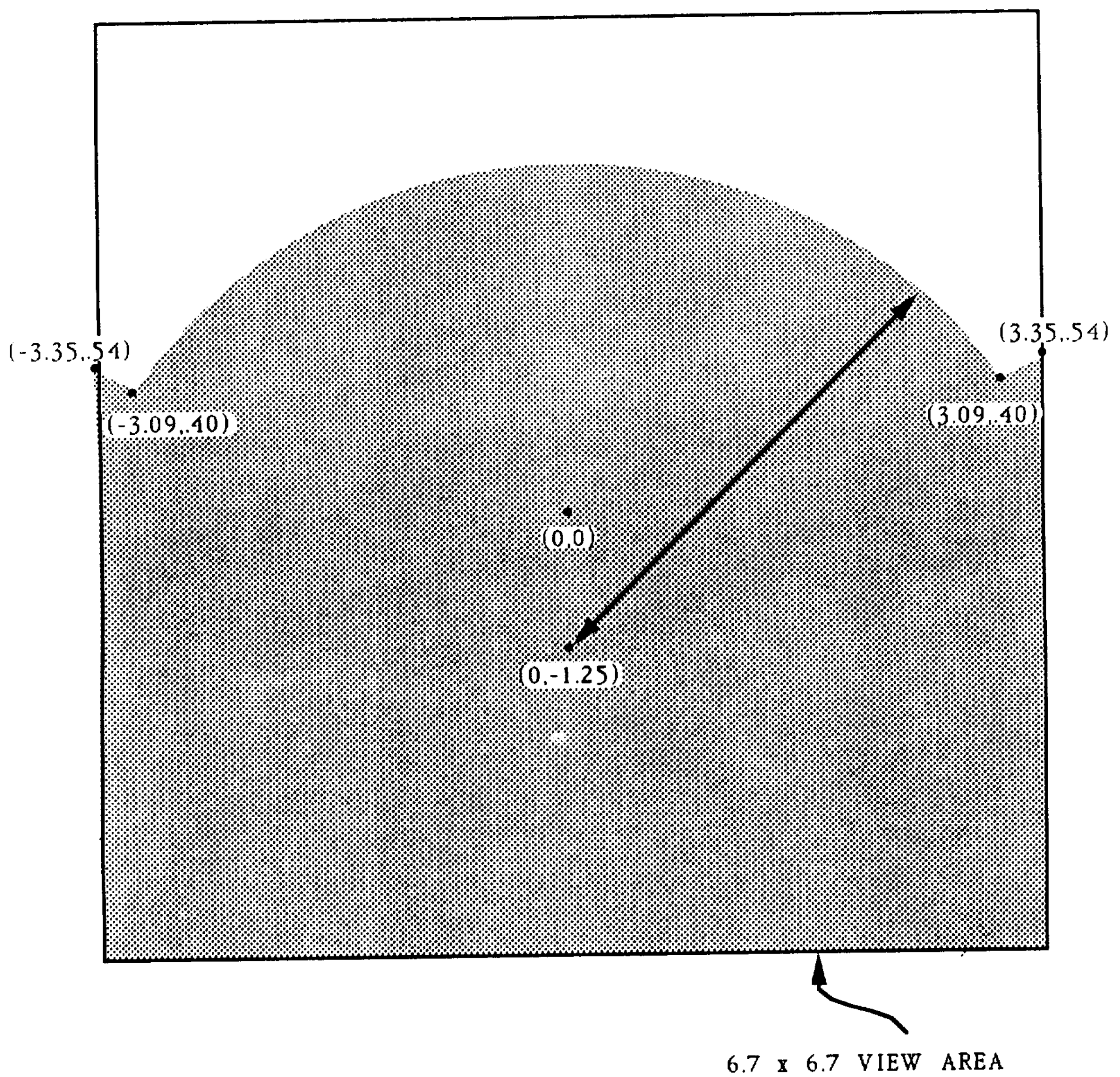
GLOBAL REFERENCES:

VARIABLES
GE_ALRM GR_ALRM

PROCEDURES
DRAW_ABS_RA DRAW_ABS_XY END_MASK_BLOCK NEW_MASK_BLOCK
SET_MASK_CHANNEL SET_MASK_PRIORITY

LOW RESOLUTION BIT PLANE

(SHADED REGION = "ON" BITS)



NOTES:

- ALL COORDINATES ARE IN INCHES
- ARC RADIUS IS 3.5 INCHES

-figure 6.9-

MODULE NAME: STATIC_WX
FILE NAME: MASKING.PLM
CALL RATE: COLD START ONLY
CALLED BY: EXECUTIVE

PURPOSE: To create the two static raster objects used in weather radar display.

REMARKS: This module calls run-time CGS subroutines to generate the data blocks for the raster mask and line pair objects. Note that these are both static objects which are normally created at compilation, however no CGS macros exist for these operations. These objects are needed as part of the raster generation package. The CGS manual should be consulted for a complete understanding of the techniques.
In general the masking object forms the visible boundary for raster color fill on the raster background patch. The object generated here (GR_WXM) creates a rectangular weather radar window for the bottom and both sides. The top of the view area however is an arc.

GLOBAL REFERENCES:

VARIABLES
CLK_HIGH DLAY FINE GE_RASLIN1 GE_WXM GR_RASLIN1 GR_WXM SETL_BOT SETL_TOP XMOV X_DELTA X_GAP X_SCALE YMOV Y_GAP Y_SCALE

PROCEDURES
DRAW_ABS_RA DRAW_REL_XY END_MASK_BLOCK END_OBJECT_BLOCK MOVE_ABS_XY MOVE_REL_XY NEW_MASK_BLOCK NEW_OBJECT_BLOCK RASTER_LINEPAIR_2 SET_MASK_CHANNEL SET_MASK_PRIORITY U_MULT

MODULE NAME:	VERT_DEV_SCALE
FILE NAME:	MAP_MGR.PLM
CALL RATE:	0.2_SECONDS
CALLED BY:	MAP_EXC

PURPOSE: To create and enqueue the graphics object for the vertical deviation scale.

REMARKS: When the vertical deviation scale is requested, VRT_ENG not zero, the vertical deviation scale will be displayed in the lower right corner of the NAV display. The variable VRT_ENG is set by the module INPUT_DATA from the I/O RAM index supplied to the PFD as the current engaged vertical guidance mode. The following chart shows how the I/O RAM index is used by the NAV format.

PFD index	PFD mode	VRT_ENG	NAV mode
0	none	0	none
1	FPA	0	none
2	ALT	2	ALT
3	VRT	3	VRT
4	ILS	4	G/S
5	MLS	4	G/S
6	GPS	4	G/S
7	FLARE	0	none

The vertical deviation scale object consists of a stationary scale, a vertical speed arrow, a reference altitude indicator, and lines detailing the scale's mask boundaries.

The stationary scale consists of two white dots positioned on either side of the origin mark, which is a small white rectangle. The dots are positioned so that one dot is the equivalent of 500 feet or .35 degrees of vertical deviation depending upon the current pointer mode.

The reference altitude indicator is a box with a pointer on the left side and it is green in color. Text labels are displayed inside the box and vary on the basis of an index, VRT_ENG, described above. The variables which dictate the position of the indicator are the same ones that position the reference altitude pointer on the Primary Flight Display so they must be scaled properly to be meaningful on the smaller NAV format scale.

If VRT_ENG is "2", the reference altitude value will be displayed inside the box in the form XX.X, which represents thousands of feet. The reference altitude indicator will be positioned in relation to the difference between reference altitude and actual altitude. In this mode, the origin mark will represent actual altitude.

If VRT_ENG equal to "3", "VRT" will be displayed in the box, indicating that the airplane is in vertical guidance mode. The pointer is positioned in terms of the number of feet of vertical deviation from the active flight plan. The origin indicates zero feet of deviation in this mode.

If VRT_ENG is a "4", "G/S" will be displayed in the box, indicating that the airplane is in either ILS, MLS, or GPS glideslope mode. The reference altitude indicator is positioned on the basis of glideslope error with the origin indicating zero degrees of error.

The vertical speed arrow is yellow and is displayed on the left side of the stationary scale. It emerges from the origin mark of the scale. Its length is obtained from the variable HDOT_SEGMENT, which is sent by the host computer to both the PFD and NAV. The NAV reduces the length by a factor of .8765 because of the smaller display area.

The vertical deviation scale mask area is bounded by two green lines. Any objects not related to the scale but positioned inside the area when the scale is being displayed are masked out (see DYNAMIC_MASK).

GLOBAL REFERENCES:

VARIABLES
  ALT_ACTUAL FCB GE_VERT_DEV GR_VERT_DEV_NIL GS_REF
  HDOT_SEGMENT REF_ALT_VALUE VRT_ENG VRT_REF

PROCEDURES
  DRAW_REL_XY_BLANK DRAW_REL_XY_VIS END_OBJECT_BLOCK
  ENQ_OBJECT_BLOCK IPVGSUB_CSTRING MOVE_ABS_XY_BLANK
  NEW_OBJECT_BLOCK U_BNRBCD U_INSERT_DEC_PT U_MULT
  VID_PRI

MODULE NAME:	WIND_EXEC
FILE NAME:	MAP_MGR.PLM
CALL RATE:	0.2 SECONDS
CALLED BY:	MAP_EXC

PURPOSE: To display the graphics object which indicates wind direction and velocity.

REMARKS: The wind direction will be represented by a small yellow arrow in the upper left corner of the NAV display. The wind velocity will be indicated just under the the arrow, also in yellow. In order for the wind graphics to be displayed, the following conditions must be satisfied. Wind, track and heading must all be valid and the wind speed must be greater than 4 knots. WIND_EXEC is called only in MAP mode.

GLOBAL REFERENCES:

VARIABLES
FCB GE_WIND GR_WIND_NIL ND_CMPSBRNG ND_VAL_BUGS
WIND_DIR WIND_SPD

PROCEDURES
END_OBJECT_BLOCK ENQ_OBJECT_BLOCK IPVGSUB_CSTRING
MOVE_ABS_XY_BLANK NEW_OBJECT_BLOCK ROTANG ROTANG_GL
U_BNRBCD VGSUB_CSTRING VID_PRI

MODULE NAME: WX_MGR
FILE NAME: MASKING.PLM
CALL RATE: 0.8 SECONDS
CALLED BY: EXECUTIVE

PURPOSE: To enqueue specific raster objects which either turn weather radar on or off.

REMARKS: This module determines if the weather radar data should be displayed. One necessary condition is the proper weather radar status from DP23 (Section 3.3.3), which is the current weather radar range. Also the weather radar bezel button must have been selected and the current map orientation must be "track-up". When the weather radar display is valid all the static weather radar objects are enqueued. On the first frame of the valid condition the DU weather radar bus must be enabled. The weather radar channel is enabled by setting the appropriate bit of the ID word sent in the format initialization call (FCB_INIT). Since this call is made at cold start only, the cold start flag is set to reenable the call. When weather radar is not valid the "nil" transform object is enqueued to remove the weather radar display. If the weather radar button was selected, the "WEATHER RADAR INVALID" message is placed on the display screen. Note that on the first pass of invalid display the weather radar channel is turned off in the DU by clearing the ID bits and reenabling the cold start condition.

GLOBAL REFERENCES:

VARIABLES
  COLD_START* FCB GR_RASLIN1 GR_RASMAIN GR_RASSUB1
  GR_WXDELAY GR_WXM GR_WX_NIL GR_WX_TEXT ND_CSERNG
  ND_LABEL* ND_MODE TR_RASTER TR_RASTER_NIL WX_CHANNEL
  WX_RANGE WX_SEL

PROCEDURES
  ENQ_OBJECT_BLOCK

MODULE NAME: XFRM_COMPASS
FILE NAME: COMPASS.PLM
CALL RATE: .05 SECONDS
CALLED BY: MAP_EXC

PURPOSE: To create the compass arc transform, call the code that builds the compass graphics, and generate the graphic object for the bearing digits and text.

REMARKS: XFRM_COMPASS is in charge of calling the procedure COMP_ARC which contains the code to build and display the compass arc, also known as the track tape. COMP_ARC is normally called only one of every four frames. However if the turn rate of the airplane is so high that the difference between current bearing and the nominal bearing is greater than 30 degrees, this code will execute early because a new portion of the track tape will have to be displayed.

The transform for the compass arc is generated each frame. The transform rotates the compass such that it is centered about the current bearing. The rotation angle will have a small factor added to it so that the two-digit ASCII representation of the bearing that is output every 30 degrees will be centered around the tick for that bearing. The center of rotation for the compass arc is the fixed airplane position of (0, -1.25").

XFRM_COMPASS also creates the bearing digits object. This consists of the bearing value displayed within the box at the top of the compass arc, and the text "TRK" and "MAG". The bearing value is displayed within the range 0 - 360 degrees, and is output in large yellow characters. The text appears in medium sized characters.

The bearing digits object contains some code that logically does not belong here, but for lack of any other good place to put it, it was included in this object. This code displays the text "MLS ON" or "MLS VLD" below the text lines in the upper right hand corner of the display if either MLS mode is true or MLS is valid. The text is displayed in medium sized characters and is colored cyan.

GLOBAL REFERENCES:

VARIABLES
CMPSRES* DP_FRAME FCB GE_BRNG_DGTS GPS_MODE LINK_CMD
MLS_MODE MLS_VALID ND_CMPSBRNG OLDBRNG

PROCEDURES
COMP_ARC END_OBJECT_BLOCK ENQ_OBJECT_BLOCK
IPVGSUB_CSTRING MAKE_XFORM_ONE NEW_OBJECT_BLOCK
U_BNRBCD U_MULT VID_PRI

Section 6.6 MEMORY ALLOCATION FILES

Most of the memory allocation modules reserve memory locations in RAM for various system and applications variables. VAR_RAM.PLM, GLBL_RAM.PLM, and IO_RAM.PLM allocate memory between the address ranges of 800H and 7FFFH in the display processor memory. IO_RAM.ASM reserves memory specifically between the ranges of 4000H and 7FFFH. INT_TBL.ASM allocates memory from 0000H to 03FFH for the interrupt vectors and TABLES.ASM reserves memory locations within the range C000H to FFFFH in EEPROM for data constants. Memory locations 0400H through 07FFH are reserved for the stack, therefore the stack pointer is initialized to 0800H.

FILE: GLBL_RAM.PLM

This module allocates memory locations in RAM for the system level variables used by the format executive. The types of variables declared globally in this module include system error flags, counters and checksum variables.

FILE: INT_TBL.ASM

This module reserves memory locations 0000H through 03FFH for the interrupt vectors. There are two words of data per interrupt. The first contains the segment to which control is passed if a given interrupt occurs and the other contains the offset within that segment. Because 3FFH words of memory are reserved, the possibility of defining 512 different interrupts exists. However, only a few are presently used, and two of them are initialized in this module.

FILE: IO_RAM.ASM

This module allocates memory for globally defined I/O variables. The variables defined here contain information from the I/O converter card as well as data sent on the internal I/O bus from other processors within the same DEU, mainly the DP4 processor. The DP4 makes available a 704 word buffer that originates at the host computer. Communications between the host computer and the three applications processors are performed by this buffer. The variables are used for interprocessor communication within a DEU and some also contain values of discretes such as bezels and pots. They are accessible by labels that are placed at offsets into I/O RAM. This is done so that the format can fetch data by name directly from I/O RAM as it is needed. The only exception is with packed discretes, which are fetched once by a processing routine. The routine then unpacks the discretes into boolean bytes which are local to the individual processor.

FILE: TABLES.ASM

TABLES.ASM contains fixed data for the NAV format. This data is accessed through global labels. The two data tables used by COMP_ARC to create the compass arc and digits, ND_CMPSS_TICK_PIE and ND_CMPSS_IP, are stored here. Also found in this file are the character codes and global labels for a one-character backspace, six-character backspace, and a line feed. These have been created as user defined characters mapped into the small character set through the translation tables, so that they can be imbedded in strings of small character text. TABLES.ASM also contains the text for the "MAPFAIL" message which is displayed whenever a new background buffer has not been received within fifteen seconds. Finally, ND_XMIT_DEFER, the transmission deferral list, is included here. It indicates what objects can be deferred to the next transmission if the high-speed transmission buffer fills up. In this case all bits are set which means all objects can be deferred.

FILE: VAR_RAM.PLM

This module allocates memory for applications software variables and runtime graphic object buffers. Many of the variables declared in this module serve as global parameters to processing routines.

Section 7.0 THE PRIMARY FLIGHT DISPLAY FORMAT

The PFD format shows the current aircraft attitude and provides other critical "aircraft state" information to the pilot. Refer to the PFD format drawing at the end of this section.

The most outstanding section of the PFD format is the rectangular area around the screen center that is topped by a 106 degree arc segment. This area is referred to as the PFD view window. Within the window a number of symbols appear that depict aircraft roll, pitch, yaw, actual and reference flight path angle, angle of attack, and track angle information. Three dimensional representations of the "TO" waypoint and the destination runway are displayed in the window along with a flare guidance cue, radar altitude, and alert messages.

Angular perspective in the window is provided by the pitch grid and horizon ticks. The pitch grid has a double solid line representing the horizon which separates the sky from the ground, along with parallel grid bars spaced in 5 degree increments. Along the horizon line, tick marks are spaced to show 10 degree steps of horizontal displacement. The other window symbology is interpreted against the grid and ticks to ascertain proper angular readings. The area from the horizon line to the top of the view window is raster filled in blue to easily distinguish the sky/ground boundary formed by the horizon line. At the top of the window along the arc is a roll scale which uses a triangular pointer to designate current aircraft roll angle. The roll angle read from the scale corresponds to the amount of rotation applied to the horizon line within the view window.

On either side of the view window are gray raster filled rectangular areas called the airspeed and altitude tapes. They have tick marks and numeric values which can slide vertically giving the appearance of a rolling measurement tape.

The airspeed tape, on the left side of the view window, has the current aircraft airspeed value in the blacked out area at the center of the tape. A blue, amber, or green pointer box may also appear at the appropriate spot on the tape representing the current airspeed selection from the pilot's mode control panel. When airspeed is changing an elongated arrow will grow from the tape center vertically along the outside of the tape ticks and point to the airspeed that will be reached in ten seconds at the current rate of acceleration or deceleration. Also along the same edge of the airspeed tape is a wedge marker that indicates the upper airspeed suggested for the current aircraft flap settings. Directly below the airspeed tape the selected airspeed value, that corresponds to the airspeed pointer box, is shown in either green or amber. The aircraft mach number is shown above the airspeed tape when the value exceeds 0.5.

On the right hand side of the view window is the altitude tape. Similar to the airspeed tape, the current airplane altitude is shown in the blacked out area at the tape center. Alongside the sliding altitude tape on the right is the vertical speed scale. A yellow arrow grows from the center indicating the rate of change in altitude in units of thousands of feet per minute. A blue, amber, or green pointer box may also appear on the altitude tape representing the selected altitude from the pilot's mode control panel. An amber or green triangular pointer which represents the vertical profile of the aircraft's flight plan may also appear along the altitude tape edge. The glideslope pointer and scale are shown just to the left of the altitude tape when selected. These appear as a set of deviation dots with a diamond shaped pointer. Included in the pointer are the letters "GP" or "GS" standing for glide path (MLS) or glideslope (ILS) respectively. The dots and pointer may be green or amber. Immediately below the altitude tape area is the barometric pressure setting from the pilot's control display unit (CDU). The selected decision height value appears above the altitude tape.

The horizontal deviation indicators and scales are presented below the PFD view window. Horizontal deviation from the aircraft's flight plan is shown by an amber or green box pointer, entitled "HOR", placed above a deviation scale. MLS azimuth or ILS localizer deviations are shown by a triangular pointer placed directly below the HOR deviation scale. The pointer indicates position relative to five deviation "dots" which appear along the bottom edge of the HOR scale. The pointer and dots may be shown in amber or green.

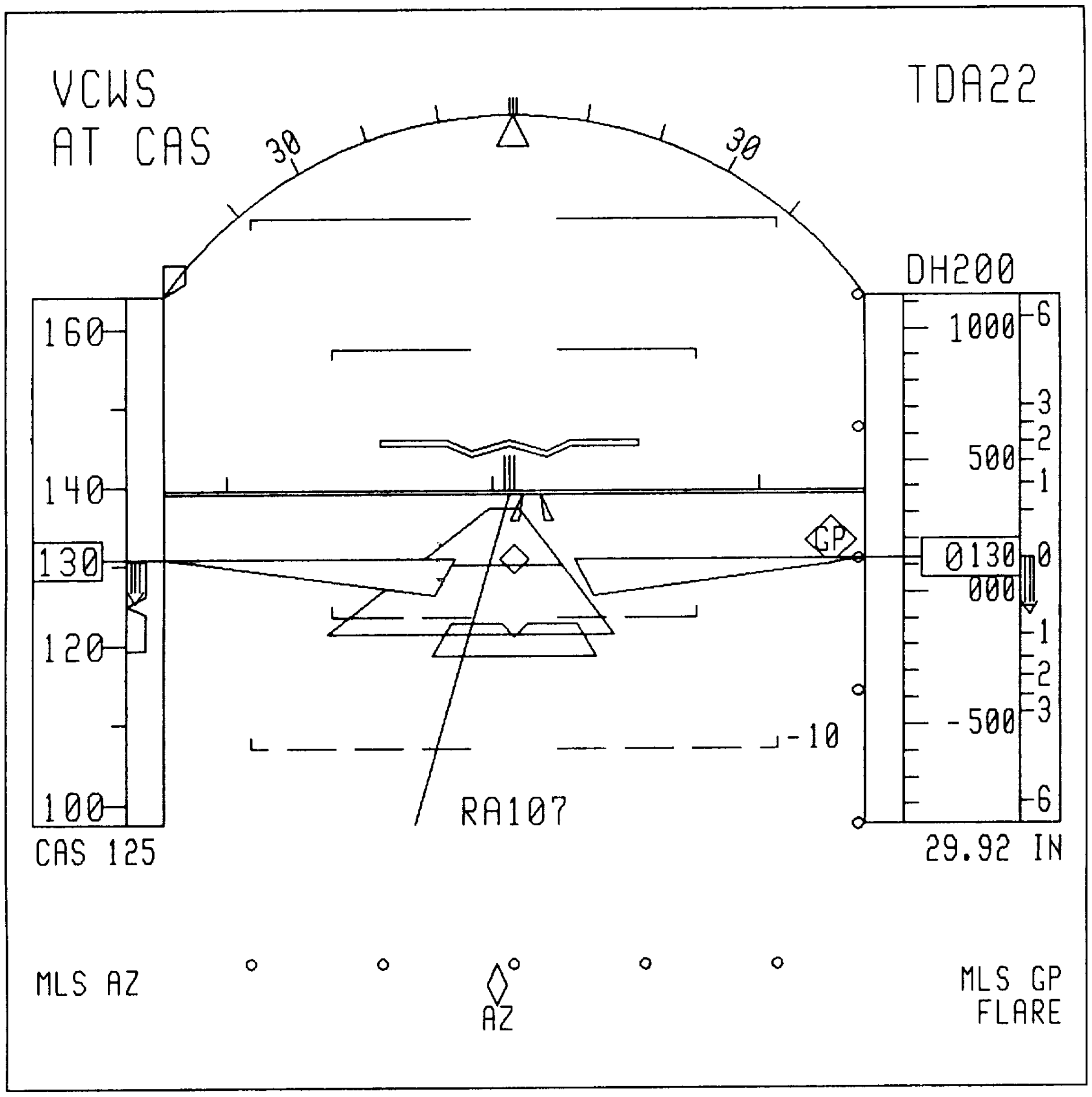
The corners of the PFD display contain information pertaining to the current control and guidance modes of the airplane. The upper left corner shows the current control and auto-throttle mode. The destination waypoint of the flight plan is shown in the upper right corner. The currently selected horizontal and vertical guidance modes are shown in the lower left and right corners respectively. Both armed and engaged modes are announced, color coded in amber (armed) and green (engaged).

The three upper right bezel panel buttons are active on the PFD format. They are used to select the waypoint STAR, perspective runway, and alert messages in order from the top.

Only the right hand potentiometer, which controls the value of the decision height, is used for the PFD format.

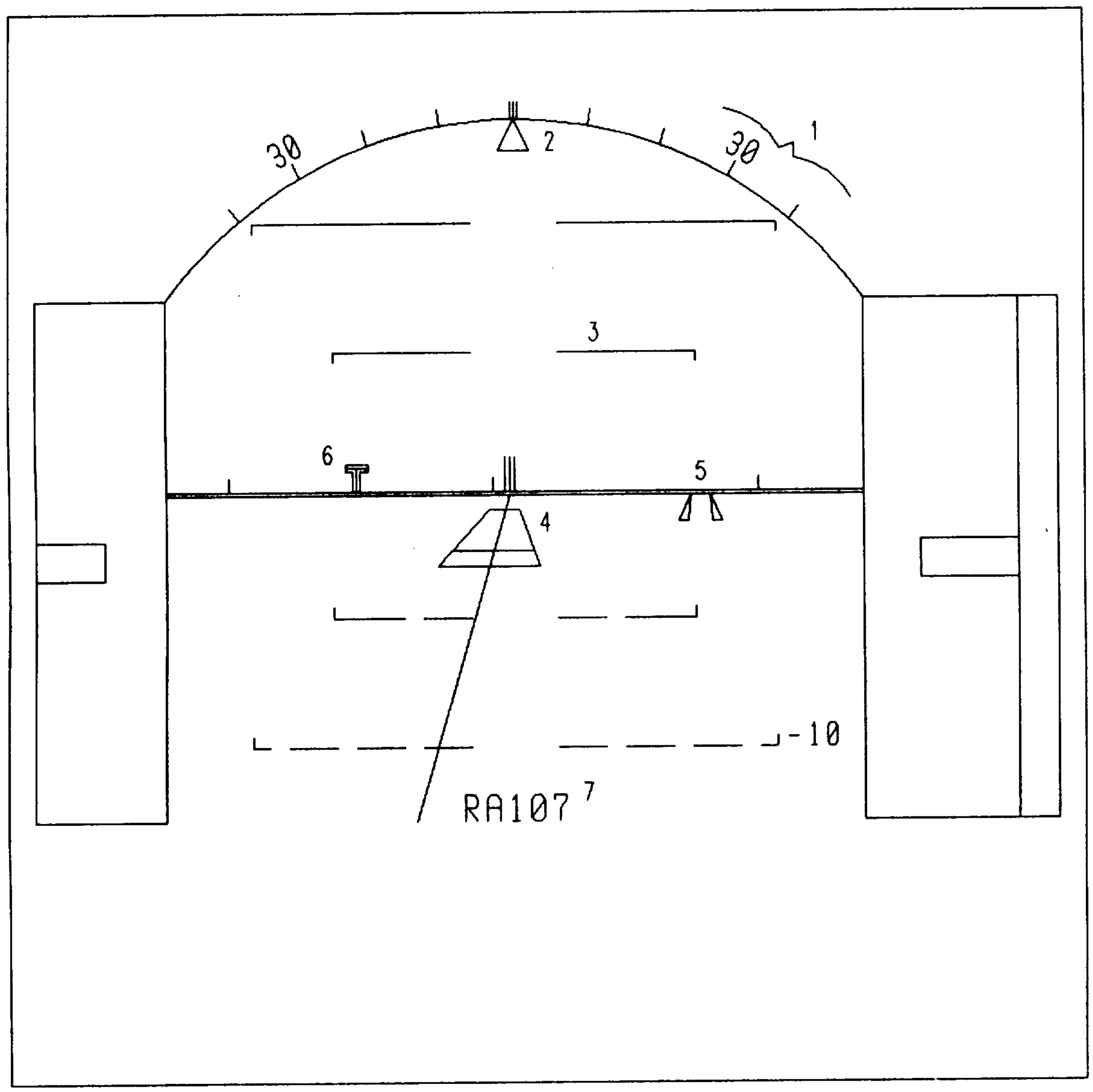
Section 7.1 INDIVIDUAL PFD SYMBOLS

This section has drawings of the PFD format followed by brief descriptions of individual symbology. The first drawing represents a typical PFD format. The remaining drawings break down the format into individual sections of the display with numbered items for identification.



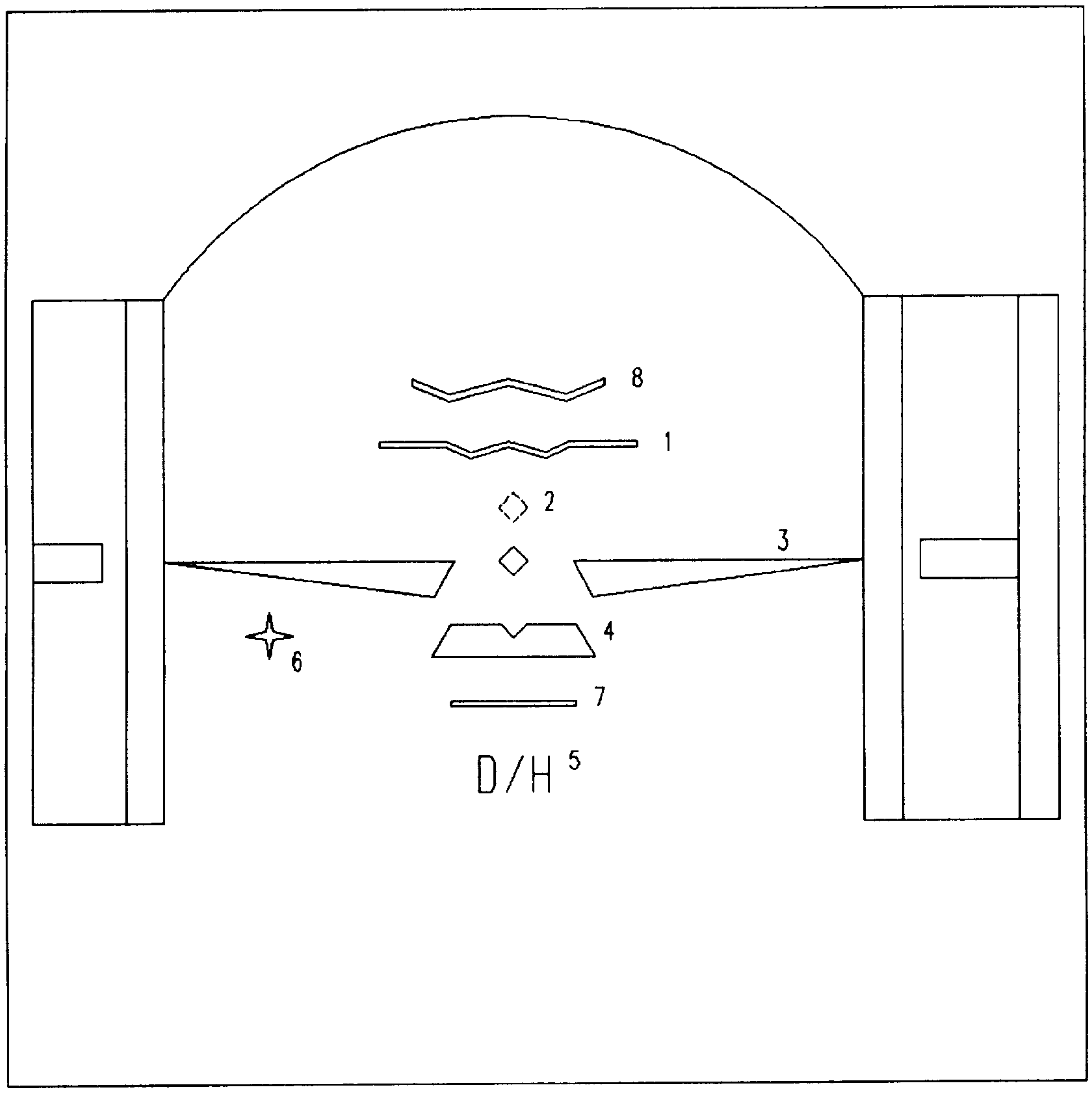
PRIMARY FLIGHT DISPLAY

-figure 7.1-



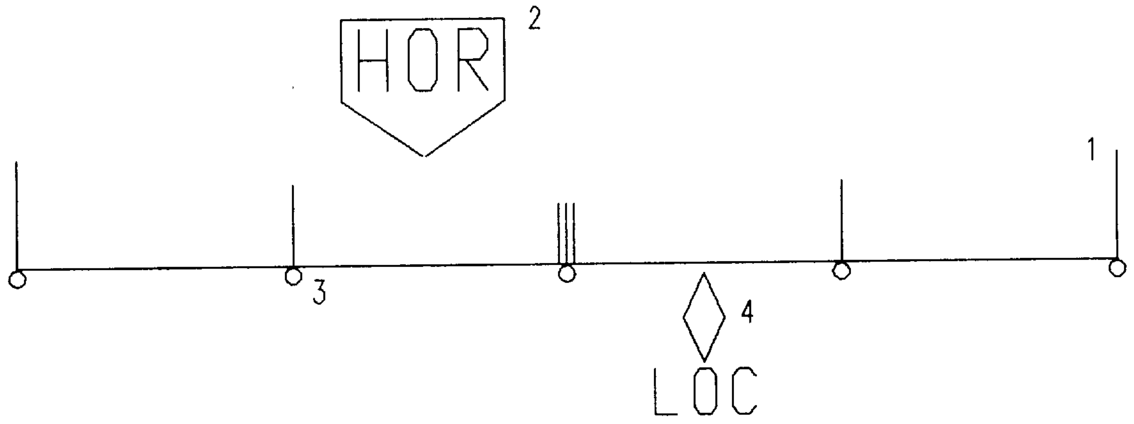
PFD WINDOW SYMBOLOGY (Part 1)

-figure 7.2-



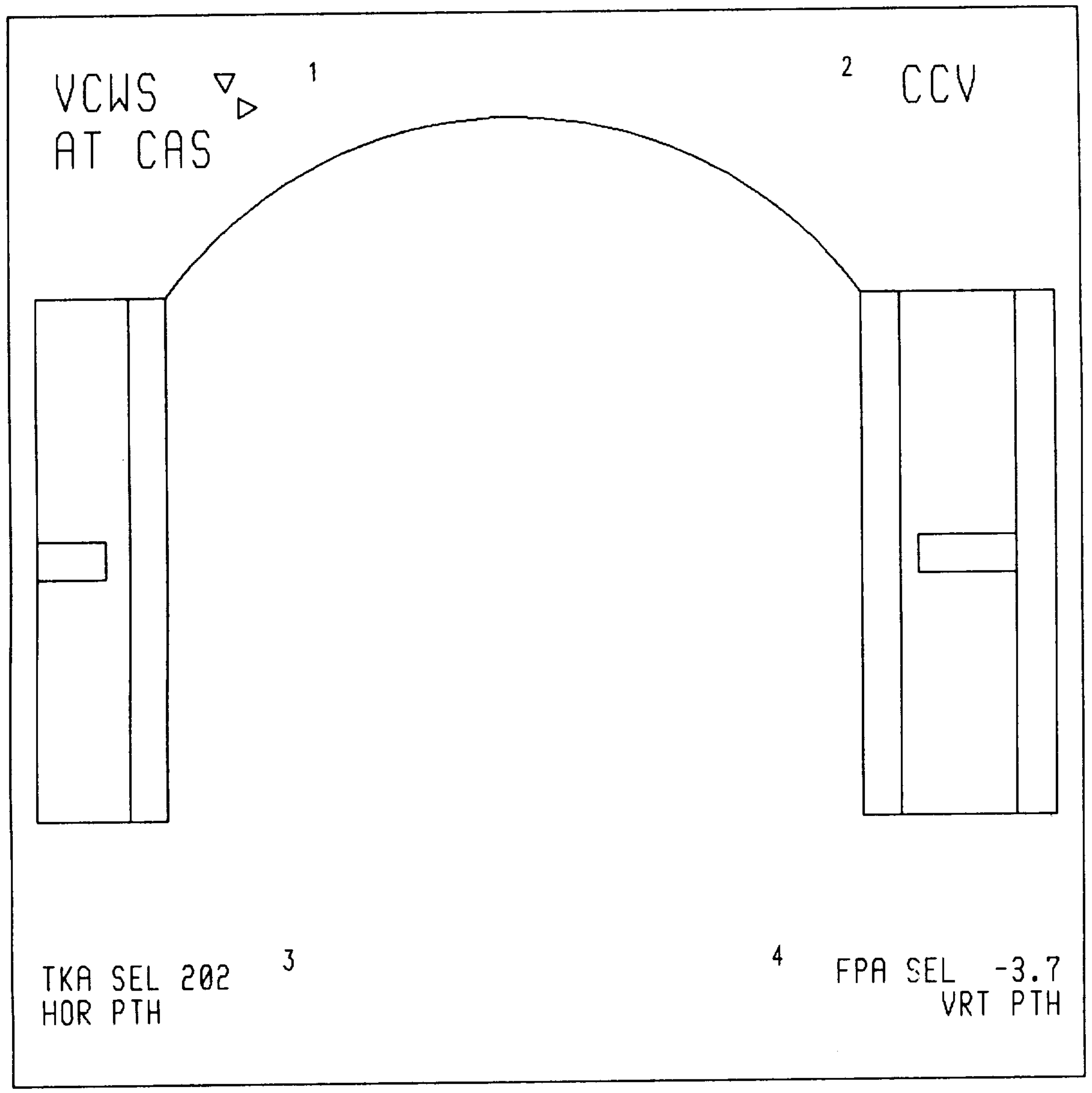
PFD WINDOW SYMBOLOGY (Part 2)

