A/F 37A-T88-06-2-0002
15 AUGUST 1988
Change 1, 15 December 1995

IDENTIFYING TECHNICAL PUBLICATION SHEET

I. PURPOSE.

This technical publication sheet is issued for the purpose of identifying commercial off-the-shelf manuals for support of the KC-135 Operational Flight Trainer System.

A/F 37A-T87/T88

BOEING MILITARY AIRPLANES
F33657-85-C-0020

SOFTWARE USERS MANUAL SP1/TEXTURE IMAGE GENERATION SYSTEM

901181-607

January 1989

Manufacturer:	Evans and Sutherland Computer Corp 580 Arapeen Dr Salt Lake City, Utah 84108

II. SUPPLEMENTAL DATA.

Supplemental data attached.

LIST OF EFFECTIVE PAGES

INSERT LATEST CHANGED PAGES, DESTROY SUPERSEDED PAGES

NOTE: The portion of text affected by the changes is indicated by a vertical line in the outer margins of the page. Changes to illustrations are indicated by miniature pointing hands. Changes to wiring diagrams are indicated by shaded areas.

Dates of issue for original and changed pages are:

Original	0	January 1989
Change	1	15 December 1995; PCR 950119110

TOTAL NUMBER OF PAGES IN THIS SUPPLEMENT IS 4 CONSISTING OF THE FOLLOWING:

†Page No.	*Change No.	†Page No.	*Change No.	†Page No.	*Change No.	†Page No.	*Change No.
Title	1						
A	1						
i	1						
ii Blank	0						

* Zero in this column indicates an original page

2-1. DISCLAIMER STATEMENT.

2-1.1 The following discrepancies have been observed in this technical manual:

The following pages are included in the manual but are not listed on page ii, List of Effective Pages:

3-7
3-8
4-11 through 4-14
6-27
6-28
8-7 through 8-12

Since this is a COTS manual, FlightSafety Services Corp. is not at liberty to change it. Evans and Sutherland Computer Corporation will not make corrections or changes to this manual; therefore, this information has been added for any further observations of this manual.

CONTENTS

ILLUSTRATIONS

TABLES

BLANK

SECTION 1

INTRODUCTION

1.1 OBJECTIVE AND ORGANIZATION OF DOCUMENT

This manual contains information on the basic operation of the operating system, utilities, and modeling programs for the SP1/T system. All messages or prompts output by the computer to the terminal are underscored throughout this document. A carriage return is indicated by the symbol [CR]. The lower case XX at the end of some file names represents the current version numbers of those files. The manual is divided into nine sections, with each section containing the particular type of information described below:

SECTION 1	INTRODUCTION -- Contains a brief description of the manual contents.
SECTION 2	MONITOR AND UTILITY PROGRAMS -- Contains an overview and operating instructions on the monitor and associated utility programs.
SECTION 3	MODEL CONCEPTS AND ORGANIZATION -- Discusses basic model concepts and organization.
SECTION 4	PROGRAM CONVENTIONS -- Introduces modeling conventions and modeling program execution.
SECTION 5	LIGHT MODULE FUNCTIONS -- Explains detailed modeling procedures for the light module.

SECTION 6	SURFACE MODULE FUNCTIONS -- Covers detailed modeling procedures for the surface module.
SECTION 7	ENVIRONMENT MODULE FUNCTION -- Contains detailed modeling procedures for the environment module.
SECTION 8	GROUP MODULE FUNCTION -- Explains detailed modeling procedures for the group module.
SECTION 9	MODEL MERGE PROGRAM (MERXxx) -- Covers the operating instructions for the model merge program.

SECTION 2

MONITOR AND UTILITY PROGRAMS

2.1 INTRODUCTION

This section discusses the procedures necessary to boot the system and load the monitor for basic system control. Monitor commands are then detailed in alphabetic order. Next, a discussion of command files is given (with a sample command file in appendix B). Finally, several utility programs which are used in association with the monitor commands and are useful for file manipulation and control are described.

2.2 INITIAL PROGRAM LOADING

All software is on floppy diskettes containing executable object programs for NOVOVIEW SP1/TEXTURE operation and maintenance. These programs are loaded into computer memory from the diskette using the primitive boot loader and system loader, both of which are also stored on the diskette. A "bootstrap" procedure performs initial program loading. It includes the following three steps:

1. Transferring the boot loader from diskette to computer memory using the Initial Program Load (IPL) feature of the floppy disk interface.

2. Executing the boot loader to transfer the system loader from diskette to computer memory.

3. Executing the system loader to transfer the desired executable program from diskette to computer memory.

There are two types of executable object files or programs: stand-alone programs and user-space programs. Stand-alone programs are those programs which do not require the use of the monitor or operating system during execution. The monitor or operating system itself falls into this category. These stand-alone programs are loaded by following the above outlined "Bootstrap" procedure.

User-space programs are executable object programs which require the monitor to be loaded first and are loaded and executed under the control of the monitor. Procedures for loading and executing user-space programs are given in Section 2.3. Examples of the two types of programs are given in Table 2-1. The lower-case xx at the end of some program names represents the current version numbers of those programs.

TABLE 2-1
EXAMPLES OF EXECUTABLE OBJECT FILES

STAND-ALONE PROGRAMS	USER SPACE PROGRAMS
MONXxx	BLDXxx
FLYXxx	NDLXxx
DMPDTS	P1Txxx
TASPDT	DSKXxx
INTMER	OBJCPY
MRAXXX	CATXxx
RRAXXX	TSE980
SKIPAX	DSCXxx
SHIFTA	E1Txxx
PIFAXX	G1TXxx
BYTEAX	S1TXxx
MEMAXX	LT1Xxx
ROMAXX	
M980RX	
FDBXxx	

2.2.1 BOOTSTRAP LOADER

The Bootstrap Loader is a small program used to read the system loader from diskette. The bootstrap loader resides on the first sector of the first track of a properly formatted diskette. See Section 2.2.5 for procedures for storing the bootstrap loader onto a diskette. During initial program load, the bootstrap loader is transferred from the first sector of diskette zero into memory locations 0000 through 007F by the use of the IPL switch on the AED unit. Execution of the program begins at address 0000. When the program is executed, it locates the system loader in the directory of the diskette in drive zero, then loads and executes the system loader.

2.2.2 BOOTSTRAP LOADER ERROR INDICATIONS

There are two places where the bootstrap loader encounters errors: locating the system loader on the diskette or reading the system loader. If the bootstrap loader encounters an error, the computer will idle at location 0024 and the M register will contain 0007. If this error occurs, it requires starting the load procedures from the beginning.

2.2.3 SYSTEM LOADER

The system loader is used in either a stand-alone mode or by the operating system to load executable object files into memory. The system loader itself is stored on diskette as an unblocked binary file for easy loading by the bootstrap loader. Procedures for placing the system loader on a diskette are given in Section 2.2.6.

When the system loader is executed in its stand-alone mode during initial program loading it offers the option of automatic loading or operator intervention, depending on the position of sense switch 1. When the sense switch is down, the previously specified default object program is automatically loaded into memory starting at address 0220 and then executed.

When sense switch 1 is up, operator intervention is allowed at three points in the loading process:

1. Operator types in the directory name of the program to be loaded.

2. Operator may inspect and/or change program offset (if program is relocatable) and memory limits to be used during the loading process.

3. System loader stops at the end of the loading and waits for operator to start program execution.

Sense switch 1 may be placed in the down position at either of the first two points of operator intervention. Remaining points of intervention will be bypassed.

Detailed procedures for use of the system loader during the initial load process are given in the following sections. Procedures for loading user-space programs using the operating system are described in Section 2.3.

2.2.4 SYSTEM LOADER ERROR INDICATIONS

When the system loader detects an error during an initial program load, the program will idle at either location 0093 or 0199; the M register will contain the error code. See Table 2-2 for an explanation of load error codes.

When the system loader is being used as an integral part of the operating system, errors detected by the system loader are printed out on the terminal. Error codes are identical with those returned in the M register described above and listed in Table 2-2.

**TABLE 2-2
LOAD ERROR CODES**

M REGISTER	LOADER STATUS CODES
0000	No error, load complete
0001	Input read error
0002	Checksum error
0003	Missing identification record
0004	Illegal format code
0005	Program too large
0006	No entry address
0007	Disk file name error
0008	Disk date error
0009	System file not found (initial program load only)

2.2.5 INITIAL PROGRAM LOAD PROCEDURE

A. Automatic load with sense switch 1 down.

STEP	INSTRUCTION	RESPONSE
1.	Ensure that all units are powered ON.	Power indicators illuminate.
2.	Insert diskette into drive 0 and close door.	
3.	Place sense switch 1 DOWN.	
4.	Set MODE switch to HALT.	
5.	Press RESET twice.	All red panel lights go out.
6.	Press IPL on disk unit.	Disk head loads for transfer of Bootstrap loader to computer memory.
7.	Set MODE switch to RUN	
8.	Press START	Disk head loads for transfer of system loader to computer memory. Default program is located in directory. Program is loaded into memory with 0220 offset. Program is executed.

B. Initial program load with operator intervention (sense switch 1 up).

STEP	INSTRUCTION	RESPONSE
1.	Ensure all units are powered ON.	Power indicators illuminate.
2.	Place diskette in drive 0 and close door.	

STEP	INSTRUCTION	RESPONSE
3.	Place sense switch 1 UP	
4.	Set MODE switch to HALT	
5.	Press RESET	
6.	Press IPL on disk unit	All red front panel lights go out. Disk-head loads for transfer of bootstrap loader to computer memory.
7.	Set MODE switch to RUN	
8.	Press START	Disk head loader for transfer of system loader to computer memory. Terminal prints: "?"
*9.	Type 6-character name of object file to be loaded. (If name is less than 6 characters, type RUBOUT for remaining characters.)	Computer idles at 008A. (If computer idles at 0011, name was not typed correctly or the file was not found on the diskette. Begin again at Step 5.)

NOTE

USE DELETE ON TI 820 terminal.

STEP	INSTRUCTION	RESPONSE
10.	If memory offset other than 0220 is desired, load desired offset into A register.	
11.	If memory limits (other than full memory limits) are desired, examine E register and load desired memory limits into E register.	
*12.	Press START	Program is loaded from disk. Computer idles at location 008E.
13.	Press START	Program begins execution.

* If sense switch 1 is placed in the down position prior to performing this step, all steps following the current step will be bypassed.

2.2.6 INSTALLING THE BOOTSTRAP AND SYSTEM LOADERS ON DISKETTE

The program FDBXxx is used to initially install or to reinstall the bootstrap and system loaders on a diskette. During this process, 1) the directory file name of the system loader is defined within the bootstrap loader and 2) the directory name of a specified system object file (to be loaded by default) is defined within the system loader. Thus, one reason for reinstalling the bootstrap and system loaders on a diskette is that they may be located, loaded, and executed during an automatic load procedure.

Before FDBXxx is executed, the file into which the system loader will be written (normally named SYSLOD) must have been previously defined on the diskette. The system file whose name will be defined within the system loader by FDBXxx need not be defined on the disk before FDBXxx is executed. However, this system file must be defined on the diskette previous to performing an automatic load procedure, or a load error 0007 will result.

Consider the following example of placing the bootstrap loader, system loader, and system monitor on an initialized diskette.

1. Mount a system diskette in drive 0 and an initialized diskette in drive 1.

2. Boot the operating system.

3. Define the system loader file on the diskette in drive 1, using the following command:

```
DEFINE,FD1,SYSLOD,64,1,12
```

4. Define the system monitor file using the following command:

```
DEFINE,FD1,MONXxx,64,3,0
```

5. Using the program OBJCPY, copy the file MONXxx from the diskette 0 to 1.

6. Boot the program FDBXxx, using the sense switch 1 up option as described in Subsection 2.2.5. When the program begins execution, the loader file name will be requested by the message:

```
FDBASR   VXX
LOADER FILE NAME =
```

7. Remove the system diskette from unit 0 and replace it with the diskette from unit 1. Enter the six character file name "SYSLOD". The program will then request the system file name by the message:

   <u>SYSTEM FILE NAME =</u>

8. Enter the six character file name MONXxx.

9. Both the bootstrap and system loaders are written on the diskette and the following message is printed, indicating completion of the operation:

   <u>LOADER INSTALLED ON DISKETTE</u>
   <u>PERFORM DISKETTE BOOT TO START SYSTEM</u>

2.3 BASIC OPERATING SYSTEM - MONXxx

The basic operating system, MONXxx, provides for simplified operation and control of the programs provided with the NSP system. MONXxx is provided as a custom configuration of the TI 980B/SPC9800 basic system software and provides the following facilities:

1. System loader for operating system and program loading.

2. Computer functions for power fail/restart processing, memory protect/privileged instruction monitoring, memory address mapping and protection, and control of program execution.

3. I/O processing functions and association of logical units to physical devices.

4. Operator communication functions for program loading, assignment of logical/physical units, and program execution.

5. File management for allocation and maintenance of disk files.

The MONXxx program is an updated version of older TI980 monitors such as BSMONT, SYSMON, SYSX02, or NEWMON.

Features of this new monitor include:

1. Double buffered input and output for high-speed reading and writing to disk.

2. Additional monitor commands and new command file processing.

3. Simplification of command format, eliminating the need for enclosing the commands between // and period. Full six-character commands are not required; three characters are sufficient.

4. Inclusion of all device drivers in the monitor, which gives a capacity for handling three keyboard devices: TI733, TI820, and video terminal VT100. The system starts up in slower TI733 mode and must be set for TI820 use with the HSP command or for VT100 use with the CRT command.

5. Expansion of user protocol for inputting characters using the PRB (physical record block); this allows a desired maximum character count to be specified. When the desired number of characters are input, control returns to the user program automatically, without the need for a carriage return or an escape.

CAUTION

This change may cause incompatibility with some previous user programs.

6. Prevention of RENAME command from forming duplicate file names on a disk.

2.3.1 MONITOR COMMANDS

The following commands will be discussed in this section:

ASM - ASSEMBLE	EXE - EXECUTE
ASS - ASSIGN	HSP - High Speed Printer
COP - COPY	LIN - LINK
CRT - CRT terminal	LOA - LOAD
DEF - DEFINE	REL - RELEASE
DEL - DELETE	REN - RENAME
EDI - EDIT	REW - REWIND
END - ENDFILE	Other commands

Monitor commands for MONXxx are similar to earlier monitor commands with the following exceptions:

1. Slashes (//) are optional at the beginning of the command sequence, and a period is optional when terminating a sequence.

NOTE

If slashes are used at the beginning, a period is required at the end of the sequence. However, in command files the slashes and period are required, not optional.

2. Only the first three characters of a command are required; the rest are optional. (Examples: EXEcute, LINk, ASMble, ASSign, EDIt, LOAd. Lower case letters indicate optional characters.)

3. Assignments of source disk and file, destination disk and file, and listing device, as well as execution of a program, can be done with one commmand sequence. (i.e., EDI,FDO,FILE1,FD1,FILE2,KEY).

4. Either commas or spaces can be used to separate the entries in the command sequence. (i.e., ASS,4,KEY or ASS 4 KEY).

5. To edit or assemble into a file that already exists on the disk without creating a new file use the :EF command after the disk number (i.e., FD1:EF).

The monitor will recognize the following physical device abbreviations:

CLP	Centronics line printer
CS1	Cassette tape, unit one
CS2	Cassette tape, unit two
DMY	Dummy assignment for required assignments when no physical device assignment is desired.
FDO-FD3	Floppy disk, top side
FTO-FD3	Floppy disk, top side
FBO-FB3	Floppy disk, bottom side
HAZ	Hazeltine CRT
KEY	Keyboard or terminal
TEK	Tektronix
TIP	TI printer (810)
UNL	Unloaded command file: refers to a section of a command file that has been unloaded from disk into TI memory.
VES	Vessel driver: refers to a buffer in the monitor which is used to provide command inputs to user programs.

Internal I/O drivers adjust the data formats and baud rates to interface with these devices.

2.3.1.1 ASMble

The assemble command is used when assembling source code into object files. The command format is:

```
ASM,(disk),(file name out),(disk),(file name in),(listing device)
```

The file name out and the preceding disk number specify the output object file. The file name in and the preceding disk number specify the source file input. The listing device specifies where the listing will be printed.

Example 1:

```
ASM,FD1,OBJ000,FD0,SRC000,TIP
```

In this example the source file SRC000 on floppy disk 0 will be assembled. The object file will be written into OBJ000 on floppy disk 1 and the listing will be printed on the TI 810 or TI 820 line printer.

Example 2:

```
ASM,DMY,,FD0,SRC000,TIP
```

In this example no object file will be created, but a listing file will be printed.

Example 3:

```
ASM,DMY,,FD0,SRC000,DMY
```

In this example the source file is assembled, but no listing or object file is created.

NOTE

Either commas or spaces may be used.

Example 4:

```
ASM FD1:EF OBJ000 FD0 SRC000
```

In this example the object is assembled into an existing file OBJ000 on disk 1. Since no listing device is specified, the device previously assigned to logical unit 6 will be used as the listing device. If no device has been assigned to logical unit 6, an error message will be printed out.

NOTE

Care should be taken when assembling or editing into an existing file. If the old file size is exceeded, the assembled data will be lost.

2.3.1.2 ASSign

The assign command is used to assign any physical device to a logical unit. The command format is:

```
    ASS,(logical unit),(physical unit),(file name)
```

The physical unit may consist of a disk number and name, or a device name.

```
    Examples:
                    ASS,8,FDO,INFILE
                    ASS 4 KEY
```

If ASS[CR], is typed, all of the current logical unit assignments will be listed. (Throughout this document [CR] means carriage return.)

2.3.1.3 COPy

The copy command is used when copying an object file from one disk to another or when concatenating object files together to form a library. The format is:

```
    COP,(disk),(file name out),(listing device)[CR]
    ?(disk:name 1),(disk:name 2),....(disk:name n),[CR]
    ?(disk:name m)(. or #)[CR]
```

After typing the first line the operator will give a [CR], whereupon the system will prompt with a question mark for each succeeding command line.

A period or a pound sign (#) must follow the list of object files to be copied and/or concatenated. A period causes two end-of-file records to be appended to the file. A pound sign causes a single end-of-file record to be written. Single end-of-file records are used to terminate the TI computer diagnostic programs.

```
    Example 1:
                    COP,FDO,MONXO5,KEY
                    ?FD1:MONXO5.
```

In this example the program MONXO5 will be copied from FD1 to FDO. The IDT names and any error messages will be printed on the terminal.

```
    Example 2:
                    COP FD1:EF TSE980
                    ?TSE980#
```

In this example TSE980 will be copied from FD0 (the default when FD0: is not typed) to an existing file on FD1. IDT name will be printed on the device last assigned to logical unit 6. A single end-of-file record will terminate the new object file.

Example 3:

```
COP FD1 LIB000 KEY
?OBJ000,OBJ001,OBJ002.
```

In this example an object library will be formed by, concatenating from disk 0, the files OBJ000, OBJ001, and OBJ002 into a file LIB000 on disk 1. Two end-of-file records will terminate the library after OBJ002 is copied. Files OBJ000 and OBJ001 will have no end-of-file marks.

NOTE

The LINKG program requires two end-of-file records to terminate all library input files.

2.3.1.4 CRT

The CRT command is used to change the terminal I/O driver to interface with the 9600-baud CRT terminal. The command is simply CRT [CR].

2.3.1.5 DEFine

The define command will open a file on the specified disk and give it the specified name, record size, type, and extent.

The general format is:

```
DEF,(disk),(file name),(record size),(type),(extent)
```

The record size refers to the number of characters or bytes per record. The standard record size is 64 characters, which allows exactly four records to be packed into each 256-byte disk sector. However, if the record size were specified as 80, only 3 records could be written per sector with 16 bytes unused or wasted.

File types are: 0- unblocked temporary; 1- unblocked permanent; 2- blocked temporary; 3- blocked permanent. The file extent is specified as the number of sectors for blocked files or the number of records (should be evenly divisible by 4) for unblocked files. A file extent of 0 produces an open-ended file which takes up all remaining space on the disk. Such open-ended files are reduced to proper length when the file is closed.

Example:

```
DEF,FD0,SYSLOD,64,1,12
```

This will define a file called SYSLOD on disk 0 having a record size of 64 characters or bytes, unblocked permanent, and an extent of 12 records (3 sectors with 4 records per sector).

2.3.1.6 DELete

The delete command will change the file name on the specified disk to a file name of VACANT. When the disk is copied using the DSC (disk save and compress) program, the VACANT files will be omitted and the new copy will be compressed. Deleted files can be recovered using the REN command, if done before the disk is compressed. (See REN command.) If an open-ended file is deleted using this command, it is completely removed from the directory.

Example:

```
DEL FDO SRC000
```

In this example the DELete command will change the name of SRC000 to VACANT in the disk 0 directory.

2.3.1.7 EDIt

The edit command is used when creating or editing source files. The command format is:

```
EDI,(disk),(file name out),(disk),(file name in),(listing device)
```

The listing device is the device on which the editor will print when the "P" editor command is used. The insert device assignment defaults to the terminal.

Example 1:

```
EDI FD1 SRCOUT FDO SRCIN CLP
```

In this example a file called SRCIN will be read in from floppy disk 0. When the edit session is completed the edited source file will be written into a file SRCOUT on disk 1. When editing, the "P" command will cause a printout on the Centronics line printer.

Example 2:

```
EDI,DMY,,FDO,@EDI,TIP
```

In this example the file @EDI will be opened for examination. A "P" command in the editor will cause the source information to be printed on the TI printer. No output file will be written.

Example 3:

```
REL
EDI FD1 FILE3 FD1 FILE2
LUN 06 ERR 0010 AT 00BF LL 3516 is printed
ASS 8 FD1 FILE1
REW 8
ASS 6 KEY
EXE
EDIT is printed
?D9999
?K
?I
?Q
```

In this example the files FILE2 and FILE1 will be combined together to form FILE3. FILE3 will contain the information of FILE2, followed by the information that is in FILE1. Since the EDI command defaults the editor's insert file assignment to the keyboard, not assigning KEY as the listing device after doing a REL will kick the monitor out of the EDI command sequence after the files have been opened and TSE980 has been loaded. At this time a new assignment can be made to the insert file. Remember to rewind the insert file and assign 6 to the listing device before executing TSE980

Example 4:

```
EDI,FD1:EF,SRC000,FD1,SRC000,KEY
```

An existing file can also be edited, as in this example, if the user is careful not to exceed the file size. If the file size is exceeded, the data in the file may be lost.

Example 5:

```
EDI,FD1,SRC000,DMY,,KEY
```

In this example a new file called SRC000 will be created on FD1 and the terminal will be assigned as the listing device.

2.3.1.8 ENDfile

The endfile command is used to place an end-of-file record in an existing file. This command is rarely used and the user must be careful when using it to ensure that the file pointer is positioned at the end of the file. The format is:

```
END,(logical unit number)
```

2.3.1.9 EXEcute

The execute command is used to load and execute an executable object file from disk, or to execute a program already in user space of TI memory. The format is:

```
    EXE,(disk),(file name)

    Example 1:
                        EXE FD1 RUNX01
```

In this example the file RUNX01 will be loaded in from disk 1 and then executed.

```
    Example 2:
                       EXE
```

In this example the user program already in TI memory will be executed.

2.3.1.10 HSP

The HSP command is used to change the monitor's I/O driver to interface with the high-speed printer terminal (TI 820). The command format is simply HSP [CR].

2.3.1.11 LINk

The link command is used when linking up to three concatenated object files into one executable file. The format is:

```
    LIN,FD0,NDLX01,FD1,NDLL01,FD1,NDLT02,DMY,,TIP
    00,MAIN02,NDLX01,1,2,0
    /*
```

In this example an executable object file called NDLX01 will be created on FD0 by linking together NDLL01 and NDLT02 from FD1. The third library is not used, so DMY is inserted in the disk entry and two commas follow. The load map will be listed on the TI printer. The module where program execution will begin is given as MAIN02. The IDT name of the linked file will be NDLX01. The libraries will be linked in the order specified (i.e., with the first input file linked first and the second input file linked second). Since no third file is specified, a 0 is used to terminate the input order sequence. The /* followed by a [CR] starts the link process.

2.3.1.12 LOAd

The load command will cause an object program to be loaded into TI memory user space but will not be executed. The format is:

```
LOA,(disk),(program name)
```

2.3.1.13 RELease

The release command is used to release a logical device from a physical device assignment, or to release all logical device assignments. The format is:

```
REL,(logical unit)
```

Example 1:

```
REL 6
```

In this example logical device 6 is released from its previous assignment.

Example 2:

```
REL
```

In this example all logical devices from logical device 5 and above are released from their physical device assignments.

2.3.1.14 REName

The rename command is used to change a file name that appears in the directory on a disk. The command format is:

```
REN,(disk),(old file name),(new file name)
```

Example:

```
REN FDO TEMP1 TEMP2
```

In this example the file TEMP1 will be changed to TEMP2 as it appears in the directory on disk 0.

When renaming file names that appear in the directory more than once, only the first occurrence of the old name will be changed (i.e., when renaming VACANT files to recover a file inadvertantly deleted).

2.3.1.15 REWind

The rewind command is used to rewind a disk file.

Example:

```
REW FDO SRC000
```

In this example the source file SRC000 on disk 0 will be rewound causing its file pointer to be moved to the top of the file. This command may also be used to see whether a file exists on a disk.

2.3.1.16 OTHER MONITOR COMMANDS

There are two other monitor commands that are recognized and documented in the TI monitor documentation. They are the SWAPCS command and the SKIPCS command. The use of these commands has been replaced by the new command file structure, but the commands have been left in the monitor so that old job control files can be run.

2.4 COMMAND FILES

A command is created by using the editor. A command file allows the operator to execute a sequence of monitor commands together, which may be cumbersome or time consuming to type individually. All command files have names of the form @XXXXX where XXXXX can be any five alpha or numeric characters.

2.4.1 EXECUTION MODES

The command file can be built to run in two execution modes, either disk or unloaded mode. In the disk mode, commands are executed as they are read in from the disk file, one line at a time. In the unloaded mode, the entire command file is unloaded from the disk into TI memory before it is executed.

The unloaded mode allows both disk drives to be used, once the command file is unloaded. In the unloaded mode the command file over-writes the DBUG portion of the monitor. This requires that the monitor be reloaded if DBUG must be used.

The keyboard entry to execute a command file is @XXXXX where @XXXXX is the command file's file name. Command files can also be executed by using the CMD monitor command.

Examples:

```
            @XXXXX
       will run command file @XXXXX on FD0.

           CMD FD1 XXXXXX
       will run command file XXXXXX on FD1.
```

2.4.2 COMMAND FILE CONVENTIONS

The command file has the following conventions and restrictions:

1. The first line is always
   //ECM.
   to cause the monitor to enter the command mode.
2. Command file commands begin with a $.
3. Other commands must begin with // and end with a period.
4. Either spaces or commas may be used as delimiters.
5. Text lines to be printed have no preceding $.
6. Comments can be included on the command line after the required input fields have been satisfied. Separate comments from input fields with a space.
7. Command files to be unloaded from the disk must be enclosed between $U commands.
8. The command mode is reset with a $B command at the beginning of the file, usually on the second or third line.
9. There are ten internal registers which are denoted <<0 through <<9. Each can contain up to six ASCII characters.
10. A carriage return response to the $Q command will cause the next command to be executed. Any other response will cause the next command to be skipped.
11. Non-printing ACSII characters can be designated by giving the ASCII code preceded by a carat >. For example:

    Tab = 89, Escape = 9B.

    This is useful for control purposes when used by the $T command.

2.4.3 COMMAND DESCRIPTIONS

The following are legal commands for the command file structure:

$B	Begin the command file mode.
$E	Exit the command file and return to the monitor.
$F $+/-n	Failure. On failure, proceed +/-n lines from the current line. If no failure, continue.
$G $+/-n	Go to. Proceed +/-n lines from the present line.
$J $+/-n $+/-m	Jump to subroutine. Go to subroutine located +/-m lines from current line and, when $R is executed, return +/-n lines from the $R line.
$M	Monitor command line. In the command file, all monitor commands must be preceded with // and will end with a period.
$Q <<n m	Question command. Input m characters into register n. The integer m is optional and is used to terminate the input and precede after m characters have been input. The range of m is from 1-6. If m is not specified, it will default to 6. If the user reply to the question is a carriage return, the next line will be executed. Any other response will result in the next line being skipped. With this feature a $E command can immediately follow the $Q command, giving the user an opportunity to abort the process.
$R	Return from subroutine. Return +/-n lines from present line as previously specified in the $J command.
$S << ...<<r	Input a sequence of register names to the internal registers indicated. Up to 5 internal registers may be filled using one $S command.
$T <<n g $+/-m	Test the register n for single character g and, if equal, proceed at the current line +/-m. If not equal, continue.
$T -<<n g $+/-m	Test the register n for single character g and, if not equal, proceed at the current line +/-m. If equal, continue.

$U	Unload command. Command files to be unloaded must have $U as the second and last commands in the file.
$V	Vessel command. Used as a subcommand file when executing a user program from the command file. A $V 0 will flush the vessel buffer. A $V n will input the next n lines to the buffer. Limit n<10. A $V will rewind the vessel buffer and assign 4 to VES. See examples contained in appendix B.
$W	Without. The $W command will allow the text on that line to be printed on the terminal without preceding it with a carriage return and line feed.

Appendix B contains a sample command file. Most command file command types are used in this file.

2.4.4 USER PROGRAM CONTROL COMMAND FILE EXAMPLES

Control inputs to user programs can be provided by the command file in either of two ways: first, by assigning control to UNL or second, by using the vessel driver or $V commands. The following examples will illustrate how to use these two methods.

Example 1:

```
   :
   :
   $M //ASS,4,UNL.
   $M //EDI,DMY,,FDO,,SRC,TIP.
   D9999
   T
   P9999
   Q
   $E
```

In this example, control is given to the unloaded command file. After the editor is executed, it will do the D9999,T,P9999, and Q operations as the unloaded command file requests, then return control to the monitor.