-figure 7.3-



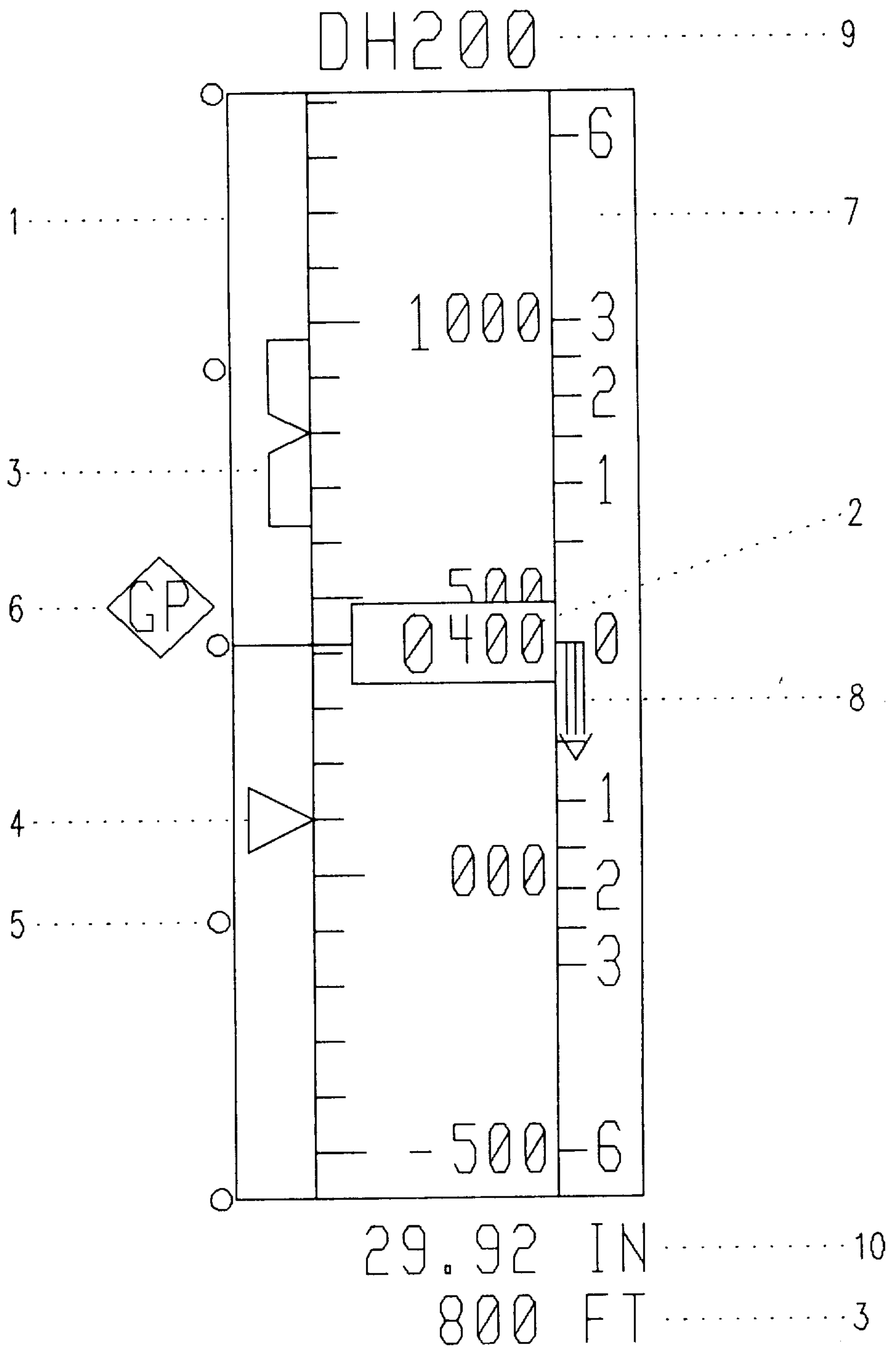
HORIZONTAL DEVIATION INDICATOR

-figure 7.4-



INFORMATIONAL READOUTS

-figure 7.5-

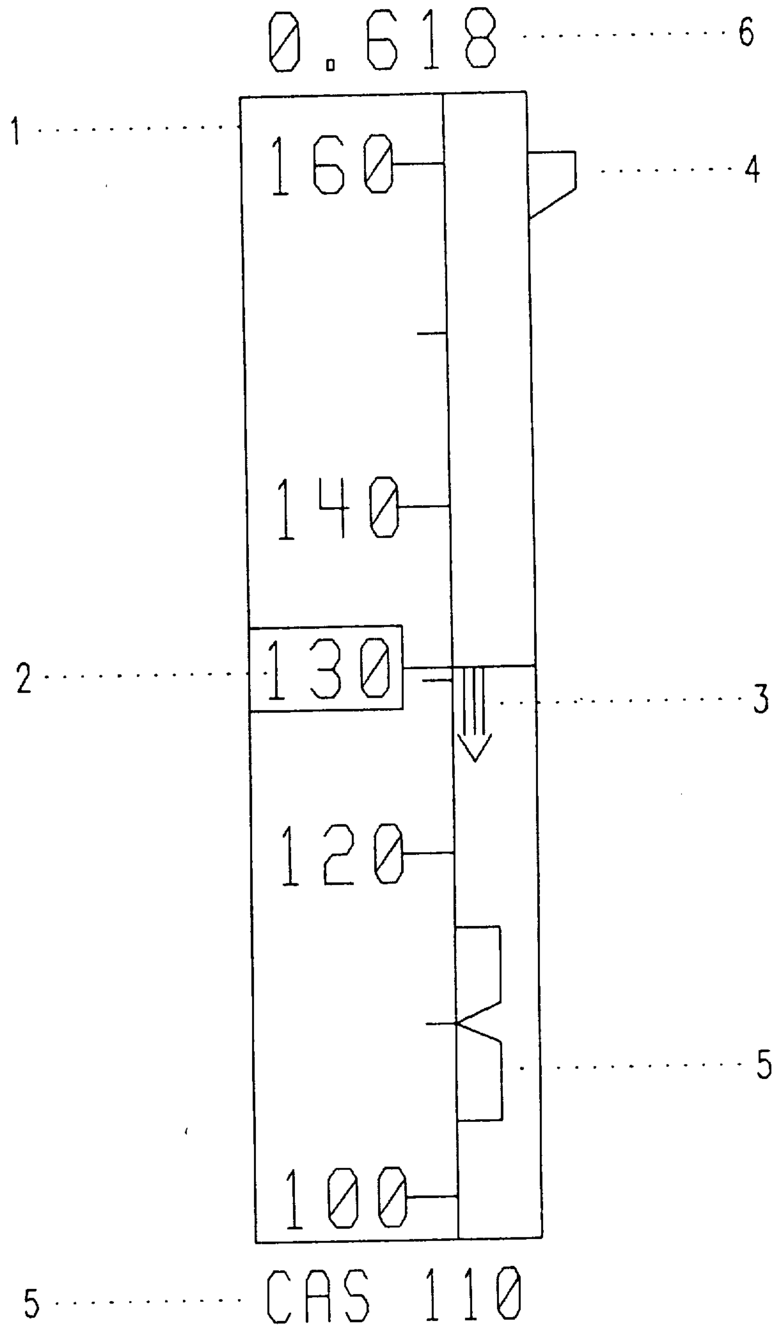


ALTITUDE TAPE

-figure 7.6-

PRECEDING PAGE BLANK NOT FILMED

178



AIRSPEED TAPE

-figure 7.7-

PFD WINDOW SYMBOLOGY
PART 1

1. ROLL SCALE

The roll scale is a 106.4 degree white arc, divided into ten degree segments. The current roll of the airplane is indicated by the position of the roll scale pointer along the arc. (See COMPILE.ASM Section 7.2)

2. ROLL POINTER

The cyan triangular pointer indicates either commanded or actual airplane roll relative to the roll scale. Commanded roll is shown in control wheel steering modes of guidance. The pointer moves right to indicate a right hand turn (or clockwise roll). (See ROLL_MGR, Section 7.3)

3. PITCH GRID

The pitch grid consists of a thick white horizon line and thinner white grid lines positioned parallel to the horizon line, both above and below it. The grid lines are spaced at intervals which represent 5-degree increments. The values represented by the gamma wedges and aircraft symbols are displayed relative to the pitch grid. The grid also rotates to reflect the current roll of the airplane. A series of small white ticks are displayed perpendicular to the horizon line at even ten degree intervals (ie. 240, 250, 260). The ticks are positioned along the horizon line based on aircraft track when in a velocity vector mode, or aircraft heading otherwise. The center of the screen, at the horizon, represents current track or heading while the ticks depict the proximity to to the nearest ten degree intervals. (See PITCH_MGR, Section 7.3).

4. PERSPECTIVE RUNWAY

A three dimensional representation of the runway will be displayed in green, when the runway bezel has been pressed on and the runway is within the view area. The runway centerline extends a distance representative of one nautical mile from the beginning of the runway. Another line drawn horizontally across the runway symbol

indicates the touchdown point and is drawn a distance of 1000 feet from the start of the runway. A symbol consisting of three vertical green lines which represents runway heading, is displayed at the point that the runway centerline intersects the horizon line.
(See RWY_MGR, Section 7.3).

5. TRACK BUG

The two small yellow triangles form a pointer which indicates track error of the aircraft. The track bug indicates the difference between the actual track of the aircraft and the desired track that was commanded through pilot trim switch inputs while in the "track hold" sub-mode of velocity control wheel steering. (See PITCH_MGR, Section 7.3)

6. TRACK "T"

This small green "T" shaped symbol rides above the pitch grid horizon line. It indicates either the track angle selected on the pilot's mode control panel or the flight plan bearing of the current leg of the path when no mode panel track selection has been made. When engaged in track select guidance, the numerical value corresponding to the "T" position is shown in the lower left corner of the PFD (see figure 7.4, informational readouts). Note that the "T" is colored blue when a mode panel preselection is made.
(see PITCH_MGR, Section 7.3)

7. RADAR ALTITUDE

The radar altitude value is displayed when the aircraft is less than 1000 feet from the ground. The color of the digits are cyan until decision height is reach, at which time they are drawn in amber. The value is shown to the nearest 10's of feet when above 500 feet. The resolution of the value is 5's of feets until 200 feet is reached, at which time each foot of radar altitude is shown.

PFD WINDOW SYMBOLOGY
PART 2

1. AIRCRAFT

The large white aircraft symbol remains stationary and indicates either the commanded or actual pitch of the airplane. It is displayed in a non-velocity vector mode. The smaller version of the symbol (shown in figure) is also white and is displayed in a velocity vector mode. The smaller aircraft symbol represents the actual pitch of the airplane and is positioned appropriately within the PFD window. (See AIRPLANE_MGR, Section 7.3)

2. GAMMA STANDOFF

The gamma standoff is red and represents the actual flight path angle of the airplane relative to the pitch grid. It is displayed when the actual and commanded flight path angles differ by more than 1.5 degrees, the control column is in detent, and the display is in a velocity vector mode. Because the display is in a velocity vector mode, the larger of the two gamma wedge symbols is also displayed and it represents the commanded flight path angle.
(See AIRPLANE_MGR, Section 7.3)

3. GAMMA WEDGES

The gamma wedges are raster filled white and can be displayed in one of two possible sizes. A large gamma wedge (shown in figure) is displayed when in a velocity vector mode and in this case the wedges represent commanded flight path angle. Otherwise, the small gamma wedges are displayed and they indicate actual flight path angle (quickened).
(See AIRPLANE_MGR, Section 7.3)

4. FLARE GUIDANCE CUE

The flare guidance cue is aqua and rises from the bottom of the screen toward the bore sight of the gamma wedge symbol. Once the cue reaches the bore sight, the pilot begins the flare maneuver and in doing so, keeps the cue and sight joined as a single unit. The flare guidance cue is displayed when the plane is being flown in VCWS and an altitude appropriate for flare procedures has been reached.
(See FLARE_MGR, Section 7.3)

5. ALERT/WARNING MESSAGES

There are thirteen possible messages which can be displayed if the message bezel has been pressed and conditions exist under which they should be displayed. The messages and their meanings are as follows:

"ALERT"	YELLOW	PLANE IS WITHIN 10 SECONDS OF NEXT WAYPOINT
"AOA"	RED	ANGLE OF ATTACK EXCEEDS THRESHOLD
"CLIMB"	YELLOW	NEXT PATH SEGMENT INDICATES ASCENT (displayed only during alert)
"DESCEND"	YELLOW	NEXT PATH SEGMENT INDICATES DESCENT (displayed only during alert)
"D/H"	YELLOW	DECISION HEIGHT HAS BEEN REACHED
"FLAPS"	YELLOW	SPEED NOT WITHIN SUGGESTED RANGE FOR CURRENT FLAP SETTING OR FLAPS AT 40 WITH WEIGHT IN EXCESS OF 95,000 LBS.
"FLAPS"	RED	STRUCTURAL FLAP LIMITS OF AIRCRAFT HAVE BEEN EXCEEDED OR FLAPS OUT OF SYNC BETWEEN AFT & FORWARD FLIGHT DECK.
"GEAR"	YELLOW	GEAR NEEDS TO BE LOWERED
"MM"	YELLOW	MIDDLE MARKER ENCOUNTERED
"OM"	YELLOW	OUTER MARKER ENCOUNTERED
"TURNL"	YELLOW	NEXT PATH SEGMENT INDICATES LEFT TURN (displayed only during alert)
"TURNR"	YELLOW	NEXT PATH SEGMENT INDICATES RIGHT TURN (displayed only during alert)
"LNK MSG"	YELLOW	DATA LINK MESSAGE RECEIVED.

(See MESSAGE_MGR, Section 7.3)

6. STAR

The STAR symbol is cyan and marks the three dimensional position of the destination waypoint within the PFD view window. When the airplane comes within 3500 feet of the waypoint, the STAR begins growing and continues until the symbol is 20 times its original size. The STAR symbol will be displayed when the STAR bezel has been pressed on and the airplane is in the proper orientation relative to the destination waypoint. (See STAR_MGR, Section 7.3)

7. FPA BAR

The flight path angle reference bar is a color coded bar placed parallel to the horizon line at the appropriate position relative to the pitch grid. It represents the reference flight path angle from either the glideslope, flight plan leg, or the mode control panel FPA selection. Its color can be green, amber, or blue to depict when a mode is engaged, armed, or preselected respectively. The guidance mode driving the symbol is shown in the mode information box at the lower right corner of the PFD. (see figure 7.4, Informational Readouts; Section 7.3, PITCH_MGR)

8. PITCH STANDOFF

The aircraft standoff symbol represents the actual pitch, while the larger aircraft symbol, which accompanies it, represents commanded pitch. When the commanded and actual pitch differ by at least 1.5 degrees, ACWS is engaged, and no control wheel inputs are being made the standoff symbol is displayed in red. The standoff indicator is a white "W" shaped symbol that fits over the large version of the aircraft symbol (not shown in figure 7.3). When the smaller aircraft symbol appears the pitch standoff never appears. (See AIRPLANE_MGR, Section 7.3)

HORIZONTAL DEVIATION INDICATOR

1. HORIZONTAL DEVIATION SCALE

This scale is shown on the PFD when the horizontal flight plan guidance is either armed or engaged. It is shown in green when engaged and amber when armed. The center of the grid represents zero deviation. Full scale indicates 7,500 feet of path error until the aircraft is within range of a destination runway localizer. At that time the range of the scale is adjusted to match the angular displacement of a localizer. The calibration is made so the range is +/-350 feet at the runway threshold, at which time the scale resolution remains constant.
(See HOR_LOC_MGR, Section 7.3)

2. HORIZONTAL DEVIATION POINTER

This pointer slides along the top of the horizontal deviation scale. The color matches the scale. See #1 above.

3. LOCALIZER/AZIMUTH SCALE

The five dots shown below the horizontal deviation scale form the localizer/azimuth scale. These dots will be shown as green or amber, depending on the engaged or armed status of the localizer/azimuth. The deviation represented by the dots is identical to the standard localizer deviation; two dots for 350 feet of error at the runway threshold. The angular deviation represented by each dot depends on the distance from the localizer antenna to the runway threshold.
(See HOR_LOC_MGR, Section 7.3)

4. LOCALIZER/AZIMUTH POINTER

This diamond shaped pointer slides along below the localizer/azimuth scale. The color matches that of the scale. The pointer is tagged with "LOC" for ILS and GPS approaches, while "AZ" is used for MLS landings.
(See item #3 above)

INFORMATIONAL READOUTS

1. CONTROL MODES

The control modes area of the PFD is found in the upper right corner of the display screen. All of the symbology associated with this area is shown in green. The current aircraft flight control mode is shown in the first text line. The various modes are shown below.

"ACWS"	aft flight deck attitude control wheel steering
"VCWS"	aft flight deck velocity control wheel steering
"AUTO"	aft flight deck auto-pilot
" "	aft flight deck disengaged

Directly below the flight control mode is the throttle mode. These modes are shown next.

"AT CAS"	auto-throttle; airspeed select
"AT 4D"	auto-throttle; time guidance
" "	auto- throttle disengaged

Pilot control wheel inputs are denoted by the two arrow heads which may appear directly to the right of the control modes. These depict when pitch up, pitch down, roll left, or roll right are being commanded. (See BORDER_MGR, Section 7.3)

2. DESTINATION WAYPOINT NAME

The name of the destination waypoint is shown in green on the upper right corner of the PFD. An active flight plan must exist for any name to be displayed, if none exists the area will remain blank. (See BORDER_MGR, Section 7.3)

3. HORIZONTAL GUIDANCE MODE

The area in the lower left corner of the PFD is used for horizontal guidance modes. Two lines of text may appear in this area. The upper line, shown in green, will display the engaged horizontal guidance mode. The armed horizontal guidance mode is shown in amber letters on the bottom line. The various mode labels are shown below.

"TKA SEL ---"	track select with chosen value
"HOR PTH"	horizontal path
"ILS LOC"	ILS localizer
"MLS AZ"	MLS azimuth
"GPS LOC"	GPS simulated localizer
" "	no horizontal guidance mode armed/engaged

(See BORDER_MGR, Section 7.3)

4. VERTICAL GUIDANCE MODE

The area in the lower right corner of the PFD is used for vertical guidance modes. Two lines of text may appear in this area. The upper line, shown in green, will display the engaged vertical guidance mode. The armed vertical guidance mode is shown in amber letters on the bottom line. The various mode labels are shown below.

"FPA SEL ---"	flight path angle select with chosen value
"ALT SEL"	altitude hold
"VRT PTH"	vertical path
"ILS GS"	ILS glideslope
"MLS GP"	MLS glide-path
"GPS GS"	GPS simulated glideslope
"FLARE"	flare manuever
" "	no vertical guidance mode armed/engaged

(See BORDER_MGR, Section 7.3)

ALTITUDE TAPE SYMBOLS

1. ALTITUDE TAPE

The altitude tape displays approximately a 2000 foot range of altitudes, centered at the value which corresponds to the current altitude. The altitude values and tick marks of the tape are displayed in white against a raster filled gray background. (See ALT_BLOCKS_MGR and, ALTITUDE_MGR, Section 7.3)

2. ACTUAL ALTITUDE READOUT

The actual altitude readout consists of a rectangular cut-out which is centered vertically on the altitude tape. The cutout is filled in black and masks out everything else on the altitude tape. The actual altitude value is displayed in white and represents aircraft altitude to the nearest ten feet. (See ALTITUDE_MGR, Section 7.3)

3. SELECTED ALTITUDE POINTER AND READOUT

Altitude values selected on the pilot's mode control panel are displayed below the altitude tape. An "M" shaped pointer is also placed along the left side of the altitude tape at the position which corresponds to the selected altitude. The color of both the pointer and the readout match the color of the lighted altitude select button on the mode control panel. Preselected, armed, and engaged are shown as blue, amber, and green respectively. When no altitude value has been selected, both items are removed from the PFD. (See ALTITUDE_MGR, Section 7.3)

4. VERTICAL PATH POINTER

This triangular pointer is placed alongside the altitude tape at the altitude position corresponding to the active flight plan. The separation of this pointer from the tape center depicts the vertical deviation from the current leg of the active flight plan. The symbol is drawn in amber when vertical path guidance is armed, and is shown in green for vertical path guidance engaged. When vertical path guidance is not selected this symbol is removed from the PFD. (See ALTITUDE_MGR, Section 7.3)

5. GLIDESLOPE ERROR SCALE

The glideslope error scale consists of five dots located just left of the altitude tape. Each dot indicates a glideslope angle error of .35 degrees from the center position. The scale is drawn in either amber or green depending on the armed or engaged status of the glide-slope mode. When no glideslope mode has been selected, this symbology (along with the GS pointer) is removed from the PFD. (See ALTITUDE_MGR, Section 7.3)

6. GLIDESLOPE DEVIATION POINTER

This pointer box is placed to the left of the glideslope scale at the position corresponding to the aircraft's current deviation from the glideslope on a runway approach. The symbol's color matches that of the scale (See #5 above). The box is labeled "GS" for ILS and GPS approaches, while MLS landings are denoted by the "GP" label. (See ALTITUDE_MGR, Section 7.3)

7. VERTICAL SPEED SCALE

This thin white rectangular scale is appended onto the right side of the altitude tape. The single digit values shown by some of the scale tick marks stand for rate of change in altitude in 1000's of feet per minute. The tick mark spacing is exponential to provide greater resolution at the more common lower vertical speeds. (See COMPILE.ASM, Section 7.2)

8. VERTICAL SPEED ARROW

The current vertical speed of the aircraft is shown by the position of the arrow head on the vertical speed scale. The yellow arrow extends from the scale zero point to the appropriate scale position. (See ALTITUDE_MGR, Section 7.3)

9. DECISION HEIGHT INDICATOR

This indicator displays the altitude value at which the pilot must decide whether to land or go-around on an instrument landing. It reflects the value that he has selected via the right hand potentiometer of the pilot's PFD. It is placed above the altitude tape and is colored gray until the decision height is reached, at which time the color changes to amber.
(See BORDER_MGR, Section 7.3)

10. BAROMETRIC PRESSURE READOUT

This value is displayed in gray letters directly below the altitude tape. It reflects the altimeter setting entered by the flight crew on the CDU.
(See BORDER_MGR, Section 7.3)

AIRSPEED TAPE SYMBOLS

1. CAS TAPE

The calibrated airspeed tape displays a 66 knot range of airspeeds, and is centered at the value which corresponds to the aircraft's current airspeed. The airspeed values and tick marks of the tape are displayed in white against a raster filled gray background. (See CAS_BLOCKS_MGR and CAS_MGR, Section 7.3)

2. ACTUAL AIRSPEED READOUT

The actual airspeed readout consists of a rectangular cut-out which is centered vertically on the CAS tape. The cut-out is filled in black and masks out everything else on the CAS tape. The actual airspeed value is displayed in white and represents actual airspeed in knots. (See CAS_MGR, Section 7.3)

3. ACCELERATION SEGMENT ARROW

The acceleration arrow is yellow and its tip is positioned along the CAS tape at the speed the aircraft will reach in the next 10 seconds, given the current acceleration. (See CAS_MGR, Section 7.3)

4. AIRSPEED LIMIT WEDGE

The rose colored wedge marker is positioned along the CAS tape to delineate the upper airspeed suggested for the current flap settings. The yellow flap alert message is displayed when the current airspeed is above the airspeed limit wedge. (See CAS_MGR, Section 7.3)

5. REFERENCE AIRSPEED POINTER AND READOUT

The airspeed selected on the pilot's mode control panel is shown directly below the CAS tape. An "M" shaped pointer is also placed along the right edge of the CAS tape at the position corresponding to the selected airspeed. The color of both items is either blue, amber, or green which matches the pre-selected, armed, or engaged status of the airspeed select mode. If no airspeed has been selected the symbology is removed from the PFD.
(See CAS_MGR and BORDER_MGR, Section 7.3)

6. MACH READOUT

The Mach number is shown, in gray numbers, directly above the airspeed tape when its value is at least 0.5.
(See BORDER_MGR, Section 7.3)

Section 7.2 COMPILE TIME GRAPHICS FILES

The files covered in this section define core graphics data constants. They create system level data base tables, which define the format to the display units, and constant CGS objects for static graphic items. The blocks of data defined in these files will be loaded into the processor's EEPROM memory for data integrity. All the data described in this section resides on the two files, COMPILE.ASM and SYS_TABS.ASM.

FILE: COMPILE.ASM

COMPILE.ASM contains the hard coded data for the PFD format's compile time graphic objects (SYS_TABS.ASM contains the system level compile time objects). There are a large number of objects that can be displayed on the PFD. Certain information concerning these objects can be fixed at compile time such as the color, rotation angle, masks in effect, and PROM or user defined characters used to build the object. This fixed set of data is contained in COMPILE.ASM. Information about a graphic object that varies or is computed can be found in the file of run time modules called MANAGER.PLM.

The data contained in this module is divided into four categories; compile time objects, compile time transformations, nil graphic objects, and nil transforms. On the following page the objects are listed under their appropriate headings. Some graphic objects have more than one variant. This is true, for example, of the alert message graphic object where one of fourteen different messages can be shown depending on current aircraft conditions. Other graphic objects with variants are the gamma wedges, LOC/HOR pointer, aircraft, and the pitch and roll out-of-detent objects.

One other important set of data found in COMPILE.ASM is the binary data containing the moves and draws for the user defined PFD characters. These are special characters created in addition to the NASA PFD characters in PROM. They are split into two user character set objects due to the size of the binary data.

```
GRAPHIC OBJECTS
---------------
STAR
PFD skeleton
alert messages
roll pointer
pitch grid and horizon
flare cue
raster scan lines
raster patch
raster delay

TRANSFORMS
----------
raster patch

CHARACTER SET #1
----------------
track "T"
flare cue
pitch standoff
flight path standoff
roll tick
horizon line
roll pointer
gamma wedge (large)
alt/CAS select pointer
GS pointer
CAS tape tick
altitude tape tick
CAS tape outline
```

```
NIL TRANSFORMS
--------------
runway
STAR
CAS tape
altitude tape
roll pointer
pitch grid and horizon
flare cue

NIL GRAPHIC OBJECTS
-------------------
alert messages

CHARACTER SET #2
----------------
altitude tape outline
roll scale arc
track bug
arrow head (up)
arrow head (down)
pitch grid down (small)
pitch grid up (small)
pitch grid up (large)
pitch grid down (large)
dot
flight path angle bar
vertical speed scale tick
```

FILE: SYS_TABS.ASM

The system tables file SYS_TABS.ASM contains six basic compile time objects required by CGS to build a new format. These objects are enqueued by PFD_EXC at the slowest rate, and are as follows:

(1) TRANSLATION TABLE
(2) EVENT LIST
(3) OBJECT TABLE DEFINITION
(4) TOP OBJECT
(5) SCREEN CENTER TRANSFORMATION
(6) NEW FORMAT SETUP

(1) TRANSLATION TABLE. Translation tables are used to define character sets. They can combine characters from several sources to form a new set. For the PFD format three sets are defined; small, medium, and large. Note that the PFD special characters in DU PROM are mapped into the standard large alphanumeric character set. In PROM, several different sizes of standard alphanumeric characters are stored, from which the small, medium, and large PFD character sets map their entries. Also in PROM is a set of characters specially designed for the NASA PFD. The object table definition will associate a unique character set number to each set defined. For further information on translation tables consult Section 4.4 in the CGS manual.

(2) EVENT LIST. The event list is used to describe objects that are affected by dynamically changing transforms. Each entry in the event list represents a transform, and within that entry a list of objects that will be repositioned based on that particular transform.

For the PFD format a large number of graphic objects are moving symbols. This set includes the roll pointer, altitude and airspeed tapes, perspective runway, pitch grid, STAR guidance, and the flare cue. The object table definition will associate these graphic objects with a transform. The transforms also have entries in the OTD which will map them to the proper entry in the event list.

Some graphic objects have been associated with a transform in the OTD but are not listed in the event list. This would include compile time transforms since they do not dynamically change. For more information on event lists consult Section 3.5 in the CGS manual.

(3) OBJECT TABLE DEFINITION. The object table definition defines the PFD format to CGS, and the resources needed to support it. It consists of a series of object table entry macros, one for each object in the format. Entries are listed in ascending order of object numbers with the new format setup always as object number 1. Special macros are used to define the new format setup, OTD table, event list, translation tables, and the top object. Also found are entries for the screen center transformation and a user defined character set CS_SPC which contains symbols unique to the PFD.

Following these entries are listed the transformation and graphic objects for the PFD format. As mentioned before, graphic objects are associated with a transform (the screen center transformation if nothing else), and transforms are mapped to event list entries (except for compile time transforms which use the special code EVLO_NIL to indicate no action taken). Transforms must have lower object numbers than the graphic objects they affect. Due to the number of OTE (object table) entries, two OTD tables are used for the PFD format.

The buffering method required for the objects, static or pingpong, is also set up here. The pingpong method is used for objects that are constantly changing, however it requires more VG memory than the static method. To conserve memory, switching between static and pingpong is done whenever possible. For further details on building an OTD, and individual OTE macros, consult Section 3.4 in the CGS manual.

(4) TOP OBJECT. The top object identifies all the graphic objects in the PFD format. Each entry actually becomes a call to the DU code to display that object on the DU screen. Entries exist for both run time and compile time objects. Objects must be listed in the top object in order to be displayed. Section 4.1 in the CGS manual describes the top object in more detail.

(5) SCREEN CENTER TRANSFORMATION. The screen center transformation is a required CGS object. It can also be used in the OTD by graphic objects that do not move. In the PFD format its position is set to (0,0).

(6) NEW FORMAT SETUP. The new format setup starts the initialization of a new format. It is always the first object block processed by CGS, and contains resource information necessary to display the format - such as the total number of objects, the unique format ID, the number of OTD and event list tables, and the object number of the top object. It also specifies the colors that are used for stroke and raster symbology. Refer to Section 3.6 for further details on the new format setup and its parameters.

Section 7.3 RUN TIME GRAPHICS ROUTINES

This section describes the PLM86 modules that perform graphic operations for dynamic display objects on the PFD format. Below is a list of the modules covered in this section with the PFD symbology each affects.

AIRPLANE_MGR	aircraft pitch symbol, flight path angle wedge, standoff symbols, reference flight path angle bar
ALTITUDE_MGR	altitude tape transform, glideslope scale and pointer, vertical path pointer, selected altitude pointer, altitude value readout, vertical speed arrow
ALT_BLOCKS_MGR	altitude tape
ARROW_BAR	utility used by ALTITUDE_MGR and CAS_MGR
BORDER_MGR	control modes, 'TO' waypoint name, control wheel "out-of-detent" indicators, Mach number, decision height, selected CAS readout, radar altitude, selected altitude readout, baro-altimeter setting, horizontal guidance mode, vertical guidance mode
CAS_BLOCKS_MGR	airspeed tape
CAS_MGR	CAS tape transform, actual CAS readout, selected CAS pointer, acceleration arrow, airspeed limit
CREATE_MASK	low and high resolution stroke mask bit planes
FLARE_MGR	flare cue transform
HOR_LOC_MGR	horizontal deviation scale and pointer, localizer deviation scale and pointer
INVALID_DATA	removes all symbology except the basic PFD skeleton from the display screen
MESSAGE_MGR	alert messages
PITCH_MGR	pitch grid transformation, track "T", track bug, horizon line tick marks, FPA reference bar
RASTER_MASK	raster channel boundaries for horizon line and gamma wedges (channel A & B)

ROLL_MGR	roll pointer transformation
RWY_MGR	perspective runway and transform
STAR_MGR	waypoint STAR transform
STATIC_RASTER	raster scan lines, raster channel boundaries for the tapes and PFD window (channels C & D)

MODULE NAME: AIRPLANE_MGR
FILE NAME: MANAGER.PLM
CALL RATE: 0.05 SECONDS
CALLED BY: PFD_EXC

PURPOSE: To create and enqueue the aircraft object containing the aircraft pitch symbol, flight path angle wedge, standoff symbol, and reference flight path angle.

REMARKS: The graphics object "GR_AIRPLANE" is created and enqueued by this procedure. If the aircraft attitude is not valid just a NIL object is placed in the queue. Otherwise the object block is filled with data for the aircraft pitch symbol, flight path angle wedge and optionally a pitch or gamma standoff indicator. Which variants of the symbology will be used is determined by the velocity vector mode flag. Refer to figure 7.3 for the symbols in this graphics object.

When in VV_MODE the large version of the gamma wedge symbol is fixed at the center of the PFD window. The small variant of the aircraft handle bar symbol is moved to the position indicated by the global inputs AIRCRAFT_X and AIRCRAFT_Y. When a non-zero standoff position exists, the red gamma standoff diamond is stored in the indicated position relative to the gamma wedge.

If not currently in a velocity vector mode the large version of the aircraft symbol is displayed in a fixed position within the PFD window. It is centered horizontally and raised vertically five degrees above the center. Note that this bias is applied to all window symbology to change the window center in non-velocity vector modes. The small variant of the gamma wedge is moved to the position indicated by the gloabl positions. Optionally, if a non-zero standoff value was supplied, the aircraft "W" standoff symbol is shown also.

GLOBAL REFERENCES:

VARIABLES
AIRCRAFT_X AIRCRAFT_Y ATT_VLD FCB GE_AIRPLANE
GR_AIRPLANE_NIL STANDOFF VV_MODE

PROCEDURES
END_OBJECT_BLOCK ENQ_OBJECT_BLOCK IPVGSUB_CSTRING
NEW_OBJECT_BLOCK VID_PRI

MODULE NAME:	ALTITUDE_MGR
FILE NAME:	MANAGER.PLM
CALL RATE:	0.05 SECONDS
CALLED BY:	PFD_EXC

PURPOSE: To create and enqueue display objects for the altitude tape transform and altitude tape pointers.

REMARKS: Two display objects are generated by this module for the PFD. The first is the transformation object for the altitude tape. The other is a graphics object created for the glideslope scale and pointer, vertical path pointer, actual altitude readout, selected altitude pointer, and vertical speed arrow. Refer to figure 7.6 to identify the symbology in this graphics object.

When the altitude information from the host computer is not valid, NIL objects are enqueued for both the transformation and graphics objects. Otherwise fresh data is created and stored into the object blocks via CGS calls.

First the altitude tape transformation is created. This object centers the altitude tape, created by the module ALT_BLOCKS_MGR, at the current aircraft altitude. This is done by scaling the difference between the actual altitude and the tape center to find the tape displacement.

The altitude pointers object uses the screen center transformation for its position. This means the offsets used for each of the included symbols must be relative to the screen center.

The first piece of this graphics object is the actual altitude readout. These digits are displayed in white using the medium character set for the thousands digits and the small character set for the three least significant numbers.

Next the vertical speed arrow is created in the object block through a call to the procedure ARROW_BAR. The length of the arrow shaft is passed using the variable HDOT, which is sent from the host computer.

The selected altitude pointer is stored in the object block when ALT_REF_VLD is set. The symbol will be colored green if Altitude Select is the currently engaged vertical guidance mode (VRT_ENG=2) or colored amber when it is the currently armed vertical mode (VRT_ARM=2). Otherwise altitude preselection must be in effect, which is shown with a blue pointer. The "Y" axis position along the edge of the tape is computed by the local procedure TAPE_Y, described below. The ALT_REF value from the host computer is passed to TAPE_Y to determine the proper position.

If the engaged or armed vertical guidance mode is vertical path guidance, the vertical path deviation pointer will be stored into the object block. Green is used for engaged and amber for armed. The value VRT_REF from the host computer is passed to the local procedure TAPE_Y to compute the position along the altitude tape where the pointer should be placed.

The last symbology that may be stored in the object block is for the glideslope scale and pointer. The engaged and armed mode variables are tested to determine if any of the glideslope modes are in use. If active, the glideslope scale and pointer box are stored in either green or amber, depending on the engaged or armed status respectively. The box is filled with the label "GS" unless the MLS guidance is in use, in which case the label is "GP". The variable GS_REF from the host computer has the pointer position. A special code value is passed (F000 hex) to signify an invalid glideslope signal. In this situation the glideslope scale is dislayed without the pointer. Otherwise GS_REF is passed to the procedure TAPE_Y to compute the display position along the glideslope scale of the glideslope pointer. Note that the glideslope scale consists of the "DOT" character, from the user defined character set, placed five times along the left boundary of the altitude tape.

A local procedure is defined within ALTITUDE_MGR to compute the "Y" axis position along the altitude tape of the various pointers. The input parameter is the pointer's reference position in feet. The display screen units, one thousandths inches, is returned as the value of the typed procedure. The difference between the reference position and the actual altitude provides the proper displacement in feet. This is scaled to screen units of displacement by multipling by 1.67 units per foot.

GLOBAL REFERENCES:

VARIABLES
ACT_ALT ALT_REF ALT_REF_VLD ALT_VLD FCB GE_ALT_PTRS
GR_ALT_PTRS_NIL GS_REF HDOT TR_ALT_TAPE_NIL VRT_ARM
VRT_ENG VRT_REF

PROCEDURES
ARROW_BAR END_OBJECT_BLOCK ENQ_OBJECT_BLOCK
IPVGSUB_CSTRING MAKE_XFORM_TWO NEW_OBJECT_BLOCK
U_BNRBCD U_MULT VGSUB_CSTRING VID_PRI

MODULE NAME: ALT_BLOCKS_MGR
FILE NAME: MANAGER.PLM
CALL RATE: 0.2 SECONDS
CALLED BY: PFD_EXC

PURPOSE: To create and enqueue the graphic object for the altitude tape ticks and digits.

REMARKS: The altitude tape range is approximately 2000 feet. An altitude value and a large tick mark are displayed every 500 feet. Four smaller ticks are displayed at 100 foot intervals in between. The altitude tape is built by piecing together sections of the tape known as "blocks". A block consists of one large tick and four smaller ticks distanced by their 100-foot differences, and an altitude value which will be a multiple of 500. Five blocks compose the altitude tape. Note that this represents more than 2000 feet, which is the length of the visible tape area. This is because the tape is repositioned, by the transformation in ALTITUDE_MGR, as actual altitude changes. Enough overlap must exist to always fill the entire tape view area.

The local subroutine BLOCKER formats the tape scale digits for one 500 foot block. The block that contains the aircraft actual altitude is determined. The procedure BLOCKER is called to generate data for that block and the two blocks immediately above and below to make the five block tape. The digits are shown to the right side of the larger ticks. The thousands digits are displayed in larger characters than the lesser digits. No digits are displayed below 0 feet.

The altitude tape graphic object is a run time object but is enqueued at a slow rate since the blocks required do not change rapidly. It's transform, enqueued at the fastest rate by ALTITUDE_MGR, is also run time and allows for the movement of the tape. A variable computed here (ALT_CENTER) that specifies the center of the altitude tape (this will actually be the middle 250-foot interval of the block containing current altitude) is used for the tape transformation by ALTITUDE_MGR.

GLOBAL REFERENCES:

VARIABLES
ACT_ALT FCB GE_ALT_TAPE

PROCEDURES
END_OBJECT_BLOCK ENQ_OBJECT_BLOCK IPVGSUB_CSTRING
NEW_OBJECT_BLOCK U_BNRBCD VID_PRI

MODULE NAME: ARROW_BAR
FILE NAME: MANAGER.PLM
CALLED BY: ALTITUDE_MGR, CAS_MGR
PARAMETERS: X, Y, BAR_LEN, OBJ_PTR

PURPOSE: To generate object block data for either the vertical speed or acceleration arrow bars.

REMARKS: This procedure is passed position coordinates and length to create graphics data in an object block whose address is also passed through the parameter list. The created symbol has a shaft consisting of three vertical yellow lines. If the shaft is long enough, an arrowhead replaces the last .073 inches of the length. CGS library procedures are called to perform the actual data generation.

GLOBAL REFERENCES:

PROCEDURES
DRAW_REL_XY_BLANK DRAW_REL_XY_VIS MOVE_ABS_XY_BLANK
VGSUB_CSTRING VID_PRI

MODULE NAME:	BORDER_MGR
FILE NAME:	MANAGER.PLM
CALL RATE:	0.05 SECONDS
CALLED BY:	PFD_EXC

PURPOSE: To create and enqueue the "Border Text" graphics object for the PFD.

REMARKS: This procedure calls CGS library routines to store data in a graphics object block. The information included consists of the flight control mode, throttle mode, 'TO' waypoint name, control stick detent indicators, Mach number, decision height value, selected airspeed, radar altitude, selected altitude, barometric altimeter setting, horizontal guidance mode, and vertical guidance mode. Refer to figures 7.2, 7.5, 7.6, and 7.7 for these items.

The first item placed in the object block is the flight contol mode. The index variable from the host computer CONTROL is used to select the proper label from the local table CTL_DAT. The label is displayed in large green characters in the upper left corner of the PFD. No text is stored in the object block when the aft flight deck is not engaged (CONTROL=0).

The control wheel detent indicators are placed into the object block next when valid. The variables from the host computer ROLL_FLAG and PITCH_FLAG are tested to determine if roll or pitch attitude changes are being commanded. When commands are in progress up/down or left/right green arrowheads are placed next to the flight control mode to signify pitch or roll commands respectively. The arrowheads are obtained from the user defined character set.

The throttle mode identifier is placed in the upper left corner of the PFD in large green letters when the auto-throttle is engaged (THROTTLE not 0). The variable THROTTLE is used to index into the mode name table THR_DAT to select the proper throttle text.

The name of the destination waypoint on the active flight plan is contained in the byte array TO_WPT. The host computer stores the character count in the first byte to conform to the format used by the CGS library. A zero count passed by the host signifies no valid 'TO' waypoint. Large green text is used for the waypoint name, which is placed in the upper right corner of the PFD.

The aircraft's Mach number is displayed in medium gray numbers atop the airspeed tape when its value is .5 or greater. The value from the host computer MACH is scaled as Mach number multiplied by 1000 to retain resolution in the fixed point value. The format of the displayed number is "0.DDD" where D is a single digit.

The barometric altimeter setting is shown in small gray text directly below the altitude tape. "IN" is appended to the end to signify the units of inches. The input variable from the host computer BARO_SET is scaled by a factor of 100 to provide two fractional decimal positions.

Decision height is sent to the PFD from the host computer in units of feet, rounded to the nearest 10 feet. The value is shown in medium characters above the altitude tape as long as the dialed in value is not less than 50 feet. The color is gray until the radar altitude passes below decision height, at which time the color is changed to amber. Note when radar altitude is invalid the color is always gray. The label "DH" is prefixed to the decision height digits.

When airspeed select mode is either engaged or armed the chosen airspeed value is shown directly beneath the airspeed tape. The variable from the host computer CAS_CODE is an index designating armed, engaged, or not selected. The color of the text is green for engaged and amber for armed. The airspeed value is written in small letters with the label "CAS" appended to the beginning. The airspeed value from the MicroVAX, CAS_REF, is scaled as knots multiplied by 64 for accurate CAS tape positioning. The value displayed is rounded to the nearest knot for the numerical readout.

The flag ALT_REF_VLD indicates the use of altitude select mode by the pilot. The vertical guidance variables VRT_ENG, and VRT_ARM are sampled to determine if the altitude select mode is either engaged or armed. If neither one is true then the ALT_REF_VLD setting must signify altitude pre-selection. The color coded value is shown directly below the barometric altimeter setting in small letters with "FT" appended to the rear to denote units of feet. The color will be either blue, amber, or green depending on the pre-select, armed, or engaged status of the mode.

The enaged and armed guidance modes are placed on the PFD by four calls to the local procedure GUID_TEXT. The horizontal modes are placed in the lower left corner and the vertical modes are shown in the lower right corner. A call to GUID_TEXT is skipped for any of the guidance modes that are not in use. The engaged modes are shown in green and the armed modes in amber. See the description for the local procedure GUID_TEXT given below.

Radar altitude is shown on the PFD when the value is valid and below 1000 feet. It is shown in medium cyan digits with the label "RA" prefixed for identification. The color changes to amber once below the displayed decision height. The value is displayed to the nearest 10 feet until below 500 feet, at which time a resolution of 5 feet is shown. Once below 200 feet radar altitude is displayed to the nearest foot.

The local procedure GUID_TEXT is defined for placing horizontal and guidance modes, both armed and engaged, on the PFD. It is called with the mode index and the position coordinates for the text field. GUID_TEXT differentiates between horizontal and vertical mode indices by the sign of the "X" coordinate. It is the responsibility of the caller to set the proper color before making the call. The mode index is used to fetch the proper text from mode name tables defined within GUID_TEXT. When vertical guidance modes are shown padding is added to the "X" coordinate to right justify the text. Both track and flight path angle select modes have the selected value appended to the mode text.

GLOBAL REFERENCES:

VARIABLES
ALT_REF ALT_REF_VLD BARO_SET CAS_CODE CAS_REF CONTROL DEC_HT FCB FPA_DIAL GE_BORDER HRAD_VLD HRZ_ARM HRZ_ENG MACH PITCH_FLAG RADAR_ALT ROLL_FLAG THROTTLE TO_WPT TRK_DIAL VRT_ARM VRT_ENG

PROCEDURES
END_OBJECT_BLOCK ENQ_OBJECT_BLOCK IPVGSUB_CSTRING NEW_OBJECT_BLOCK ROTANG U_BNRBCD U_INSERT_DEC_PT U_MULT VGSUB_CSTRING VID_PRI

MODULE NAME: CAS_BLOCKS_MGR
FILE NAME: MANAGER.PLM
CALL RATE: 0.2 SECONDS
CALLED BY: PFD_EXC

PURPOSE: To create and enqueue the graphic object for the CAS tape ticks and digits.

REMARKS: The CAS tape range is approximately 66 knots. A computed airspeed value and a large tick mark are displayed at each 20-knot interval. A smaller tick is also displayed between the ten knot intervals. The CAS tape is built by piecing together sections of the tape known as "blocks". A block consists of a small and large tick mark distanced by their 10-knot difference, and an airspeed value which will be a multiple of 20. Five blocks compose the CAS tape. Note that this represents more than 66 knots, which is the length of the visible tape area. This is because the tape is repositioned, by the transformation in CAS_MGR, as actual airspeed changes. Enough overlap must exist to always fill the entire tape view area.

The local subroutine BLOCKER formats the tape scale digits for one 20 knot block. The block that contains the aircraft actual airspeed is determined. The procedure BLOCKER is called to generate data for that block and the two blocks immediately above and below to make the five block tape. The digits are shown to the left side of the larger ticks. No digits are displayed below 0 knots.

The CAS tape graphic object is a run time object but is only enqueued at a slow rate since the blocks required do not change rapidly. It's transform, computed at the fastest rate by CAS_MGR, is also run time and allows for the movement of the tape. A variable computed here (CAS_CENTER) that specifies the center of the CAS tape (this will actually be the middle tens unit value of the block that contains current CAS) is used for the tape transformation in the module CAS_MGR.

The CAS tape is displayed in white. The host computer supplies the actual computed airspeed scaled by a factor of 64. Computations in this procedure have taken this scale into account. The CAS tape graphics are not shown if the CAS signal is determined to be invalid.

GLOBAL REFERENCES:

VARIABLES
ACT_CAS FCB GE_CAS_TAPE

PROCEDURES
END_OBJECT_BLOCK ENQ_OBJECT_BLOCK IPVGSUB_CSTRING
NEW_OBJECT_BLOCK U_BNRBCD VID_PRI

MODULE NAME:	CAS_MGR
FILE NAME:	MANAGER.PLM
CALL RATE:	0.05 SECONDS
CALLED BY:	PFD_EXC

PURPOSE: To create and enqueue airspeed tape objects.

REMARKS: This procedure creates two CGS objects used in the PFD. The first is the transformation object which positions the airspeed tape, produced by CAS_BLOCKS_MGR. The "Y" displacement used in the airspeed tape transformation is the difference between actual CAS and the center-of-tape CAS (a variable computed in CAS_BLOCKS_MGR).

The other object created by CAS_MGR is a graphics object containing the CAS readout, selected CAS pointer, acceleration arrow, and speed limit chevron (see figure 7.7). This object uses the screen center transformation, therefore all of the items stored in the object block have coordinate displacements realtive to the screen center. Note that NIL objects are enqueued for both CAS_MGR objects when airspeed is not valid.

The first item stored in the graphics object block is the actual airspeed readout. The value from the MicroVAX is scaled by a factor of 64 to provide accurate CAS tape positioning. The scaling is removed from the input variable and it is rounded to the nearest knot for the digital readout. Medium sized white digits are used on the display.

The white acceleration arrow runs along the outer right boundary of the CAS tape. Its symbol consists of a small up or down arrowhead followed by three vertical lines that extend to the actual airspeed pointer's location. The tip of the acceleration arrow points to the speed along the tape that the host computer has specified the aircraft will reach in 10 seconds given the current acceleration rate. The actual CGS calls to generate the arrowhead are performed by the procedure ARROW_BAR.

The selected airspeed pointer is a color coded symbol which moves vertically in the right side rectangular area of the airspeed tape. It is shown when a desired airspeed has been chosen on the pilot's mode control panel. The variable CAS_CODE will contain a non-zero index when the CAS select mode is active. Indices of 1, 2, and 3 represent pre-select, armed, and engaged status respectively. The corresponding symbol colors are blue, amber, and green. The difference between selected and actual airspeed is scaled by a factor of 19.724 knots per inch, limited to the tape window area, to find the "Y" axis position of the selected airspeed pointer. The scale factor of 64 used for both actual and reference airspeed is accounted for in the calculations.

The host computer provides an upper airspeed limit for the current flap setting. The difference between the speed limit value and the actual CAS is used to displace the airspeed limit symbol along the tape in the same manner as the selected airspeed pointer. The speed limit variable from the MicroVAX is first converted to the 64 units per knot scaling before the difference can be made. The original scaling of FLAP_UP is 65.536 units per knot.

GLOBAL REFERENCES:

VARIABLES
  ACC_SEGMENT ACT_CAS CAS_CODE CAS_REF CAS_VLD FCB
  FLAP_UP GE_CAS_PTRS GR_CAS_PTRS_NIL THROTTLE
  TR_CAS_TAPE_NIL

PROCEDURES
  ARROW_BAR END_OBJECT_BLOCK ENQ_OBJECT_BLOCK
  IPVGSUB_CSTRING MAKE_XFORM_TWO NEW_OBJECT_BLOCK ROTANG
  U_BNRBCD U_DIV U_MULT VID_PRI

MODULE NAME: CREATE_MASK
FILE NAME: MASKING.PLM
CALL RATE: cold start only
CALLED BY: PFD_EXC

PURPOSE: To create the high and low resolution bit planes used in masking visible lines on the display.

REMARKS: When performing masking with the Core Graphics system two bit-planes are used to define regions of the display screen. A bit-plane can be thought of as a rectangular grid filled with evenly spaced bits, known as pixels. Regions can be denoted by having some bits "ON" and some "OFF" in a bit-plane. When two bit-planes are involved more areas can be bounded by the intersection of regions defined in each bit-plane. See the "Core Graphics System" manual for information on stroke masking.

This procedure creates both the low and high resolution bit-planes through the use of CGS library calls. Refer to figure 7.9 and 7.10 for the areas bounded by these bit-planes.

The regions are created in the bit-planes by the use of mask channels. Each bit-plane has four channels to use for bounding regions that are created in a left to right sweep across the bit-plane. These channels are identified by the labels A through D. For the PFD only channel A is used in either bit-plane. The other three channels are simply cleared by defining the ON and OFF borders as the same. The user units defined in CREATE_MASK for generating mask regions is 20,000 units full screen (2,985 units per inch).

Channel A in the Low-Res object is defined ON in the actual CAS and altitude readout regions and also in the entire PFD view window (see figure 7.9). The mask priority code is set such that the bits in the low resolution bit-plane are set whenever channel A is on and cleared everywhere else.

Channel A in the High-Res object is defined ON over the PFD tape areas (see figure 7.10). The high resolution mask priority is set the same as the low resolution mask, forcing bits set when channel A is ON and clearing them elsewhere.

GLOBAL REFERENCES:

VARIABLES
GE_HIGH_RES GE_LOW_RES GR_HIGH_RES GR_LOW_RES

PROCEDURES
DRAW_ABS_RA DRAW_ABS_XY DRAW_REL_XY END_MASK_BLOCK
MOVE_ABS_XY MOVE_REL_XY NEW_MASK_BLOCK
SET_MASK_CHANNEL SET_MASK_PRIORITY

MODULE NAME: FLARE_MGR
FILE NAME: MANAGER.PLM
CALL RATE: 0.05 SECONDS
CALLED BY: PFD_EXC

PURPOSE: To process the flare cue transform.

REMARKS: The flare cue transformation object, which positions the flare cue symbol (figure 7.3), is created and enqueued by this procedure. The flare cue position is sent directly from the MicroVAX computer (FLARE). When a non-zero value is supplied and aircraft attitude is valid (ATT_VLD) the transformation is made. Otherwise a nil transform is enqueued to remove the symbol from the display.

GLOBAL REFERENCES:

VARIABLES
ATT_VLD FCB FLARE TR_FLARE_NIL

PROCEDURES
ENQ_OBJECT_BLOCK MAKE_XFORM_TWO

MODULE NAME: HOR_LOC_MGR
FILE NAME: MANAGER.PLM
CALL RATE: 0.05 SECONDS
CALLED BY: PFD_EXC

PURPOSE: To create and enqueue the object containing the horizontal deviation scale and pointer, and the localizer dots and pointer.

REMARKS: The HOR/LOC deviation object uses the screen center transformation for its positioning, therefore all coordinate offsets used in positioning the symbology are relative to the screen center.

The armed and engaged modes of the horizontal guidance axis (HRZ_ARM, HRZ_ENG) are examined to determine if either horizontal flight plan guidance or a localizer mode has been selected. If neither has, a NIL object is enqueued to remove the symbology.

When a horizontal flight plan mode is either armed or engaged the horizontal deviation scale and pointer are placed in the object block (see figure 7.4). The color of the symbols is amber when the mode is armed, and green when it is engaged. The scale and pointer are both characters in the special DU PROM character set.

When an auto-land mode is selected (ILS, MLS, or GPS) the localizer scale and pointer are placed in the object block (see fiqure 7.4). The color coding is the same as the horizontal scale and pointer. The scale is made up of five "DOT" symbols from the user defined character set and the pointer is a character in the DU PROM set. The pointer is labeled with either "AZ" or "LOC" by the local procedure HRZ_LABEL which is described next.

The local procedure HRZ_LABEL is passed the auto-land mode index from HOR_LOC_MGR. It returns the address pointer of a character string containing either "LOC" (ILS, GPS) or "AZ" (MLS) and a "X" offset for centering the text under the localizer deviation pointer.

GLOBAL REFERENCES:

VARIABLES
FCB GE_HORSCALE GR_HORSCALE_NIL HOR_X HRZ_ARM HRZ_ENG LOC_X

PROCEDURES
END_OBJECT_BLOCK ENQ_OBJECT_BLOCK IPVGSUB_CSTRING NEW_OBJECT_BLOCK ROTANG VGSUB_CSTRING VID_PRI

```
MODULE NAME:   INVALID_DATA
FILE NAME:     MANAGER.PLM
CALL RATE:     0.05 SECONDS (when enabled)
CALLED BY:     PFD_EXC
```

PURPOSE: To alert the user that the DEU is not receiving valid data from the host computer.

REMARKS: The routine INPUT_DATA determines if data is being transmitted from the host computer, and if it is being sent in the proper format. If either of these events are false, INPUT_DATA will clear the boolean that indicates a valid format exists. PFD_EXC then checks that boolean and calls this procedure to remove PFD symbology.

The sole purpose of this routine is to put up a display which immediately tells the user that valid data is not being sent from the host computer to the DEU (although data is being transferred correctly between the DEU and DU or the display would only show a red "X"). NIL objects are enqueued to remove most of the PFD, leaving only the PFD outline skeleton. A list of the NIL objects is provided below.

runway transform
waypoint STAR transform
airspeed tape transform
roll pointer transform
altitude tape transform
pitch grid transform
flare cue transform
alert message graphics
altitude tape pointers
airplane/gamma window symbology
airspeed tape pointers
horizontal/localizer deviation symbology

GLOBAL REFERENCES:

VARIABLES
FCB GR_AIRPLANE_NIL GR_ALERT_NIL GR_ALT_PTRS_NIL
GR_CAS_PTRS_NIL GR_HORSCALE_NIL TR_ALT_TAPE_NIL
TR_CAS_TAPE_NIL TR_FLARE_NIL TR_PITCH_GRID_NIL
TR_ROLL_PTR_NIL TR_RWY_NIL TR_STAR_NIL

PROCEDURES
ENQ_OBJECT_BLOCK

```
MODULE NAME:  MESSAGE_MGR
FILE NAME:    MANAGER.PLM
CALL RATE:    0.2 SECONDS
CALLED BY:    PFD_EXC
```

PURPOSE: To display the appropriate alert or warning messages.

REMARKS: The host computer contains logic that determines if an alert or warning condition exists, and will send to MESSAGE_MGR an index which it uses to enqueue the appropriate message. The messages are displayed in large characters inside the PFD window (see fiqure 7.3). There are twelve different messages that can appear (ALERT, CLIMB, DESCEND, TURNL, TURNR, AOA, D/H, FLAPS, OM, MM, LNK MSG). One message, FLAPS, can be shown in either red, or yellow. All other messages are yellow, except AOA which is always red. If no alert or warning conditions exists an index of 0 will be sent and the message area of the screen remains blank. The message FLAPS will not be displayed even when selected if the computed airspeed signal is determined to be invalid. The same is true for D/H (decision height) if the radar altitude signal is invalid. Each of the variants of the object are constant data defined in COMPILE.ASM.

GLOBAL REFERENCES:

VARIABLES
ALERT_FLAG* ATT_VLD CAS_VLD FCB GR_ALERT GR_ALERT_NIL
GR_AOA GR_BLANK GR_CLIMB GR_DESCEND GR_DH GR_FLAPS
GR_FLAPS_RED GR_GEAR GR_LINK GR_MID_MARK GR_OUT_MARK
GR_TURNL GR_TURNR HRAD_VLD

PROCEDURES
ENQ_OBJECT_BLOCK

MODULE NAME:	PITCH_MGR
FILE NAME:	MANAGER.PLM
CALL RATE:	0.05 SECONDS
CALLED BY:	PFD_EXC

PURPOSE: To create and enqueue the transformation for the pitch grid and the graphics object for the horizon line symbols.

REMARKS: This procedure creates and enqueues two CGS objects. The first is the transformation object used to position the pitch grid and horizon line symbols. The second is the horizon line symbols object containing the track "T" bar, VCWS track bug, horizon ticks, and the flight path angle reference bar. Figure 7.2 shows these items.

The pitch grid is a moving symbol based on aircraft roll and flight path angle. The graphic object for the pitch grid is a fixed compile time object enqueued by PFD_EXC at the slowest rate. However, the transformation which controls its movement is a run time object created and enqueued in PITCH_MGR at the fastest rate. The host computer sends a pitch delta based on the flight path angle which is used to perform an initial y-translation. The pitch grid is then rotated according to the actual roll angle. Another y-displacement is performed if the current mode of flight is not velocity vector mode (the pitch grid will be biased by +5 degrees).

The selected track "T" bar depicts the selected track along the horizon line relative to the current aircraft track indicated by the gamma wedge. The symbol is shown when a track value is selected on the pilot's mode control panel or horizontal flight plan guidance is engaged. When track select mode is used the color of the symbol may be blue or green to match the preselect or engaged status of the mode. The symbol is always green when it represents the horizontal flight plan desired track. The symbol is displaced along the horizon line by the difference between the reference track value and the aircraft's current track, scaled by a factor of .1675 inches per degree. When the PFD is not in the velocity vector mode the track "T" is also displaced by the aircraft yaw angle. This must be done because the center of the PFD screen depicts aircraft heading, not track, in this mode. The yaw angle is obtained by computing the difference between heading and track. The symbol is gotten from the user defined character set.

The track bug is a yellow colored symbol that moves along the bottom edge of the horizon line. The x-value is supplied by the host based on a track delta converted to screen units. Note that the y-value is not 0. Due to the thickness of the horizon line the symbol is drawn slightly below it in order to make it fully visible. The track bug is a selectable symbol which is drawn or removed based on a discrete from the MicroVAX. The track bug symbol is a character from the user defined character set.

The flight path angle reference bar is placed onto the pitch grid to indicate the desired flight path angle sent in FPA_REF from the host computer. Its color may be blue, amber, or green to match the guidance mode status (preselect, armed, or engaged). The current vertical guidance mode shown in the lower right portion of the display may be examined to determine which flight path angle reference is in use. When no reference exists the symbol is removed from the screen. This symbol is gotten from the user defined character set.

Also contained in PITCH_MGR is the graphic object for the horizon ticks. They also use the pitch grid transform and the PFD window mask. The horizon ticks consist of four white tick marks output at ten degree intervals. The host supplies an offset relative to the PFD screen center (track or heading) which is used to position the ticks.

GLOBAL REFERENCES:

VARIABLES
ACT_ROLL ATT_VLD FCB FPA_BAR FPA_REF GE_HORIZON
HOR_TICK_CTR HRZ_ENG MAG_TRK PITCH_Y PTH_TRK
TRACK_BUG_X TRK_BUG_SEL TRK_DIAL TRK_SEL TRU_HDG
TRU_TRK TR_PITCH_GRID_NIL VV_MODE

PROCEDURES
END_OBJECT_BLOCK ENQ_OBJECT_BLOCK IPVGSUB_CSTRING
MAKE_XFORM_ZERO NEW_OBJECT_BLOCK ROTANG U_MULT VID_PRI

MODULE NAME: RASTER_MASK
FILE NAME: MASKING.PLM
CALL RATE: 0.05 SECONDS
CALLED BY: PFD_EXC

PURPOSE: To create the dynamic raster fill object using raster channels "A" and "B".

REMARKS: Since the raster regions associated with the "Gamma Wedge" and "Horizon Line" change, raster channels "A" and "B" must be updated continually. Each call to this subroutine updates the graphic object containing channels A and B. Figures 7.11 and 7.12 at the end of this section depict typical boundaries for these channels.

To properly understand raster fill techniques, Sections 2.5 - 2.7, 6.0 - 6.2, and 7.0 - 7.6 of The Core Graphics System manual should be read thoroughly. A detailed description of the software in this subroutine is given only for segments of code that are not solely Core Graphics calls.

The bottom boundary of channel A is made to align along the current position of the horizon line of the pitch grid object. The coordinates, in 1/1000 inches, of the two points where the horizon line intersects the view rectangle (X = 2241, -2241; Y = 2798, -1675) are calculated from the airplane roll angle and the offset from screen center of the pitch grid, ACT_ROLL and PITCH_DELTA respectively. The values are saved in LEFT_X, LEFT_Y, RIGHT_X, and RIGHT_Y. The diagram on the following page shows the layout of a typical horizon line within the view rectangle. Note the horizon is shown both with a roll angle and an offset from screen center. The intersection points are found by first solving for the parameters to the standard line equation.

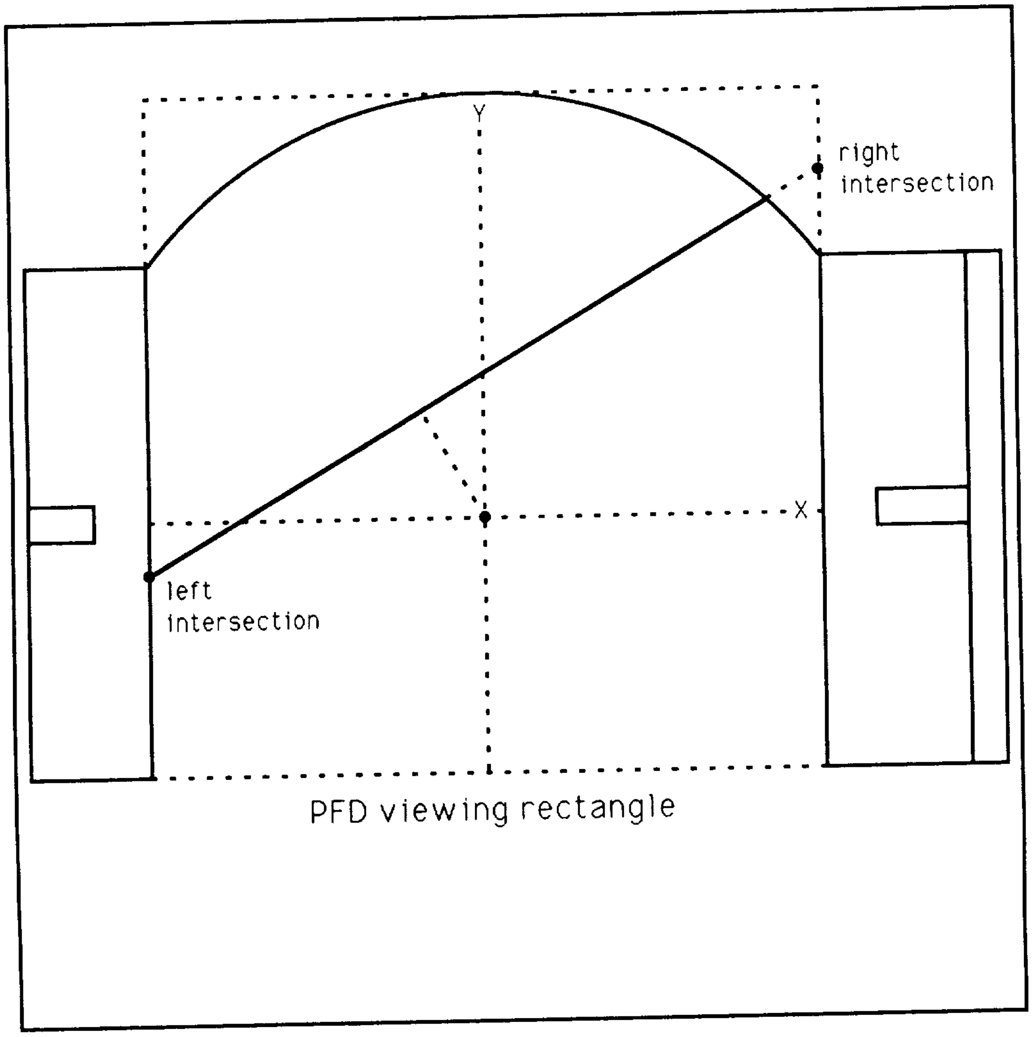
$$Y = m * X + b$$

"m" = slope of line = TANGENT(ACT_ROLL)
"b" = Y axis intercept = PITCH_DELTA / COSINE(ACT_ROLL)

Once the horizon line equation is obtained side boundary intersections are found by using X values of -2241 and 2241. If the horizon intersects at either the top or bottom of the view rectangle, Y values of -1675 and 2798 are used in the equation;

$$X = (Y - b) / m$$

PFD Horizon Line Computation



-figure 7.8-

to produce the desired coordinates. It is possible to have no intersection between the horizon line and the view rectangle. When this condition is detected the bottom channel "A" boundary is made to coincide with either the top or bottom edge of the view rectangle, depending on the position of the horizon line.

When the coordinate values are found CGS routines are called to define channel "A". Raster lengths are supplied in 1/5000 inch units, so each length obtained earlier is multiplied by 5.

Channel "B" is used to define the area within the "gamma wedge" symbol. There are two different gamma wedge symbols that appear depending on PFD mode. One is a large symbol that never moves and the other is a smaller version that can move about the display screen. The coordinates of the various points on these characters have been predetermined so the raster boundaries can be drawn once the character's screen position is known. The variables GAMMA_X and GAMMA_Y contain the small wedge position relative to screen center. The large wedge is always placed at screen center.

When the small moving wedge is positioned on the screen where the left "wing" extends off the left edge of the raster patch, special computations must be made to clip the raster boundary since raster channel initial positions are always at the lower left edge of the patch area. The raster channel boundaries may extend past the right edge of the patch area so special clipping is not needed when the wegde boundary extends beyond the right edge.

GLOBAL REFERENCES:

VARIABLES
ACT_ROLL ALT_HOLD ATT_VLD FCB GE_RASTER_DYNAMIC
PITCH_Y VV_MODE X_SLOP Y_SLOP

PROCEDURES
DRAW_REL_XY END_MASK_BLOCK ENQ_OBJECT_BLOCK
MOVE_ABS_XY MOVE_REL_XY NEW_MASK_BLOCK
SET_MASK_CHANNEL SET_MASK_PRIORITY U_COSINE U_DIV
U_MULT U_SINE

PAGE 226 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

MODULE NAME: ROLL_MGR
FILE NAME: MANAGER.PLM
CALL RATE: 0.05 SECONDS
CALLED BY: PFD_EXC

PURPOSE: To compute and enqueue the roll pointer transformation.

REMARKS: The roll pointer symbol is a small triangle that moves along the inner edge of the roll scale arc. Since it may be constantly moving, it must have a run time transformation which is computed in this procedure. The transform is enqueued at the fastest rate possible. The graphic object for the roll pointer does not change, and in fact is a compile time object. It is enqueued by PFD_EXC at the slowest rate. When the aircraft attitude is not valid a NIL transform is enqueued to remove the pointer from the PFD.

GLOBAL REFERENCES:

VARIABLES
ATT_VLD FCB SKY_PTR TR_ROLL_PTR_NIL

PROCEDURES
ENQ_OBJECT_BLOCK MAKE_XFORM_ONE

MODULE NAME: RWY_MGR
FILE NAME: MANAGER.PLM
CALL RATE: 0.05 SECONDS
CALLED BY: PFD_EXC

PURPOSE: To create and enqueue the graphic object and transformation for the perspective runway.

REMARKS: The runway symbol is a selectable option via a bezel switch on the DU. Once it has been selected, the host computer determines if the other conditions have been met for the runway to be displayed and sets a discrete accordingly. A set of eight points are available to draw the symbol. Two of the points will be used to draw a line through the center of the runway that will extend to the horizon line. The runway heading symbol is displayed at the point where the two lines intersect. An index is also sent by the host which tells if the runway threshold and touchdown lines are to be drawn. They are not displayed after the aircraft passes these points. Since the runway becomes larger as the aircraft approachs it, a dynamic scale factor for the runway points is computed by the host to prevent overflows. The scale factor is used in creating the runway transform. The runway symbol is colored green and masked by the PFD window.

GLOBAL REFERENCES:

VARIABLES
ACT_ROLL ATT_VLD FCB GE_RWY RWY_INDEX RWY_SCALE
RWY_SEL RWY_X1 RWY_X2 RWY_X3 RWY_X4 RWY_X5 RWY_X6
RWY_X7 RWY_X8 RWY_Y1 RWY_Y2 RWY_Y3 RWY_Y4 RWY_Y5
RWY_Y6 RWY_Y7 RWY_Y8 TR_RWY_NIL

PROCEDURES
DRAW_ABS_XY_BLANK DRAW_ABS_XY_VIS END_OBJECT_BLOCK
ENQ_OBJECT_BLOCK IPVGSUB_CSTRING MAKE_XFORM_TWO
MOVE_ABS_XY_BLANK NEW_OBJECT_BLOCK ROTANG U_DIV
VID_PRI

MODULE NAME: STAR_MGR
FILE NAME: MANAGER.PLM
CALL RATE: 0.05 SECONDS
CALLED BY: PFD_EXC

PURPOSE: To create and enqueue the transformation for the moving "to" waypoint STAR.

REMARKS: The STAR is a selectable option on the PFD via a bezel switch. The host computer sends a discrete which will be set if the conditions to display the STAR have been met. If the valid bit is set, STAR_MGR computes the run time transformation for the STAR graphics. It will be en-queued at the fastest rate possible. The STAR graphic ob-ject is a compile time object which does not change. It is enqueued by PFD_EXC at the slowest rate.

Since the STAR may grow up to 20 times its original size, a dynamic scale factor is computed from the host zoom factor for use in the STAR transformation object. Since the STAR also has X,Y position offsets sent from the host computer they will need to be adjusted by the changing scale factor to position the STAR at the appropriate screen location.

If the STAR valid bit sent from the host is not set, the STAR will be removed from the display.

GLOBAL REFERENCES:

VARIABLES
ATT_VLD FCB STAR_VLD STAR_X STAR_Y STAR_ZOOM
TR_STAR_NIL

PROCEDURES
ENQ_OBJECT_BLOCK MAKE_XFORM_TWO U_DIV U_MULT

MODULE NAME: STATIC_RASTER
FILE NAME: MASKING.PLM
CALL RATE: COLD START ONLY
CALLED BY: PFD_EXC

PURPOSE: To create the static graphics objects for the PFD raster fill.

REMARKS: Since most operations performed in raster fill have no compile time counterparts, a subroutine that creates static raster data blocks is necessary. The subroutine is called once after each system cold start up. Thereafter the static objects created must be transmitted to the DU at a slow rate to prevent object timeouts. Sections 2.5 - 2.7, 6.0 - 6.2, and 7.0 - 7.6 of The Core Graphics System manual should be understood before attempting to become familiar with raster software.

The first object created is the "Quad Scan" object. All formats performing raster fill operations must have this object which defines the length and separation of raster scan lines.

The other object created by this subroutine contains the definitions of raster channels "C" and "D". Figures 7.13 and 7.14 at the end of this section depict the regions of the display screen bounded by these channels. Channel C bounds the regions used by the altitude and airspeed tapes. The areas of both tapes blocked out for the actual altitude and airspeed values and the entire PFD view window are defined by channel D.

Another subroutine, RASTER_MASK, is called every 0.1 seconds to define dynamic raster fill areas using channels "A" and "B".

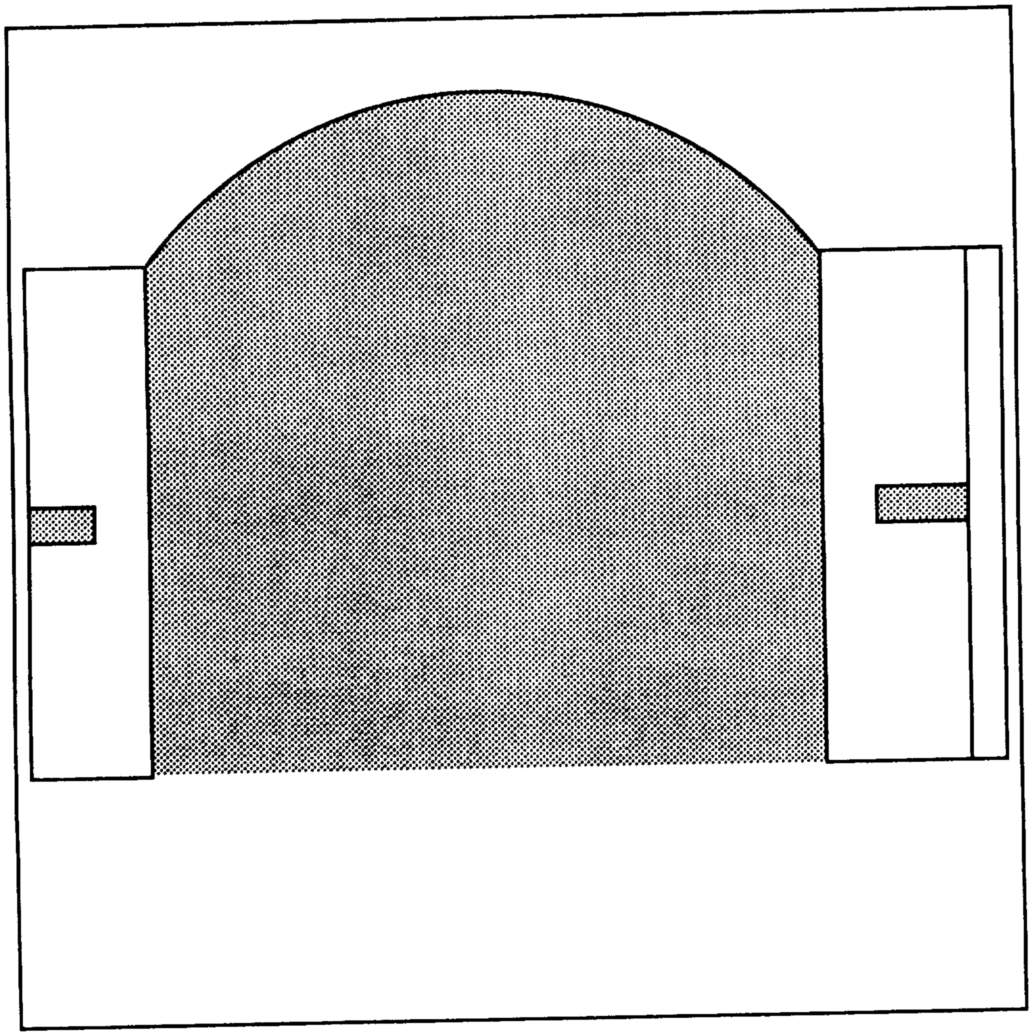
Since raster mask commands are always relative to the lower left corner of the raster patch area, position adjustment factors (X_SLOP, Y_SLOP) must be empirically obtained so the start of the raster fill area is properly aligned with the stroke drawn lines on the display screen. Once these values are set the origin of the raster commands is the lower left corner of the desired fill area.