Example 2:

```
   :
   :
   $VO                          will reset the vessel
                                driver.
   $V3                          next 3 lines to vessel
                                buffer.
   D9999                        read input file into buffer.
   T                            position pointer at top of file.
   P23                          print 23 lines.
   $V                           rewind VES and assign it
                                control.
   $M //EDI,DMY,,FDO,SRCIN,KEY.
   $E
```

In this example the vessel driver is used. When EDI is executed, the three commands in the vessel will be executed. Since no Q command has been executed, control will be given to the keyboard. When the editing task is completed and when the operator types in Q, control will return to the command file. When the $E command is executed, control will return to the monitor.

NOTE

Using the vessel driver in this way will allow operator intervention during command file execution. This cannot be done using the UNL control assignments.

Example 3:

```
   :
   :
   $S <<0,<<1,<<2                 Input names from keyboard.
   $V 0                           Reset the vessel driver.
   $V 4                           Next 4 lines to vessel
                                  buffer.
   D9999
   T
   P9999
   Q
   $V                             Rewind VES and assign it
                                  control.
   $M //ASS,6,TIP.                Listing device assignment.
   $M //REW,6.                    Cause a page eject.
   $M //EDI,DMY,,FDO,<<0.         Edit file in <<0.
   $M //REW,6.                    Page eject.
   $V
   $M //EDI,DMY,,FDO,<<1.         Edit file in <<1.
   $M //REW,6.
   $V
   $M //EDI,DMY,,FDO,<<2          Edit file in <<2.
   $E                             Return to monitor.
```

In this example the vessel is reset and four lines are input. The vessel is then rewound and control is assigned to the vessel driver. The editor is executed and the four vessel commands are executed. This will cause a listing of the file specified in <<0 to be printed on the TI line printer. The edit will end on "Q", the last command from the vessel. Control will return to the command file. The second $V will again rewind and assign control to the vessel driver. The process will repeat until the two other files have been listed. $E will return control to the monitor. The $M //REW,6. commands will cause a page eject on the printer.

Note the value of the vessel driver for repetitive operations and user control in the examples given. Also, note the simplicity and ease of use of the UNL control mode.

The vessel driver is also useful when an internal register needs to be part of a user program command.

Example 4:

```
   :
   :
   $M //ASS,5,VES.                5 is control for LINKG.
   $V 0                           reset the vessel driver.
   $V 2                           input 2 lines to VES.
   00,<<0,RUNX01,1,0
   /.
   $V
   $M //LIN,FDO,OUT,FDO,IN,DMY,,DMY,,TIP.
   $E
```

In this example the vessel contains the 2 control inputs to the link program. Note that the user can specify the entry module name earlier in the command file by putting the module name in the internal register <<0. Also, note that a special control assignment needs to be made for the linker. Normally, user programs assign control to logical unit 4. Therefore, the $V command, when it does its automatic control assignment, will ASS 4 VES. Since the linker uses logical unit 4 for control, a special assignment needs to be made in the command file.

This example also shows a /. for a start link command instead of the /* which is normally used. The /. will be interpreted by the vessel driver as a /* and will pass a /* to the linker whenever this user command is seen. This is done since, when editing the command file, a /*, in the editor is interpreted as an end-of-file and will not appear in the edited file.

2.5 VOLUME LISTING PROGRAM – CATXxx

The program CATLOG provides the user the facility to list the directory of a diskette volume. In order to run the program, assign logical unit 6 to the printing device, assign logical unit 32 to the diskette volume, then load and execute the program. For example:

```
ASSIGN,6,KEY
ASSIGN,32,FDO
EXECUTE,FDO,CATXxx
```

2.6 DISK SAVE AND COMPRESS

The Disk Save and Compress (DSC) utility program copies permanent files contained on the diskette in the input drive into the diskette in the output drive for backup and storage. DSC can also initialize, certify that the disk surface is good, zero a directory, and list a directory for a diskette.

2.6.1 TO INITIATE DSC

To initiate DSC, execute the following command strings after the system monitor has been loaded and executed.

```
1.   ASSIGN, 4, KEY.              ; Console input
     ASSIGN, 5, KEY.              ; Console output
     ASSIGN, 6, KEY.              ; List device
     EXECUTE,FDO,DSCXxx.          ; Load and execute DSC

2.   @DSC                         ; Assigns logical units 4 and 5 to
                                    KEY; asks for the operator to enter
                                    the LIST device; and then executes
                                    DSC.
```

2.6.2 DSC COMMANDS

FORMAT	DESCRIPTION
? [CR]	Prints a list and brief description of all program recognized commands.
ID (DEV) [CR]	Sets the input drive to DEV where DEV is a valid diskette drive name. If DEV is omitted, prints the current device assignments. The input drive is initially set to FD0.
OD (DEV) [CR]	Sets the output drive to DEV where DEV is a valid diskette drive name. If DEV is omitted, prints the current device assignments. The output drive is initially set to FD1.
PD (DEV) [CR]	Change the list device to DEV where DEV is a valid output device. If DEV is omitted, prints the current device assignment.
ZE (disklabel) [CR]	Zero the directory on diskette in the output drive. If specified, write the disklabel.
IN (disklabel) [CR]	Format the diskette in the output drive by writing timing marks for 32 sectors per track. Certify the diskette and write the disklabel, if specified.
CE (disklabel) [CR]	Certify diskette surface in the output drive by writing to and reading from every sector. If specified, write disklabel.
CO (disklabel) [CR]	Copy permanent files from the diskette in the input drive to the diskette in the output drive. Skip temporary and VACANT files. This single option is used to copy and/or compress diskettes. If specified, writes the disklabel. Otherwise, copies the input disklabel.
DI (disklabel) [CR]	Output a brief directory for diskette in the output drive.
DII (disklabel) [CR]	Output a directory for the diskette in the input drive.
BY [CR]	Return control to the monitor.

LA (disklabel) [C/R]	Write a label on the output diskette.
LAI (disklabel) [C/R]	Write a label on the input diskette.
FC (filename) [C/R]	Copy a file from the input diskette to the output diskette.

2.6.3 DISKLABEL

The disklabel field specifies the label or disk identification to be stored on the diskette as part of the directory. This information is then printed at the top of the directory listing using the DI or DII commands. The disklabel field can contain up to 64 characters.

In the case of the DI or DII option, the disklabel field is not used to specify the disk identification record, but may be used to specify the date or some other comment to be printed on the directory listing. A maximum of twelve characters is allowed in the directory options.

2.6.4 DIRECTORY FORMAT

The directory format is composed of five columns of the following:

Name type size in decimal

Types are as follows:

.	permanent/blocked
*	permanent/unblocked
.	temporary/blocked
*	temporary/unblocked
??	bad type (includes open-ended files)

The final line of the directory is a summary of the total sectors and files used.

2.6.5 DSC ERROR MESSAGES

MESSAGE	DESCRIPTION
DISK FAILURE*	This is a FATAL error for both drives. One may recover the diskette only on drive 1, by executing initialize option.
SIZE FAILURE	This is a WARNING error for the input drive. However, all files are still transferred.
VERIFY FAILURE	This is a FATAL error for the output drive. The diskette has a bad surface.

2.6.6 DSC EXAMPLE

The following is an example of the uses of DSC. In this example, a new diskette is initialized and certified and then the contents of another diskette are copied (and compressed) onto the new diskette.

STEP	INSTRUCTION	RESPONSE
1.	Initiate DSC following the instructions in Section 2.6.1.	Terminal prints: DSCXxx TURN [WP] AND [INIT] SWITCH ON *** DSC>.
2.	Turn WP and INIT switches on.	
3.	Place new diskette in the output drive and diskette to be copied into the input drive.	
4.	Type IN[CR].	Diskette in the output drive is initialized. Since no disklabel was given, a null label is specified on the diskette. Terminal responds with DSC>.
5.	Type CE[CR].	Diskette surface is certified. The null label is again used. Terminal prints: DSC>.
6.	Type CO NEWNAME[CR].	Diskette 0 is copied onto diskette 1. Diskette 1 is given the identification label of "NEWNAME". Terminal prints: DSC>.
7.	Type DI 8MAY80[CR].	Brief directory is printed on listing device. Terminal prints: DSC>.
8.	Type BY[CR].	Terminal prints: ***TURN [WP] AND [INIT] SWITCHES OFF ***. Monitor prompt is given.
9.	TURN WP and INIT switches OFF.	

2.7 TERMINAL SOURCE EDITOR - TSE980

TSE980 is a terminal source editor that has been provided for the movement of source format files. It is executed whenever the EDI command is used. Refer to the Model 980 Computer, Basic System Use and Operation Manual, or the SPC9800 Software User's Manual for a complete description of its operation.

2.8 GENERALIZED OBJECT COPY PROGRAM - OBJCPY

OBJCPY is a general purpose program for moving and concatenating object format files. It is executed whenever the COP command is used. OBJCPY is a modification of Texas Instruments program CPYOBJ and when used with the EXE command, executes identically to CPYOBJ. Refer to the Model 980 Computer, Basic System Use and Operation Manual for a complete description of its operation.

2.9 ASSEMBLER - SAPFGL

SAPFGL is a program used to assemble source files into executable object files. It is executed whenever the ASM command is used.

2.10 OVERLAY LINK EDITOR

LINKG concatenates and links several object files into a single executable object file. It is executed whenever the LIN command is used.

2.11 SYSTEM ERROR MESSAGES

Error messages output by MONXxx are divided into general system errors and abort errors.

Improper use of a logical unit or physical device causes system errors. When such an error occurs, an error message in the following format is output:

```
LUN xxxx ERR yyyy AT zzzz LL ssss
```

```
Where:     xxxx is the logical unit number.
           yyyy is one of the error codes shown in Table 2-3.
           zzzz is the memory location of the request which
                resulted in the error.
           ssss is the contents of the lower limit register.
                or 0 is the error occurred in the monitor.
```

The user should correct the cause of the error and execute the program again.

TABLE 2-3
GENERAL SYSTEM ERROR CODES

ERROR CODE	DESCRIPTION
0000	Undefined operation code
0001	File not opened
0002	File not defined
0003	End of file encountered
0004	Internal file management buffers full
0005	Hardware error
0006	Disk volume full
0007	Illegal operation code
0008	Directory full
0009	Duplicate file
0010	Logical unit not defined
0011	Buffer outside user memory space
0040	EIA interface error

Abort error messages are output on the system logging device on the occurrence of one of the events described in Table 2-4. The message output is of the form:

```
ABORT ERR yyyy AT zzzz U ssss
```

where: yyyy is one of the error codes shown in Table 2-4.
zzzz is the memory location where the event occurred.
ssss is the contents of the lower limit register or 0 if the event occurred in the monitor.

TABLE 2-4
ABORT ERROR CODES

ERROR CODE	DESCRIPTION
0000	Power failure (printed on power restart)
0001	Illegal operation
0002	Privileged instruction violation
0003	Memory protect violation

BLANK

SECTION 3

MODEL CONCEPTS AND ORGANIZATION

3.1 OVERVIEW OF MODEL ORGANIZATION

The data base, referred to from now on as the "model", consists of three types of primitives (edges, faces, and lights), and several kinds of abstract objects. These abstract objects reference a single primitive or a collection of the primitives in order to achieve certain important effects, such as real-time hidden surface elimination and dynamic movement within the model.

Model primitives and organization are discussed as follows:

	SECTION	FIGURE
Light Strings	3.2	3-1
Edges	3.3	3-2
Faces	3.4	3-3
Module Contents and Structure	3.5	3-4

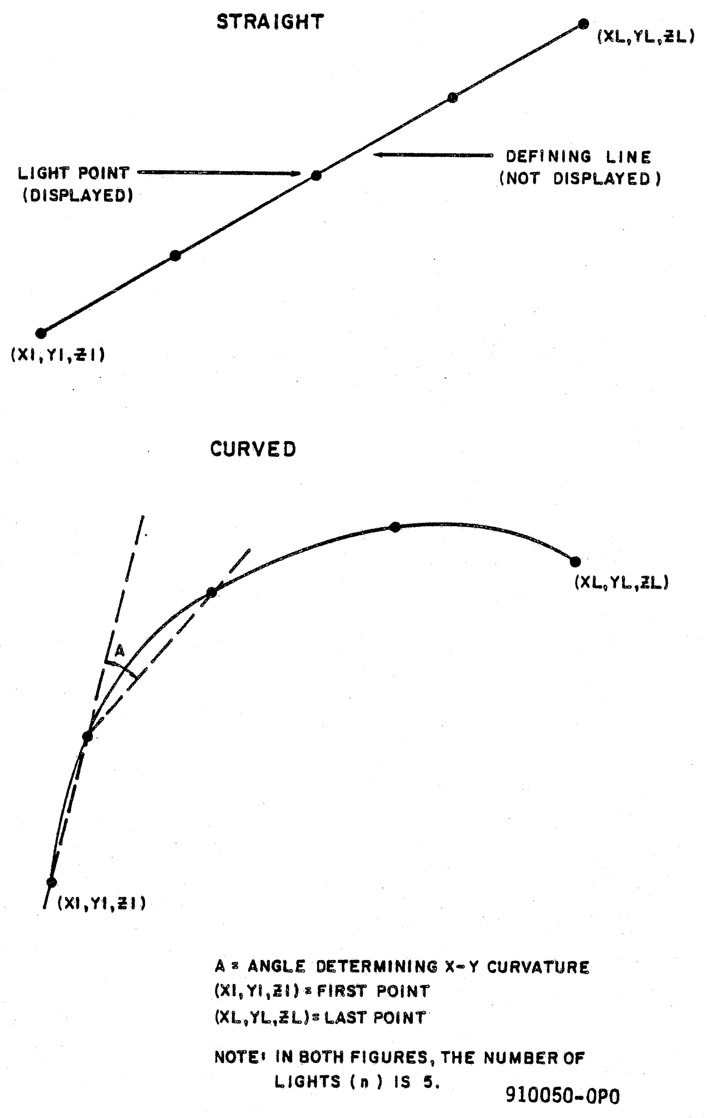


FIGURE 3-1
GENERIC LIGHT STRING

3.2 LIGHT STRINGS

Lights are not modeled individually but as collections which have the same color and lie on the same straight or curved line segment. Straight lines are defined by their endpoints. Curved lines are defined by their endpoints and by an angle determining the curvature in the X-Y plane. The number of lights determines their distribution along the segment between the endpoints.