GLOBAL REFERENCES:

VARIABLES
GE_QUAD_SCAN GE_RASTER_STATIC GR_QUAD_SCAN
GR_RASTER_STATIC NCLK_DLAY NCLK_FINE NCLK_SETL_BOT
NCLK_SETL_TOP NDIX_DELTA NDPC_XMOVMAX NDPC_YMOVMIN
X_SLOP Y_SLOP

PROCEDURES
DRAW_ABS_RA DRAW_ABS_XY DRAW_REL_XY END_MASK_BLOCK
END_OBJECT_BLOCK MOVE_REL_XY NEW_MASK_BLOCK
NEW_OBJECT_BLOCK RASTER_LINEPAIR_2 SET_MASK_CHANNEL
U_MULT

Low Resolution Bit Plane For NASA PFD



(shaded region denotes "ON" bits)

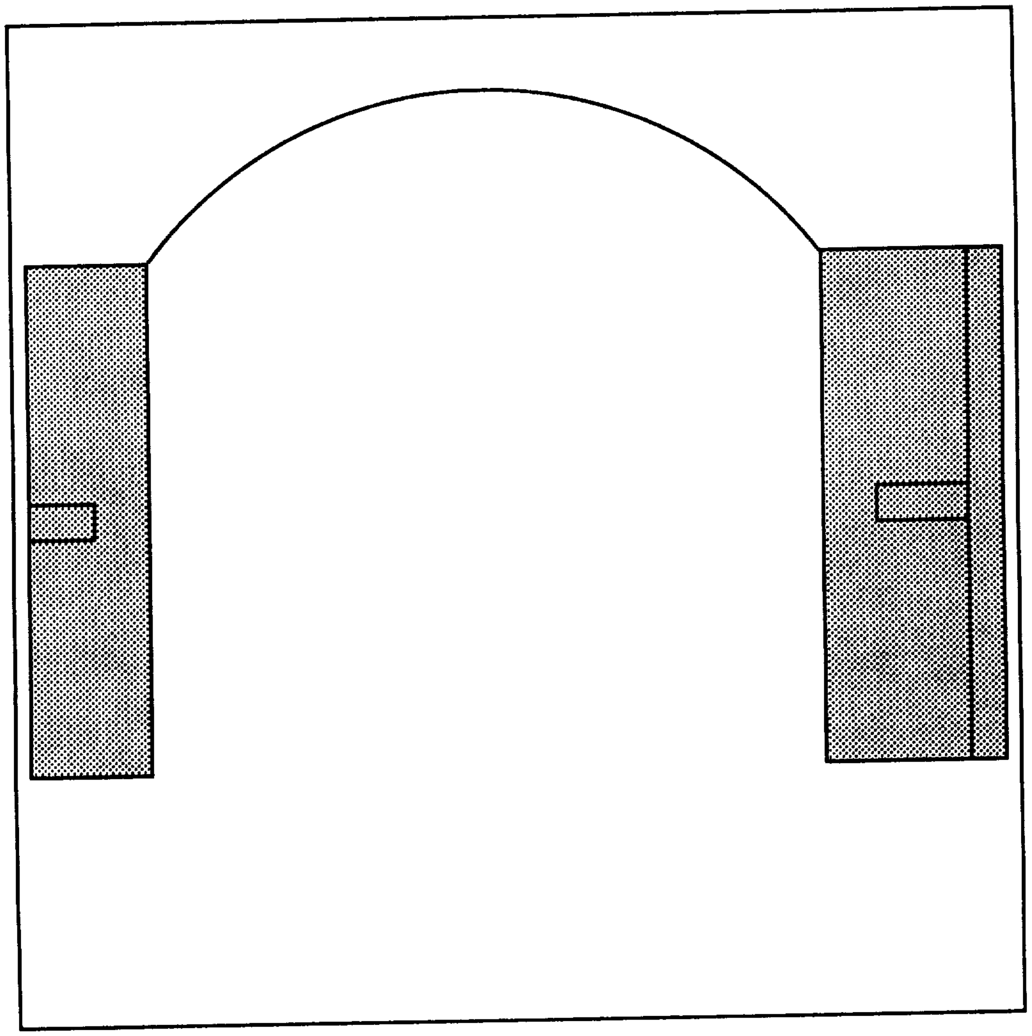
-figure 7.9-

232

PAGE ______ INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

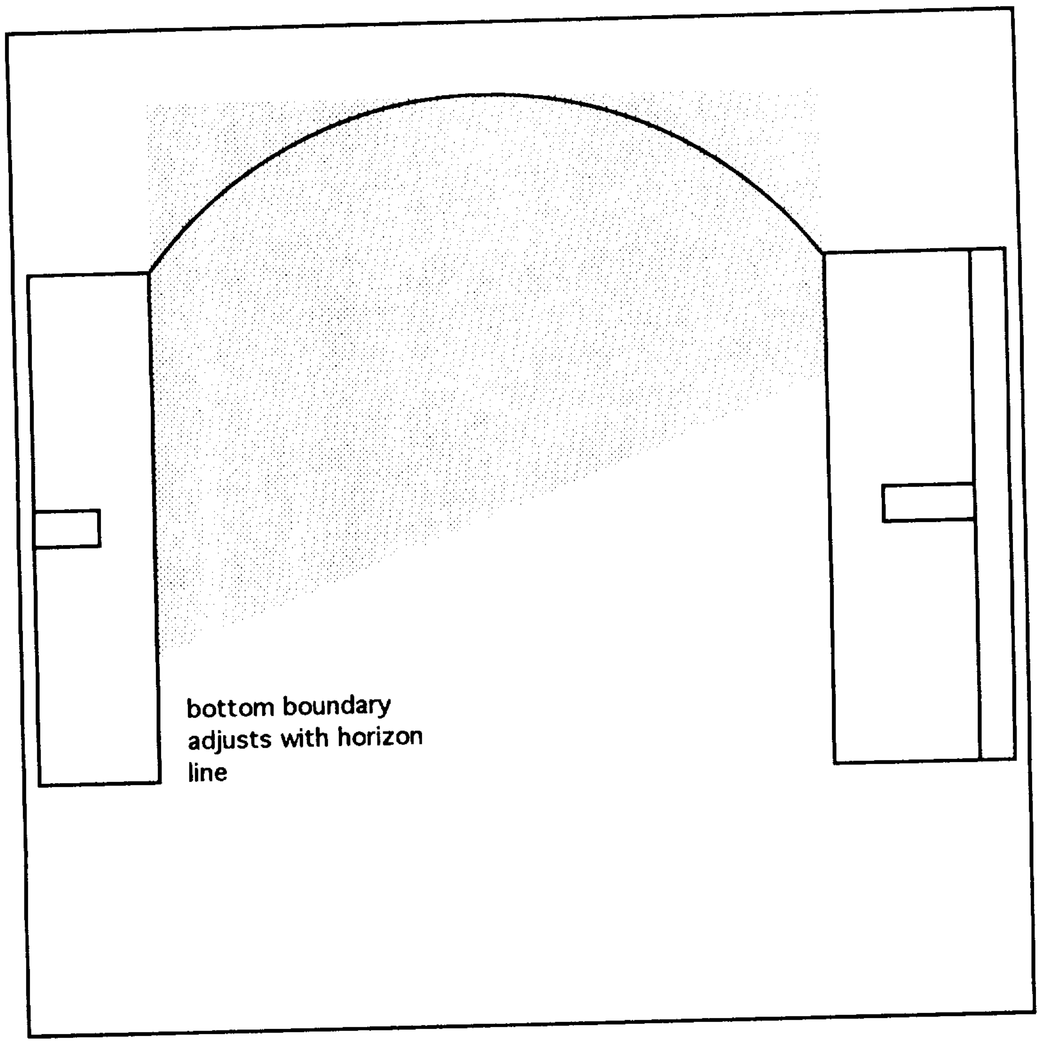
High Resolution Bit Plane For NASA PFD



(shaded region denotes "ON" bits)

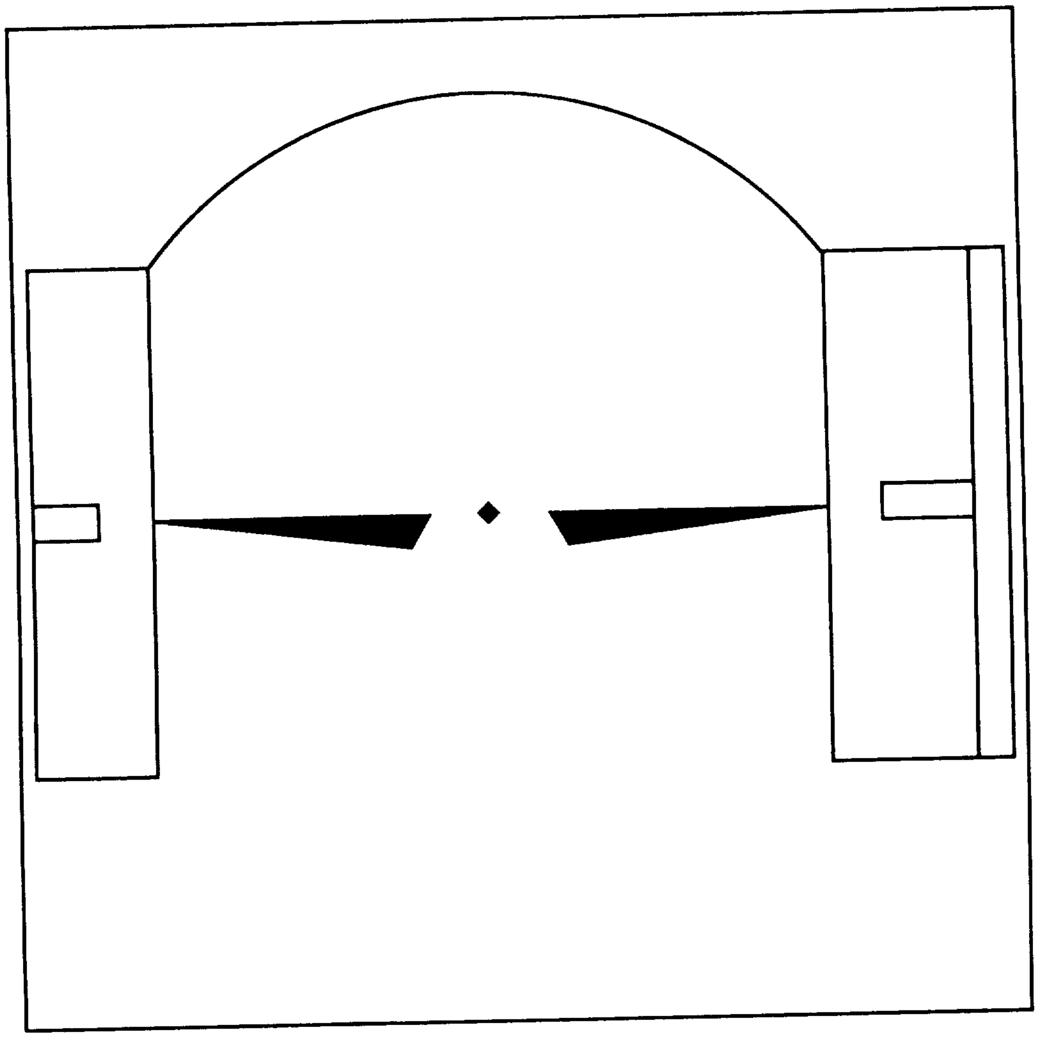
-figure 7.10-

"A"

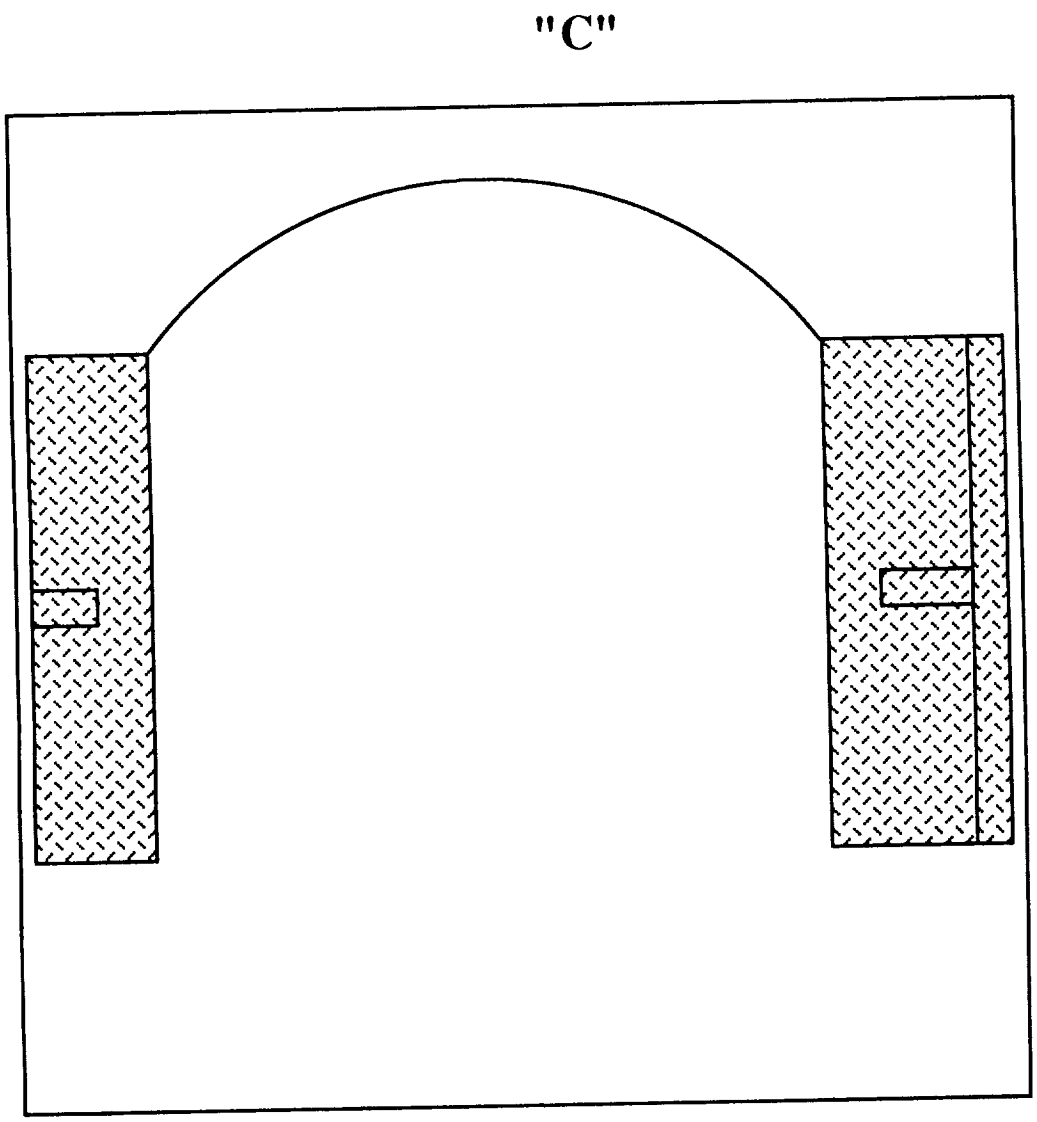


-figure 7.11 (A)-

"B"

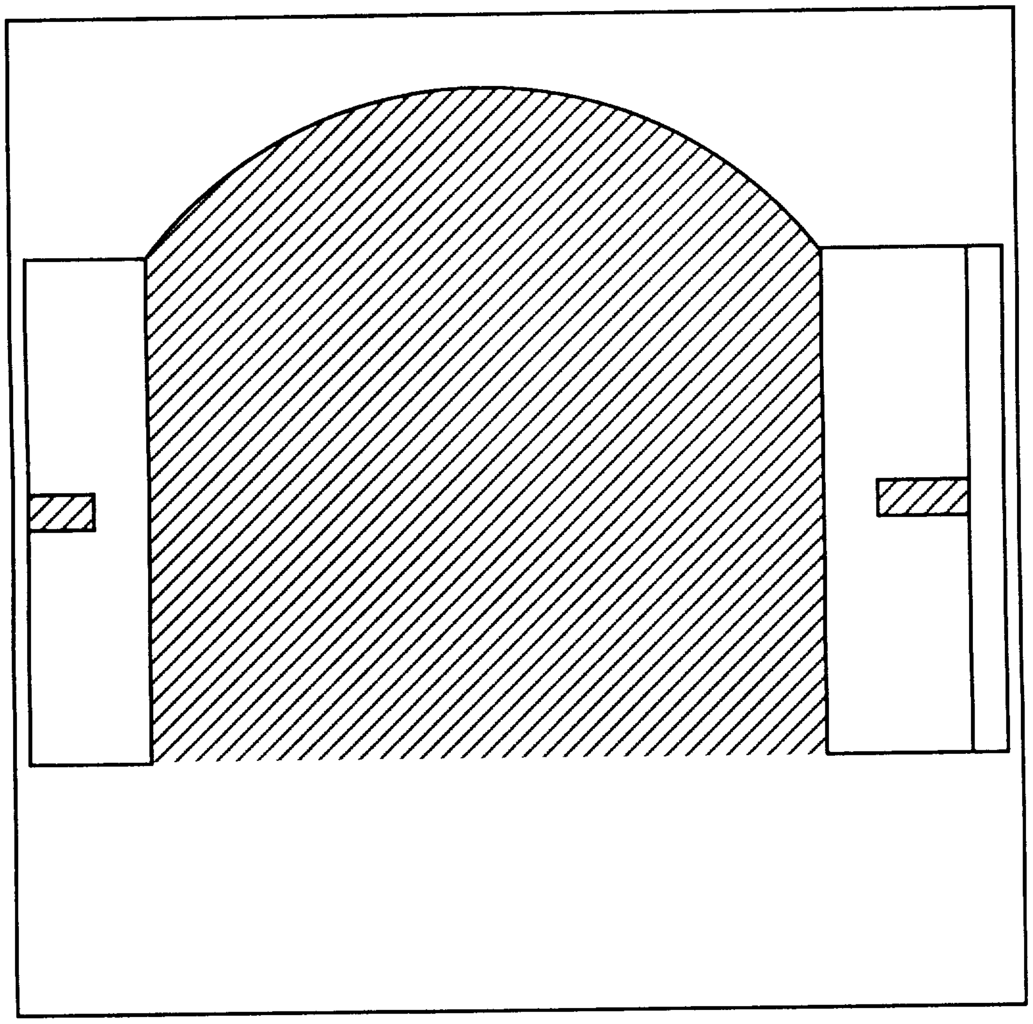


-figure 7.11 (B)-



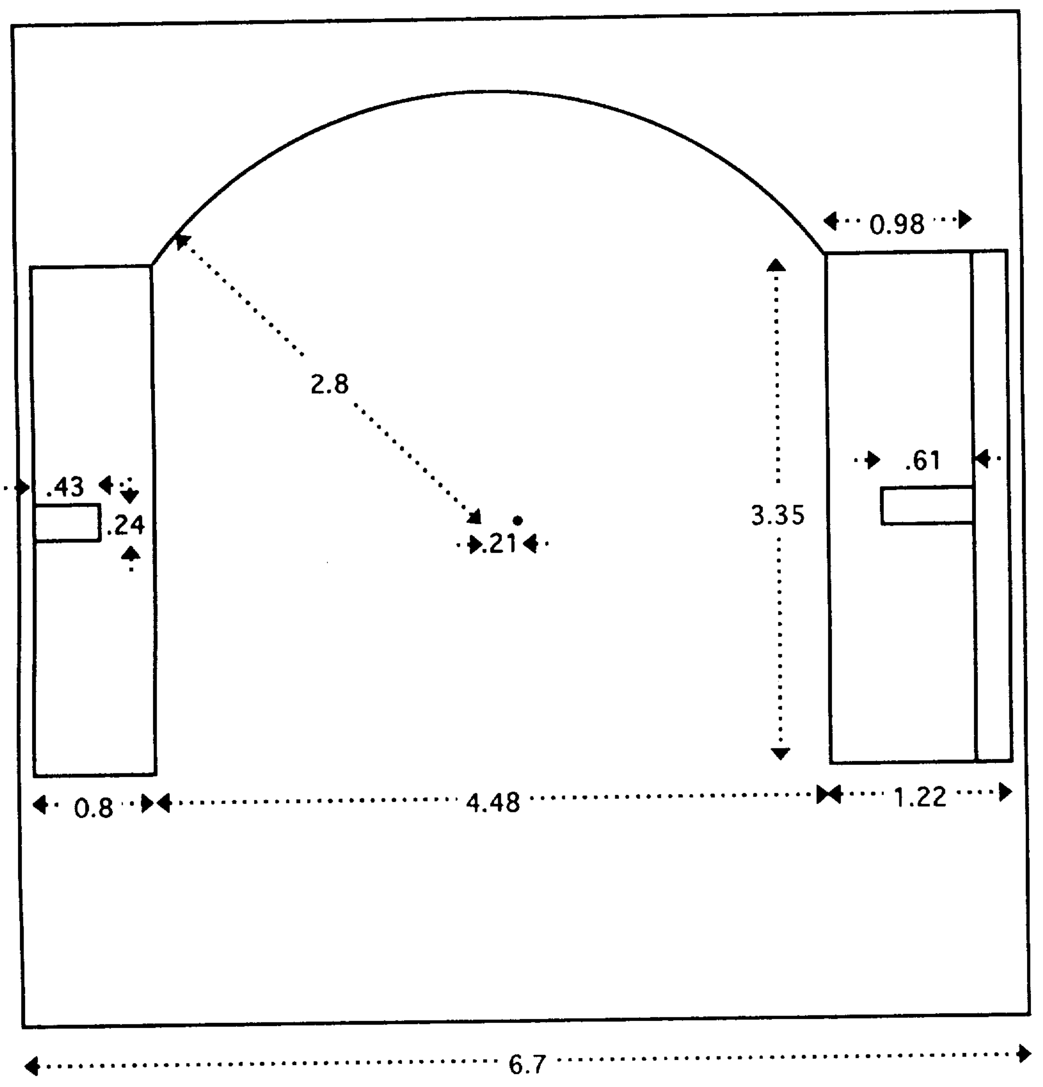
-figure 7.11 (C)-

"D"



-figure 7.11 (D)-

NASA PFD
Raster Channel



(all measurements in inches)

-figure 7.11 (E)-

PAGE 244 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

Section 7.4 MEMORY ALLOCATION FILES

Most of the memory allocation modules reserve memory locations in RAM for various system and applications variables. VAR_RAM.ASM, GLBL_RAM.PLM, and IO_RAM.ASM allocate memory between the address ranges of 800H and 7FFFH in the display processor memory. IO_RAM.ASM reserves memory specifically between the ranges of 4000H and 7FFFH. Memory locations 0400H through 07FFH are reserved for the stack and the stack pointer is initialized to 0800H.

FILE: GLBL_RAM.PLM

This module allocates memory locations in RAM for the system level variables used by the format executive. The types of variables declared globally in this module include system error flags, counters and checksum variables.

FILE: INT_TBL.ASM

This module reserves memory locations 0000H through 03FFH for the interrupt vectors. There are two words of data per interrupt. The first contains the segment to which control is transferred if a given interrupt occurs and the other contains the offset within that segment. Because 3FFH words of memory are reserved, the possibility of defining 512 different interrupts exists. However, only a few are presently used, and two of them are initialized in this module.

FILE: IO_RAM.ASM

This module allocates memory for globally defined I/O variables. The variables defined here contain information from the I/O converter card as well as data sent on the internal I/O bus from other processors within the same DEU, mainly the DP4 processor. The DP4 makes available a 704 word buffer that originates at the host computer. Communications between the host computer and the three applications processors are performed by this buffer. The variables are used for interprocessor communication within a DEU and some also contain values of discretes such as bezels and pots. They are accessible by labels that are placed at offsets into I/O RAM. This is done so that the format can fetch data by name directly from I/O RAM as it is needed. The only exception is with packed discretes, which are fetched once by a processing routine. The routine then unpacks the discretes into boolean bytes which are local to the individual processor.

FILE: VAR_RAM.ASM

This module allocates memory for applications software variables. Most of the variables declared here are discretes which indicate if certain signals are valid or certain graphic objects have been selected.

Section 8.0 THE ENGINE DISPLAY FORMAT

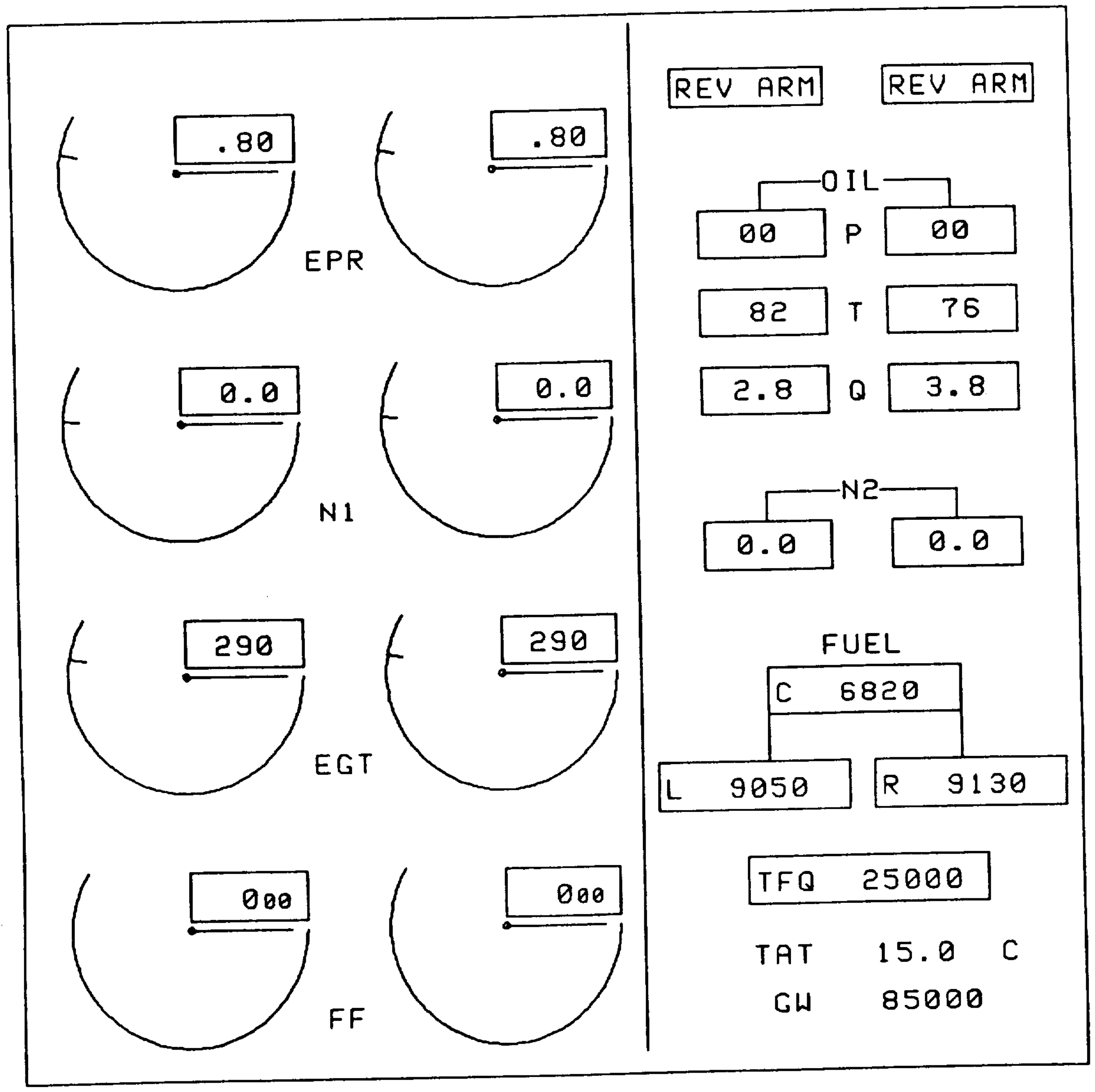
The ENG format provides a graphic representation of quantities associated with the aircraft engines. The format is split into two distinct sections; simulated engine gauges on the left and numeric displays on the right.

The gauges are displayed as four sets of pairs, situated side by side. Each set depicts the same engine parameter, but pertaining to the left and right engines. A gauge has a digital readout, an arc section representing the valid range of values for the particular quantity, and a radial pointer positioned along the arc at the appropriate location for the current value of the quantity. The four engine parameters shown in this manner are the engine pressure ratio, N1 RPM percentage, exhaust gas temperature, and fuel flow rate.

The right hand section of the display screen has numeric values of engine quantities displayed within rectangular boxes. Four miscellaneous items, left/right thrust reverser armed messages, total air temperature, and aircraft gross weight, are also shown in this section. The engine oil pressure, temperature, and quantity along with the N2 ratio are shown in left and right pairs. The amount of fuel remaining is given for the left, right, and center tanks along with the total amount of the three tanks.

This format does not utilize any of the bezel panel buttons or potentiometers. On the next page is a drawing of the ENG format. A chart provided after the drawing shows the modules that handle the various parts of the format. For information about sections of the ENG format, read the relevant module description in Section 8.2.

ENGINE DISPLAY FORMAT



-FIGURE 8.1-

ENGINE INDICATOR NAME	MODULE DESCRIPTIONS
ENGINE PRESSURE RATIO	F3GP_EPR_MGR F3GP_EPR_ARC F3GP_EPR_DRAW_ARC F3GP_EPR_DIGITS F3TP_EPR
N1 PERCENTAGE	F3GP_N1_MGR F3GP_N1_ARC F3GP_N1_DRAW_ARC F3GP_N1_DIGITS F3TP_N1
EXHAUST GAS TEMPERATURE	F3GP_EGT_MGR F3GP_EGT_ARC F3GP_EGT_DRAW_ARC F3GP_EGT_DIGITS F3TP_EGT
FUEL FLOW	F3GP_FF_MGR F3GP_FF_DIGITS
REVERSE THRUSTERS	F3XP_RA_MGR
OIL (PRESSURE, TEMPERATURE, QUANTITY)	F3GP_OIL_MGR F3GP_OILPSI_DATA F3GP_OILTMP_DATA F3GP_OILQTY_DATA F3GP_OIL_FAIL_ANN
N2 PERCENTAGE	F3GP_N2_MGR F3GP_N2 F3GP_N2_FAIL_ANN
FUEL QUANTITY (CENTER, LEFT, RIGHT, TOTAL)	F3GP_FUELQ_MGR F3GP_FUELQ_CRL F3GP_FUELQ_FAIL_ANN
TOTAL AIR TEMPERATURE	F3XP_TAT_MGR
GROSS WEIGHT	F3XP_GW_MGR

Section 8.1 COMPILE TIME GRAPHICS FILES

The files covered in this section define core graphics data constants. They create system level data base tables, which define the format to the display units, and constant CGS objects for static graphic items. The blocks of data defined in these files will be loaded into the processor's EEPROM memory for data integrity.

FILE: COMPILE.ASM

COMPILE.ASM contains the hardcoded data for some of the engine display's compile time graphic objects (SYS_TABS.ASM contains the system level compile time objects). The graphic objects included in the file consist of the radials and limit ticks for the engine pressure ratio (EPR), exhaust gas temperature (EGT), and N1 graphics functions; the fuel flow (FF) radial and the EPR command objects; and the reverse thruster armed object. The EPR, EGT, and N1 radials have three variants each for red, amber, and white radials.

COMPILE.ASM also contains two graphic objects which will output the labels for the different functions on the engine display. The first contains the labels for the left side of the engine display - the EPR, EGT, N1, and FF functions. The other will output the labels for the right side of the engine display - the reverse thrusters, oil, N2, fuel quantity, air temperature, and gross weight functions.

Nil variants of some of the graphic objects are also found here. When enqueued they will take away the appropriate graphics from the screen. The nil objects in this file are for the EPR, EGT, and N1 radials and limit ticks, the FF radial, the reverse thruster graphics, and the total air temperature and gross weight values.

One other important graphic object found in COMPILE.ASM is the engine character set. This consists of the binary data that defines the characters unique to the engine display. They will be used in building the graphic objects. Some of the characters in this set (CS3_DEU) have already been referenced in the compile time objects mentioned above.

FILE: SYS_TABS.ASM

The system tables file SYS_TABS.ASM contains six basic compile time objects required by CGS to build a new format. These objects are enqueued by F3_EXC at the slowest rate, and are as follows:

(1) TRANSLATION TABLE
(2) EVENT LIST
(3) OBJECT TABLE DEFINITION (OTD)
(4) TOP OBJECT
(5) SCREEN CENTER TRANSFORMATION
(6) NEW FORMAT SETUP

(1) TRANSLATION TABLE. Translation tables are used to define character sets. They can combine characters from several sources to form a new set. For the engine format three sets are defined; small, medium, and large. In PROM, several different sizes of standard alphanumeric characters are stored, from which the engine character sets just mentioned map their entries. The object table definition will associate a unique character set number to each set defined. For further information on translation tables consult Section 4.4 in the CGS manual.

(2) EVENT LIST. The event list is used to describe objects that are affected by dynamically changing transforms. Each entry in the event list represents a transform, and within that entry a list of objects that will be repositioned based on that particular transform.

For the engine format the following graphic objects will be moving symbols: the EPR, N1, and EGT left and right radials, and left and right limit symbols; the EPR left and right command symbols; and the fuel flow left and right radials. The object table definition will associate these graphic objects with a transform. The transforms also have entries in the OTD which will map them to the proper entry in the event list.

Some graphic objects have been associated with a transform in the OTD but are not listed in the event list. This would include compile time transforms since they do not dynamically change. For more information on event lists consult Section 3.5 in the CGS manual.

(3) OBJECT TABLE DEFINITION. The object table definition defines the engine format to the CGS, and the resources needed to support it. It consists of a series of object table entry macros, one for each object in the format.

Entries are listed in ascending order with the new format setup always as object number 1. Special macros are used to define the new format setup, OTD table, event list, translation tables, and the top object. Also found are entries for the screen center transformation and a user defined character set, CS3_DEU, which contains symbols unique to the engine display.

Following these entries are listed the transformation and graphic objects for the engine format. As mentioned before, graphic objects are associated with a transform (the screen center transformation if no other transform affects it), and transforms are mapped to event list entries (except for compile time transforms which use a special code indicating no action is taken). An important note is that transforms must have lower object numbers than the graphic objects they affect.

The buffering method required for the objects, static or pingpong, is also set up here. Basically, objects that move require more Vector Generator (VG) memory to account for their constant changes in size - unfortunately the pingpong method requires double the VG memory. Therefore, switching from pingpong to static is done wherever possible to conserve memory.

For further information on building an OTD, and the individual OTE macros consult Section 3.4 of the CGS manual.

(4) TOP OBJECT. The top object identifies all the graphic picture objects for the engine format. Each entry essentially is a call to display that object on the DU. Entries are made up of both run time and compile time objects. Despite the fact that an object may be enqueued, if it is not listed in the top object it will not be displayed. Section 4.1 in the CGS manual describes the top object in more detail.

(5) SCREEN CENTER TRANSFORMATION. The screen center transformation is a required CGS object. It can also be used in the OTD by graphic objects that do not move. In the engine format its position is set to (0,0).

(6) NEW FORMAT SETUP. The new format setup starts the initialization of a new format. It is always the first object block processed by CGS, and contains resource information necessary to display the format - such as the total number of objects, the unique format ID, the number of OTD and event list tables, and the object number of the top object. It also specifies the colors that are used for stroke and raster symbology, even though raster is not used currently in the engine format. Refer to Section 3.6 of the the CGS manual for further details on the new format setup and its parameters.

Section 8.2 RUN TIME GRAPHICS ROUTINES

This section describes the PLM86 modules that perform graphic operations for dynamic display objects on the ENG format.

MODULE NAME: F3GP_EPR_MGR
FILE NAME: EPR_MGR.PLM
CALL RATE: 0.1 SECONDS
CALLED BY: EXECUTIVE

PURPOSE: To control the processing required to display the engine pressure ratio data for the left and right engines.

REMARKS: This subroutine acts as the manager for the engine pressure ratio (EPR) display, which is a subset of the engine format. The EPR graphic function is located on the upper left side of the picture. The left and right EPR displays are output side by side. Each EPR display consists of an arc, a radial pointing to some place along the arc, and a box containing the EPR value. Code exists to display a commanded EPR value as a small triangle whose apex runs along the outer edge of the arc, but the code is not referenced in our current system. All the symbology just mentioned is only shown as long as the input EPR value is within the valid range. When the input EPR value is not valid, a red "X" is shown in place of the dial symbology. Although only a boolean is checked within this routine to see if the input is good, the acceptable range is known to be between 0.8 and 2.5 (which correspond to the start and end positions of the arc).