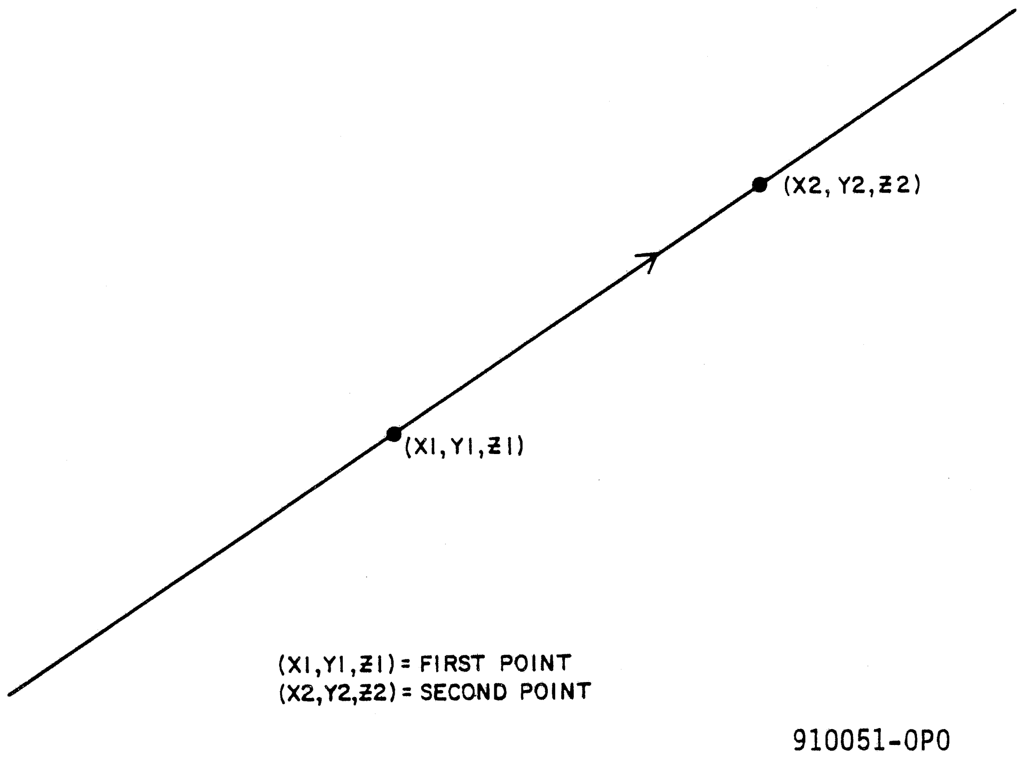


FIGURE 3-2
GENERIC EDGE

3.3 EDGES

An edge is an infinite line with a direction. It is defined by specifying any two points through which the line passes. The direction is along the line from the first point towards the second. Edges are used to define faces.

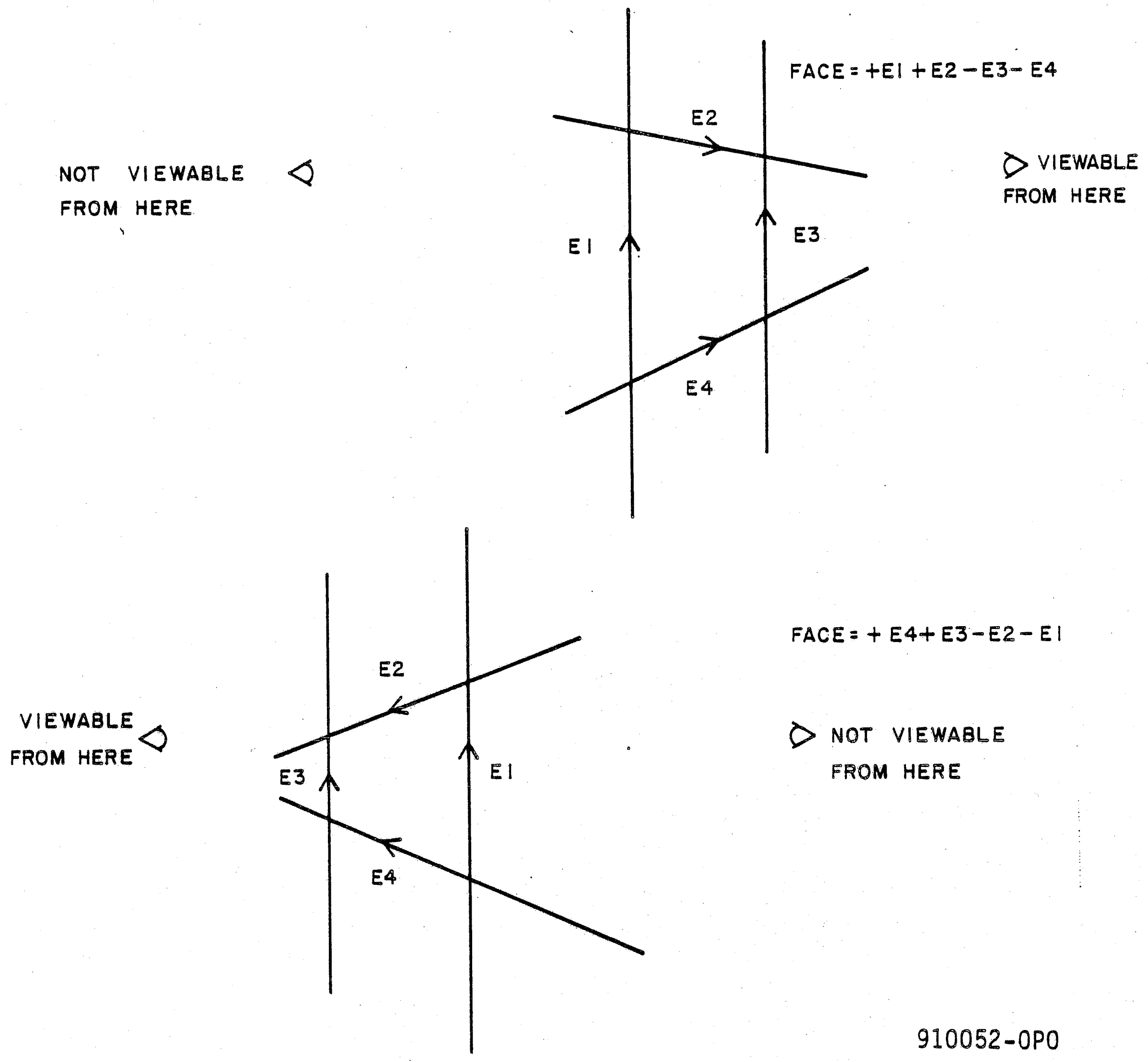
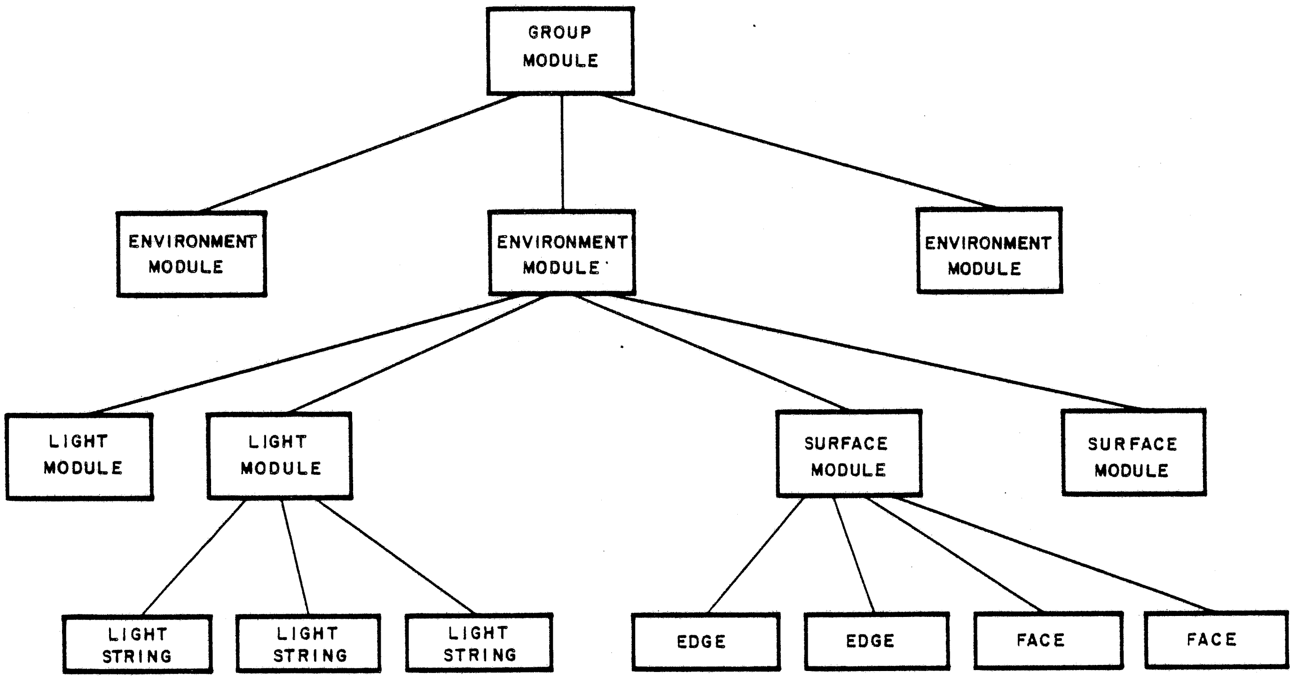


FIGURE 3-3
FACE DEFINITION

3.4 FACES

Faces are the displayable surfaces for SP1/T. They are planar and single sided. The boundary of a face is determined by up to four edges. The direction from which a face is viewable is determined by the direction or sense (+ or -) of its constituting edges and the left hand rule. Specifically, the order in which the edges are entered into the face definition and the sense on those edges (either + or - depending on the direction of the edge's definition) must be clockwise from the viewing position.



910053-0P0

FIGURE 3-4
MODEL ORGANIZATION

3.5 MODULE CONTENTS AND STRUCTURE

A complete SP1/T model is a pyramidic structure. At its base are the primitives consisting of light strings, edges, and faces. The primitives are defined and stored in modules. There is a light module for light strings and a surface module for edges and faces. Modules contain not only primitives but other objects as well. For example, the surface module contains two command lists: one for edges and one for faces. A command list is a list of instructions which tells the geometric processor in what order and in what way a group of primitives are to be displayed. The command list is linked to the primitives by entry pointers which delineate the previously mentioned groups. Light modules possess entry pointers, but no command list.

The environment module contains the information necessary to associate a collection of light and surface modules into a displayable model. Besides the list of modules which compose it, the environment module possesses a command list to direct the integration of light and surface data, a priority tree for hidden surface management, and a vector for sun shading.

At the peak of the pyramid is the group module. It consists of a list of environments and the information necessary to enable a smooth change of a model in real-time. The group module also contains texture-related data that applies to all environments of the model. Figure 3-4 shows the organization of model data.

Each of the various types of modules is defined and manipulated by a corresponding section of the modeling program. The user will select which type of module is to be manipulated, then proceed to work on that module. This approach provides a structured organization to both the model and the commands needed to manipulate the data.

3.6 CONVERTING SP1 MODELS TO SP1/TEXTURE

3.6.1 INTRODUCTION

Texture increases scene content and adds realism to the scene for higher quality pilot training.

Without texture, the faces are one solid color, flat and featureless. With a texture pattern across the faces, they look more like something out of the real world (fields, treetops, water, concrete, etc.) This gives the pilot motion and height cues as well as rate of closure and turn cues.

Texturing is done without adding any polygons to the data base. The texture is created by modulating the intensity of the face color across the face according to the pattern in a texture map.

3.6.2 SURFACE MODULE TEXTURE

The surface module has been expanded to 17 sectors to contain texture information for each face. The modeler converts a standard SP1 surface module to SP1/T by reading in the SP1 surface module and writing the module using a different surface module filename. The conversion assures only that any baseline-compatible SP1 surface module is compatible with the baseline SP1/T realtime program. the conversion process does not add texture values to the model; that means that the model does not have any textured faces.

3.6.3 ENVIRONMENT MODULE TEXTURE

The environment module has been changed to allow the insertion of a new command list instruction, TXTR. The texture command is inserted into the environment command list after the PVEC, SCHN, and GCS (MTX) commands.

The SUN command has been disabled. If a SUN command was in the SP1 environment module, it appears in the SP1/T environment module as the command NOP.

To update from SP1 to SP1/T, the modeler reads the SP1 environment, inserts the TXTR instruction into the command list, deletes the NOP instruction if it appears, and writes the SP1/T environment. The same file name can be used because the environment's module size has not changed.

3.6.4 GROUP MODULE TEXTURE

The group module has been expanded to ten sectors so that it can contain both group data and texture data items. These items are lambdas, movement vectors, heights, cloud data and texture map names.

To update from SP1 to SP1/T, the modeler reads the SP1 group, defines the texture data items, and writes the module using a new group module filename.

BLANK

SECTION 4

PROGRAM CONVENTIONS

4.1 INTRODUCTION

This section discusses the prompts, replies, and program execution for the SP1/Texture Modeling System. It contains the following sections:

4.2 PROMPTS AND REPLIES

The dialogue between the modeling program and the user is basically that of the user responding to a prompt message from the program. User responses to these prompts are such things as single alphabetic characters, short text strings, names, integer numbers, or real or fractional numbers. Tables 4-1 through 4-5 provide a summary of all prompts and the type of response that is expected. Each user reply to a prompt is terminated by either a tab character (control-I) or a carriage return. In some instances, a tab or carriage return alone is an appropriate response. When such inputs are legal, entering a tab in response to a prompt will result in the program using the existing definition, echoing that definition as the response to the prompt, and continuing. A carriage return will perform the same function without echoing the existing definition. If an item (edge, face, node, etc.) is edited and all following item information is currently correct, it can all be skipped at once by typing the Escape or Alt-mode key. This facility is particularly useful when entering many similar definitions and/or making minor changes to existing definitions.

Names that are used by the system are intended to help the user in associating and identifying various data elements. Names that are used in each type of module, the interpretation of that name, and any special characteristics are as follows:

Light Module

P1 thru P8	Entry points names.
R1 thru R110	Red light string names.
A1 thru A110	Amber light string names.
O1 thru O110	Orange light string names.
W1 thru W110	White light string names.
G1 thru G110	Green light string names.

Since light strings must be maintained in color order and sequential locations in the light module, specification of a string name that is higher than currently exists will result in the next available string name being used. For example, inserting R110 will always result in the definition of the next red string. If the user specifies an existing string name when doing an insert, it will be inserted as requested and all string names for that color above that number will be incremented by one.

Surface Module

P1 thru P8	Entry point names. There are eight entry points each for edges and faces.
E1 thru E64	Edge names. Since edge definitions must be maintained in sequential locations in the surface module, specification of an edge name that is higher than currently exists will result in the next available edge name being used. For example, inserting E64 will always result in the definition of the next edge.
F1 thru F64	Face names. Since face definitions must be maintained in sequential locations in the surface module, specification of a face name that is higher than currently exists will result in the next available face name being used. For example, inserting F64 will always result in the definition of the next face.
N1 thru N5	Priority tree node names.
I1 thru I15	Command list instruction names. Since command list instructions must be maintained in sequential locations in the surface module, specifications of an instruction name that is higher than currently exists will result in the next available instruction being used. For example, inserting I16 will always result in the definition of the next instruction. There are 15 command list names each for edges and faces.

Environment Module

M1 thru M11	Light and Surface module names. M1 - M4 are switchable surface modules. M5 - M7 are non-switchable surface modules. M8 - M11 are light modules.
R1 thru R4	Runway offset names.
P1 thru P3	Converging DCS names.
L1 thru L3	Routed DCS names.
G1 thru G4	LAT/LON reference names.