There are four procedures other than the manager that are used specifically to build the EPR display. They are: F3TP_EPR (creates transform for limit/radial/command), F3GP_EPR_DRAW_ARC (determines the number of arc segments), F3GP_EPR_ARC (draws arc), and F3GP_EPR_DIGITS (outputs EPR value). This routine calls F3GP_EPR_ARC which in turn calls the other procedures.

GLOBAL REFERENCES:

VARIABLES
ENG_EPR_L ENG_EPR_LVAL ENG_EPR_R ENG_EPR_RVAL F3GE_EPR
F3TC_EPR_LCOMMAND F3TC_EPR_LLIMIT_IN
F3TC_EPR_LRADIAL_I F3TC_EPR_RCOMMAND
F3TC_EPR_RLIMIT_IN F3TC_EPR_RRADIAL_I F3_FCB

PROCEDURES
END_OBJECT_BLOCK ENQ_OBJECT_BLOCK F3GP_EPR_ARC
IPVGSUB_CSTRING NEW_OBJECT_BLOCK U_MULT VID_PRI

MODULE NAME: F3GP_EPR_ARC
FILE NAME: EPR_MGR.PLM
CALL RATE: 0.1 SECONDS (if EPR valid)
CALLED BY: F3GP_EPR_MGR
PARAMETERS: ENG_FLG, ANGLE, EPR_VALUE

PURPOSE: To perform the processing necessary to display the EPR arc, limit tick, readout value, moving radial, and commanded EPR symbols.

REMARKS: F3GP_EPR_ARC is called twice within a frame, once for each engine, but only when the EPR input value is within the valid range. This module is responsible for coordinating and calling three smaller procedures which are used to draw the EPR display symbols. They are: F3GP_EPR_DRAW_ARC, which draws the colored arc segments; F3GP_EPR_DIGITS, which outputs the appropriately colored readout value; and F3TP_EPR, which creates the transforms that move the radial arm, limit tick, and commanded EPR symbols to their correct positions along the arc.

F3GP_EPR_ARC determines where the normal, caution, and warning areas are on the arc, if their limit values are valid. It passes as arguments to the routine that draws arc segments, F3GP_EPR_DRAW_ARC, the appropriate color and the beginning and ending points of each region. F3GP_EPR_ARC also determines where the radial and commanded EPR symbols point to on the arc. The radial arm and EPR readout box and digits will be colored the same as the region pointed to by the radial. The commanded EPR is represented as a triangle that runs along the outer edge of the arc with its apex touching it, and is always colored cyan.

The EPR limit tick is always drawn in red at the starting position of the warning region.

There are three parameters to this routine: the left/right engine flag, the radial symbol's rotation angle, and the input EPR value.

GLOBAL REFERENCES:

VARIABLES
AFD_ENGAGED ENG_EPRCAU ENG_EPRCAUV ENG_EPRCOM
ENG_EPRCOMV ENG_EPRLIM ENG_EPRLIMV F3GC_EPR_LRADIAL_A
F3GC_EPR_LRADIAL_R F3GC_EPR_LRADIAL_W
F3GC_EPR_RRADIAL_A F3GC_EPR_RRADIAL_R
F3GC_EPR_RRADIAL_W F3TC_EPR_LCOMMAND_
F3TC_EPR_LLIMIT_IN F3TC_EPR_RCOMMAND_
F3TC_EPR_RLIMIT_IN F3_FCB

PROCEDURES
ENQ_OBJECT_BLOCK F3GP_EPR_DIGITS F3GP_EPR_DRAW_ARC
F3TP_EPR MOVE_ABS_XY_BLANK U_MULT

MODULE NAME: F3GP_EPR_DRAW_ARC
FILE NAME: EPR_MGR.DOC
CALLED BY: F3GP_EPR_ARC
PARAMETERS: COLOR, START, STOP

PURPOSE: To draw all or part of the arc for the EPR display function.

REMARKS: F3GP_EPR_DRAW_ARC may be called by F3GP_EPR_ARC a number of times to draw the complete EPR arc for both the left and right engines. This module references the CGS utility procedure DRAW_ARC_REL_1 which outputs an arc section relative to the current screen position. The inputs to the utility are: direction to draw (counterclockwise or clockwise), start angle, stop angle, radius from start position, and number of line segments used to form the arc. Separate calls to F3GP_EPR_DRAW_ARC are made to draw each colored area of the arc (ie., normal, caution, and warning regions). The number of line segments used to draw each arc section is figured based on the size of the section. For each 8.4 degrees of difference between the start and stop angles of the arc an extra line segment is used, with a minimum of at least three lines. When all the arc sections have been combined, the complete EPR arc will have been drawn a fixed distance, clockwise, from 0 to 150 degrees.

There are three parameters to this routine: the color of the arc section to be drawn, and the starting and ending angles of the arc.

GLOBAL REFERENCES:

PROCEDURES
DRAW_ARC_REL_1 VID_PRI

MODULE NAME: F3GP_EPR_DIGITS
FILE NAME: EPR_MGR.PLM
CALL RATE: 0.1 SECONDS (if EPR valid)
CALLED BY: F3GP_EPR_ARC
PARAMETERS: ENG_FLG, COLOR, EPR_VALUE

PURPOSE: To output the digital engine pressure ratio value and its display box.

REMARKS: F3GP_EPR_DIGITS is called twice within a frame, once for each engine, and only when the input EPR value is inside the valid range. The box and digits are output right above the starting position of the EPR arc. Their color will be white, amber, or red depending on whether the input value is in the normal, caution, or warning area respectively. The format for the numeric value is X.XX, and it is displayed in medium size characters.

Code exists in this module to display a commanded EPR value directly above the input EPR readout box, however this code is not referenced in our current system. If the commanded value were shown it would be displayed in small size cyan colored characters, provided that it was inside the valid range.

There are three parameters to this routine: the left/right engine flag, the color of the box and readout, and the input EPR value.

GLOBAL REFERENCES:

PROCEDURES
IPVGSUB_CSTRING U_BNRBCD U_INSERT_DEC_PT U_MULT
VGSUB_CSTRING VID_PRI

MODULE NAME: F3TP_EPR
FILE NAME: EPR_MGR.PLM
CALLED BY: F3GP_EPR_ARC
PARAMETERS: ENG_FLG, XFRM_TYPE, ANGLE

PURPOSE: To create and enqueue the transform for the engine pressure ratio radial symbol, limit tick, and commanded EPR triangle.

REMARKS: F3TP_EPR can potentially be called six times within a single frame by F3GP_EPR_ARC to create the transforms for the EPR radial, limit tick, and commanded EPR triangle, for both the left and right engines. However, even though the code exists to display and transform the commanded EPR symbol, it is not referenced in our current system. The other two symbols are displayed. The radial is transformed to point to the place along the EPR arc that represents the input EPR value. The limit tick is moved to the spot on the arc that indicates the start of the warning region.

There are three parameters to this routine: the left/right engine flag, an index indicating which object is to be transformed (radial, limit tick, or commanded epr triangle), and the object's rotation angle.

GLOBAL REFERENCES:

VARIABLES
  F3_FCB

PROCEDURES
  ENQ_OBJECT_BLOCK MAKE_XFORM_ONE ROTANG

MODULE NAME: F3GP_N1_MGR
FILE NAME: N1_MGR.PLM
CALL RATE: 0.1 SECONDS
CALLED BY: EXECUTIVE

PURPOSE: To control the processing required to display the N1 percentage data for the left and right engines.

REMARKS: This subroutine acts as the manager for the N1 display, which is a subset of the engine format. The N1 graphic function is located on the upper middle side of the picture. The left and right N1 displays are output side by side. Each N1 display consists of an arc, a radial pointing to some place along the arc, and a box containing the N1 value; as long as the input N1 value is within the valid range. When the input N1 value is not valid, a red "X" is shown in place of the dial symbology. Although only a boolean is checked within this routine to see if the input is good, the acceptable range is known to be between 0 and 115 percent (which corresponds to the start and end positions of the arc).

There are four procedures other than the manager that are used specifically to build the N1 display. They are: F3TP_N1 (creates transform for limit/radial), F3GP_N1_ARC (draws arc), F3GP_N1_DRAW_ARC (determines number of arc segments), and F3GP_N1_DIGITS (outputs N1 value). This routine call F3GP_N1_ARC which in turn calls the other procedures.

GLOBAL REFERENCES:

VARIABLES
ENG_N1_L ENG_N1_LVAL ENG_N1_R ENG_N1_RVAL F3GE_N1
F3TC_N1_LLIMIT_INV F3TC_N1_LRADIAL_IN
F3TC_N1_RLIMIT_INV F3TC_N1_RRADIAL_IN F3_FCB

PROCEDURES
END_OBJECT_BLOCK ENQ_OBJECT_BLOCK F3GP_N1_ARC
IPVGSUB_CSTRING NEW_OBJECT_BLOCK U_MULT VID_PRI

MODULE NAME:	F3GP_N1_ARC
FILE NAME:	N1_MGR.PLM
CALL RATE:	0.1 SECONDS (if N1 valid)
CALLED BY:	F3GP_N1_MGR
PARAMETERS:	ENG_FLG, ANGLE, N1_VALUE

PURPOSE: To perform the processing necessary to display the N1 arc, limit tick, readout value, and moving radial symbol.

REMARKS: F3GP_N1_ARC is called twice within a frame, once for each engine, but only when the N1 input value is within the valid range. This module is responsible for coordinating and calling three smaller procedures which are used to draw the N1 display symbols. They are: F3GP_N1_DRAW_ARC, which draws the colored arc segments; F3GP_N1_DIGITS, which outputs the appropriately colored readout value; and F3TP_N1, which creates the transforms that move the radial arm and limit tick to the correct positions along the arc.

F3GP_N1_ARC determines where the normal, caution, and warning areas are on the arc, if their limit values are valid. It passes as arguments to the routine that draws arc segments, F3GP_N1_DRAW_ARC, the appropriate color and the beginning and ending points of each region. F3GP_N1_ARC also determines where the radial symbol points to on the arc. The radial symbol represents the actual input N1 value. The radial arm and N1 readout box and digits will be colored the same as the region pointed to by the radial. However, there is one exception to that rule. When the aft flight deck is not engaged, the radial and readout are prevented from turning red even though the radial may point to somewhere in the warning zone.

The N1 limit tick is always drawn in red at the starting position of the warning region.

There are three parameters to this routine: the left/right engine flag, the angle to rotate the radial symbol, and the input N1 value.

GLOBAL REFERENCES:

VARIABLES
AFD_ENGAGED ENG_N1CAU ENG_N1CAUVAL ENG_N1LIM
ENG_N1LIMVAL F3GC_N1_LRADIAL_AM F3GC_N1_LRADIAL_RE
F3GC_N1_LRADIAL_WH F3GC_N1_RRADIAL_AM
F3GC_N1_RRADIAL_RE F3GC_N1_RRADIAL_WH
F3TC_N1_LLIMIT_INV F3TC_N1_RLIMIT_INV F3_FCB

PROCEDURES
ENQ_OBJECT_BLOCK F3GP_N1_DIGITS F3GP_N1_DRAW_ARC
F3TP_N1 MOVE_ABS_XY_BLANK U_MULT

MODULE NAME: F3GP_N1_DRAW_ARC
FILE NAME: N1_MGR.PLM
CALLED BY: F3GP_N1_ARC
PARAMETERS: COLOR, START, STOP

PURPOSE: To draw all or part of the arc for the N1 display function.

REMARKS: F3GP_N1_DRAW_ARC may be called by F3GP_N1_ARC a number of times to draw the complete N1 arc for both the left and right engines. This module references the CGS utility procedure DRAW_ARC_REL_1 which outputs an arc section relative to the current screen position. The inputs to the utility are: direction to draw (counterclockwise or clockwise), start angle, stop angle, radius from start position, and number of line segments used to form the arc. Separate calls to F3GP_N1_DRAW_ARC are made to draw each colored area of the arc (ie., normal, caution, and warning regions). The number of line segments used to draw each arc section is figured based on the size of the section. For each 8.4 degrees of difference between the start and stop angles of the arc an extra line segment is used, with a minimum of at least three lines. When all the arc sections have been combined, the complete N1 arc will have been drawn a fixed distance, clockwise, from 0 to 150 degrees.

There are three parameters to this routine: the color of the arc section to be drawn, the starting angle, and the ending angle.

GLOBAL REFERENCES:

PROCEDURES
DRAW_ARC_REL_1 VID_PRI

MODULE NAME: F3GP_N1_DIGITS
FILE NAME: N1_MGR.PLM
CALL RATE: 0.1 SECONDS (if N1 valid)
CALLED BY: F3GP_N1_ARC
PARAMETERS: ENG_FLG, COLOR, N1_VALUE

PURPOSE: To output the digital N1 value and its display box.

REMARKS: F3GP_N1_DIGITS is called twice within a frame, once for each engine, and only when the N1 input value is inside the valid range. The box and digits are output right above the starting position of the N1 arc. Their color will be white, amber, or red depending on whether the input value is in the normal, caution, or warning area respectively. Two formats are possible for the numeric value. If it is below 100 percent the format is XX.X, with at least two characters displayed. If the N1 value is 100 percent or more then the format is XXX with no fractional part to the text. The numeric value is output in medium size characters.

There are three parameters to this routine: the left/ right engine flag, the color used to draw the readout and box, and the input N1 value.

GLOBAL REFERENCES:

PROCEDURES
IPVGSUB_CSTRING U_BNRBCD U_INSERT_DEC_PT U_MULT
VGSUB_CSTRING VID_PRI

MODULE NAME: F3TP_N1
FILE NAME: N1_MGR.PLM
CALLED BY: F3GP_N1_ARC
PARAMETERS: ENG_FLG, XFRM_TYPE, ANGLE

PURPOSE: To create and enqueue the transform for the N1 radial symbol and limit tick.

REMARKS: F3TP_N1 can potentially be called four times within a frame by F3GP_N1_ARC; twice to create the transforms for the left and right engine radial symbols, and twice to create the transforms for the left and right engine limit tick symbols. The radial is transformed to point to the place along the N1 arc that represents the input N1 value. The limit tick is moved to the spot on the arc that indicates the start of the warning region.

There are three parameters to this routine: the left right engine flag, the object to be transformed (radial or limit tick, and the object's rotation angle.

GLOBAL REFERENCES:

VARIABLES
F3_FCB

PROCEDURES
ENQ_OBJECT_BLOCK MAKE_XFORM_ONE

MODULE NAME:	F3GP_EGT_MGR
FILE NAME:	EGT_MGR.PLM
CALL RATE:	0.1 SECONDS
CALLED BY:	EXECUTIVE

PURPOSE: To control the processing required to display the exhaust gas temperature data for the left and right engines.

REMARKS: This subroutine acts as the manager for the exhaust gas temperature (EGT) display, which is a subset of the engine format. The EGT graphic function is located on the lower middle left side of the picture. The left and right EGT displays are output side by side. Each EGT display consists of an arc, a radial pointing to some place along the arc, and a box containing the EGT value; as long as the input EGT value is within the valid range. When the input EGT value is not valid, a red "X" is shown in place of the dial symbology. Although only a boolean is checked within this routine to see if the input is good, the acceptable range is known to be between 0 and 600 degrees centigrade (which corresponds to the start and end positions of the arc).

There are four procedures other than the manager that are used specifically to build the EGT display. They are: F3TP_EGT (creates transform for limit/radial), F3GP_EGT_ARC (draws arc), F3GP_EGT_DRAW_ARC (determines number of arc segments), and F3GP_EGT_DIGITS (outputs EGT value). This routine calls F3GP_EGT_ARC which in turn calls the other procedures.

GLOBAL REFERENCES:

VARIABLES
ENG_EGT_L ENG_EGT_LVAL ENG_EGT_R ENG_EGT_RVAL F3GE_EGT
F3TC_EGT_LLIMIT_IN F3TC_EGT_LRADIAL_I
F3TC_EGT_RLIMIT_IN F3TC_EGT_RRADIAL_I F3_FCB

PROCEDURES
END_OBJECT_BLOCK ENQ_OBJECT_BLOCK F3GP_EGT_ARC
IPVGSUB_CSTRING NEW_OBJECT_BLOCK U_MULT VID_PRI

MODULE NAME: F3GP_EGT_ARC
FILE NAME: EGT_MGR.PLM
CALL RATE: 0.1 SECONDS (if EGT valid)
CALLED BY: F3GP_EGT_MGR
PARAMETERS: ENG_FLG, ANGLE, EGT_VALUE

PURPOSE: To perform the processing necessary to display the EGT arc, limit tick, readout value, and moving radial symbol.

REMARKS: F3GP_EGT_ARC is called twice within a frame, once for each engine, but only when the EGT input value is within the valid range. This module is responsible for coordinating and calling three smaller procedures which are used to draw the EGT display symbols. They are: F3GP_EGT_DRAW_ARC, which draws the colored arc segments; F3GP_EGT_DIGITS, which outputs the appropriately colored readout value; and F3TP_EGT, which creates the transforms that move the radial arm and limit tick to the correct positions along the arc.

F3GP_EGT_ARC determines where the normal, caution and warning areas are on the arc, if their limit values are valid. It passes as arguments to the routine that draws arc segments, F3GP_EGT_DRAW_ARC, the appropriate color and the beginning and ending points of each region. F3GP_EGT_ARC also determines where the radial symbol points to on the arc. The radial symbol represents the actual input EGT value. The radial arm and EGT readout box and digits will be colored the same as the region pointed to by the radial. The EGT limit tick is always drawn in red at the starting position of the warning region.

There are three parameters to this module: the left/right engine flag, the angle of rotation for the radial symbol, and the input EGT value.

GLOBAL REFERENCES:

VARIABLES
```
  ENG_EGTCAU ENG_EGTCAUV ENG_EGTLIM ENG_EGTLIMV
  F3GC_EGT_LRADIAL_A F3GC_EGT_LRADIAL_R
  F3GC_EGT_LRADIAL_W F3GC_EGT_RRADIAL_A
  F3GC_EGT_RRADIAL_R F3GC_EGT_RRADIAL_W
  F3TC_EGT_LLIMIT_IN F3TC_EGT_RLIMIT_IN F3_FCB
```

PROCEDURES
```
  ENQ_OBJECT_BLOCK F3GP_EGT_DIGITS F3GP_EGT_DRAW_ARC
  F3TP_EGT MOVE_ABS_XY_BLANK U_MULT
```

MODULE NAME: F3GP_EGT_DRAW_ARC
FILE NAME: EGT_MGR.PLM
CALLED BY: F3GP_EGT_ARC
PARAMETERS: COLOR, START, STOP

PURPOSE: To draw all or part of the arc for the EGT display function.

REMARKS: F3GP_EGT_DRAW_ARC may be called by F3GP_EGT_ARC a number of times to draw the complete EGT arc for both the left and right engines. This module references the CGS utility procedure DRAW_ARC_REL_1 which outputs an arc section relative to the current screen position. The inputs to the utility are: direction to draw (counterclockwise or clockwise), start angle, stop angle, radius from start position, and number of line segments used to form the arc. Separate calls to F3GP_EGT_DRAW_ARC are made to draw each colored area of the arc (ie., normal, caution, and warning regions). The number of line segments used to draw each arc section is figured based on the size of the section. For each 8.4 degrees of difference between the start and stop angles of the arc an extra line segment is used, with a minimum of at least three lines. When all the arc sections have been combined, the complete EGT arc will have been drawn a fixed distance, clockwise, from 0 to 150 degrees.

There are three parameters to this routine: the color of the arc section, the starting angle of the arc, and the ending angle of the arc.

GLOBAL REFERENCES:

PROCEDURES
DRAW_ARC_REL_1 VID_PRI

MODULE NAME: F3GP_EGT_DIGITS
FILE NAME: EGT_MGR.PLM
CALL RATE: 0.1 SECONDS (if EGT valid)
CALLED BY: F3GP_EGT_ARC
PARAMETERS: ENG_FLG, COLOR, EGT_VALUE

PURPOSE: To output the digital exhaust gas temperature value and its display box.

REMARKS: F3GP_EGT_DIGITS is called twice within a frame, once for each engine, and only when the EGT input value is within the valid range. The box and digits are output right above the starting position of the EGT arc. Their color will be white, amber, or red depending on whether the input value is in the normal, caution, or warning area respectively. Up to three digits can be displayed with at least one always showing. The numeric text is output in medium size characters.

There are three parameters to this routine: the left/right engine flag, the color that the box and readout will be displayed in, and the input EGT value.

GLOBAL REFERENCES:

PROCEDURES
IPVGSUB_CSTRING U_BNRBCD U_MULT VGSUB_CSTRING VID_PRI

MODULE NAME: F3TP_EGT
FILE NAME: EGT_MGR.PLM
CALLED BY: F3GP_EGT_ARC
PARAMETERS: ENG_FLG, XFRM_TYPE, ANGLE

PURPOSE: To create and enqueue the transform for the exhaust gas temperature radial symbol and limit tick.

REMARKS: F3TP_EGT can potentially be called four times within a frame by F3GP_EGT_ARC; twice to create the transforms for the left and right engine radial symbols, and twice to create the transforms for the left and right engine limit tick symbols. The radial is transformed to point to the place along the EGT arc that represents the input EGT value. The limit tick is moved to the spot on the arc that indicates the start of the warning region.

There are three parameters to this module: the left/right engine flag, the object to be transformed (radial or limit tick), and the object's rotation angle.

GLOBAL REFERENCES:

VARIABLES
F3_FCB

PROCEDURES
ENQ_OBJECT_BLOCK MAKE_XFORM_ONE

MODULE NAME:	F3GP_FF_MGR
FILE NAME:	ENG_MGRS.PLM
CALL RATE:	0.1 SECONDS
CALLED BY:	EXECUTIVE

PURPOSE: To display the rate of fuel consumption symbology for the left and right engines.

REMARKS: The fuel flow graphic function is a subset of the engine format, and is displayed on the lower left side of the picture. The left and right fuel flow displays are output side by side. Each fuel flow display consists of an arc, a radial pointing to some place along the arc, and a box containing the fuel flow value; as long as the input fuel flow value is within the valid range. Although only a boolean is checked within this routine to see if the input is good, the acceptable range is known to be between 0 and 12000 pounds per hour (which corresponds to the start and end positions of the arc). The arc is drawn clockwise from 0 to 150 degrees, and is always shown in white. If the input fuel flow value falls within the valid range it is displayed in the box and the radial points to the appropriate place along the arc. The transformation for the radial symbol is contained in this module. A positive change in value causes the radial to move clockwise. If invalid input is received a large red "X" replaces the dial symbology.

F3GP_FF_MGR calls the procedure F3GP_FF_DIGITS to output the fuel flow value and box.

GLOBAL REFERENCES:

VARIABLES
ENG_FF_L ENG_FF_LVAL ENG_FF_R ENG_FF_RVAL F3GE_FF
F3TC_FF_LRADIAL_IN F3TC_FF_RRADIAL_IN F3_FCB

PROCEDURES
DRAW_ARC_REL_1 END_OBJECT_BLOCK ENQ_OBJECT_BLOCK
F3GP_FF_DIGITS IPVGSUB_CSTRING MAKE_XFORM_ONE
MOVE_ABS_XY_BLANK NEW_OBJECT_BLOCK U_MULT VID_PRI

```
MODULE NAME:   F3GP_FF_DIGITS
FILE NAME:     ENG_MGRS.PLM
CALL RATE:     0.1 SECONDS (if fuel flow valid)
CALLED BY:     F3GP_FF_MGR
PARAMETERS:    ENG_FLG, FUEL_FLOW
```

PURPOSE: To output the digital fuel flow value and box.

REMARKS: F3GP_FF_DIGITS is called twice by F3GP_FF_MGR, once for each engine, and only when the fuel flow input value is within the valid range. The box and digits are output right above the starting position of the arc, and both are shown in white. The fuel flow value can have up to five digits displayed. The two least significant digits are output in small size characters, and the three upper digits are output in medium size characters. The three least significant digits will always be shown even if they are zeroes. There are two input parameters to this routine: the left/right engine indicator, and the fuel flow value to be displayed.

GLOBAL REFERENCES:

PROCEDURES
  IPVGSUB_CSTRING U_BNRBCD U_MULT VGSUB_CSTRING VID_PRI

```
MODULE NAME:  F3XP_RA_MGR
FILE NAME:    ENG_MGRS.PLM
CALL RATE:    0.2 SECONDS
CALLED BY:    EXECUTIVE
```

PURPOSE: To display the thrust reversers armed symbology.

REMARKS: The left and right thrust reversers armed messages appear on the upper right side of the engine display. This module simply checks to see if either engine is in an armed state, and enqueues a compile time object to display the message symbology if armed, or blanks out the message area if unarmed. The symbology consists of a box with the text 'REV ARM' shown inside. The box and text are displayed in cyan, and the text is output in medium size characters.

GLOBAL REFERENCES:

VARIABLES
ENG_LREVARM ENG_RREVARM F3GC_REVARML_NIL
F3GC_REVARMR_NIL F3GC_REVARM_L F3GC_REVARM_R F3_FCB

PROCEDURES
ENQ_OBJECT_BLOCK

MODULE NAME: F3GP_OIL_MGR
FILE NAME: OIL_MGR.PLM
CALL RATE: 0.2 SECONDS
CALLED BY: EXECUTIVE

PURPOSE: To control the processing required to display the oil pressure, oil temperature, and oil quantity.

REMARKS: The engine oil display is located on the upper right side of the engine format. It consists of six boxes which display the left and right engine oil pressure, temperature, and quantity readouts. The boxes and readouts change colors depending on what range the input value is in (normal, caution, or warning). Part of the oil function symbology is contained inside a compile time graphic object which is enqueued, by the module EXECUTIVE, every 0.8 seconds. The compile time object puts out the labels 'OIL', 'P', 'T', and 'Q', and also the two arms that extend from each side of the 'OIL' header to the left and right engine oil boxes. The oil readouts (which include the digital values or a failure annunciation) and the oil boxes are put out by the procedures that this routine calls. F3GP_OIL_FAIL_ANN is called to output the failure annunciation. F3GP_OILPSI_DATA, F3GP_OILTMP_DATA, and F3GP_OILQTY_DATA are called to output the oil pressure, temperature, and quantity readouts respectively. Input flags are checked to determine if the oil parameters are within their valid ranges. Although it is not explicitly stated in this module, the valid ranges are as follows: oil pressure, 0 to 99 psi; oil temperature, 0 to 180 degrees centigrade; oil quantity, 0 to 5 gallons.

GLOBAL REFERENCES:

VARIABLES
ENG_EOP_L ENG_EOP_LVAL ENG_EOP_R ENG_EOP_RVAL
ENG_EOQ_L ENG_EOQ_LVAL ENG_EOQ_R ENG_EOQ_RVAL
ENG_EOT_L ENG_EOT_LVAL ENG_EOT_R ENG_EOT_RVAL F3GE_OIL
F3_FCB

PROCEDURES
END_OBJECT_BLOCK ENQ_OBJECT_BLOCK F3GP_OILPSI_DATA
F3GP_OILQTY_DATA F3GP_OILTMP_DATA F3GP_OIL_FAIL_ANN
NEW_OBJECT_BLOCK

MODULE NAME:	F3GP_OILPSI_DATA
FILE NAME:	OIL_MGR.PLM
CALL RATE:	0.2 SECONDS (if oil pressure valid)
CALLED BY:	F3GP_OIL_MGR
PARAMETERS:	OIL_PRESSURE, SCREEN_X

PURPOSE: To display an engine oil pressure readout and its box.

REMARKS: F3GP_OILPSI_DATA is called by the engine oil display manager, F3GP_OIL_MGR, when an input oil pressure value is within its valid range. It may be called twice within a frame, once each for the left and right engines. This module formats the scaled input value, and calls the graphic functions necessary to display the digital readout and box. The readout is output in medium size characters in the format XX which represents pounds per square inch. The readout and box are colored white, amber, or red depending on whether the input oil pressure value is within the normal, caution, or warning range respectively. The ranges are as follows: normal, 40 to 55 psi; caution, 35 to 40 psi; warning, 0 to 35 psi, or 55 to 99 psi. If the input oil pressure limit has been determined to be invalid then the box and readout will be colored amber regardless of what the input oil pressure value is.

There are two parameters to this routine: the input oil pressure value, and the screen x-coordinate used to position the readout and box.

GLOBAL REFERENCES:

VARIABLES
ENG_EOLIMVAL ENG_EOLLIM ENG_EOULIM

PROCEDURES
IPVGSUB_CSTRING U_BNRBCD U_MULT VGSUB_CSTRING VID_PRI

MODULE NAME:	F3GP_OILTMP_DATA
FILE NAME:	OIL_MGR.PLM
CALL RATE:	0.2 SECONDS (if oil temperature valid)
CALLED BY:	F3GP_OIL_MGR
PARAMETERS:	OIL_TEMPERATURE, SCREEN_X

PURPOSE: To display an engine oil temperature readout and its box.

REMARKS: F3GP_OILTMP_DATA is called by the engine oil display manager, F3GP_OIL_MGR, when an input oil temperature value is within its valid range. It may be called twice within a frame, once each for the left and right engines. This module formats the scaled input value, and calls the graphic functions necessary to display the digital readout and box. The readout is output in medium size characters in the format XXX which represents degrees centigrade. The readout and box are colored white, amber, or red depending on whether the input oil temperature value is within the normal, caution, or warning range respectively. The ranges are as follows: normal, 40 to 120 degrees C; caution, 120 to 157 degrees C; warning, 0 to 40 degrees C, and 157 to 180 degrees C. If the input oil temperature limit has been determined to be invalid then the box and readout will be colored amber regardless of what the input oil temperature value is.

There are two parameters to this routine: the input oil temperature value, and the screen x-coordinate used to position the readout and box.

GLOBAL REFERENCES:

VARIABLES
ENG_EOTLIM ENG_EOTLIMVA

PROCEDURES
IPVGSUB_CSTRING U_BNRBCD U_MULT VGSUB_CSTRING VID_PRI

MODULE NAME: F3GP_OILQTY_DATA
FILE NAME: OIL_MGR.PLM
CALL RATE: 0.2 SECONDS (if oil quantity is valid)
CALLED BY: F3GP_OIL_MGR
PARAMETERS: OIL_QTY, SCREEN_X

PURPOSE: To display an engine oil quantity readout and its box.

REMARKS: F3GP_OILQTY_DATA is called by the engine oil display manager, F3GP_OIL_MGR, when an input oil quantity value is within its valid range. It may be called twice within a frame, once each for the left and right engines. This module formats the scaled input value, and calls the graphic functions necessary to display the digital readout and box. The readout is output in medium size characters in the format X.X which represents gallons of oil. The readout and box are colored white or red depending on whether the input oil quantity value is within the normal or warning range respectively. The ranges are as follows: normal, 1 to 5 gallons; warning 0 to 1 gallons. There is no caution range for the oil display, however, if the input oil quantity limit has been determined to be invalid then the box and readout will be colored amber regardless of what the input oil quantity value is.

There are two parameters to this routine: the input oil quantity value, and the screen x-coordinate used to position the readout and box.

GLOBAL REFERENCES:

VARIABLES
ENG_EOQLIM ENG_EOQLIMVA

PROCEDURES
IPVGSUB_CSTRING U_BNRBCD U_INSERT_DEC_PT U_MULT
VGSUB_CSTRING VID_PRI

MODULE NAME:	F3GP_OIL_FAIL_ANN
FILE NAME:	OIL_MGR.PLM
CALLED BY:	F3GP_OIL_MGR
PARAMETERS:	SCREEN_X, SCREEN_Y

PURPOSE: To display the failure annunciator for the engine oil parameters.

REMARKS: F3GP_OIL_FAIL_ANN is called by the engine oil display manager, F3GP_OIL_MGR, when an input oil pressure, oil temperature, or oil quantity value for either the left or right engine has been determined to be outside its valid range. The oil failure annunciator consists of a white readout box with a red 'X' displayed in place of the digital value. There are two parameters to this routine: the screen x-coordinate, and screen y-coordinate positions that specify where the failure text is to be displayed.

GLOBAL REFERENCES:

PROCEDURES
  IPVGSUB_CSTRING VID_PRI

MODULE NAME: F3GP_N2_MGR
FILE NAME: ENG_MGRS.PLM
CALL RATE: 0.2 SECONDS
CALLED BY: EXECUTIVE

PURPOSE: To control the processing required to display the N2 readouts and symbology.

REMARKS: The N2 display is located on the middle right side of the engine format. Its symbology consists of a box and readout for both the left and right engines, and a tag 'N2' above the readouts with arms extending outward from the tag to each box. The graphics for the tag and arms are contained in a compile time object that is enqueued by the module EXECUTIVE every 0.8 seconds. The readout, failure annunciation, and color choice are determined real-time by this routine and the ones it calls.

F3GP_N2_MGR is responsible for calling the routines F3GP_N2 or F3GP_FAIL_ANN to output either the N2 readout or failure annunciation respectively. The failure annunciation consists of a large red 'X' output inside the readout box, and is shown when the input N2 value is outside its valid range. Although it is not explicity stated in this module, the valid N2 range is 0 to 115 percent.

GLOBAL REFERENCES:

VARIABLES
ENG_N2_L ENG_N2_LVAL ENG_N2_R ENG_N2_RVAL F3GE_N2
F3_FCB

PROCEDURES
END_OBJECT_BLOCK ENQ_OBJECT_BLOCK F3GP_N2
F3GP_N2_FAIL_ANN NEW_OBJECT_BLOCK

MODULE NAME: F3GP_N2
FILE NAME: ENG_MGRS.PLM
CALLED BY: F3GP_N2_MGR
PARAMETERS: N2_VALUE, SCREEN_X

PURPOSE: To display the N2 digital readout and its box.

REMARKS: F3GP_N2 is called by the N2 display manager, F3GP_N2_MGR, when an input N2 value for either the left or right engine is within its valid range. This module formats the input value, and calls the graphic functions necessary to display the digital readout and box. The readout and box will be colored white, amber, or red depending on whether the input N2 value is within the normal, caution, or warning areas respectively. The ranges are as follows: normal, 0 to 94 percent; caution, 94 to 100 percent; warning, 100 to 115 percent. The digital readout will be displayed in medium sized characters in the format XX.X when the N2 value is below 100 percent, and in the format XXX when it is equal to or above 100 percent.

There are two parameters to this routine: the input N2 value, and the screen x-coordinate position for the N2 box and readout.

GLOBAL REFERENCES:

VARIABLES
ENG_N2LIM ENG_N2LIMV

PROCEDURES
IPVGSUB_CSTRING U_BNRBCD U_INSERT_DEC_PT U_MULT
VGSUB_CSTRING VID_PRI

MODULE NAME: F3GP_N2_FAIL_ANN
FILE NAME: ENG_MGRS.PLM
CALLED BY: F3GP_N2_MGR
PARAMETERS: SCREEN_X

PURPOSE: To display the N2 failure annunciator.

REMARKS: F3GP_N2_FAIL_ANN is called by the N2 display manager, F3GP_N2_MGR, when an input N2 value for either the left or right engine has been determined to be outside its valid range. The N2 failure annunciator consists of a white readout box with a red 'X' in place of the digital value. There is one parameter to this routine, which is the screen x-coordinate position for the failure text.

GLOBAL REFERENCES:

PROCEDURES
IPVGSUB_CSTRING VID_PRI

MODULE NAME: F3GP_FUELQ_MGR
FILE NAME: ENG_MGRS.PLM
CALL RATE: 0.2 SECONDS
CALLED BY: EXECUTIVE

PURPOSE: To control the processing required to display the left, right, and center tank fuel quantities, and also the total fuel quantity.

REMARKS: The fuel quantities are displayed on the lower right side of the engine display above the total air temperature and gross weight. Part of the fuel quantity function symbology is contained inside a compile time graphic object which is enqueued by the module EXECUTIVE every 0.8 seconds. The compile time object puts out the four fuel quantity boxes and the labels 'FUEL', 'C', 'L', 'R', and 'TFQ'. The fuel readouts (which includes the digital value, or dashes indicating the input value is invalid) are put out by the two procedures that this module calls. Those procedures are F3GP_FUELQ_CRL and F3GP_FUELQ_FAIL_ANN. For the left, right, and center fuel tanks F3GP_FUELQ_MGR checks the appropriate input valid flags to determine which procedure to call. The processing for the total fuel quantity digital value is contained inline in this routine. The total quantity is the sum of the left, right, and center tanks.

Although it is not explicity stated in this module, the valid ranges for the four fuel quantities are as follows: left and right tanks - 0 to 9260 pounds; center tank - 0 to 8930 pounds; total quantity - 0 to 27300 pounds.

GLOBAL REFERENCES:

VARIABLES
ENG_CFLQVAL ENG_CFUELQ ENG_LFLQVAL ENG_LFUELQ
ENG_RFLQVAL ENG_RFUELQ F3GE_FUEL F3_FCB

PROCEDURES
END_OBJECT_BLOCK ENQ_OBJECT_BLOCK F3GP_FUELQ_CRL
F3GP_FUELQ_FAIL_AN IPVGSUB_CSTRING NEW_OBJECT_BLOCK
U_BNRBCD

MODULE NAME: F3GP_FUELQ_CRL
FILE NAME: ENG_MGRS.PLM
CALLED BY: F3GP_FUELQ_CRL
PARAMETERS: FUEL_QTY, SCREEN_X, SCREEN_Y

PURPOSE: To display the digital value for an input fuel quantity.