W1 thru W11	Way-Segment names. If any Routed DCS names are defined, the Way-Segments pointed to by those Routed DCS definitions must also be defined.
N1 thru N6	Priority tree node names. Priority processing begins with node N1.
I1 thru I64	Command list instruction names. Since command list instructions must be maintained in sequential locations in the environment module, specification of an instruction name that is higher than currently exists will result in the next available instruction being used. For example, inserting I64 will always result in the definition of the next instruction.
C1 thru C8	Coordinate system names.
T1 thru T7	Transformation names.
S1 thru S4	Submodel option names S1 references one of the four switchable submodels which is selected by the real-time program. S2-S4 correspond directly to the last three (non-switchable) surface modules (see Section 7.1.8.4).

<u>Group Module</u>

G0 thru G255	Group number names. Group numbers are used to associate the model select number received from the host computer with the actual data of the diskette.
M1 thru M256	Environment module names. These module names are used to order the environment module references used for data base management.
D1	Texture cloud managements delta.
L1	Lambdas (texture map size).
H1	Texture plane heights.
T1 thru T4	Texture map file names.
X1	Dynamic texture vector, X component.
Y1	Dynamic texture vector, y component.

TABLE 4-1
LIGHT MODULE ENTRY PARAMETERS

PROMPT	DATA TYPE	RANGE MIN	RANGE MAX	UNITS	NOTES
LIT>>	CHAR				R,Z,X,E,P,I,C,K,K,S,W, B,D,V,F,A,T,N,H,O,M
VER>	Y/N				
FILENAME=	FILE SPEC.				CS0:,CS1:,FT0-FT3:X. FB0-FB3:X,FD0-FD3:X Where X represents a one to six character file name.
ENTRY=	NAME	P1	P8		
STR #= STR #1= STR #2=	NAME	R1 A1 O1 W1 G1	R110 A110 O110 W110 G110		
FLS,ROT,VLA, SFL,NML	ALPHABET				F,R,V,S,N,X
ONT=	INTEGER	0	255	FRAMES	
PER=	INTEGER	ON TIME VALUE	32,767 32,767	FRAMES	Flashing lights Strobe lights
PHA=	INTEGER	0	255	FRAMES	
OFT=	INTEGER	0	255	FRAMES	
UP ANG=	REAL	0	360	DEGREES	
HDG=	REAL	0	360	DEGREES	
#MASTERS=	INTEGER	2	3		
#REPEATS=	INTEGER	0	7		
ELEV=	REAL	0	360	DEGREES	
#LTS=	INTEGER	1	255		
INT=	REAL	0	1.9414		
SAW=	INTEGER	0 1 10 20 30	 13 23 33		Omnidirectional Curved Directional
SWT#=	INTEGER	0	32		
FLG>	YES/NO				
FFB= *B= *TRAN= *SP= PRI= ID= *DFO=	ZERO/ONE				
CHAN=	INTEGER	0	8		

TABLE 4-1 (CONTINUED)
LIGHT MODULE ENTRY PARAMETERS

X= Y= Z=	REAL	-2^{21}	$2^{21}-1$	FEET/ METERS	
DELTA=	REAL	0	SNMI	FEET/m	
ANG=	REAL	0	360	DEGREES	
FL/SP>	ALPHABET				F, S
SPAN=	REAL	0	$2^{21}-1$	FEET/m	
POS CROSS>	ALPHABET				Any printing character X= Exit
TEXT=	ALPHABET				String of up to 29 characters in length.
TOP STRING=	NAME				X to exit
BOTTOM STRING=	NAME				X to exit
VLA#= A: B: C: D:	INTEGER REAL REAL REAL REAL	0	1		
DCS:	Y/N				
TYPE=	INTEGER	0	4		Type of VLA
#INSTANCES=	INTEGER				
CHANGE/DELETE?:	C/D				

TABLE 4-2
SURFACE MODULE ENTRY PARAMETERS

PROMPT	DATA TYPE	RANGE		UNITS	NOTES
		MIN	MAX		
SUR>>	ALPHABET				R,Z,X,E,P,M,S,I,C,K,W,Q,L,B,D,V,F,N,A,T,H,Y,U
FILENAME=	FILE SPEC.				CSO:,CS1:,FBO-FB3:X, FTO-FT3:X,FDO-FD3:X. Where X Represents a one to six character file name.
VER>	YES/NO				
ENTRY=	NAME	P1	P8		
ITEM #= ITEM #1= ITEM #2=	NAME	E1 F1 I1 N1	E64 F64 I16 N5		
MS=	ZERO/ONE				
X= Y= Z=	REAL	-2^{21}	$2^{21}-1$	FT(M)	
MODEL STRIPES=	INTEGER	0	15		
REMAINING STRIPES=	INTEGER	0	255		
INT= PRI=	INTEGER INTEGER	0 1	63 16		
FLG>	YES/NO				
LLL? GND? 3D? TXT?	YES/NO				
*TRAN?	INTEGER	0	1		
CHAN=	INTEGER	0	8		
E1= E2= E3= E4=	NAME	E1	E64		Edge names may be preceded with a minus sign.
PLANE TEST>	YES/NO				
NEW PLANE>	YES/NO				
TSON= FSON=	NAME	N1 F1	N5 F50		
LAST FACE=	NAME	F1	F50		
EDG,FAC>	ALPHABET				E,F

TABLE 4-2 (CONTINUED)
SURFACE MODULE ENTRY PARAMETERS

INST=	TEXT				PVEC,SCHN,MTX,MMIR, TVEC,TADD,TMIR,EDGE, FACE,LINK,GCS,TXTR,GCS
MTX=	NAME	T1	T7		
OPT=	NAME	01	03		
SPAN=	REAL	0	$2^{21}-1$	FT(M)	
POS CROSS>	ALPHABET				Any printing character; X=EXIT
TEXT:	ALPHABET				String of up to 29 characters in length.
CONTRAST RATIO	INTEGER	1	3		1=Low, 2=Med, 3=High
TEXTURE MAP	INTEGER	1	4		
LAMBDA ADJUSTER=1/N	INTEGER				1, 2, 4, or 8
TEXTURE CLOUD MANAGEMENT?	ALPHABET				Y or N

TABLE 4-3 (PART 1)
ENVIRONMENT MODULE ENTRY PARAMETERS

PROMPT	DATA TYPE	RANGE		UNITS	NOTES
		MIN	MAX		
ENV>>	ALPHABET				E,R,Z,H,Q,D,I,C,K, L,W
FILENAME=	FILE SPEC.				CSOX,CSX:,FDO-FD3:X, FBO-FB3:X,FTO-FT3:X Where X represents a one to six character file name.
VER>	YES/NO				
HDG= PITCH= ROLL=	REAL	0	360	DEGREES	
ITEM #= ITEM #1= ITEM #2=	NAME	M1 L1 R1 P1 I1 W1 N1 G1	M11 L3 R4 P3 I64 W11 N6 G4		
LIT,SUR>	ALPHABET		N6		L,S
X= Y= Z=	REAL	-2^{21}	$2^{21}-1$	FEET/ METERS	
SPEED	INTEGER	1	128		
MOTION TIME IMPACT TIME VEER TIME	INTEGER	-2^{15}	$2^{15}-1$		
INST=	TEXT				PVEC,GCS,SCHN,LCHN,MTX, TVEC,STAR,SFOG,DRAW,TXTR STRT,LINK,FOG,LITE,END
MTX=	NAME	T1	T7		
OPT=	NAME	O1	O4		
MOD=	NAME	M1	M11		
ENTRY=	NAME	P1	P8		
EDG,FAC>	ALPHABET				E,F
PLANE TEST>	YES/NO				
NEW PLANE>	YES/NO				
CS=	NAME	C1	C8		
TSON= FSON=	NAME	N1 S1	N5 S4		
NODE=	NAME	N1	N6		
TEXT=	ALPHABET				String of up to 29 characters in length.

TABLE 4-3 (PART 2)
ENVIRONMENT MODULE ENTRY PARAMETERS

X=	REAL				
Y=	REAL				
DX=	REAL				
DY=	REAL				
ADDITION=	REAL				
ANGLE=	REAL				
N-OR-S:	N/S				
E-OR-W:	E/W				
DEG=	INTEGER	0	90		
MIN=	INTEGER	0	59		
SEC=	REAL	0	59.9999		
COORDINATE SYS=	NAME	C3	C7		
S1> THRU S9>	NAME	W1	W11		X to exit
RECYCLE?	YES/NO				
END POINT SAME AS START?	YES/NO				
SPEED	INTEGER				
MOTION TIME	REAL	0	1090.0		
CHANGE POSITION?	YES/NO				
CHANGE ATTITUDE?	YES/NO				
C3> THRU C6>	NAME	C1	C6		X to exit

TABLE 4-4
GROUP MODULE ENTRY PARAMETERS

PROMPT	DATA TYPE	RANGE MIN	RANGE MAX	UNITS	NOTES
GRP>>	ALPHABET				E,R,X,Z,W,I,K,L,C, H,Q,Y
FILENAME=	FILE SPEC.				CSO:X,CS1:X,FBO-FB3:X, FTO-FT3:X,FDO-FD3:X Where X represents a one to six character filename.
VER>	YES/NO				
GRP #= GRP #1= GRP #2=	NAME	GO	G255		
MODULE #= MODULE #1= MODULE #2=	NAME	M1	M256		
ITEM #= ITEM #1= ITEM #2=	NAME	GO L1 X1 Y1 H1 D1 T1	G255 L1 X1 Y1 H1 D1 T4		
GND/SKY?	ALPHABET				G,S
X VECTOR = Y VECTOR =	REAL	-128	+127.99	FEET/ FRAME	
DELTA =	REAL	-128	+127.99	FEET	Texture cloud management
LAMBDA =	REAL	64	65,536	FEET	Map size
TEXTURE MAP FILE NAME =	ALPHABET				One to six character filename.
HEIGHT=	REAL	-32768	32767	FEET	Ground height
HEIGHT=	REAL	0	32767	FEET	Sky height
SEARCH #=	INTEGER	0	256		Module number

TABLE 4-5
MERGE ENTRY PARAMETERS

PROMPT	DATA TYPE	RANGE MIN	RANGE MAX	UNITS	NOTES
MER>>	ALBHABET				M,X,Z,W
FILENAME=	FILE SPEC.				CSO:,CS1:,FDO:X,FD1:X, Where X represents a one to six character file name.

4.3 PROGRAM EXECUTION

To load and execute the model build program put the program disk in FDO and type:

```
@1TSOx
```

The program description will be the output:

```
SP1 Texture Build Program Overlay Controller.

SP1-TEXTURE BUILD and DSC ADVANTAGE
    Support -
        Tektronics - yes
        Tablet     - XXX
```

The user must then provide the control and listing devices in response to the prompts: Control device - and Listing device -. The user must then specify which program he wishes to use. The prompt provided here is LIT,SUR,ENV,GRP,MER,DSC>>. The valid responses are 'L', 'S', 'E', 'G', 'M', 'D', 'X', and a carriage return only. [By responding with only a carriage return or 'X' the user is taken back to the Listing device - prompt. By responding again with a carriage return the user is provided the Control device - prompt. By responding with another carriage return the user is returned to the monitor.] The disk drive will then be engaged to load and execute the specified program.

NOTE

At this time the Disk Drive numbers should be changed or Diskette moved so that the command disk ends up in Drive #1
Example: FDØ = Model Disk, FD1: = Build Disk
otherwise a disk error will result: "command disk not in Drive #1"

NOTE

Cursors are not available when no TEK is used. Also if the TEKTRONICS is connected to the system it must be powered up because of hardware interrupt lines; otherwise, the system will hang up during the initialization sequence.

The program will be loaded and will start execution.

If the system used includes tablet support, the first question asked is TABLET USE>> to which the user should reply Y if he intends to use a tablet and N if he does not.

The next question asked is UNITS= to which the user would reply with E for English, M for Metric, or X to exit the program. After he specifies the units, the prompt PAGE LENGTH > will be output to request the length of a page of output. Default is 52 lines. Five is the minimum page length specification. After entering this information, the prompt for that module will be output. Refer to the appropriate section for further instruction.

BLANK

SECTION 5

LIGHT MODULE FUNCTIONS

5.1 INTRODUCTION

Each light module contains all the information required to define a set of light strings. Each module also contains entry points that group the light strings.

Light modules are defined and manipulated with the following functions. To select a function, type the appropriate character in response to the prompt LIT>>.

A	Angle rotation	M	Map of VLA
B	Blank screen (option)	N	Print statistics
C	Change light string	P	Define entry point
D	Display screen (option)	Q	List entire module
E	Edit light module	R	Read light module
F	Find light string (option)	S	Select first/second or first/last entry mode
G	Generalized visual Landing Aid (VLA)	T	Translation
H	Insert module description	V	Set view position (option)
I	Insert light string	W	Write module
K	Kill light string	X	Return to module
L	List light string	Z	Zero light module

Each of these functions is discussed separately below.

5.1.1 EDIT LIGHT MODULE

<u>LIT>></u> E

With the edit module function the user can declare the current module in memory to be active without destroying the data. By using this function after a program abort or a module write function, the user can gain access to the light module data. It is assumed by the system that the user is enabling access to valid light module data.

5.1.2 READ LIGHT MODULE

<u>LIT>></u> R

Previously generated light modules are read into the modeling system using the read function. In response to the selection of the read function, the system will prompt the user by typing:

<u>FILE NAME=</u>

The user reply depends on the type of device that the file is to be read from, as follows:

CS1: or CS2:	-Cassette tape drive (foreground mode only)
FD0: or FD1:	-Floppy disk drive (single-sided)
FT0: or FT1:	-Floppy disk drive (double-sided, top)
FB0: or FB1:	-Floppy disk drive (double-sided, bottom)

The device name is to be followed immediately by a file name, from one to six characters in length. If the output is from cassette tape, the user should have rewound the tape. If a file name is entered and the device name is omitted, FD0: is assumed. If an active module currently exists, the system will ask for verification of the read command by displaying <u>VER></u>. The user responds with either "Y" to complete the read command or "N" to abort it.

5.1.3 ZERO LIGHT MODULE

<u>LIT>></u> Z

The zero function, which is the first command executed when defining a new light module, clears all string definitions and resets all of the pointer structures for the light module. If an active module currently exists, the system requests verification of the zero command by displaying <u>VER></u>. The user responds with either "Y" to complete the zero command or"N" to abort it.

5.1.4 EXIT

LIT>>   X

After the user has completed light module editing, he selects the exit function. When the user selects this function the system returns to the editing mode for another type of module. If an active module still exists when the exit function is selected, the system responds with the message:

ACT MOD
VER>

the user must reply with either a "Y" to exit or a "N" to abort the command.

5.1.5 INSERT MODULE DESCRIPTION

LIT>>   H

When the user selects the H function, the system returns with the prompt:

TEXT:

The user then enters or changes the text which is printed at the top of every complete module listing.

5.1.6 ENTRY POINT DEFINITION

LIT>>   P

Entry points are used to associate groups of light strings so they can be referenced in the command list; the command list directs their processing in the hardware. Each light module has a minimum of one and a maximum of eight defined entry points. After the user selects the entry point definition function, the user receives the prompt:

ENTRY=

He responds by entering the name of the entry point to be defined, P1 through P8. A tab or carriage return terminates the entry. If a legal entry point was entered, the prompt STR #1= appears. After the user enters a legal string name, the prompt STR #2=. The user enters a second string number.