REMARKS: The manager of the fuel quantity display, F3GP_FUELQ_MGR, calls this procedure when an input left, right, or center tank fuel quantity is received that is within its predetermined valid range. F3GP_FUELQ_CRL calls the utility procedures that will display the digital value to the tenths accuracy, in white medium sized characters, within the appropriate readout box.

There are three parameters to this routine: the input fuel quantity value, and the screen x-coordinate and y-coordinate positions that specify where to display the value.

GLOBAL REFERENCES:

PROCEDURES
IPVGSUB_CSTRING U_BNRBCD VID_PRI

MODULE NAME: F3GP_FUELQ_FAIL_ANN
FILE NAME: ENG_MGRS.PLM
CALLED BY: F3GP_FUELQ_MGR
PARAMETERS: SCREEN_X, SCREEN_Y

PURPOSE: To display the fuel quantity invalid annunciator.

REMARKS: F3GP_FUELQ_FAIL_ANN is called by the fuel quantity display manager, F3GP_FUELQ_MGR, when an input fuel value for either the left, right, or center tanks has been determined to be outside its valid range. The fuel failure annunciator consists of four amber colored dashes, which are displayed inside the appropriate readout box in place of the digital value. There are two parameters to this routine: the screen x-coordinate, and screen y-coordinate positions that specify where the invalid text is to be displayed.

GLOBAL REFERENCES:

PROCEDURES
  IPVGSUB_CSTRING VID_PRI

MODULE NAME: F3XP_TAT_MGR
FILE NAME: ENG_MGRS.PLM
CALL RATE: 0.2 SECONDS
CALLED BY: EXECUTIVE

PURPOSE: To display the total air temperature.

REMARKS: The total air temperature appears on the lower right side of the engine display. It is displayed in the format XX.X which is a value in degrees centigrade. A '+' or '-' prefixes the numeric value. (Note that the plus sign had to be created and put into a user defined character set since it doesn't exist in PROM - although the hyphen does.) The tag 'TAT' is shown to the left of the numeric value. Both the tag and value are output in white medium size characters. Although only a boolean is checked within this routine to see if the input TAT value is valid, the acceptable range is known to be between -99.9 and +99.9 degrees centigrade. The numeric value will be blanked when it is invalid.

GLOBAL REFERENCES:

VARIABLES
ENG_TAT ENG_TATVAL F3GC_TAT_NIL F3GE_TAT F3_FCB

PROCEDURES
END_OBJECT_BLOCK ENQ_OBJECT_BLOCK IPVGSUB_CSTRING
NEW_OBJECT_BLOCK U_BNRBCD U_INSERT_DEC_PT U_MULT
VID_PRI

```
MODULE NAME:  F3XP_GW_MGR
FILE NAME:    ENG_MGRS.PLM
CALL RATE:    0.2 SECONDS
CALLED BY:    EXECUTIVE
```

PURPOSE: To display the gross weight of the airplane.

REMARKS: The gross weight is displayed on the lower right side of the engine display. It is displayed in the format XXXXXX which is a value in pounds. The tag 'GW' is shown to the left of the numeric value. Both the tag and value are output in white medium size characters. Although only a boolean is checked within this routine to see if the input weight value is valid, the acceptable range is known to be between 0 and 200,000 pounds. The numeric value will be blanked when it is invalid. Due to the scaling factor applied to the input gross weight by the host computer, the displayed numeric value will only update for every eight pounds of change.

GLOBAL REFERENCES:

VARIABLES
ENG_GW ENG_GWVAL F3GC_GW_NIL F3GE_GW F3_FCB

PROCEDURES
END_OBJECT_BLOCK ENQ_OBJECT_BLOCK IPVGSUB_CSTRING
NEW_OBJECT_BLOCK U_BNRBCD VID_PRI

Section 8.3 MEMORY ALLOCATION FILES

Most of the memory allocation modules reserve memory locations in RAM for various system and applications variables. VAR_RAM.ASM, GLBL_RAM.PLM, and IO_RAM.PLM allocate memory between the address ranges of 800H and 7FFFH in the display processor memory. IO_RAM.ASM reserves memory specifically between the ranges of 4000H and 7FFFH. Memory locations 0400H through 07FFH are reserved for the stack and the stack pointer is initialized to 0800H.

FILE: GLBL_RAM.PLM

This module allocates memory locations in RAM for the system level variables used by the format executive. The types of variables declared globally in this module include system error flags, counters and checksum variables.

FILE: INT_TBL.ASM

This module reserves memory locations 0000H through 03FFH for the interrupt vectors. There are two words of data per interrupt. The first contains the segment to which control is transferred if a given interrupt occurs and the other contains the offset within that segment. Because 3FFH words of memory are reserved, the possibility of defining 512 different interrupts exists. However, only a few are presently used, and two of them are initialized in this module.

FILE: IO_RAM.ASM

This module allocates memory for globally defined I/O variables. The variables defined here contain information from the I/O converter card as well as data sent on the internal I/O bus from other processors within the same DEU, mainly the DP4 processor. The DP4 makes available a 704 word buffer that originates at the host computer. Communications between the host computer and the three applications processors are performed by this buffer. The variables are used for interprocessor communication within a DEU and some also contain values of discretes such as bezels and pots. They are accessible by labels that are placed at offsets into I/O RAM. This is done so that the format can fetch data by name directly from I/O RAM as it is needed. The only exception is a series of packed discrete words processed by the routine F3RINPUT.PLM. F3RINPUT.PLM unpacks the discretes into boolean bytes local to the individual processor.

FILE: VAR_RAM.ASM

This module allocates memory, local to the ENG display processor, for copies of the host computer I/O variables. Decoded input discretes and values which drive the dials and gauges of the engine display are contained in the variables declared in this module.

Section 9.0 THE TAKEOFF FORMAT (TOPMS)

The Takeoff Performance Monitoring System (TOPMS) display format is used only during the takeoff phase of flight. It is loaded into the same microprocessor memory as the navigation format to allow automatic format switching from TOPMS to NAV immediately after takeoff. Refer to section 4.6 for information about format switching. The flight crew must manually request the TOPMS format through the CDU to replace the NAV with the TOPMS format.

The TOPMS format provides critical real-time information to the flight crew during the takeoff roll. Advisory status is also provided to assist in making the decision to proceed with takeoff or abort. The TOPMS format is always in one of three configurations; Takeoff mode, abort mode, and flap alert. Typical samples of the TOPMS display are given in figures 9.1 through 9.3. The flap alert configuration is shown when the aircraft flaps are not in the setting that was indicated by pre-takeoff entries on the CDU.

The takeoff mode of the TOPMS format has the most information presented. It consists of the runway outline with an aircraft symbol marking the current position on the runway. The nose of the aircraft symbol denotes the position on the runway. Small ticks are placed along the right edge of the runway outline at 1000 foot intervals from the far end of the runway and the runway number is shown at the start point. The runway length appears at the far end of the runway whenever the stop STAR symbol is not shown. The current airspeed of the aircraft is shown in the pointer box that moves vertically with the airplane along the left edge of the runway outline.

Several symbols appear within the runway outline to indicate significant positions on the runway. A small filled triangle appears at the position on the runway where rotation speed should be reached if the current takeoff conditions are maintained. The rotate speed is shown on the right edge of the runway outline with a pointer line, adjacent to the filled triangle. Another triangle, not filled, also is placed within the runway outline. This one shows the estimated position of rotation also, however the calculated position is derived from pre-takeoff computations. The position of the filled triangle is updated constantly throughout the takeoff roll, while the other remains at a fixed position. The top of the triangles is the reference point for these markers. The decision speed (V1) position is shown with the decision speed value along the left edge of the runway outline by a number written on top of a pointer line. A circular symbol with a four point STAR enclosed may appear within the runway outline. It indicates the position on the runway where the aircraft could be brought to a stop using maximum braking, without thrust reversers. It only appears in takeoff mode when a stop or warning advisory is present, or its position falls beyond the roll limit line. When the stop position is off the end of the runway the stop STAR jumps to the top of the display and flashes. The center of the STAR marks the reference position for this symbol. The last position marked on the runway outline is the "Roll Limit" line. This line crosses the runway at the last position where rotation may occur and give proper end of runway clearance. This line is extended out the sides of the runway outline to serve as the base of the EPR bars described below.

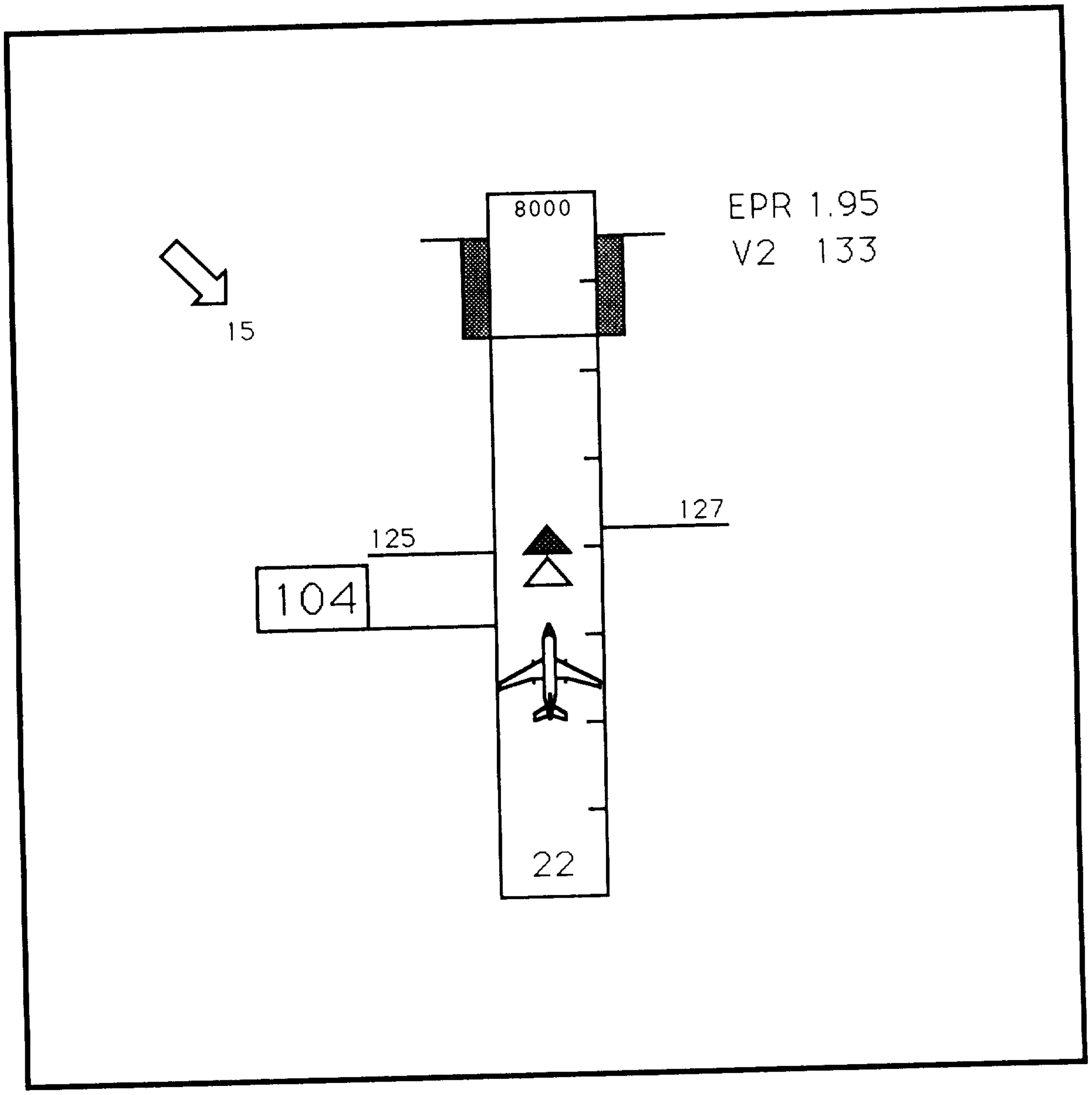
Two filled bars extend vertically along both the left and right edges of the runway outline. These bars depict the left and right engine status during the takeoff roll. The length of each bar is proportional to the corresponding engine pressure ratio (EPR) for the left and right engine. The base of the bars is always the roll limit line which represents the "1.0" EPR position. Lines are drawn directly above the EPR base lines to indicate the amount of engine EPR recommended for the current takeoff conditions. This value is displayed in the upper right area of the screen. On takeoff the throttles should be set so the EPR bars extend upward to the takeoff EPR lines. The color of the bars is blue as long as the engines are performing normally. When an engine's performance is questionable the corresponding EPR bar is changed to red.

Textual information is provided in the upper right corner of the TOPMS display. The top line contains the recommended takeoff EPR. Below is the takeoff second segment climb out speed (V2). A flashing warning may appear below the V2 line when the takeoff is being executed with air bleeds off, since the TOPMS engine model assumes air bleeds will be on.

A wind direction arrow is placed in the upper left corner of the TOPMS display. The wind speed (knots) is shown with the arrow.

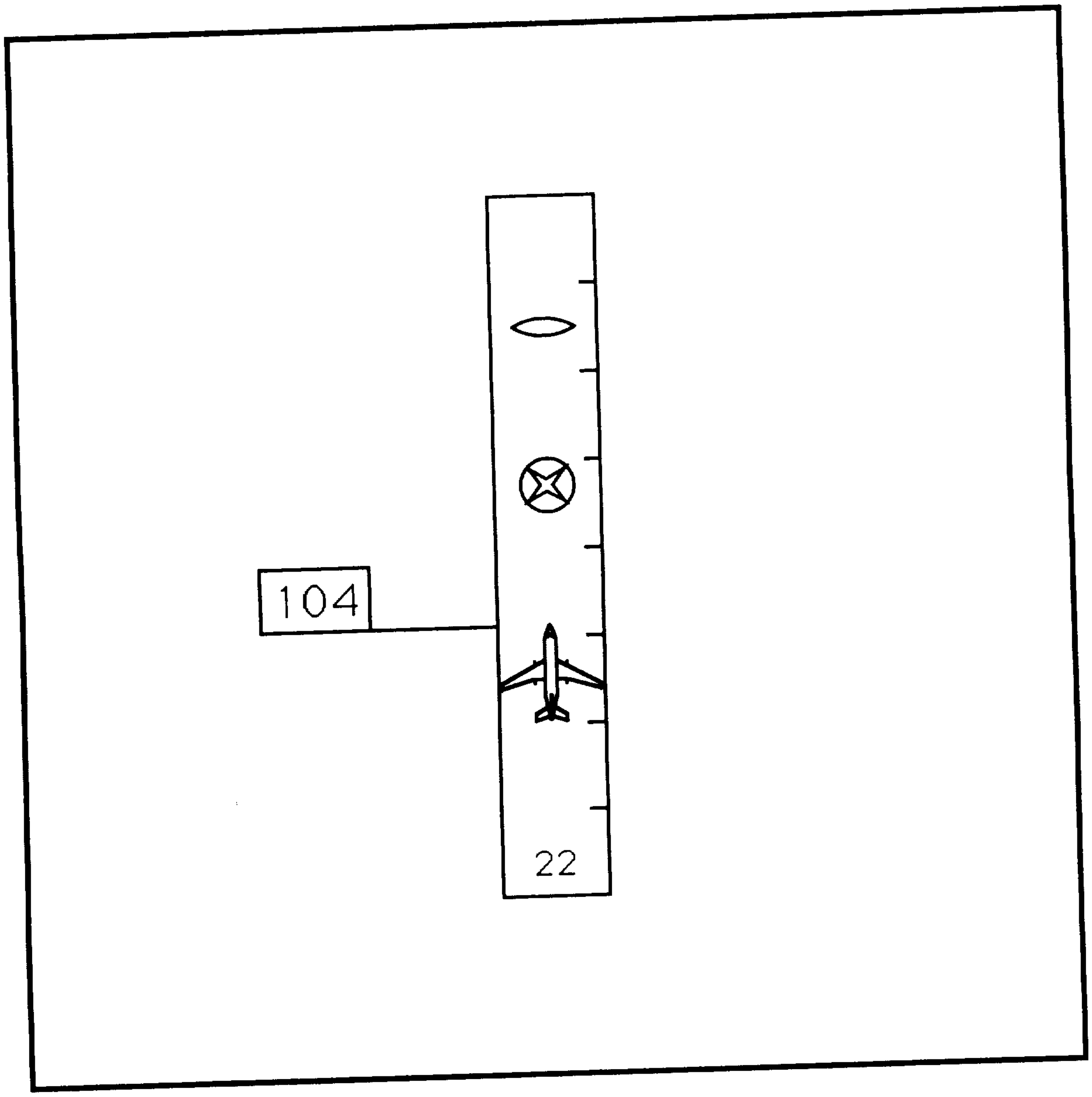
Advisory status may appear on top of the runway outline as shown in figure 9.3. When no status symbols are shown the takeoff roll is proceeding satisfactorily. When a red stop sign appears above the runway, TOPMS recommends aborting the takeoff. A flashing yellow triangle means that a problem has occured, but decision speed was already reached and a takeoff can be completed. The yellow triangle also implies that there is room to stop the aircraft if the pilot wishes to abort. When a problem occurs and there is no room left to stop, the green "Must Go" bar is placed atop the runway outline. This advisory is not given in the case of both engines out, which is always a stop advisory symbol.

When the pilot decides to abort a takeoff, signaled by a significant reduction in throttles, the abort mode TOPMS format is shown. This variation is mostly just a subset of the takeoff TOPMS format. The runway outline and aircraft symbol remain with runway ID and 1000 foot ticks. The speed tag along side the airplane is changed from airspeed to groundspeed. The maximum braking stop STAR is also shown on the abort mode display. One unique symbol appears in this mode. A football shaped object is placed at the point on the runway where the aircraft will stop with the current level of braking being applied. This allows the pilot to bring the aircraft to a stop no faster than necessary.



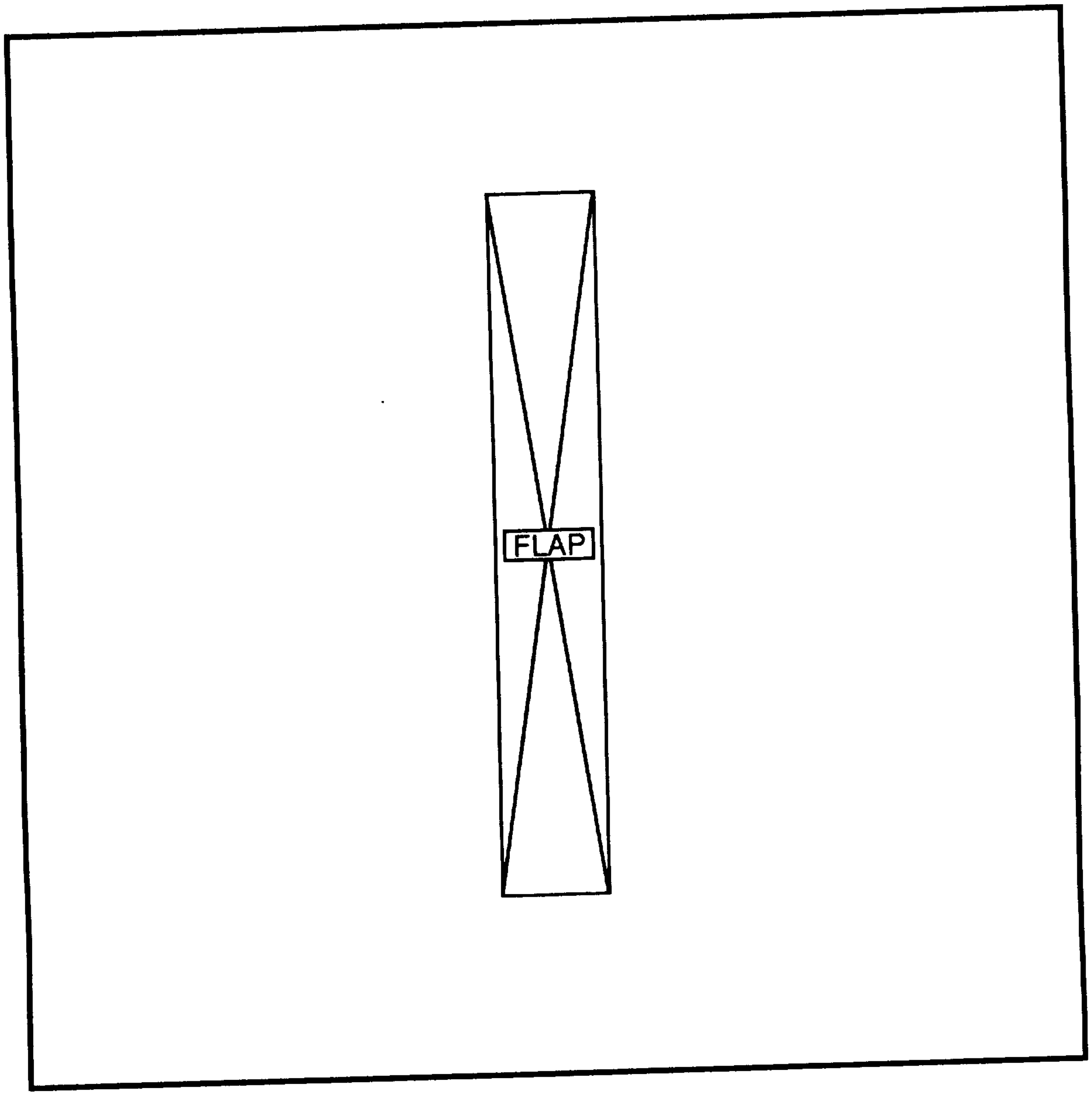
Takeoff Performance Monitoring System
(takeoff mode)

-figure 9.1-



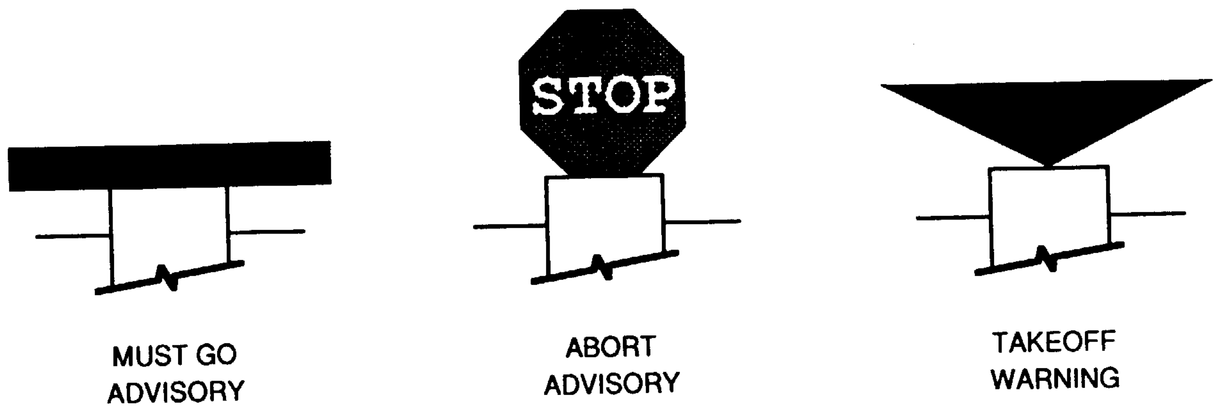
Takeoff Performance Monitoring System
(abort mode)

-figure 9.2-



Takeoff Performance Monitoring System
(Flap Alert)

-figure 9.3-



TOPMS ADVISORY SYMBOLS

-figure 9.4-

Section 9.1 COMPILE TIME GRAPHICS FILES

The files covered in this section define core graphics data constants. They create system level data base tables, which define the format to the display units, and constant CGS objects for static graphic items. The blocks of data defined in these files will be loaded into the processor's EEPROM memory for data integrity. All the data described in this section resides on the two files, COMPILE.ASM and SYS_TABS.ASM.

FILE: COMPILE.ASM

COMPILE.ASM contains the hard coded data for the TOPMS format's compile time graphic objects (SYS_TABS.ASM contains the system level compile time objects). There are a large number of objects that can be displayed on the TOPMS display. Certain information concerning these objects can be fixed at compile time such as the color, rotation angle, masks in effect, and PROM or user defined characters used to build the object. This fixed set of data is contained in the file COMPILE.ASM. Information about a graphic object that varies or is computed can be found in the file of run time modules called MANAGER.PLM.

The data contained in this module is divided into four categories; compile time objects, compile time transformations, nil graphic objects, and nil transforms. On the following page the objects are listed under their appropriate headings. Some graphic objects have more than one variant. This is true, for example, of the advisory symbology object where one of three different symbols may be shown depending on current aircraft conditions.

One other important set of data found in COMPILE.ASM is the binary data containing the moves and draws for the user defined characters. These are special characters created to be down loaded to the DU for reference by single byte character codes.

- User Characters -

runway outline, "Must go" box, runway tick, alert triangle, wind arrow, current rate stop position, airplane, max rate stop STAR, left EPR tick, right EPR tick, stop advisory, flap alert, rotate position

- Transforms -

ratser patch

- Graphics -

advisory status (3 variants), predicted rotate point, stop point (current rate), stop point (maximum rate), raster patch, raster multi-scan, raster delay

- NIL Graphics -

advisory status, text block, decision speed, wind symbology

- NIL Transforms -

airplane, roll limit line, rotate position (pretakeoff), rotate position (real-time), stop position (current rate), stop position (max rate), raster position

FILE: SYS_TABS.ASM

The system tables file SYS_TABS.ASM contains six basic compile time objects required by CGS to build a new format. These objects are enqueued by TOP_EXC at the slowest rate, and are as follows:

(1) TRANSLATION TABLES
(2) EVENT LIST
(3) OBJECT TABLE DEFINITION (OTD)
(4) TOP OBJECT
(5) SCREEN CENTER TRANSFORMATION
(6) NEW FORMAT SETUP

(1) TRANSLATION TABLES. Translation tables are used to define character sets. They can combine characters from several sources to form a new set. For the TOPMS format three sets are defined; small, medium, and large. In the DU PROM, several different sizes of standard alphanumeric characters are stored, from which the small, medium, and large TOPMS character sets map their entries. The object table definition will associate a unique character set number to each set defined. For further information on translation tables consult Section 4.4 in the CGS manual.

(2) EVENT LIST. The event list is used to describe objects that are affected by dynamically changing transforms. Each entry in the event list represents a transform, and within that entry a list of objects that will be repositioned based on that particular transform.

The TOPMS format has six graphics objects whose transformation is initiated through the event list. They include the airplane, roll limit line, real-time rotate position, pretakeoff rotate position, maximum braking stop position, and current braking stop position. The object table definition will associate these graphic objects with a transform. The transforms also have entries in the OTD which will map them to the proper entry in the event list.

Some graphic objects have been associated with a transform in the OTD but are not listed in the event list. This would include compile time transforms since they do not dynamically change. For more information on event lists consult Section 3.5 in the CGS manual.

(3) OBJECT TABLE DEFINITION. The object table definition defines the TOPMS format to CGS, and the resources needed to support it. It consists of a series of object table entry macros, one for each object in the format. Entries are listed in ascending order of object numbers with the new format setup always as object number 1. Special macros are used to define the new format setup, OTD table, event list, translation tables, and the top object. Also found are entries for the screen center transformation and a user defined character set CS_SPC which contains symbols unique to the TOPMS display.

Following these entries are listed the transformation and graphic objects for the TOPMS format. As mentioned before, graphic objects are associated with a transform (the screen center transformation if nothing else), and transforms are mapped to event list entries (except for compile time transforms which use the special code EVLO_NIL to indicate no action taken). Transforms must have lower object numbers than the graphic objects they affect.

The buffering method required for the objects, static or pingpong, is also set up here. The pingpong method is used for objects that are constantly changing, however it requires more VG memory than the static method. To conserve memory, switching between static and pingpong is done whenever possible. For further details on building an OTD, and individual OTE macros consult Section 3.4 in the CGS manual.

(4) TOP OBJECT. The top object identifies all the graphic objects in the TOPMS format. Each entry actually becomes a call to the DU code to display that object on the DU screen. Entries exist for both run time and compile time objects. Objects must be listed in the top object in order to be displayed. Section 4.1 in the CGS manual describes the top object in more detail.

(5) SCREEN CENTER TRANSFORMATION. The screen center transformation is a required CGS object. It can also be used in the OTD by graphic objects that do not move. In the TOPMS format its position is set to (0,0).

(6) NEW FORMAT SETUP. The new format setup starts the initialization of a new format. It is always the first object block processed by CGS, and contains resource information necessary to display the format - such as the total number of objects, the unique format ID, the number of OTD and event list tables, and the object number of the top object. It also specifies the colors that are used for stroke and raster symbology. Refer to Section 3.6 for further details on the new format setup and its parameters.

Section 9.2 RUN TIME GRAPHICS ROUTINES

This section describes the PLM86 modules that perform graphic operations for dynamic display objects on the PFD format. Below is a list of the modules covered in this section with the TOPMS symbology each affects.

DSPEED_MGR	decision speed value and pointer
DYNAMIC_RASTER	raster channel boundaries
EYE_TFM	current rate stop position transform
INVALID_DATA	NIL objects
PLANE_MGR	airplane and speed tag
PLANE_TFM	airplane transform
ROLL_MGR	roll limit line & EPR bars
ROLL_TFM	roll limit & EPR bar transform
RWY_MGR	runway outline, runway ticks, runway ID, and recommended EPR line
STAR_MGR	max rate stop transform
STATIC_RASTER	raster line pair definition
STATUS_MGR	advisory status
TEXT_MGR	text information block
VR1_TFM	fixed rotate position transform
VR2_MGR	real-time rotate position and speed tag
VR2_TFM	real-time rotate pos transform
WIND_MGR	wind direction and value

MODULE NAME: DSPEED_MGR
FILE NAME: MANAGER.PLM
CALL RATE: 0.05 SECONDS
CALLED BY: TOP_EXC

PURPOSE: To create the decision speed line and readout.

REMARKS: The decision speed pointer line and the associated numeric readout are stored into the decision speed object block by this procedure. Since this graphics object is positioned by the screen center transformation, the offsets used are relative to the screen center. The position of the decision speed marker changes constantly with the real-time runway distance computations performed by the MicroVax. This routine is called each displays frame to provide smooth movement of the decision speed line.

GLOBAL REFERENCES:

VARIABLES
DSPEED_POS DSPEED_VAL FCB FT_TO_TN GE_DSPEED

PROCEDURES
DRAW_REL_XY_BLANK DRAW_REL_XY_VIS END_OBJECT_BLOCK
ENQ_OBJECT_BLOCK MOVE_ABS_XY_BLANK NEW_OBJECT_BLOCK
U_BNRBCD U_MULT VGSUB_CSTRING VID_PRI

MODULE NAME: DYNAMIC_RASTER
FILE NAME: MASKING.PLM
CALL RATE: 0.05 SECONDS
CALLED BY: TOP_EXC

PURPOSE: To define raster channel boundaries.

REMARKS: Raster channels A, B, C, and D are defined by this procedure. The units used in CGS calls for the definition of the channels are 10,000 drawing units per inch.

Channel "A" is used to bound the advisory status area. The value TKOFF_STATUS from the MicroVAX is used to select the required shape. See figure 9.3 for the three shapes that may appear. When no advisory is present the channel is made null by overlaying an ON and OFF transition across the raster patch area.

Raster channel "B" bounds the areas for both EPR bars and the rotation position triangle. The positions of these areas are sent by the MicroVax software. The values provided are distances in feet from the beginning of the runway. The bottom of both EPR bars is gotten from the variable ROLL_LIMIT, and the tops by RATIO1 and RATIO2. VR2_POS is the position of the filled runway triangle. Figure 9.1 shows typical positions of these items.

The area bounded by channel "C" is only to differentiate between parts of channels "A" and "B" to provide alternate coloring. The area does not form the shapes of any particular TOPMS symbols. Figure 9.5 shows the channel "C" boundaries. Note that the area is divided into a static and a dynamic region. The static region is always created as seen in figure 9.5. Its purpose is to define the color of the rotation triangle as the "B" and "C" channel combination. The upper dynamic region is only created when the TKOFF_STATUS variable from the MicroVax was set to select the red stop or yellow caution advisories. In the case of the caution advisory the region is only defined for the OFF cycle of the blinking symbol.

Like channel "C", the "D" channel only differentiates between parts of channels "A" and "B". Refer to figure 9.6 for the "D" channel boundaries. As seen in figure 9.6, the area is separated into three dynamic regions. The actual "D" channel will consist of any combination of these three regions. The upper region is used to select the yellow caution advisory. The left and right lower areas are used to change the EPR bars to red during their respective engine failures.

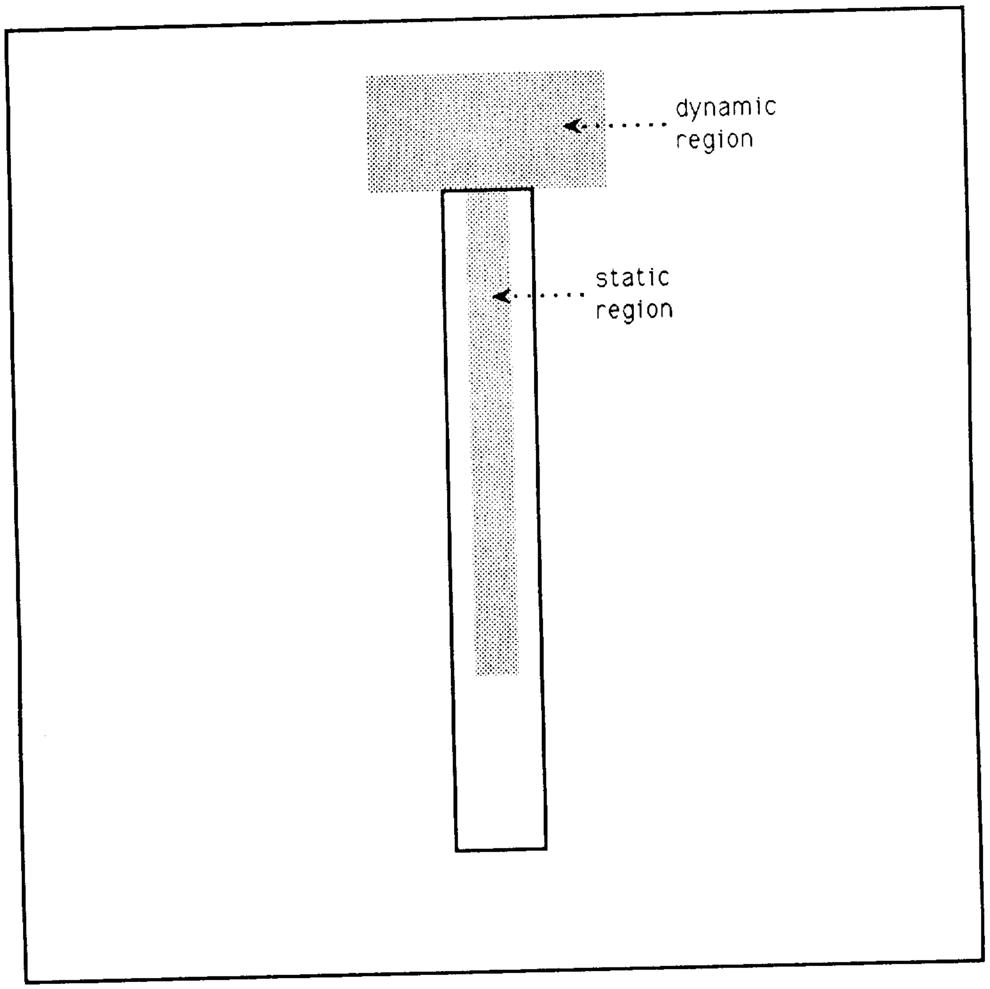
The following table lists the raster channel color definitions. Any combination of channels not listed are defined as no raster fill (black).

"A"	"B"	"C"	"D"	COLOR	USAGE
ON	OFF	OFF	OFF	green	"Must Go" advisory
OFF	ON	OFF	OFF	blue	EPR bars (engines O.K.)
ON	OFF	ON	OFF	red	stop advisory
OFF	ON	ON	OFF	cyan	rotation triangle
ON	OFF	OFF	ON	yellow	caution advisory
OFF	ON	OFF	ON	rose	EPR bars (engine failure)

GLOBAL REFERENCES:

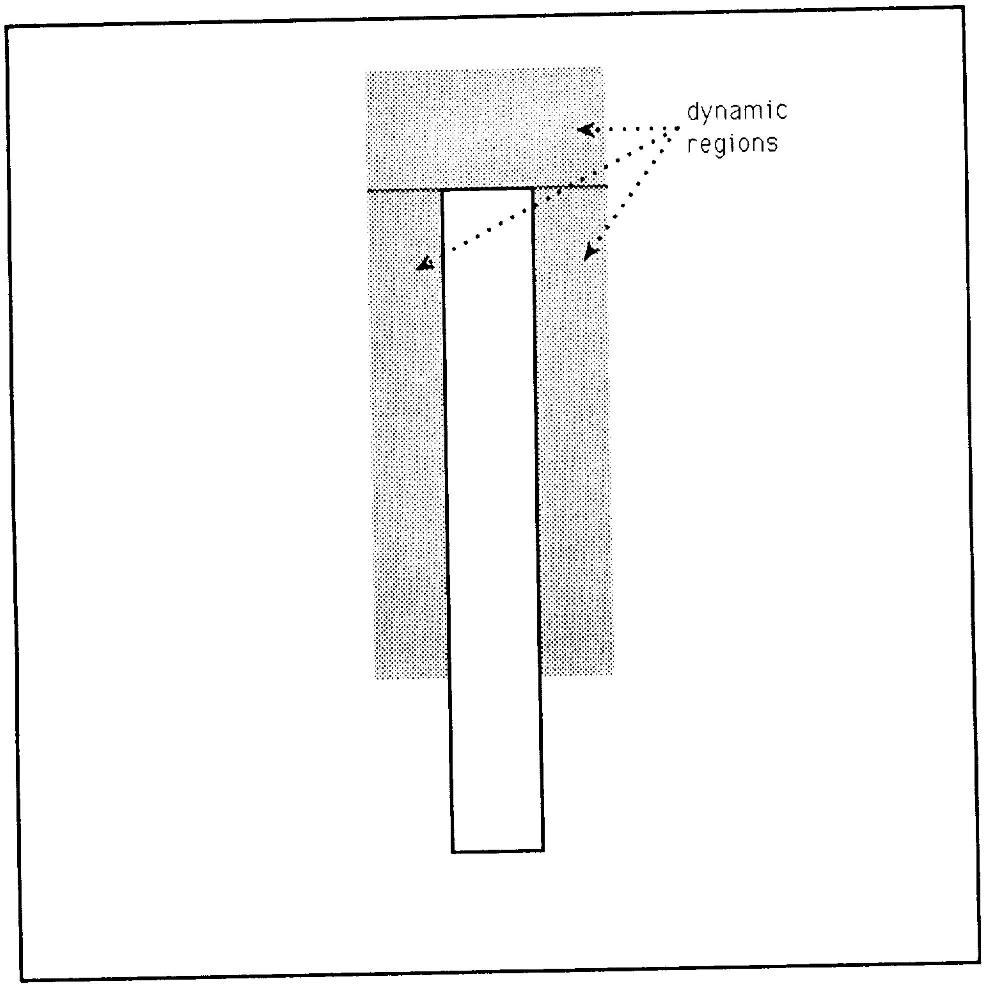
VARIABLES
ENG_LEFT ENG_RIGHT FCB FT_TO_TN GE_RASTER RATIO1
RATIO2 ROLL_LIMIT TKOFF_STATUS VR2_POS X_FDGE X_SLOP
Y_SLOP

PROCEDURES
DRAW_REL_XY END_MASK_BLOCK ENQ_OBJECT_BLOCK
MOVE_REL_XY NEW_MASK_BLOCK SET_MASK_CHANNEL
SET_MASK_PRIORITY U_MULT



TOPMS
Raster Channel "C"
(shaded area)

-figure 9.5-



TOPMS
Raster Channel "D"
(shaded area)

-figure 9.6-

MODULE NAME: EYE_TFM
FILE NAME: MANAGER.PLM
CALL RATE: 0.05 SECONDS (when selected)
CALLED BY: TOP_EXC

PURPOSE: To create and enqueue the "EYE" symbol transform.

REMARKS: The aircraft stop position marker, current rate of braking, is shaped like an eye. See figure 9.2 for this symbol. The positioning transform is created by this module using the global variable EYE from the MicroVax. This variable comes as the number of feet from the beginning of the runway to the EYE symbol. The distance is converted to one-thousandth inches and reoriented to be relative to the DU screen center.

GLOBAL REFERENCES:

VARIABLES
EYE FCB FT_TO_TN

PROCEDURES
ENQ_OBJECT_BLOCK MAKE_XFORM_TWO U_MULT

MODULE NAME: INVALID_DATA
FILE NAME: MANAGER.PLM
CALL RATE: 0.05 SECONDS (when enabled)
CALLED BY: TOP_EXC

PURPOSE: To enqueue NIL objects.

REMARKS: When I/O communication with the MicroVax has failed this module is called instead of the regular symbol manager procedures. NIL objects are enqueued to remove much of the TOPMS symbology from the display. The skeleton display which remains consists of the runway outline only. The following is a list of the NIL objects enqueued.

airplane transform object
advisory status graphics object
text block graphics object
roll limit line transform object
rotation triangle transform object (pretakeoff)
rotation triangle transform object (real-time)
stop position transform object (current rate)
stop position transform object (maximum rate)
decision speed graphics object
wind information graphics object
raster patch transform object

GLOBAL REFERENCES:

VARIABLES
FCB GR_DSPEED_NIL GR_STATUS_NIL GR_TEXT_NIL
GR_WIND_NIL TR_EYE_NIL TR_PLANE_NIL TR_RASTER_NIL
TR_ROLL_NIL TR_STAR_NIL TR_VR1_NIL TR_VR2_NIL

PROCEDURES
ENQ_OBJECT_BLOCK

```
MODULE NAME:   PLANE_MGR
FILE NAME:     MANAGER.PLM
CALL RATE:     0.2 SECONDS
CALLED BY:     TOP_EXC
```

PURPOSE: To create and enqueue the airplane graphics object.

REMARKS: This procedure calls CGS utility procedures to generate the object block for the airplane symbol and its associated speed readout (see figure 9.1). The symbology is positioned by a transform created in the procedure PLANE_TFM. The global variable from the MicroVax "SPEED" contains either the airspeed or groundspeed, depending on display mode, in knots.

GLOBAL REFERENCES:

```
VARIABLES
  FCB GE_PLANE SPEED

PROCEDURES
  END_OBJECT_BLOCK ENQ_OBJECT_BLOCK IPVGSUB_CSTRING
  NEW_OBJECT_BLOCK U_BNRBCD VGSUB_CSTRING VID_PRI
```

MODULE NAME:	PLANE_TFM
FILE NAME:	MANAGER.PLM
CALL RATE:	0.05 SECONDS
CALLED BY:	TOP_EXC

PURPOSE: To create and enqueue the airplane transform.

REMARKS: The MicroVax sends the airplane position on the runway in the variable PLANE. The value is the number of feet from the beginning of the runway. This value is converted to TOPMS screen units, one-thousandths inches, and reoriented relative to the DU screen center. A type two transform is created and placed in the transmission queue.

GLOBAL REFERENCES:

VARIABLES
  FCB FT_TO_TN PLANE

PROCEDURES
  ENQ_OBJECT_BLOCK MAKE_XFORM_TWO U_MULT

MODULE NAME: ROLL_MGR
FILE NAME: MANAGER.PLM
CALL RATE: 0.05 SECONDS
CALLED BY: TOP_EXC

PURPOSE: To create and enqueue the graphics for the ground roll limit line and EPR bar outlines.

REMARKS: This procedure creates one graphics object containing the ground roll limit line and the EPR bar outlines. The symbology is actually positioned on the display screen by the transformation object created by PLANE_TFM. The EPR bars may continually change height. The MicroVAX sends the left and right EPR sizes in the global variables RATIO1 and RATIO2 respectively. Once the object block is complete it is placed in the transmission queue to be sent to the DU.

GLOBAL REFERENCES:

VARIABLES
FCB GE_ROLL RATIO1 RATIO2

PROCEDURES
DRAW_REL_XY_BLANK DRAW_REL_XY_VIS END_OBJECT_BLOCK ENQ_OBJECT_BLOCK MOVE_ABS_XY_BLANK NEW_OBJECT_BLOCK VID_PRI

MODULE NAME: ROLL_TFM
FILE NAME: MANAGER.PLM
CALL RATE: 0.8 SECONDS
CALLED BY: TOP_EXC

PURPOSE: To create and enqueue the roll limit line transformation object.

REMARKS: The MicroVax sends the roll limit line position on the runway in the variable ROLL_LIMIT. The value is the number of feet from the beginning of the runway. This value is converted to TOPMS screen units, one-thousandths inches, and reoriented relative to the DU screen center. A type two transform is created and placed in the transmission queue. Since this position remains fixed throughout the takeoff roll, refreshing of the transform object is performed at a slow rate.

GLOBAL REFERENCES:

VARIABLES
FCB FT_TO_TN ROLL_LIMIT

PROCEDURES
ENQ_OBJECT_BLOCK MAKE_XFORM_TWO U_MULT

MODULE NAME: RWY_MGR
FILE NAME: MANAGER.PLM
CALL RATE: 0.8 SECONDS
CALLED BY: TOP_EXC

PURPOSE: Create and enqueue the graphics for the runway outline and associated symbology.

REMARKS: This procedure places the runway outline into the object block. If the I/O communications with the MicroVax is working and the host computer has enabled the full TOPMS format, the runway identification number and runway tick marks are also placed in the object block. The runway ticks are placed at 1000 foot intervals from the far end of the runway. The EPR bar bases are also placed in the object block at this time.

When the TOPMS format is not valid one other piece of symbology may be stored in the object block besides the runway outline. If the host computer has flagged improper flap settings the "Flap Alert" symbol is stored (see figure 9.3).

When the object block is complete it is placed in the transmission queue to be sent to the DU.

GLOBAL REFERENCES:

VARIABLES
BAD_FLAP EPR_BAR FCB FMT_VLD FT_TO_TN GE_RWY MODE
RWY_ID RWY_LEN

PROCEDURES
END_OBJECT_BLOCK ENQ_OBJECT_BLOCK IPVGSUB_CSTRING
NEW_OBJECT_BLOCK U_DIV U_MULT VGSUB_CSTRING VID_PRI

MODULE NAME: STAR_MGR
FILE NAME: MANAGER.PLM
CALL RATE: 0.05 SECONDS
CALLED BY: TOP_EXC

PURPOSE: To create and enqueue the stop STAR transform.

REMARKS: This procedure manages the transformation object for the stop STAR symbol (Refer to figure 9.2). The Micro-Vax always sends the position to the TOPMS format in the variable STAR. The determination of when the symbol should be shown is made in this procedure. When it is not valid a NIL transform is enqueued to invalidate the object in the display unit. Otherwise a type two transformation is made to position the symbol. The position value from the host computer is sent as the number of feet from the beginning of the runway. It is converted to TOPMS screen units, one-thousandths inches, and reoriented as an offset from screen center. If the position sent by the host computer is past the far end of the runway, the symbol is shown at a fixed position above the far end of the runway.

The stop STAR is always shown in the TOPMS abort mode. In takeoff mode, the STAR is shown when the stop or warning advisory symbols are present. It is also shown in takeoff mode when its position is beyond the roll limit line. When the takeoff mode STAR is beyond the far end of the runway, sending of the valid transform and the NIL object is alternated to cause the symbol to blink. When flashing, the symbol is "ON" for 10 consecutive frames (.5 seconds) and off for 6 (.3 seconds).

GLOBAL REFERENCES:

VARIABLES
DP_FRAME FCB FT_TO_TN MODE ROLL_LIMIT RWY_LEN
SHOW_STAR* STAR TKOFF_STATUS TR_STAR_NIL

PROCEDURES
ENQ_OBJECT_BLOCK MAKE_XFORM_TWO U_MULT

MODULE NAME: STATIC_RASTER
FILE NAME: MASKING.PLM
CALL RATE: cold start only
CALLED BY: TOP_EXC

PURPOSE: To create the interlaced scan line pair object.

REMARKS: This procedure calls the CGS utility subroutine which creates the required interlaced line pair object for the DU raster fill. The object block does not change, so it is generated when the microprocessor is reset only. The actual placement of the object into the transmission queue is done by TOP_EXC at a rate of once every 0.8 seconds.

GLOBAL REFERENCES:

VARIABLES
GE_QSCAN GR_QSCAN NCLK_DLAY NCLK_FINE NCLK_SETL_BOT NCLK_SETL_TOP NDIX_DELTA NDPC_XMOVMAX NDPC_YMOVMIN

PROCEDURES
END_OBJECT_BLOCK NEW_OBJECT_BLOCK RASTER_LINEPAIR_2

MODULE NAME: STATUS_MGR
FILE NAME: MANAGER.PLM
CALL RATE: 0.2 SECONDS
CALLED BY: TOP_EXC

PURPOSE: To enqueue the advisory status objects.

REMARKS: This procedure enqueues one of the four variants of the advisory status object. the selection index from the MicroVax computer, TKOFF_STATUS, is used to determine which one is required. Each of the variants are compile time graphics objects defined in COMPILE.ASM. The screen center transformation object, enqueued by TOP_EXC, positions the symbology on the DU screen. The following list shows the values of TKOFF_STATUS and the corresponding symbology.

0	"MUST GO" rectangle
1	Caution Triangle
2	Stop Sign
3	No advisory, NIL object enqueued

GLOBAL REFERENCES:

VARIABLES
FCB GR_STATUS_AMBER GR_STATUS_GREEN GR_STATUS_NIL
GR_STATUS_RED TKOFF_STATUS

PROCEDURES
ENQ_OBJECT_BLOCK

MODULE NAME:	TEXT_MGR
FILE NAME:	MANAGER.PLM
CALL RATE:	0.8 SECONDS
CALLED BY:	TOP_EXC

PURPOSE: To create and enqueue the TOPMS text block object.

REMARKS: This procedure creates a graphics object which contains up to four informational text strings. It is positioned by the screen center transformation, so all coordinate offsets are relative to the DU screen center.

The runway length is displayed on the far end of the runway outline whenever the stop STAR symbol is not shown. The recomended takeoff engine pressure ratio (EPR) is always displayed as the top line of the text block appearing in the upper right corner of the screen. The second segment takeoff speed (V2) is always shown directly below the EPR. Optionally a "BLEED OFF" warning may be shown underneath the V2 speed. When selected, this message flashes on a 3.2 second cycle. It is on for 2.4 seconds and off for .8.

The MicroVax computer sends the following information for the text block object.

RWY_LEN	runway length in feet
EPR	recomended EPR scaled by 8192
V2	second segment speed in knots
BLEED_WARN	air bleed warning discrete

Note that medium size characters are used for all text except the runway length, which is shown in small letters.

GLOBAL REFERENCES:

VARIABLES
BLEED_WARN EPR FCB GE_TEXT RWY_LEN SHOW_STAR V2

PROCEDURES
END_OBJECT_BLOCK ENQ_OBJECT_BLOCK IPVGSUB_CSTRING
NEW_OBJECT_BLOCK U_BNRBCD U_INSERT_DEC_PT U_MULT
VGSUB_CSTRING VID_PRI

MODULE NAME: VR1_TFM
FILE NAME: MANAGER.PLM
CALL RATE: 0.8 SECONDS
CALLED BY: TOP_EXC

PURPOSE: To create and enqueue the rotation triangle (pretakeoff) transformation object.

REMARKS: The MicroVax sends the position of the fixed (non-filled) triangle on the runway in the variable VR1. The value is the number of feet from the beginning of the runway. This value is converted to TOPMS screen units, one-thousandths inches, and reoriented relative to the DU screen center. A type two transform is created and placed in the transmission queue. Since this position remains fixed throughout the takeoff roll, refreshing of the transform object is performed at a slow rate.

GLOBAL REFERENCES:

VARIABLES
FCB FT_TO_TN VR1

PROCEDURES
ENQ_OBJECT_BLOCK MAKE_XFORM_TWO U_MULT

MODULE NAME:	VR2_MGR
FILE NAME:	MANAGER.PLM
CALL RATE:	0.2 SECONDS
CALLED BY:	TOP_EXC

PURPOSE: (real-time) To create and enqueue the rotation triangle and its associated pointer line and speed value.

REMARKS: This procedure creates the graphics object containing the real-time rotation triangle, pointer line, and rotation airspeed value. The actual positioning of the object on the DU screen is through the transform object generated in the module VR2_TFM. The value from the host computer is sent as VR2_VAL, which is in knots. Small DU PROM characters are used for the rotation airspeed.

GLOBAL REFERENCES:

VARIABLES
  FCB GE_VR2 VR2_VAL

PROCEDURES
  DRAW_REL_XY_BLANK DRAW_REL_XY_VIS END_OBJECT_BLOCK
  ENQ_OBJECT_BLOCK IPVGSUB_CSTRING MOVE_ABS_XY_BLANK
  NEW_OBJECT_BLOCK U_BNRBCD VGSUB_CSTRING VID_PRI

MODULE NAME: VR2_TFM
FILE NAME: MANAGER.PLM
CALL RATE: 0.05 SECONDS
CALLED BY: TOP_EXC

PURPOSE: (real-time) To create and enqueue the rotation position transformation object.

REMARKS: The MicroVax sends the position of the rotate position triangle on the runway in the variable VR2_POS. The value is the number of feet from the beginning of the runway. This value is converted to TOPMS screen units, one-thousandths inches, and reoriented relative to the DU screen center. A type two transform is created and placed in the transmission queue.

GLOBAL REFERENCES:

VARIABLES
FCB FT_TO_TN VR2_POS

PROCEDURES
ENQ_OBJECT_BLOCK MAKE_XFORM_TWO U_MULT

MODULE NAME: WIND_MGR
FILE NAME: MANAGER.PLM
CALL RATE: 0.8 SECONDS
CALLED BY: TOP_EXC

PURPOSE: To create and enqueue the wind arrow and value.

REMARKS: This procedure creates a graphics object for the wind arrow and value readout. The actual position of the object is defined by the screen center transformation, therefore coordinate offsets are in one-thousandth inches from the DU screen center. The host computer sends the wind direction in the global variable WIND_DIR and the wind speed in WIND_SPD (knots). DU PROM small characters are used for the wind speed value.

GLOBAL REFERENCES:

VARIABLES
FCB GE_WIND WIND_DIR WIND_SPD

PROCEDURES
END_OBJECT_BLOCK ENQ_OBJECT_BLOCK IPVGSUB_CSTRING
NEW_OBJECT_BLOCK ROTANG U_BNRBCD VID_PRI

Section 9.3 MEMORY ALLOCATION FILES

Most of the memory allocation modules reserve memory locations in RAM for various system and applications variables. VAR_RAM.ASM, GLBL_RAM.PLM, and IO_RAM.ASM allocate memory between the address ranges of 800H and 7FFFH in the display processor memory. IO_RAM.ASM reserves memory specifically between the ranges of 4000H and 7FFFH. Memory locations 0400H through 07FFH are reserved for the stack and the stack pointer is initialized to 0800H.

FILE: GLBL_RAM.PLM

This module allocates memory locations in RAM for the system level variables used by the format executive. The types of variables declared globally in this module include system error flags, counters and checksum variables.

FILE: INT_TBL.ASM

This module reserves memory locations 0000H through 03FFH for the interrupt vectors. There are two words of data per interrupt. The first contains the segment to which control is transferred if a given interrupt occurs and the other contains the offset within that segment. Because 3FFH words of memory are reserved, the possibility of defining 512 different interrupts exists. However, only a few are presently used, and two of them are initialized in this module.

FILE: IO_RAM.ASM

This module allocates memory for globally defined I/O variables. The variables defined here contain information from the I/O converter card as well as data sent on the internal I/O bus from other processors within the same DEU, mainly the DP4 processor. The DP4 makes available a 704 word buffer that originates at the host computer. Communications between the host computer and the three applications processors are performed by this buffer. The variables are used for interprocessor communication within a DEU and some also contain values of discretes such as bezels and pots. They are accessible by labels that are placed at offsets into I/O RAM. This is done so that the format can fetch data by name directly from I/O RAM as it is needed. The only exception is with packed discretes, which are fetched once by a processing routine. The routine then unpacks the discretes into boolean bytes which are local to the individual processor.

FILE: VAR_RAM.PLM

This module allocates memory for applications software variables. Most of the variables declared here are discretes which indicate if certain signals are valid or certain graphic objects have been selected.

Appendix A How to Build the Formats

To use the display software described in this document a load module, with the file extension ".CIM", must be created from the source files. The software package "UDI" must be used, when using an IBM PC or clone, to allow the use of Intel development system software. Once operating under the development package, the Intel assembler, compiler, linker, locater, and hex formatter can be used. Command files to automatically build the formats reside with the format source files. Entering the command ASSEMBLE to the UDI prompt will cause the compilation or assembly of all source files, followed by automatic running of the LINK/LOC/ HEX process.

The LINK86 program processes object files created from either compilation or assembly. The segments are built and the offsets into each segment of relocatable symbols are computed. The resultant file from the LINK86 program is then used as input to the LOC86 program. This program assigns physical addresses to all segments and symbols. The last Intel program used is OH86. This program takes the absolute object file produced by LOC86 and converts it to a file containing hexadecimal ASCII characters used in the loading process.

Since program segments are combined with word alignment, undefined byte locations within a segment will exist in the load file. This invalidates the checksum process described in Section 4.4. A program to fill gaps within segments is used as the last step of creating the load file. The program is run while under the DOS system as follows.

```
    C>GAP FILE.CIM
```

The output is a new ".CIM" file with the same name as the input file plus an additional "_CK" added to the name.

```
    PFD.CIM --> PFD_CK.CIM
```

Example command files for the LINK/LOC/HEX process follow.

```
\path\LINK86           &
FIRST_FILE.OBJ,        &
     .                 &
     .                 &
     .                 &
                       &
CGS.LNK,               &
LAST_FILE.OBJ          TO  file.LNK
```

```
\path\LOC86 file.LNK TO file.EXE SYMBOLCOLUMNS(1)
ADDRESSES (SEGMENTS( CODE(0F8000H),                                   &
INTERRUPTS(00000H),CONST(00F000H),                                    &
GDATA1(00C000H),DATA(00800H),IODATA(04000H),                          &
SIDATA(024800H),HSDATA(010000H),STACK(0400H)),                        &
GROUPS(CGROUP(0F0000H),DGROUP(00000H)))                               &
RESERVE(08000H TO 0BFFFH)                                             &
RESERVE(0FFFCAH TO 0FFFFFH)                                           &
START(0F000H,8000H)                                                   &
NOINITCODE SYMBOLS

\path\OH86 file.EXE TO file.CIM
```

Appendix B Processor memory allocations

The PLM86 applications software can access two 64K byte contiguous blocks of memory, one each for instructions and data. Refer to the PLM86 reference manual for information about the PLM "Small" model of compilation. Each processor however can address sixteen 64K byte blocks. Within the DEU only a few ranges of addresses are actually mapped to memory or device control registers. The rest are non-existent and would cause a memory access time out if addressed. Some of the various address ranges are actually shared between all the processors in the DEU, while some access local memory of the processor. All addresses not available to the PLM program are reserved for DEU system use and cannot be used by applications software. The two 64K blocks set up for applications use are 00000H through 0FFFFH (Data group) and F0000 through FFFFF (Code group). The following charts show memory allocation and usage for display processors. The first page gives the address ranges available and their given software segment names. The remaining pages depict the actual memory usage by particular formats.

DISPLAY PROCESSOR ADDRESS SPACE

16K DATA RAM ALLOCATION

Segment Name	Addresses	Size	
INTERRUPTS	00000-003FF	(0400)	
STACK	00400-007FF	(0400)	
DATA	00800-03F9F	(37A0)	
@DAT[1]	03FA0-03FFF	(0060)	<003E USED>

16K I/O RAM ALLOCATION

Segment Name	Addresses	Size
IODATA	04000-07FFF	(4000)

16K CONSTANT EEPROM ALLOCATION

Segment Name	Addresses	Size	
CONST[2]	0C000-0DFFF	(2000)	
CONST[3]	0E000-0FEBF	(1EC0)	
@CONS[1]	0FEC0-0FFFF	(0140)	<010A USED>

32K CODE EEPROM ALLOCATION

Segment Name	Addresses	Size	
CODE[2]	F8000-FB7FF	(3800)	
CODE[3]	FB800-FDCFF	(2500)	
@CGS[1]	FDD00-FFBFF	(1F00)	<1E17 USED>
<BOOT UVPROM>	FFC00-FFFFF	(0400)	

SYSTEM ALLOCATIONS

HSDATA	10000-13FFF	(HIGH SPEED BUS)
******	24000-247FF	(WEATHER RADAR)
SIDATA	24800-24FFF	(SYSTEM INTERFACE)
******	40000-407FF	(CPI MONITOR)