Entering the second legal light string defines the entry point; this entry point is composed of a group of strings from the first through the second string names. If the user wishes to delete an entry point definition, he defines the entry point with the first string name higher than the second string name.

ENTRY= P3 STR #1= R20 STR #2= R1

would cause entry point P3 to be deleted.

Since entry points are entered independently and since they define a contiguous group of light strings from the first through the second string, the user should be careful not to overlap entry point specifications. This causes incorrect light string groupings. After entry points have been set and a light string is added, the user must redefine the entry point; entry points do not change when adding more light strings.

NOTE:

If light strings containing the entry point or end-of-list are deleted then the entry point must be re-inserted.

5.1.7 INSERT LIGHT STRING

LIT>> I

With the insert light string function the user can define new database light strings in the active light module. After the user selects the insert function, the prompt to select the type of light string appears. They are as follows:

FLS,ROT,VLA,SFL,NML>>

User responses to this prompt and the type of light string selected are:

F	Flashing light string
R	Rotating light string
V	VLA (Visual Landing Aid)
S	Strobe light string (sequential flashing lights)
N	Normal light string
X	Return to command selection

After the user selects the type of light string to be inserted, the prompt STR #1 appears. The user specifies the first letter of each color desired (i.e., R=red, O=orange, A-amber, W=white, and G=green) followed by a number from 1 to 110 (i.e., R10). The user enters the light string number followed by a tab or carriage return.

If the user responds to the STR #1= prompt with a tab for all strings after the first one, then the next sequential light string of the same color is entered.

If the user specifies the string number of an existing string, the new string is inserted as designated. All string numbers of that color following the new string are incremented by one. After the light string has been selected, any special characteristics that the light string requires are requested. Then the actual light string characteristics are requested. Special characteristics are discussed by light type. The basic string light characteristics are discussed under the normal light string specification.

5.1.7.1 FLASHING LIGHT STRINGS

```
FLS,ROT,VLA,SFL,NML>>     F
```

Flashing light strings require that the user specify flash pattern characteristics (i.e., light on-time, period, and phase). The prompt ONT= requests on time; the user enters the integer number of frames that the light string is on.

The prompt PER= requests the flash period; it is the flash period in frames. If the flash period is less than the on-time, it is set equal to the on-time. In other words, off-time plus on-time equals total period.

The prompt PHA= appears; it is phase delay in frames for this light string for the beginning of the first frame only. Figure 5-1 shows two light strings set up to flash in alternating sequence. After the flash pattern characteristics have been specified, the light characteristics discussed under normal light strings are entered.

5.1.7.2 ROTATING LIGHT STRING

```
FLS,ROT,VLA,SFL,NML>>     R
```

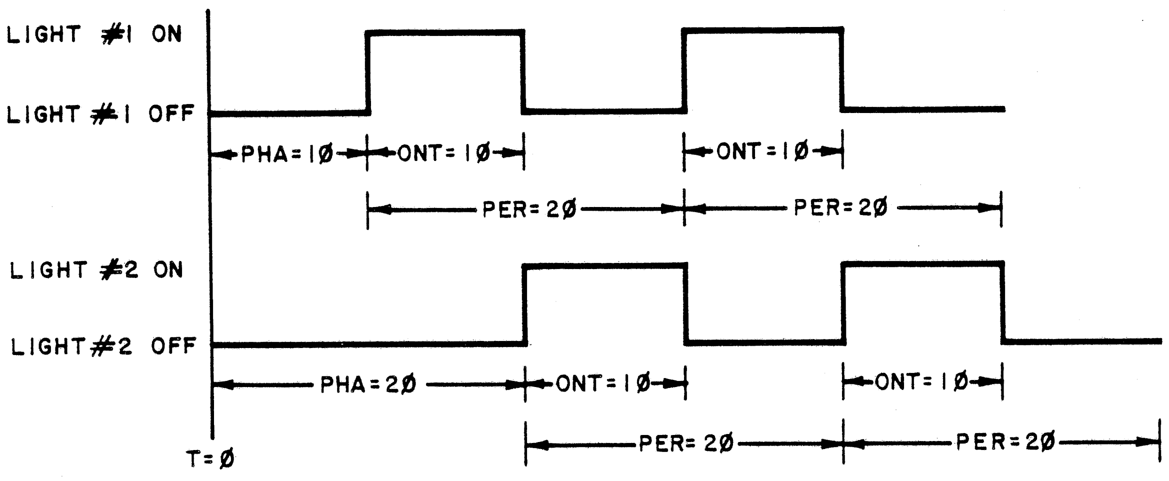
Rotating light strings require the user to specify update characteristics. The prompt UP ANG= requests the per frame update angle; the prompt HDG= requests the initial heading angle.

In other words, two light strings with the same update angle and headings of Ø and 18Ø respectively would rotate opposite each other. When one is off the other is on and vice versa. The update angle refers to how many degrees of rotation this light will rotate in 1/30 of a second or one frame. For example to rotate a light at 6 RPM use the following formula:

```
  3Ø Frames per second          36Ø = 1 RPM
 X6Ø Seconds per minute          X6
18ØØ Frames per minute         216Ø = Degrees for 6 rotation

           1.2  = Update Angle (UP ANG=)
18ØØ   |216Ø
```

After the update characteristics have been specified, the light characteristics discussed under normal light strings are entered.



910054-0P0

FIGURE 5-1
FLASHING LIGHT SAMPLES

```
#W1       FLASHING:   ONT=10       PER=20          PHA=10
     # LTS=12   INT=1.0000   SAW=0   HDG=0.0000   SWT #=1
     FFB=1  CHAN= 0   *B= 0  *TRAN?1  SP=1  PRI=0  IO= 0  *DFO= 0
     START:     X=   10400.0000     Y=  -1725.0000    Z=588.0000
     END:       X=   10400.0000     Y=  -1425.0000    Z=588.0000

#W2       FLASHING:   ONT=10       PER=20          PHA=20
     # LTS=12   INT=1.0000   SAW=0   HDG=0.0000   SWT #=1
     FFB=1  CHAN= 0   *B= 0  *TRAN?1  SP=1  PRI=0  IO= 0  *DFO= 0
     START:     X=   10400.0000     Y=  -525.0000    Z= -488.0000
     END:       X=   10400.0000     Y=  -225.0000    Z= -488.0000
```

FIGURE 5-2
FLASHING LIGHTS

5.1.7.3 VLA LIGHT STRINGS

FLS,ROT,VLA,SFL,NML>>   V

Except for two changes, defining a VLA light string is the same as defining a normal light string. The VLA light string does not require an intensity or switch number. These light strings should be directional.

5.1.7.4 STROBE LIGHT STRINGS

FLS,ROT,VLA,SFL,NML>>   S

Strobe lights can be one of two types: regular strobes or moving traffic strobes. A regular strobe is drawn several times in one frame; then no lights from the string are drawn for several frames; another light is drawn in another frame but offset from the first. The cycle repeats until a specified number of strings is drawn. The traffic strobe draws a string of lights in one frame. In the next frame, the string is drawn again; this time the string is offset from the first. The string appears to be moving slowly in a row because of the way these frames are drawn.

The # INSTANCES is the number of lights that appear in one cycle for a regular strobe or it is the number of moves that a traffic strobe makes in one cycle. The DELTA: X=, Y=, Z= for a regular traffic strobe is the offset for each light drawn. It is the per-frame movement that each light makes for a traffic strobe. The #LTS= is the number of times that a regular strobe is drawn in one frame; or it is the number of lights that are drawn in one frame for a traffic strobe.

With strobe light strings, the user must specify flash pattern characteristics, for example, off-time between sequential lights, period, and phase. The prompt OFT= requests the off-time; the user responds with the integer number of frames between a light going off and the next sequential light going on. With a traffic strobe, the off-time is Ø.

The prompt PER= requests the flash period; it is the flash period in frames for the entire strobe string. For either strobe string the period must be greater than or equal to the value of the minimum (number of instances) x (off-time + 1). If this is not true, this minimum is calculated and it is used instead of the entered period.

The prompt PHA= requests the flash phase; it is the phase delay in frames for this particular light string for the first frame only. After the flash pattern characteristics have been specified, the light characteristics discussed under normal light strings are entered. With a regular strobe the start and end x, y, z's are the same (i.e., xstart = xend, ystart = yend, etc.). With a traffic strobe the start and end refer to the start and end of the string of lights to be drawn in the first frame.

5.1.7.5 NORMAL LIGHT STRINGS

FLS,ROT,VLA,SFL,NML>> N

Normal light string characteristics define the basic character of the lights being used in the model. The first parameter requested for input is the number of lights in the string.

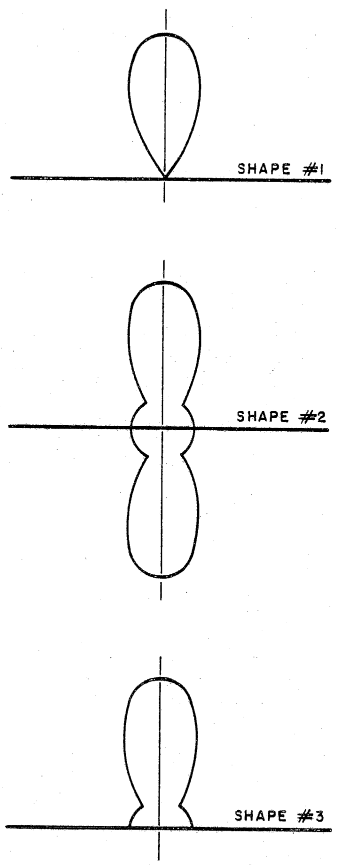
The prompt #LITS= requests the number of lights in the string; the user responds with a number between one and 255 to indicate the light count.

The prompt INT= requests the light string intensity; the user responds with a valid lights string intensity--a number between Ø and 1.9414. Maximum screen intensity occurs at model intensity 1.Ø. With values greater than 1.Ø the fog penetration capability of the lights is increased but the perceived intensity is not increased.

The prompt SAW= requests the shape and directional characteristics of the light strings. The user responds with an integer value which describes the shape, width and directional characteristics of the light string. These responses and shapes are shown in Figure 5-3.

If the user selects a curved string, the message ANG= is displayed. The user specifies the rotation angle between successive lights in the string.

If the user selects a directional string, the message HDG= is displayed. The user responds with the directional heading, in degrees, for the directional light pattern.



910056-0P0

RESPONSE TO SAW=	DIRECTIONAL CHARACTER	SHAPE	WIDTH
0	omnidirectional	-	-
1	curved	-	-
10	directional	1	12°
11	directional	1	24°
12	directional	1	48°
13	directional	1	124°
20	directional	2	12°
21	directional	2	24°
22	directional	2	48°
23	directional	2	124°
30	directional	3	12°
31	directional	3	24°
32	directional	3	48°
33	directional	3	124°

FIGURE 5-3
LIGHT STRING DIRECTION, SHAPE, AND WIDTH CHARACTERISTICS

After the user specifies the shape and the width of the light string, the prompt SWT #= appears; the user enters a number between 0 and 32 which defines the software switch that turns the light string on and off. When Ø is entered, the light string is assumed to be always turned on. See Table 5-1 for a list of switches.

TABLE 5-1
MODEL BUILDING CONTROL-TABLE LIGHT SWITCH NUMBERS
GENERIC SAMPLE

MODEL BUILDING CONTROL	LIGHT SWITCH NUMBERS		
APPROACH ------ 1	----------- 9	----------- 17	SPARE ---- 25
EDGE ---------- 2	----------- 10	----------- 18	SPARE ---- 26
CENTERLINE ---- 3	----------- 11	----------- 19	SPARE ---- 27
TAXI ---------- 4	SPARE----- 12	SPARE------ 20	ENV ------ 28
VLA ----------- 5	----------- 13	----------- 21	SPARE ---- 29
TDZ ----------- 6	----------- 14	----------- 22	LITN10 --- 30
STROBE -------- 7	----------- 15	----------- 23	LITN10 --- 31
REILS --------- 8	----------- 16	----------- 24	LITN10 --- 32

A set of flags used to control special processing functions is associated with all light string functions. The prompt FLG> asks whether the current setting of the flags is to be changed; the user responds with a Y or a N. If the user responds with a Y, all of the flags described in Table 5-2 are requested. If the user is inserting successive light strings, the values set in these flags remain set.

Next, the light string position is defined as either a first and last point or as a first point and spacing. If the user has selected the first/last point format, the prompts X=, Y= and Z= are typed for both the first and last points of the light string.

If the first/spacing point format is being used, the prompts X=, Y=, Z=, will be typed for both the first and second points of the light string. The system then computes and types out the last point coordinate of the light string.

When a curved string is being defined, the system automatically selects the first point/spacing format for that particular light string, but it does not change the format for other light strings.

NOTE:

If you have selected only one light point under the prompt # LITS= then the second X=, Y=, Z= will not be asked.

TABLE 5-2
LIGHT STRING FLAGS

FLAG PROMPT	FLAG SETTING
FFB=	0 = apply fog function A to the light string (default) 1 = apply fog function B to the light string

NOTE:

Fog function A is more commonly used for city lights, while fog function B provides greater attenuation for mid-range light strings, such as those used in the airport area.

FLAG PROMPT	FLAG SETTING
CHAN=	0 = light string to be processed by all channels (default) 1-8 = light string to be processed only by the specified channel 1 = apply fog function B to the light string
*B=	0 = apply fog function to this light string (default) 1 = do not apply fog function to this light string
*TRAN	0 = translate this string as required during hardware processing (default) 1 = do not translate this string during hardware processing (such as a wing tip light)
*SP=	0 = do same point test (default) 1 = do not do same point test

NOTE:

The same point test allows only one magnitude of brightness for lights which are defined to be at the same point. For strobe lights, the system will force *SP=1

FLAG PROMPT	FLAG SETTING
PRI=	0 = light string does not take priority over surfaces (default) 1 = light string takes priority over all surfaces
IO=	0 = light string visible in all channels (default) 1 = light string valid for instructor monitor only
*DFO=	0 = defocus lights which are closer to viewer (default) 1 = do not defocus the lights
CHAN=	0 = light string to be processed by all channels (default) 1-8 = light string to be processed only by the specified channel

5.1.8 CHANGE LIGHT STRING

LIT>> C

With the change light string function the user can modify the definition of an existing light string in the active light module. The prompt STR #= requesting the string number to be changed appears on the screen. After a legal string number has been entered, the system retrieves the string definition from the module. It then asks the questions described under the insert string function. Parameters that are to be changed are re-entered when requested. If the parameters remain unchanged, the user responds with a tab. By typing a tab when the string number is requested the user can modify successive light strings after the first one. The user should remember to reset all the flags when returning from another module; otherwise, previous data will be used.

If the user specifies a nonexistent light string, the system will display the error message NO ITEM and return to the prompt LIT>> to allow the user to select another function.

5.1.9 KILL LIGHT STRINGS

LIT>> K

With the kill light string function, the user can delete light strings from the active light module. After the user selects the kill function, the prompt STR #1 appears; the response to this prompt indicates the first string to be deleted. The prompt STR #2= appears next; the response to this prompt indicates the last string to be deleted.

NOTE:

If light strings containing the entry point or end of list are deleted then the entry point must be re-inserted.

5.1.10 LIST LIGHT STRINGS

LIT>> L

With the list light string function the user can list the light strings defined in the light module. The prompts STR #1= and STR #2= request the user to specify the first and last string to be listed.

5.1.11 LIST ENTIRE LIGHT MODULE

<u>LIT>></u> Q

The Q function creates a complete listing of the module including file name, descriptive text, statistics, entry pointers, and string blocks.

5.1.12 SELECT POSITION MODE

<u>LIT>></u> S

With the select position mode function, the user can select either the first point/last point or the first point/second point for defining the light string position. When the prompt <u>SP/LP</u> appears, the user specifies L for first point/last point mode or S for first point/spacing mode.

NOTE:

If you are inserting long complicated curved angles, the system may appear hung up. Wait for a while.

5.1.13 WRITE LIGHT MODULE

<u>LIT>></u> W

After a light module is defined, it is written on cassette tape or floppy disk by invoking the write function. After the user selects the write function, the system will request the output file name by typing:

<u>FILE NAME=</u>

The user reply depends on the type of device that the file is to be written to, as follows:

CS1: or CS2	-Cassette tape drive (foreground mode only)
FD0: or FD1	-Floppy disk drive (single-sided)
FT0: or FT1	-Floppy disk drive (double-sided, top)
FB0: or FB1:	-Floppy disk drive (double-sided, bottom)

The device name is to be followed immediately by a file name, from one to six characters in length. If output is to a cassette tape, the user is expected to have rewound the tape to load point.

If multiple output files are to be put onto cassette tape, they must be written sequentially without rewinding the tape. To add modules to the end of a cassette tape that already contains modules, the user must first read past the existing modules to position the tape. He then outputs a new module to the tape.

When the output is to be to floppy disk, the file name can be specified without the device name. The system will assume FDO: as the output device.

5.1.14 BLANK SCREEN

LIT>> B

The blank screen function erases the screen of the Tektronix display. If the system does not include the Tektronix hardware, the command is ignored.

5.1.15 DISPLAY LIGHT STRINGS

LIT>> D

With the display function the user can specify the light strings in the active light module that are to be displayed. This applies only to systems containing the Tektronix display.

After the function has been selected, the modeling system will request the strings to be displayed by outputting the prompts STR #1= and STR #2=. The user responds to these prompts with the first and last light strings to be displayed. If the system does not contain a Tektronix display, this command is ignored. View position must be set first.

5.1.16 SET VIEW POSITION

LIT>> V

The set view position function defines the viewing window for displaying light strings on the Tektronix display. To define the viewing window, the user specifies an X and Y ground position by responding to the prompts X= and Y=; the user then specifies the ground span to be covered by the long axis of the display by responding to the prompt SPAN=.

If light strings have previously been displayed from the active module, the screen will be erased and the light strings displayed using the new viewing window. For systems without a Tektronix display, this command is ignored.

5.1.17 FIND LIGHT STRING

<u>LIT>></u> F

With the find function, the user can locate strings that are displayed on the string. This applies to systems that have a Tektronix display.

To use the find function, the user positions the crosshairs displayed on the screen over or near a light in question. Any printing character except an X is then entered on the console in response to the prompt <u>POS CROSS></u>. The modeling system then prints out the position of the crosshair and compares the crosshair position to all of the displayed light strings in the active light module.

The modeling system outputs, in listing format, any light strings near the crosshair position. This process may be repeated until the character X is entered in response to <u>POS CROSS></u>. The system then returns to the <u>LIT>></u> prompt. For systems that do not have a Tektronix display this command is ignored.

5.1.18 ANGLE ROTATION FOR LIGHT STRINGS

<u>LIT>></u> A

With the angle rotation function the user can rotate all the defined light strings in the active light module by a specified angle. The prompt <u>ANG=</u> appears; the user enters the rotation angle in degrees. The system prompts the user with the message <u>ANG=</u>. The user enters the rotation angle in degrees.

5.1.19 TRANSLATION FOR LIGHT STRINGS

<u>LIT>></u> T

With the translate function the user can translate all of the defined light strings in the active light module. The user response to the prompts, <u>X=</u>, <u>Y=</u>, and <u>Z=</u> defines the translation vector.

NOTE:

Generally the sequence is best used by rotating the lights first, and then translating them.

5.1.20 PRINT STATISTICS

```
LIT>>   N
```

Statistical information concerning the light strings defined in the active light module is output when the print statistics function is selected.

5.1.21 GENERAL VLA DESCRIPTION

The VLA system is a general vertical lighting system used to model various types of visual approach slope indicators. To provide proper visual information, the lights will change. If you are using a set of reference lights, they will turn on or off.

T-bar VASI's turn on and off; VASI's and PAPI's change color.

VLA's are modeled with several angles, distances, etc. Various other items are not modeled. They are defined in the FLY program.

The following items are modeled:

1. the number of bars - the reference lights
2. the number of repeats - lights which are the same as the bar that precedes them
3. the intensity of the light - when it is fully on
4. the type number - 0 through 7
5. the aim angle - the angle at which the first bar changes from the bottom color to the top color
6. the transition angle - the width of the transition zones where a light goes from off to full intensity or where it changes from one color to another
7. the cutoff angle - the angle above which the light is always off
8. the delta angle - the angle between the aim angle and/or the cutoff angle for successive bars
9. the string number for the first top and bottom lights

Other prompts that appear are for the light switch number for the VLA and whether the VLA is on a DCS.

Two items are found in the tables in the FLY program. They are delta distances and flags. Delta distance is the distance between bars in integer feet; flags tell whether the delta angle is to be added to the aim angle and/or the cutoff angle.

Lights in a VLA are modeled so that the first set is the light closest to the approach end of the runway.

5.1.21.1 TWO- AND THREE-BAR VASI'S

These VLA's provide a pilot with a color change cue of glide slope. All red lights indicates undershot (glide slope too low). A red light on top and a white on the bottom indicates being "on glide slope." All white lights indicates glide slope being too high.

The standard distance between bars of lights in a VASI is 700 feet. A more distant bar has a higher aim angle than the nearer bar; this allows a light behind to change colors at a higher slope angle.

When modeling VASI's the first lights modeled are the ones closest to the beginning of the runway. The lights are in sets on each side of the runway with bars spaced 700 feet back from the previous bar. The type number is zero (0).

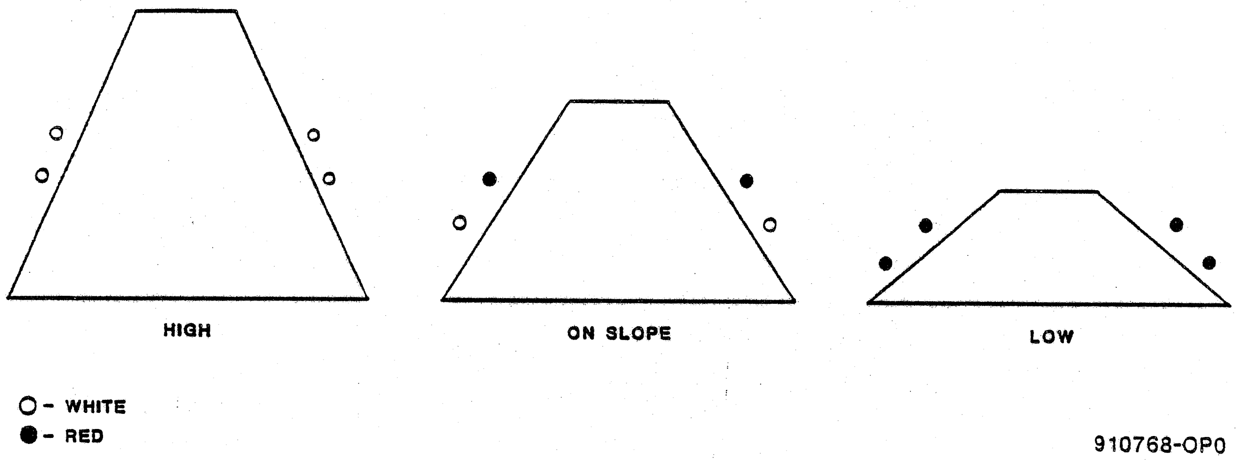
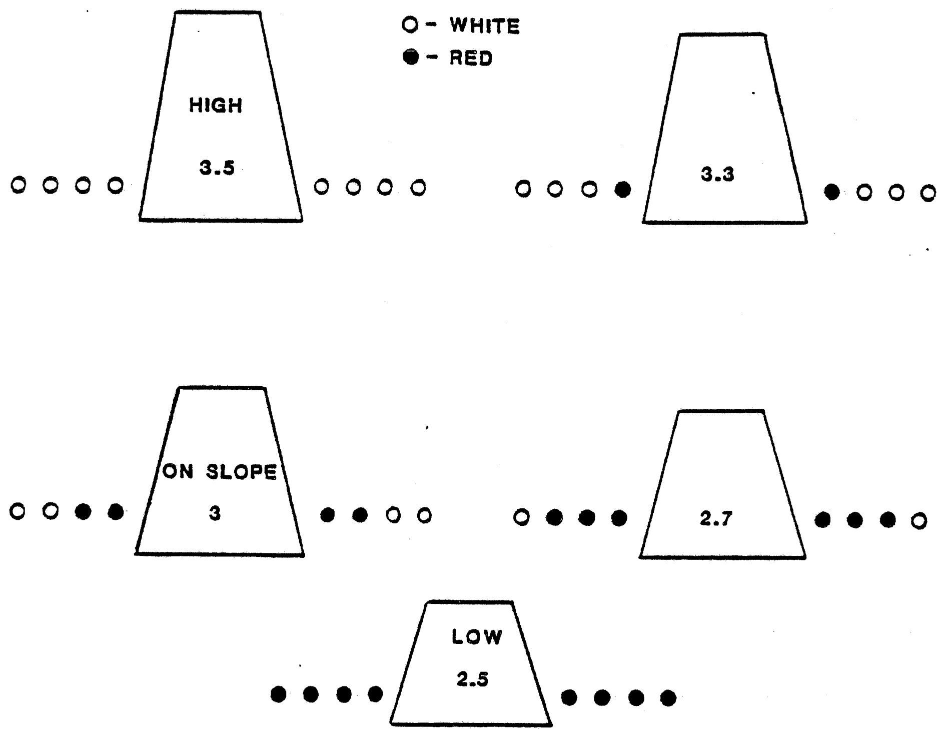


FIGURE 5-4
VASI LIGHT CONFIGURATION

5.1.21.2 PAPI'S

These VLA's provide a pilot with a color change cue of glide slope. As in VASI's, all red mean undershot, all white mean overshot. However, more lights give a better indication of the slope. The four light sets on either side of the runway are horizontally in line with one another. These lights are aimed so that the first lights to change color (going from high to low) are the ones closest to the runway. The last lights to change are the ones farthest from the runway. Two white and two red lights on a side are an indication of "on glide slope."

When modeling PAPI's the first lights modeled should be the ones farthest from the center of the runway. All lights are in a line perpendicular to the runway sitting beyond the touch-down point. The delta angle is small so that the transition from one color to another is rapid. The type number is one (1).



910769-OP0

FIGURE 5-5
PAPI LIGHT CONFIGURATION

5.1.21.3 T-BARS

T-bars are the most difficult VLA's to model because they require three VLA's each with different characteristics. The T-bar has two sets of crossbar lights and a lower set of lights turn on as needed if the aircraft is above or below the correct glide slope. If the aircraft is in severe undershot range, the full lower set and the crossbar lights all turn red. The lights change quite rapidly in the region just above and below the "on slope" angle.

The middle crossbar lights are on from ground level to the cutoff angle (probably > 10 deg.). It changes color from red to white (going from low to high) at a low angle (the same angle at which the lower sets of lights change red). Model the first lights closest to the center of the runway at about the touch-down point. Their type number is four (4).

The cutoff angle is important to the lower set of lights. These lights change from red to white going from low to high at the same angle, but the cutoff of these lights is used below the "on slope" angle (only one light is on if the angle is slightly lower than on slope, two if lower, three if even lower, etc.). Model the first lights closest to the beginning of the runway in front of the crossbar lights. Their type number is two (2).