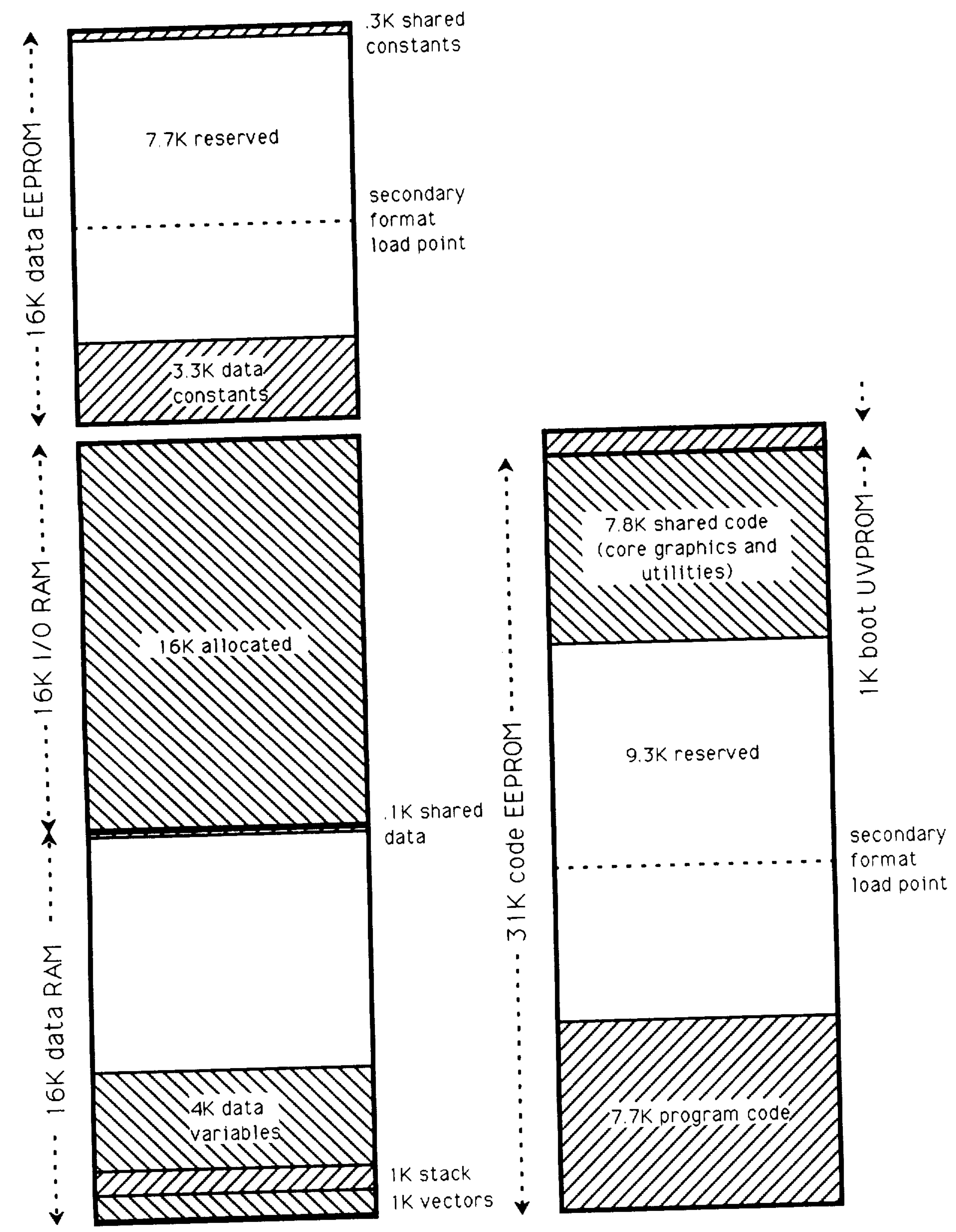
1 CORE GRAPHICS SHARED REGION
2 PRIMARY FORMAT
3 SECONDARY FORMAT

-figure C.0-

PRECEDING PAGE BLANK NOT FILMED

PAGE 340 INTENTIONALLY BLANK

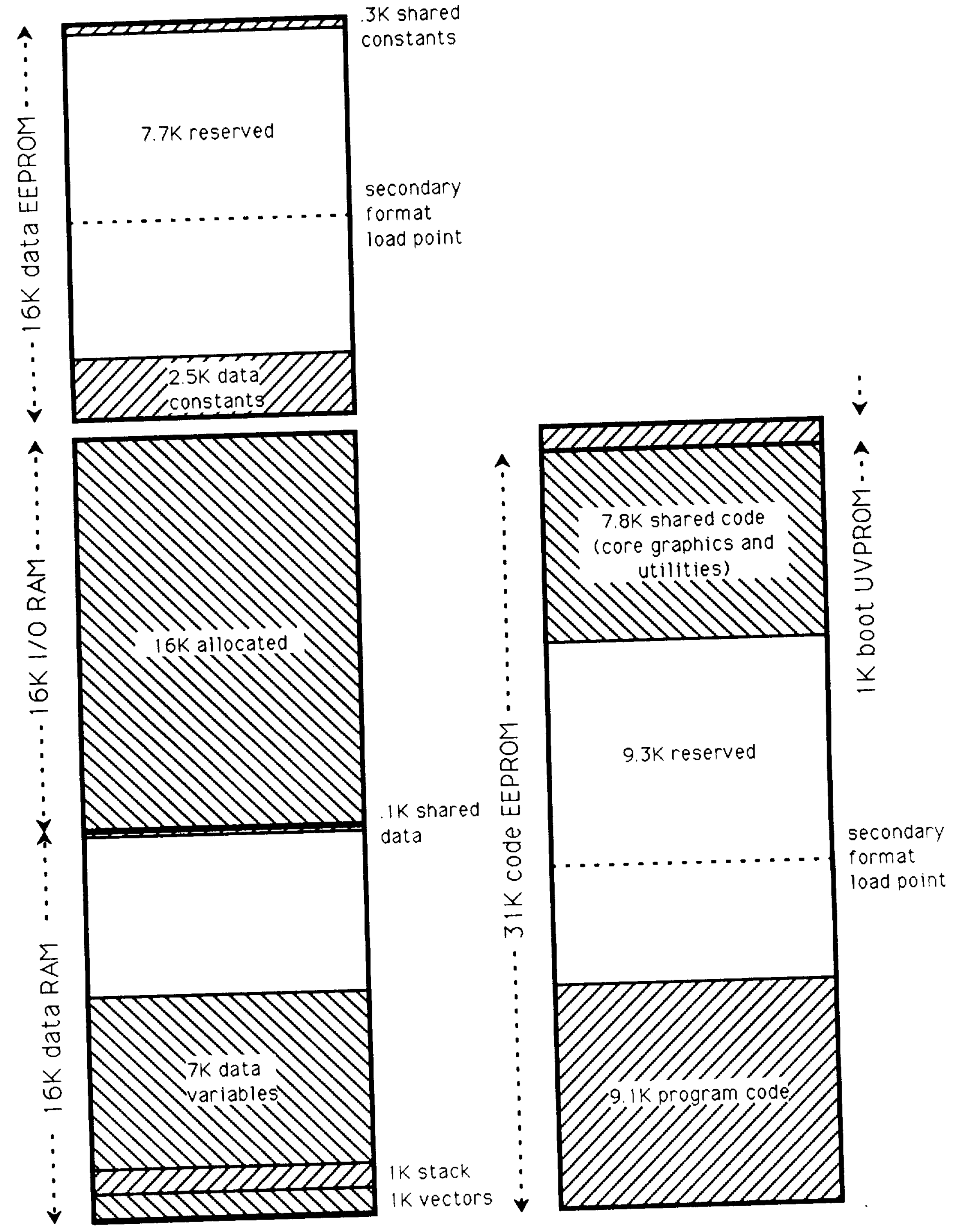
PFD Format Memory Layout



-figure C.1-

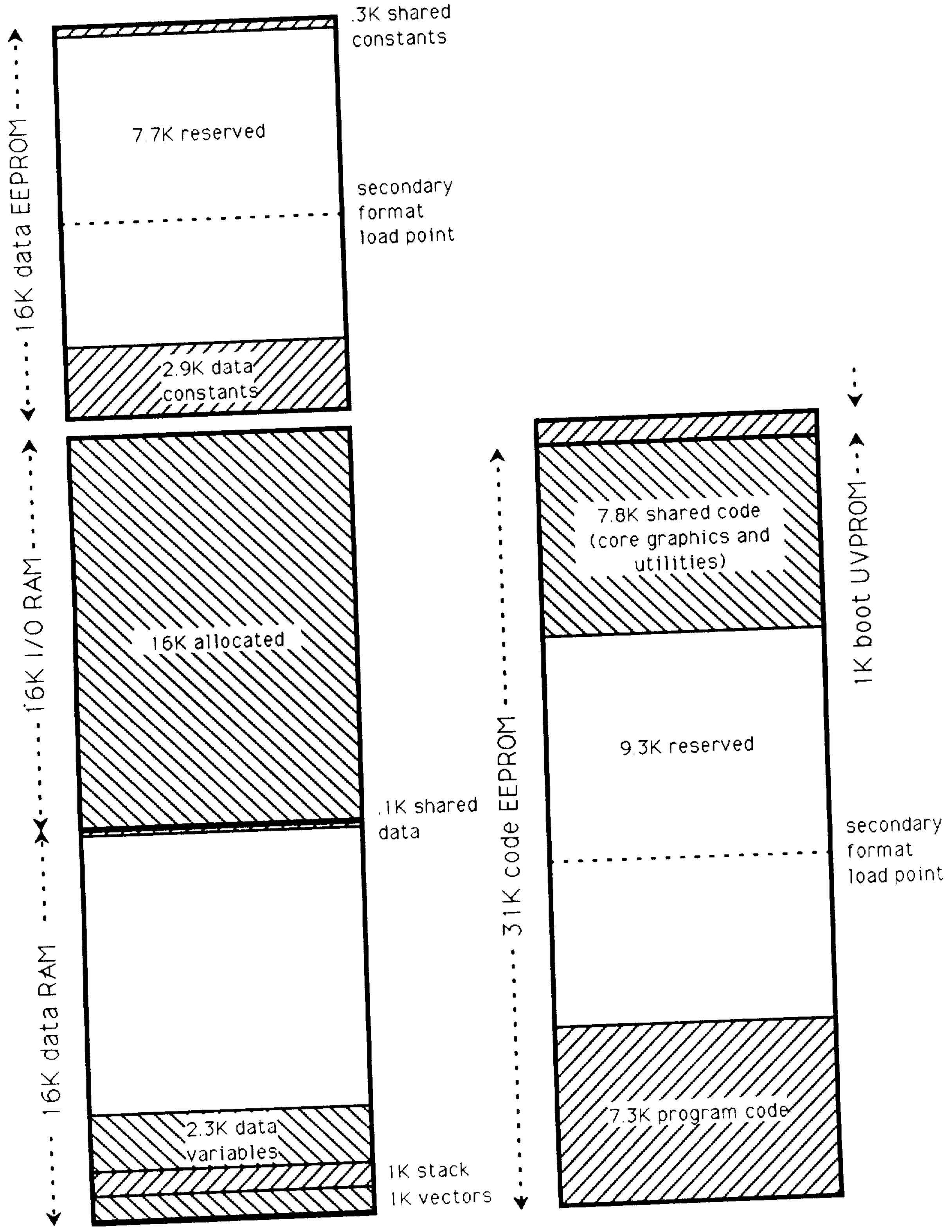
PRECEDING PAGE BLANK NOT FILMED

NAV Format Memory Layout



-figure C.2-

ENG Format Memory Layout

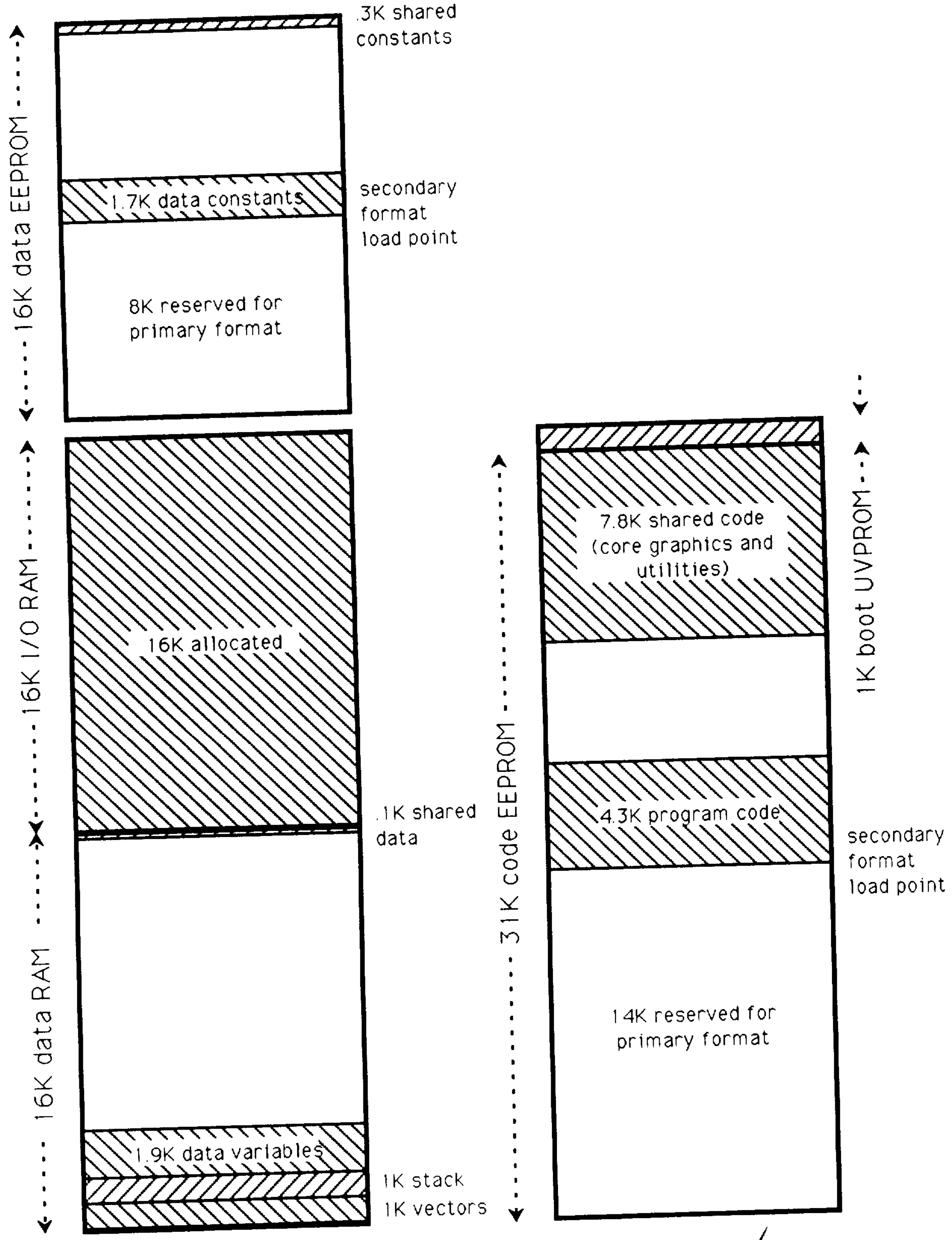


-figure C.3-

PAGE 346 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

TOP Format Memory Layout

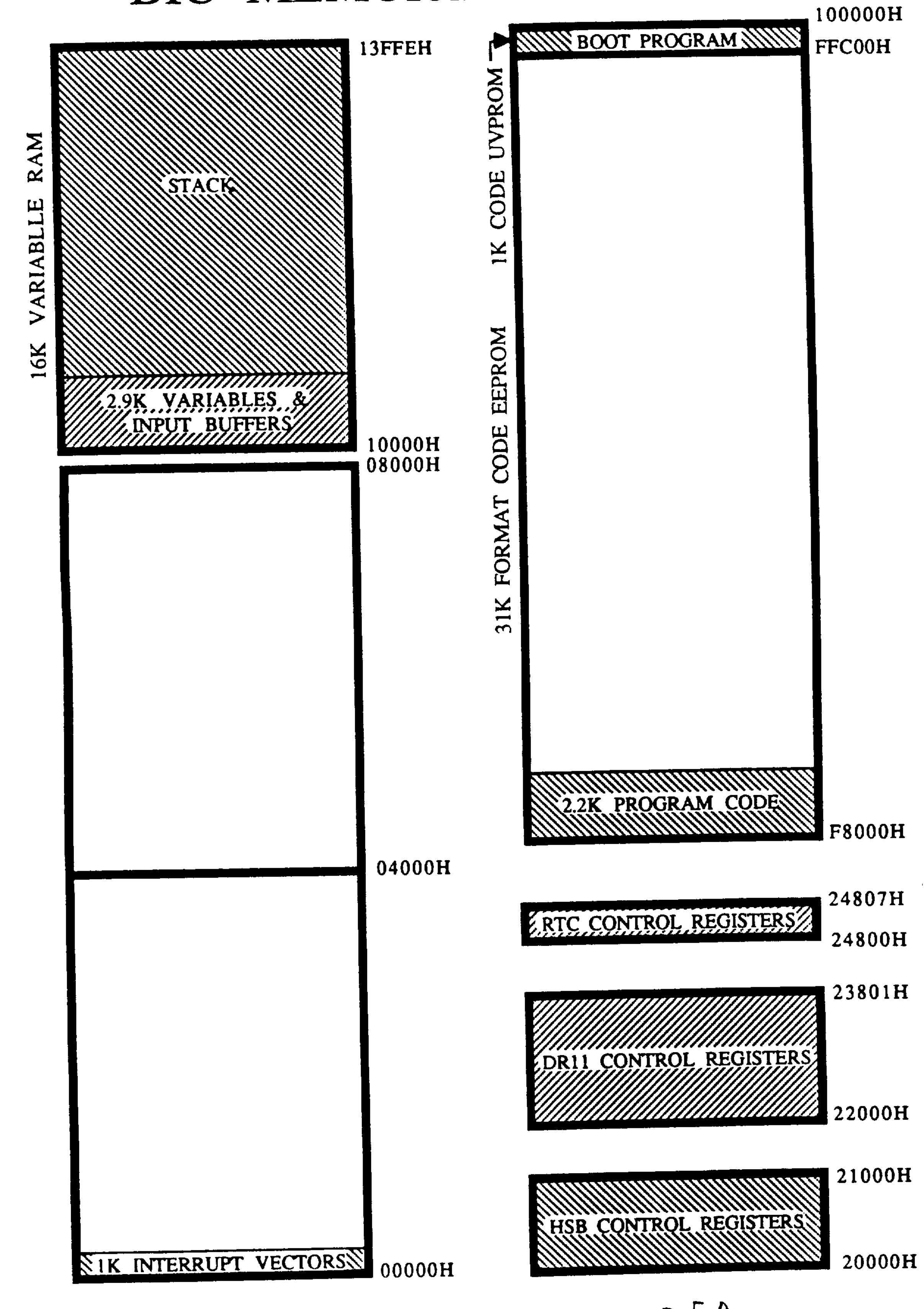


PRECEDING PAGE BLANK NOT FILMED

PAGE 348 INTENTIONALLY BLANK

-figure C.4-

BIU MEMORY LAYOUT

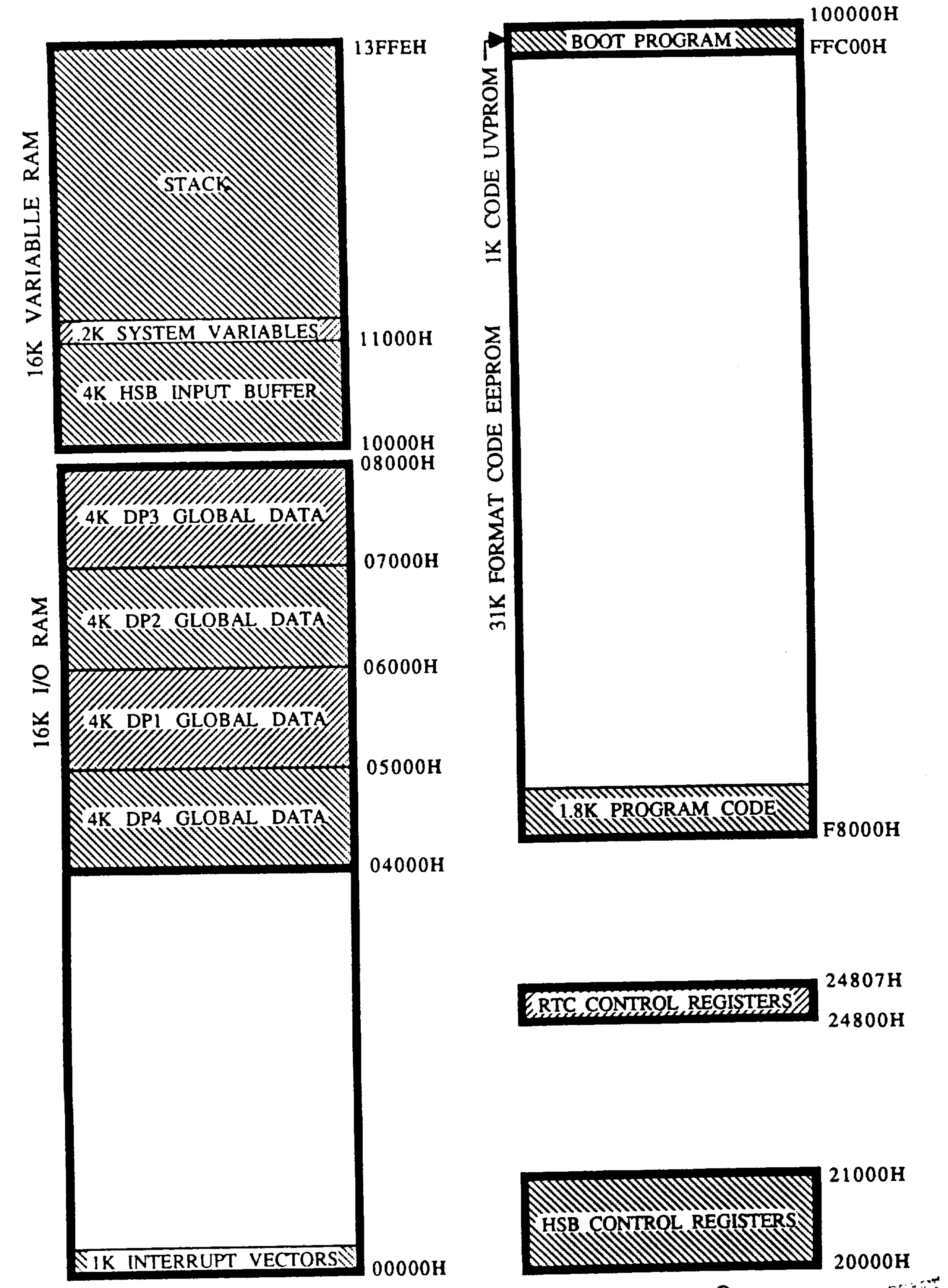


-figure C.5-

PAGE 350 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

DP4 MEMORY LAYOUT



-figure C.6

Appendix C I/O Block Layouts

Display data is sent to the microprocessor system in a 704 word contiguous block. Except for the Navigation display format's background buffer (first 400 words), specific locations within the data block have assigned contents. The following pages contain the information on the allocation of the data buffer. The first chart shows a composite usage layout for all six display formats. The ranges of memory used by the various formats is shown by assigning the byte offset values to ID mnemonics (NAV: Navigation Display; PFD: Primary Flight Display; ENG: Engine Display; SYS: System warning display; F1: Sperry Primary Display TOP: TOPMS display). The remaing pages have charts for each individual format, breaking down the I/O memory usage by specific variables. Note that the memory allocations for NAV and TOP overlay each other. This is because these display formats are mutually exclusive. The same microprocessor contains both formats, with only one active at a time.

The offset and size values in the charts are in terms of bytes. The index column contains the index into a word array used by the host computer. When a "+" is shown by the index the data actually resides on an odd byte boundary.

Following the I/O layouts is a detailed description of the various packed words contained in the charts.

MEMORY USAGE FOR ALL FORMATS

OFFSET	SIZE	INDEX	ID
0 - 801	802	1	NAV
802 - 1005	204	402	
1006 - 1007	2	504	ENG
1008 - 1079	72	505	
1080 - 1081	2	541	F1
1082 - 1085	4	542	
1086 - 1089	4	544	F1
1090 - 1091	2	546	PFD F1
1092 - 1097	6	547	F1
1098 - 1099	2	550	
1100 - 1103	4	551	SYS
1104 - 1107	4	553	
1108 - 1169	62	555	ENG
1170 - 1179	10	586	
1180 - 1181	2	591	PFD NAV
1182 - 1187	6	592	NAV
1188 - 1189	2	595	PFD NAV
1190 - 1193	4	596	NAV
1194 - 1195	2	598	PFD NAV
1196 - 1201	6	599	NAV
1202 - 1203	2	602	PFD NAV
1204 - 1207	4	603	NAV
1208 - 1241	34	605	NAV TOP
1242 - 1247	6	622	NAV
1248 - 1255	8	625	
1256 - 1263	8	629	PFD
1264 - 1267	4	633	PFD F1
1268 - 1271	4	635	PFD
1272 - 1275	4	637	PFD NAV
1276 - 1343	68	639	PFD
1344 - 1345	2	673	PFD NAV
1346 - 1347	2	674	
1348 - 1351	4	675	PFD
1352 - 1355	4	677	
1356 - 1357	2	679	NAV SYS
1358 - 1359	2	680	PFD F1
1360 - 1361	2	681	PFD NAV SYS F1
1362 - 1363	2	682	PFD NAV
1364 - 1365	2	683	PFD NAV SYS F1
1366 - 1369	4	684	TOP ENG SYS
1370 - 1373	4	686	
1374 - 1375	2	688	SYS F1
1376 - 1377	2	689	PFD F1
1378 - 1379	2	690	PFD NAV TOP ENG SYS F1
1380 - 1381	2	691	PFD F1

1382 - 1383	2	692	NAV TOP
1384 - 1395	12	693	SYS
1396 - 1401	6	699	ENG SYS
1402 - 1403	2	702	ENG
1404 - 1405	2	703	NAV SYS F1
1406 - 1407	2	704	NAV SYS

MEMORY ALLOCATION FOR PFD

OFFSET	SIZE	INDEX	VARIABLE	UNITS
1090 - 1091	2	546	RADAR_ALT	FEET * 6.5536
1180 - 1181	2	591	TRK_DIAL	DEG * 182.0444
1188 - 1189	2	595	TRU_HDG	DEG * 182.0444
1194 - 1195	2	598	TRU_TRK	DEG * 182.0444
1202 - 1203	2	602	MAG_TRK	DEG * 182.0444
1256 - 1257	2	629	ACC_SEGMENT	INCHES * 1000
1258 - 1259	2	630	STANDOFF	INCHES * 1000
1260 - 1261	2	631	AIRCRAFT_X	INCHES * 1000
1262 - 1263	2	632	AIRCRAFT_Y	INCHES * 1000
1264 - 1265	2	633	ACT_CAS	KNOTS * 64
1266 - 1267	2	634	CAS_REF	KNOTS * 64
1268 - 1269	2	635	DEC_HT	FEET
1270 - 1271	2	636	FPA_DIAL	DEG * 10
1272 - 1273	2	637	VRT_REF	FEET
1274 - 1275	2	638	GS_REF	FEET
1276 - 1277	2	639	HOR_TICK_CTR	INCHES * 1000
1278 - 1279	2	640	HOR_X	INCHES * 1000
1280 - 1281	2	641	MACH	RATIO * 1000
1282 - 1283	2	642	PITCH_Y	INCHES * 1000
1284 - 1285	2	643	LOC_X	INCHES * 1000
1286 - 1287	2	644	RWY_X1	INCHES * 1000
1288 - 1289	2	645	RWY_Y1	INCHES * 1000
1290 - 1291	2	646	RWY_X2	INCHES * 1000
1292 - 1293	2	647	RWY_Y2	INCHES * 1000
1294 - 1295	2	648	RWY_X3	INCHES * 1000
1296 - 1297	2	649	RWY_Y3	INCHES * 1000
1298 - 1299	2	650	RWY_X4	INCHES * 1000
1300 - 1301	2	651	RWY_Y4	INCHES * 1000
1302 - 1303	2	652	RWY_X5	INCHES * 1000
1304 - 1305	2	653	RWY_Y5	INCHES * 1000
1306 - 1307	2	654	RWY_X6	INCHES * 1000
1308 - 1309	2	655	RWY_Y6	INCHES * 1000
1310 - 1311	2	656	RWY_X7	INCHES * 1000
1312 - 1313	2	657	RWY_Y7	INCHES * 1000
1314 - 1315	2	658	RWY_X8	INCHES * 1000
1316 - 1317	2	659	RWY_Y8	INCHES * 1000
1318 - 1319	2	660	RWY_SCALE	(0-1) * 16384
1320 - 1321	2	661	PTH_TRK	DEG * 182.0444
1322 - 1323	2	662	FLARE	INCHES * 1000
1324 - 1325	2	663	STAR_X	INCHES * 1000
1326 - 1327	2	664	STAR_Y	INCHES * 1000
1328 - 1329	2	665	STAR_ZOOM	(1-20) * 1024
1330 - 1331	2	666	FPA_REF	INCHES * 1000
1332 - 1333	2	667	TRACK_BUG_X	INCHES * 1000
1334 - 1339	6	668	TO_WPT	ASCIC

1340 - 1341	2	671	NIBBLES_1	PACKED WORD
1342 - 1343	2	672	NIBBLES_2	PACKED WORD
1344 - 1345	2	673	NIBBLES_3	PACKED WORD
1348 - 1349	2	675	DISCRETES	PACKED WORD
1350 - 1351	2	676	SKY_PTR	DEG * 182.0444
1358 - 1359	2	680	ACT_ROLL	DEG * 182.0444
1360 - 1361	2	681	ACT_ALT	FEET
1362 - 1363	2	682	HDOT	INCHES * 1000
1364 - 1365	2	683	ALT_REF	FEET
1376 - 1377	2	689	FLAP_UP	KNOTS * 65.536
1378 - 1379	2	690	HOST_CNT	SECONDS * 20
1380 - 1381	2	691	BARO_SET	INCHES * 100

MEMORY ALLOCATION FOR NAV

OFFSET	SIZE	INDEX	VARIABLE	UNITS
0 - 799	800	1	BCKGND_BUF	BACKGROUND DATA
800 - 801	2	401	LINK_CMD	reserved
1180 - 1181	2	591	SEL_TRK	DEG * 182.0444
1182 - 1183	2	592	ALT_RANGE	NAUT. MILES * 128
1184 - 1185	2	593	XTK_GS_RATIO	(1/SEC) * 65536
1186 - 1187	2	594	AC_NORM	(FT/SEC/SEC) * 512
1188 - 1189	2	595	TRU_HDG	DEG * 182.0444
1190 - 1191	2	596	WIND_SPD	KNOTS
1192 - 1193	2	597	WIND_DIR	DEG * 182.0444
1194 - 1195	2	598	TRU_TRK	DEG * 182.0444
1196 - 1197	2	599	MAP_WORD	PACKED WORD
1198 - 1199	2	600	MLS_X	NAUT. MILES * 2048
1200 - 1201	2	601	MLS_Y	NAUT. MILES * 2048
1202 - 1203	2	602	MAG_TRK	DEG * 182.0444
1204 - 1205	2	603	BOX_X	FEET / 32
1206 - 1207	2	604	BOX_Y	FEET / 32
1208 - 1209	2	605	BOX_HDG	DEG * 182.0444
1210 - 1227	18	606	BUB_COORDS	3-(X,Y,HDG)
1228 - 1229	2	615	TANG_TIMEI	SEC * 256
1230 - 1231	2	616	OFF_VECTOR	NAUT. MILES * 128
1232 - 1239	8	617	AP_EAST	4-(INCHES * 1000)
1240 - 1247	8	621	AP_NORTH	4-(INCHES * 1000)
1272 - 1273	2	637	VRT_REF	FEET
1274 - 1275	2	638	GS_REF	FEET
1344 - 1345	2	673	NIBBLE3	PACKED WORD
1356 - 1357	2	679	GS_FPS	(FT/SEC) * 32
1360 - 1361	2	681	ALT_ACTUAL	FEET
1362 - 1363	2	682	HDOT_SEGMENT	INCHES * 1000
1364 - 1365	2	683	REF_ALT_VALUE	FEET
1378 - 1379	2	690	HOST_CNT	SECONDS * 20
1382 - 1383	2	692	FMT_ID	2, 7
1404 - 1405	2	703	DISC_WORD11	PACKED WORD
1406 - 1407	2	704	DISC_WORD12	PACKED WORD

MEMORY ALLOCATION FOR TOP

OFFSET	SIZE	INDEX	VARIABLE	UNITS
1208 - 1209	2	605	SPEED	KNOTS
1210 - 1211	2	606	WIND_SPD	KNOTS
1212 - 1213	2	607	WIND_DIR	DEG * 182.0444
1214 - 1215	2	608	RWY_LEN	FEET
1216 - 1217	2	609	PLANE	FEET
1218 - 1219	2	610	EPR	RATIO * 8192
1220 - 1221	2	611	V2	KNOTS
1222 - 1223	2	612	ROLL_LIMIT	FEET
1224 - 1225	2	613	VR1	FEET
1226 - 1227	2	614	VR2_POS	FEET
1228 - 1229	2	615	VR2_VAL	KNOTS
1230 - 1231	2	616	DSPEED_POS	FEET
1232 - 1233	2	617	DSPEED_VAL	KNOTS
1234 - 1235	2	618	STAR	FEET
1236 - 1237	2	619	PACK_WORD	PACKED WORD
1238 - 1239	2	620	EYE	FEET
1240 - 1241	2	621	RWY_ID	ASCII
1366 - 1367	2	684	EPR1	RATIO * 8192
1368 - 1369	2	685	EPR2	RATIO * 8192
1378 - 1379	2	690	HOST_CNT	SECONDS * 20
1382 - 1383	2	692	FMT_ID	2, 7

MEMORY ALLOCATION FOR ENG

OFFSET	SIZE	INDEX	VARIABLE	UNITS
1006 - 1007	2	504	SWITCH	reserved
1108 - 1109	2	555	ENG_EPRLIM	RATIO * 8192
1110 - 1111	2	556	ENG_EPRCAU	RATIO * 8192
1112 - 1113	2	557	ENG_EPRCOM	RATIO * 8192
1114 - 1115	2	558	ENG_N1_L	% * 163.84
1116 - 1117	2	559	ENG_N1_R	% * 163.84
1118 - 1119	2	560	ENG_N1LIM	% * 163.84
1120 - 1121	2	561	ENG_N1CAU	% * 163.84
1122 - 1123	2	562	ENG_EGT_L	DEG-C * 32.768
1124 - 1125	2	563	ENG_EGT_R	DEG-C * 32.768
1126 - 1127	2	564	ENG_EGTLIM	DEG-C * 32.768
1128 - 1129	2	565	ENG_EGTCAU	DEG-C * 32.768
1130 - 1131	2	566	ENG_FF_L	(LB/HR) * 1.6384
1132 - 1133	2	567	ENG_FF_R	(LB/HR) * 1.6384
1134 - 1135	2	568	ENG_EOP_L	PSI * 163.84
1136 - 1137	2	569	ENG_EOP_R	PSI * 163.84
1138 - 1139	2	570	ENG_EOLLIM	PSI * 163.84
1140 - 1141	2	571	ENG_EOULIM	PSI * 163.84
1142 - 1143	2	572	ENG_EOT_L	DEG-C * 91.022
1144 - 1145	2	573	ENG_EOT_R	DEG-C * 91.022
1146 - 1147	2	574	ENG_EOTLIM	DEG-C * 91.022
1148 - 1149	2	575	ENG_EOQ_L	GALLONS * 3276.8
1150 - 1151	2	576	ENG_EOQ_R	GALLONS * 3276.8
1152 - 1153	2	577	ENG_EOQLIM	GALLONS * 3276.8
1154 - 1155	2	578	ENG_N2_L	% * 163.84
1156 - 1157	2	579	ENG_N2_R	% * 163.84
1158 - 1159	2	580	ENG_N2LIM	% * 163.84
1160 - 1161	2	581	ENG_LFUELQ	POUNDS
1162 - 1163	2	582	ENG_RFUELQ	POUNDS
1164 - 1165	2	583	ENG_CFUELQ	POUNDS
1166 - 1167	2	584	ENG_TAT	DEG-C * 128
1168 - 1169	2	585	ENG_GW	POUNDS * .125
1366 - 1367	2	684	ENG_EPR_L	RATIO * 8192
1368 - 1369	2	685	ENG_EPR_R	RATIO * 8192
1378 - 1379	2	690	HOST_CNT	SECONDS * 20
1396 - 1397	2	699	DISC_WORD7	PACKED WORD
1398 - 1399	2	700	DISC_WORD8	PACKED WORD
1400 - 1401	2	701	DISC_WORD9	PACKED WORD
1402 - 1403	2	702	DISC_WORD10	PACKED WORD

MEMORY ALLOCATION FOR SYS

OFFSET	SIZE	INDEX	VARIABLE	UNITS
1100 - 1101	2	551	SYS_LFLAP	DEG * 182.0444
1102 - 1103	2	552	SYS_RFLAP	DEG * 182.0444
1356 - 1357	2	679	GS_FPS	(FT/SEC) * 32
1360 - 1361	2	681	PFD_ALT	FEET
1364 - 1365	2	683	PFD_SELALT	FEET
1366 - 1367	2	684	ENG_EPR_L	RATIO * 8192
1368 - 1369	2	685	ENG_EPR_R	RATIO * 8192
1374 - 1375	2	688	PFD_VSPD	(FT/SEC) * 49.152
1378 - 1379	2	690	HOST_CNT	SECONDS * 20
1384 - 1385	2	693	DISC_WORD1	PACKED WORD
1386 - 1387	2	694	DISC_WORD2	PACKED WORD
1388 - 1389	2	695	DISC_WORD3	PACKED WORD
1390 - 1391	2	696	DISC_WORD4	PACKED WORD
1392 - 1393	2	697	DISC_WORD5	PACKED WORD
1394 - 1395	2	698	DISC_WORD6	PACKED WORD
1396 - 1397	2	699	DISC_WORD7	PACKED WORD
1398 - 1399	2	700	DISC_WORD8	PACKED WORD
1400 - 1401	2	701	DISC_WORD9	PACKED WORD
1404 - 1405	2	703	DISC_WORD11	PACKED WORD
1406 - 1407	2	704	DISC_WORD12	PACKED WORD

MEMORY ALLOCATION FOR F1

OFFSET	SIZE	INDEX	VARIABLE	UNITS
1080 - 1081	2	541	PFD_PITCH	DEG * 182.0444
1086 - 1087	2	544	PFD_VDVDAT	FEET * 9.6
1088 - 1089	2	545	PFD_XLDDAT	FEET * 9.6
1090 - 1091	2	546	RADDAT	FEET * 6.5536
1092 - 1093	2	547	PFD_SPDDAT	knots * 500
1094 - 1095	2	548	PITCMDCP	DEG * 182.0444
1096 - 1097	2	549	ROLCMDCP	DEG * 182.0444
1264 - 1265	2	633	IASDAT	KNOTS * 64
1266 - 1267	2	634	ASTDAT	KNOTS * 64
1358 - 1359	2	680	PFD_ROLL	DEG * 182.0444
1360 - 1361	2	681	ALTITUDE	FEET
1364 - 1365	2	683	PFD_ASELDAT	FEET
1374 - 1375	2	688	VSPDAT	(FT/SEC) * 49.152
1376 - 1377	2	689	VMODAT	KNOTS * 65.536
1378 - 1379	2	690	HOST_CNT	SECONDS * 20
1380 - 1381	2	691	BAROSET	INCHES * 100
1404 - 1405	2	703	DISC_WORD11	PACKED WORD

PACKED WORD DESCRIPTIONS

INDEX	BIT(S)	USAGE
599	0-15	map background contol word (Section 7.1)
619	0	TOPMS format valid
	1	left engine o.k.
	2	right engine o.k.
	3-4	takeoff advisory status
	5	flap set error
	6	left or right bleeds off
671	0-3	control mode index
	4-7	throttle mode index
	8-11	roll controller detent status
	12-15	pitch controller detent status
672	0-3	perspective runway approach index
	4-7	reference CAS color code
	8-11	engaged horizontal mode index
	12-15	armed horizontal mode index
673	0-3	engaged vertical mode index
	4-7	armed vertical mode index
	8-11	flight path angle bar color code
	12-15	alert message index
675		
	0	attitude valid
	1	velocity vector mode enabled
	2	altitude hold sub-mode valid
	3	star symbol valid
	4	track bug symbol valid
	5	runway symbol valid
	6	reference altitude valid
	7	radar altitude valid
	8	airspeed valid
	9	altitude valid
	10	PFD format valid
	11	track select valid
693	0	RFDIU disengaged
	1-4	reserved
	5	RFD Nav #2 tuning
	6	DME #2 tuning
	7	RFD com tuning 1 & 2
694	0-6	reserved
	7	stabilizer out of trim
	8-15	reserved
695	0-9	reserved
	10	speed brake-do not arm
	11-15	reserved

696	0	flap placard
	1	throttle placard
	2	speed brake placard
	3	aileron cam out
	4	elevator cam out
	5	rudder cam out
	6-13	reserved
	14	anti-skid inoperative
	15	reserved
697	0-1	reserved
698	0	nose gear up
	1	nose gear down
	2	nose gear locked
	3	nose gear up to down
	4	nose gear down to up
	5	left gear up
	6	left gear down
	7	left gear locked
	8	left gear up to down
	9	left gear down to up
	10	right gear up
	11	right gear down
	12	right gear locked
	13	right gear up to down
	14	right gear down to up
699	0	left flap moving
	1	right flap moving
	2-3	reserved
	4	speed brake armed
	5-9	reserved
	10	left reverser armed
	11	right reverser armed
700	0	left EPR valid
	1	left N1 valid
	2	left EGT valid
	3	left fuel flow valid
	4	left oil pressure valid
	5	left oil temperature valid
	6	left oil quantity valid
	7	left N2 valid
701	0	right EPR valid
	1	right N1 valid
	2	right EGT valid
	3	right fuel flow valid
	4	right oil pressure valid
	5	right oil temperature valid
	6	right oil quantity valid
	7	right N2 valid

702	0	EPR limit valid
	1	EPR caution valid
	2	EPR command valid
	3	N1 limit valid
	4	N1 caution valid
	5	EGT limit valid
	6	EGT caution valid
	7	oil pressure limit valid
	8	oil temperature limit valid
	9	oil quantity limit valid
	10	N2 limit valid
	11	left fuel quantity valid
	12	right fuel quantity valid
	13	center fuel quantity valid
	14	total air temperature valid
	15	gross weight valid
703	0	attitude valid
	1	airspeed valid
	2	selected airspeed valid
	3	airspeed limit valid
	4	altitude valid
	5	selected altitude valid
	6	vertical speed valid
	8	vertical deviation valid
	9	lateral deviation valid
	10	radar altitude valid
	11-12	reserved
	13	true heading valid
	14	true track valid
	15	selected track valid
704	0	track line valid
	1	MLS mode engaged
	2	MLS mode valid
	3	GPS mode valid
	6	wind valid
	7	left flap valid
	8	right flap valid
	9	MLS airplane color variant
	13	ground speed valid
	14	position valid
	15	magnetic track valid

INDAT(N)	DESCRIPTION
1	DP11 FORMAT ID/STATUS
2	DP11 LEFT POT VALUE
3	DP11 RIGHT POT VALUE
4	DP11 BEZEL DISCRETE WORD
5:0-7	DP11 DP ID FOUND BY FORMAT FROM DISCRETES
5:8-15	DP11 DEU ID FOUND BY FORMAT FROM DISCRETES
6	DP11 FORMAT CHECKSUM
7	DP11 FORMAT SPARE TIME APPROXIMATION (msec)
8	DP11 FORMAT TIME FRAME OVERFLOW COUNT
9	DP11 FORMAT TO DP4 I/O INTERFACE MISSES
10 - 32	
33	DP12 FORMAT ID/STATUS
34	DP12 LEFT POT VALUE
35	DP12 RIGHT POT VALUE
36	DP12 BEZEL DISCRETE WORD
37:0-7	DP12 DP ID FOUND BY FORMAT FROM DISCRETES
37:8-15	DP12 DEU ID FOUND BY FORMAT FROM DISCRETES
38	DP12 FORMAT CHECKSUM
39	DP12 FORMAT SPARE TIME APPROXIMATION (msec)
40	DP12 FORMAT TIME FRAME OVERFLOW COUNT
41	DP12 FORMAT TO DP4 I/O INTERFACE MISSES
42 - 64	
65	DP13 FORMAT ID/STATUS
66	DP13 LEFT POT VALUE
67	DP13 RIGHT POT VALUE
68	DP13 BEZEL DISCRETE WORD
69:0-7	DP13 DP ID FOUND BY FORMAT FROM DISCRETES
69:8-15	DP13 DEU ID FOUND BY FORMAT FROM DISCRETES
70	DP13 FORMAT CHECKSUM
71	DP13 FORMAT SPARE TIME APPROXIMATION (msec)
72	DP13 FORMAT TIME FRAME OVERFLOW COUNT
73	DP13 FORMAT TO DP4 I/O INTERFACE MISSES
74 - 96	
97	DP21 FORMAT ID/STATUS
98	DP21 LEFT POT VALUE
99	DP21 RIGHT POT VALUE
100	DP21 BEZEL DISCRETE WORD
101:0-7	DP21 DP ID FOUND BY FORMAT FROM DISCRETES
101:8-15	DP21 DEU ID FOUND BY FORMAT FROM DISCRETES
102	DP21 FORMAT CHECKSUM
103	DP21 FORMAT SPARE TIME APPROXIMATION (msec)
104	DP21 FORMAT TIME FRAME OVERFLOW COUNT
105	DP21 FORMAT TO DP4 I/O INTERFACE MISSES

106 - 128	
129	DP22 FORMAT ID/STATUS
130	DP22 LEFT POT VALUE
131	DP22 RIGHT POT VALUE
132	DP22 BEZEL DISCRETE WORD
133:0-7	DP22 DP ID FOUND BY FORMAT FROM DISCRETES
133:8-15	DP22 DEU ID FOUND BY FORMAT FROM DISCRETES
134	DP22 FORMAT CHECKSUM
135	DP22 FORMAT SPARE TIME APPROXIMATION (msec)
136	DP22 FORMAT TIME FRAME OVERFLOW COUNT
137	DP22 FORMAT TO DP4 I/O INTERFACE MISSES
138 - 192	
193	DP31 FORMAT ID/STATUS
194	DP31 LEFT POT VALUE
195	DP31 RIGHT POT VALUE
196	DP31 BEZEL DISCRETE WORD
197:0-7	DP31 DP ID FOUND BY FORMAT FROM DISCRETES
197:8-15	DP31 DEU ID FOUND BY FORMAT FROM DISCRETES
198	DP31 FORMAT CHECKSUM
199	DP31 FORMAT SPARE TIME APPROXIMATION (msec)
200	DP31 FORMAT TIME FRAME OVERFLOW COUNT
201	DP31 FORMAT TO DP4 I/O INTERFACE MISSES
202 - 224	
225	DP32 FORMAT ID/STATUS
226	DP32 LEFT POT VALUE
227	DP32 RIGHT POT VALUE
228	DP32 BEZEL DISCRETE WORD
229:0-7	DP32 DP ID FOUND BY FORMAT FROM DISCRETES
229:8-15	DP32 DEU ID FOUND BY FORMAT FROM DISCRETES
230	DP32 FORMAT CHECKSUM
231	DP32 FORMAT SPARE TIME APPROXIMATION (msec)
232	DP32 FORMAT TIME FRAME OVERFLOW COUNT
233	DP32 FORMAT TO DP4 I/O INTERFACE MISSES
234 - 256	
257	DP33 FORMAT ID/STATUS
258	DP33 LEFT POT VALUE
259	DP33 RIGHT POT VALUE
260	DP33 BEZEL DISCRETE WORD
261:0-7	DP33 DP ID FOUND BY FORMAT FROM DISCRETES
261:8-15	DP33 DEU ID FOUND BY FORMAT FROM DISCRETES
262	DP33 FORMAT CHECKSUM
263	DP33 FORMAT SPARE TIME APPROXIMATION (msec)
264	DP33 FORMAT TIME FRAME OVERFLOW COUNT
265	DP33 FORMAT TO DP4 I/O INTERFACE MISSES
266 - 288	
289	BIU REAL TIME INTERRUPT COUNT

290	COUNT OF PASSES THROUGH BIU MAIN LOOP
291	BIU HIGH-SPEED BUS TRANSMITTER TIMEOUT COUNTER
292	COUNTER OF DEU RESPONSE FAILURES
293	BIU HSB RECEPTIONS WITH WRONG NUMBER OF BYTES
294	HSB CRC ERROR COUNTER
295	DISCRETE INDICATING BIU ROUTINES EXECUTED
296	NUMBER OF BIU REINITIALIZATIONS AFTER POWER ON
297	NUMBER OF BIU HARDWARE RESETS SINCE POWER ON
298	NUMBER OF UNEXPECTED INTERRUPTS IN THE BIU
299	HSB SECONDARY ADDRESS (DESTINATION OF FRAME)
300	HSB PRIMARY ADDRESS (FILTERS INCOMING FRAMES)
301	BIU HARDWARE INPUT DISCRETE
302	NORDEN TO BIU INPUT COUNTER
303	BIU TO NORDEN OUTPUT COUNTER
304	NUMBER OF TIMEOUTS DURING INPUT FROM DR11 TO BIU
305	NUMBER OF TIMEOUTS DURING OUTPUT FROM BIU TO DR11
306	CONTENTS OF DR11 ADDRESS REGISTER
307	BIU TIMEOUT COUNTER (CLEARED EACH FRAME)
308	NO DR11 ACTIVITY COUNTER
309 - 311	DP14 GENERAL PURPOSE STATUS WORDS
312 - 314	DP24 GENERAL PURPOSE STATUS WORDS
315 - 317	DP34 GENERAL PURPOSE STATUS WORDS
318	UNEXPECTED DR11 WRITE COMMAND COUNTER
319	UNEXPECTED DR11 READ COMMAND COUNTER
320	

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** January 1992	**3. REPORT TYPE AND DATES COVERED** Contractor Report

4. TITLE AND SUBTITLE Advanced Transport Operating System (ATOPS) Color Displays Software Description - Microprocessor System	**5. FUNDING NUMBERS** C NAS1-19038 WU 505-64-13-11
6. AUTHOR(S) Christopher J. Slominski Valerie E. Plyler Richard W. Dickson	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Computer Sciences Corporation 3217 North Armistead Avenue Hampton, Virginia 23666-1379	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Langley Research Center Hampton, Virginia 23665-5225	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER** NASA CR-189605

11. SUPPLEMENTARY NOTES

Langley Technical Monitor: Dr. James R. Schiess (COTR)
Robert A. Kudlinski

12a. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified - Unlimited Subject Category 06	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(Maximum 200 words)*

This document describes the software created for the Sperry Microprocessor Color Display System used for the Advanced Transport Operating Systems (ATOPS) project on the Transport Systems Research Vehicle (TSRV). The software delivery of April 1991, known as the "baseline display system", is the one described in this document. Throughout this publication, module descriptions are presented in a standardized format which contains module purpose, calling sequence, detailed description and global references. The global reference section includes procedures and common variables referenced by a particular module.

The system described supports the Research Flight Deck (RFD) of the TSRV. The RFD contains eight Cathode Ray Tubes (CRTs) which depict a Primary Flight Display, Navigation Display, System Warning Display, Takeoff Performance Monitoring System Display, and Engine Display.

14. SUBJECT TERMS Electronic Flight Instrumentation System Glass Cockpit Primary Flight Display Multifunction Display Navigation Display Flight Display Software	**15. NUMBER OF PAGES** 371 **16. PRICE CODE** A16

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT**

NSN 7540-01-280-5500

C-4

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std Z39-18
298-102