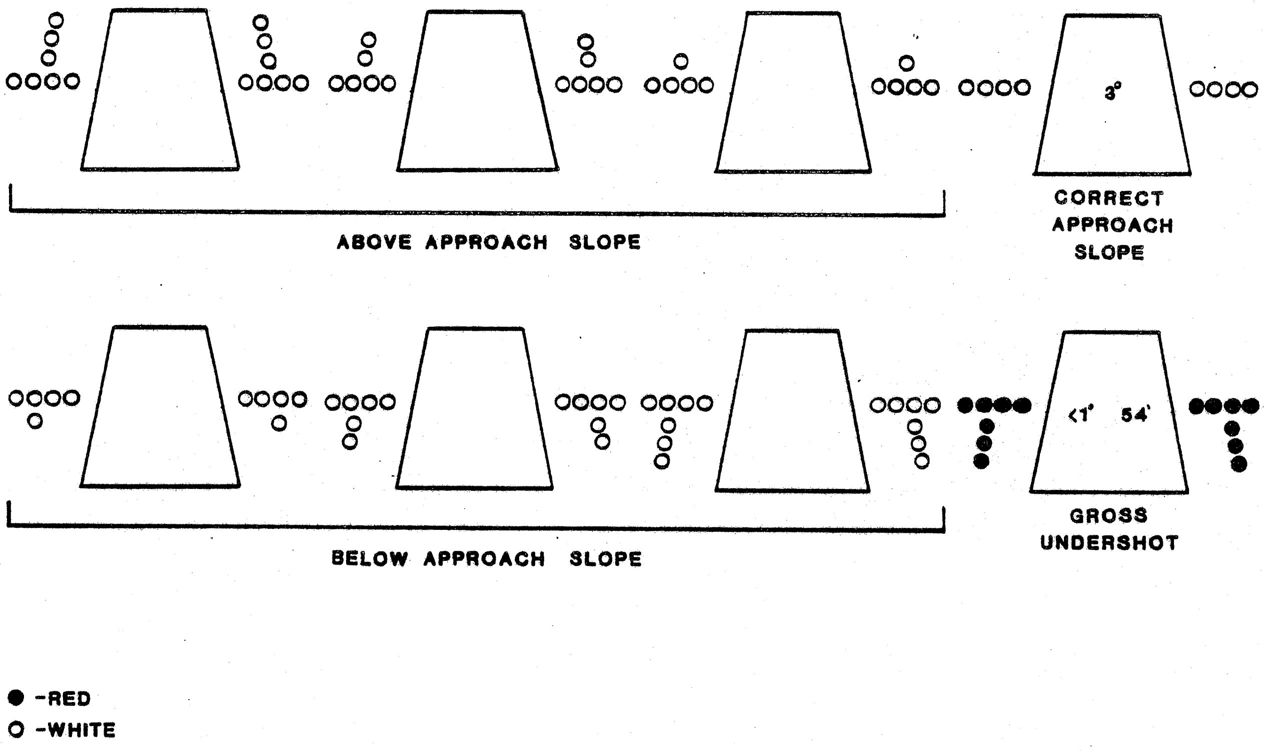


FIGURE 5-6
T-BAR LIGHT CONFIGURATION

910770-OPO

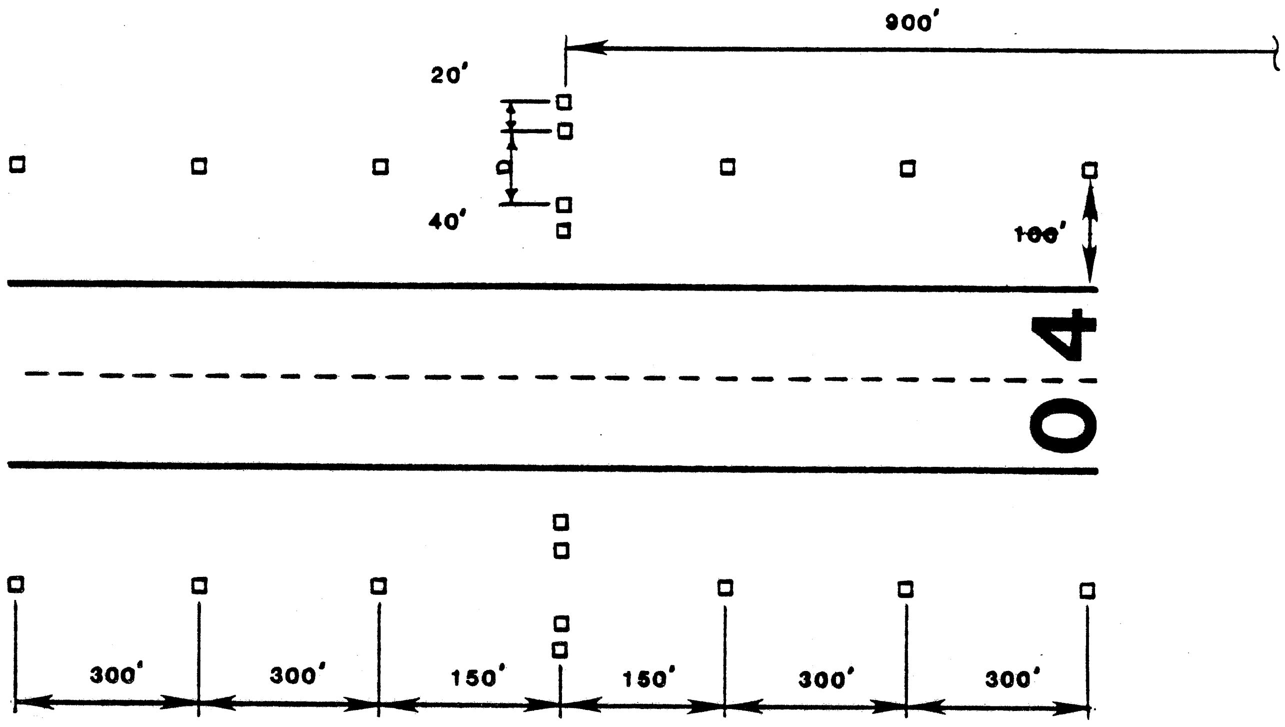


FIGURE 5-7
T-BAR LIGHT PLACEMENT

The upper set of lights comes on as the angle increases (one lights on a little above "on slope", two when a little higher, etc.). There is no bottom color lights set for the upper light sets. They are either white or off. Model the first lights closest to the beginning of the runway behind the cross-bar lights. Their type number is three (3).

The light sets on a side of the runway are placed 240 feet apart.

5.1.21.4 BARS AND REPEATS

A bar is a master set of lights on which the behavior of several sets of lights is based. The first top and bottom strings of a VLA are masters. The strings that follow, which are to behave the same way as the masters, are the repeats of the master. The master and repeats occur in sequential order, except for PAPI's which occur side by side. The VLA lights must be modeled in this way--master followed by its repeats, followed by a master and its repeats, etc. Top and bottom strings must match each others' positions; for example, the first top and the first bottom string must be modeled in the same x, y, z position. The second top and bottom strings must then be modeled in the same position, etc. (Note: A master, or repeat, consists of a top color and a bottom color light string which have the same x, y, z coordinates.)

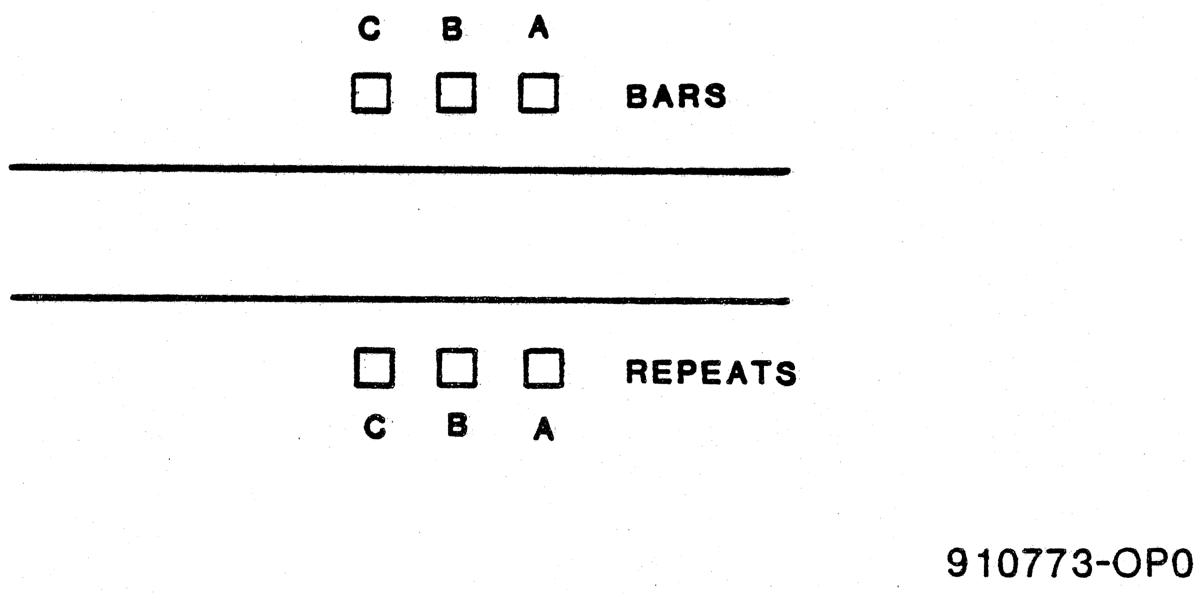


FIGURE 5-8
BARS AND REPEATS

5.1.21.5 AIM ANGLE

The VLA aim angle (A) is the angle between the ground (horizontal plane) and the middle of the transition zone from bottom color to top color for the first light modeled in a VLA. For subsequent bars, the aim angle is calculated by adding the delta angle for each bar back from the first bar. The delta angle may be zero, as in the case of the bottom set of lights in a T-bar VASI.

A common glide slope is three degrees. The aim angle should be less than this for the first master and the delta angle should be specified so that the "on slope" configuration of the lights corresponds to the "on slope" angle desired.

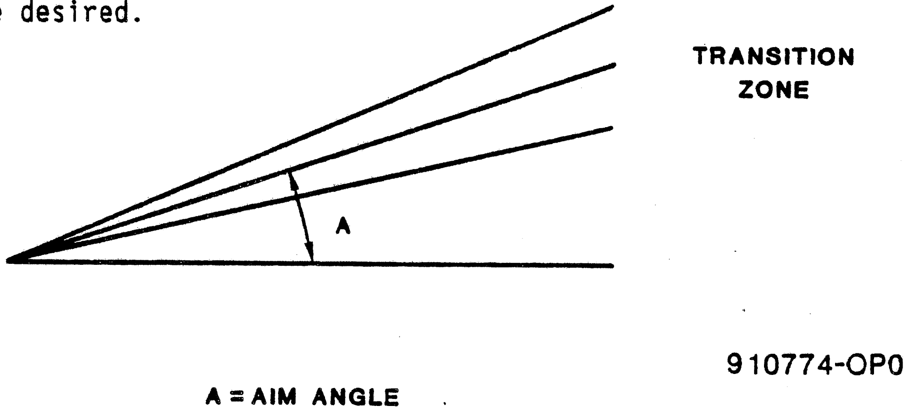


FIGURE 5-9
AIM ANGLE

5.1.21.6 TRANSITION ANGLE

The transition angle (B) is the angle over which the light changes from one color to another, or from full on to full off or vice-versa. A transition angle of zero will provide for an abrupt change. A transition angle greater than zero will allow the colors to change slowly.

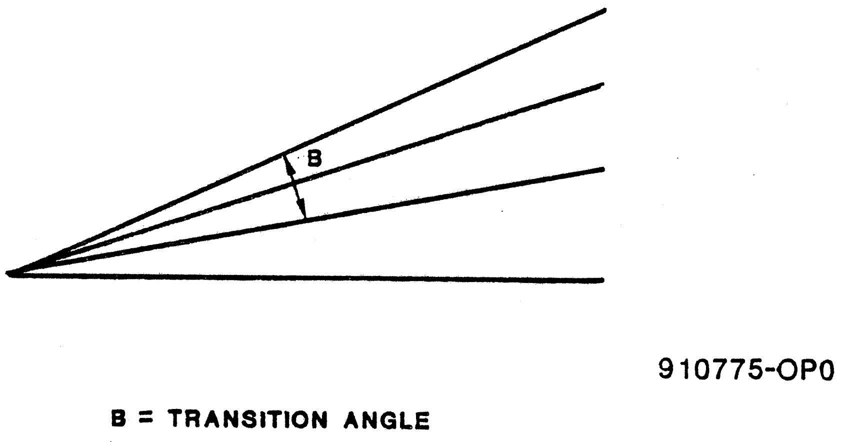
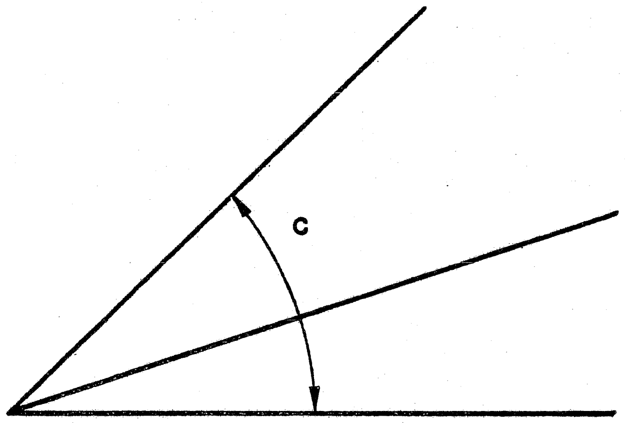


FIGURE 5-10
TRANSITION ANGLE

5.1.21.7 CUTOFF ANGLE

The VLA cutoff angle (C) is the angle from the ground at which the top color light of the first bar turns off completely. The cutoff angle of subsequent bars has the delta angle in depending on the "type" of the VLA.



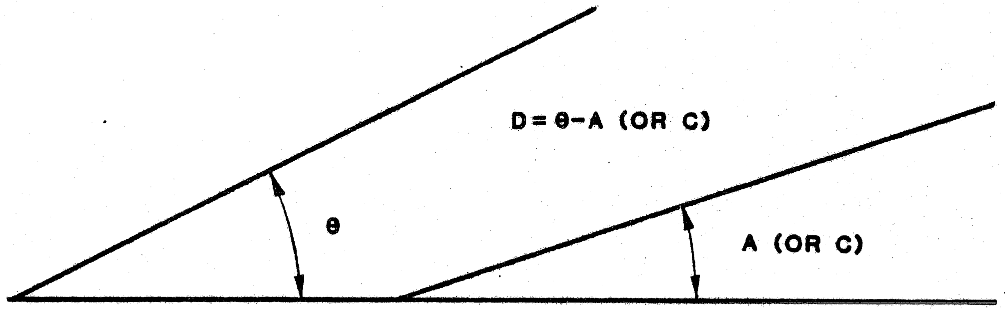
C =CUTOFF ANGLE

FIGURE 5-11
CUTOFF ANGLE

910776-OP0

5.1.21.8 DELTA ANGLE

The delta angle (D) is the angle between the aim and/or cutoff angles of subsequent bars of the VLA. The type of VLA determines whether the delta angle is added to the aim angle for successive bars and/or to the cutoff angle.



- DELTA ANGLE

910777-OP0

FIGURE 5-12
DELTA ANGLE

5.1.22 GENERALIZED VISUAL LANDING AID (VLA)

```
LIT >>   G
```

With the 'G' command, the user can insert, change or delete a VLA definition. A maximum of two VLA's can be defined in a light module--the two modules' numbers are 0 and 1. The prompt <u>VLA #:</u> appears; the user responds with the number of the VLA he wants to work with. If it has previously been defined, the prompt <u>CHANGE OR DELETE?:</u> appears. If the user wishes to only delete it, he responds with 'D'. If he responds with 'C', the program will allow him to change any parameter of the VLA.

The prompts <u>BOTTOM STRING:</u> and <u>TOP STRING:</u> are then output. Top strings are those which are turned on when viewing from a high angle. Bottom strings are those which are turned on from a low angle. Top strings and bottom strings must be different colors. If the VLA is to have only one color then an 'X' response to the top (or bottom) string prompt will specify that there are no top (or bottom) strings. The set of light strings which make up the bottom strings and top strings are begun with a master light closest to the approach end of the runway. These light string names are entered in response to the above prompts.

The next prompt, <u>SWITCH #:</u> is the same as <u>SWT #=</u> for a normal light string.

The <u># MASTERS=</u> prompt requires an integer from 1 to 8. The <u># REPEATS=</u> prompt requires an integer from 0 to 7. <u>INT=</u> is the same as <u>INT=</u> in a normal light string definition. Next the user will be asked for the type of VLA to be inserted with the prompt <u>TYPE=</u> to which the user should enter the corresponding number from the following list.

The types currently defined are:

0 - VASI
1 - PAPI
2 - T-Bar lower lights
3 - T-Bar upper lights
4 - T-Bar datum lights
5 - Undefined
6 - Undefined
7 - Undefined

The angles (<u>A:</u>, <u>B:</u>, <u>C:</u>, <u>D:</u>) are real number angles between 0 and 45 degrees.

Respond to the DCS: prompt with a 'Y' if the VLA is to be on a DCS; and with a 'N' if it does not.

NOTE:

Before entering the VLA definition the light strings should be entered into the module as described in the VLA section.

5.1.23 LIST VLA MAP

```
LIT >>   M
```

The 'M' command produces a listing of a given VLA and its associated light strings. The prompt VLA #: is responded to with 0 or 1 and the VLA information will be printed on the listing device.

SECTION 6

SURFACE MODULE FUNCTIONS

6.1 INTRODUCTION

The surface module contains all of the information that the system requires to define the surfaces in a model. A surface module contains edge and face definitions, along with control and grouping information such as entry points, command lists, and priority trees.

Surface modules are defined and manipulated with the following functions. To select a function, type the appropriate character in response to the prompt <u>SUR>></u>.

A	Angular rotation
B	Blank screen (option)
C	Change item
D	Display faces (option)
E	Edit surface module
F	Find face (option)
H	Insert header description
I	Insert item
J	Join surface modules
K	Kill item
L	List item
M	Define mirror point vector
N	Print statistical information
P	Define entry point
Q	List entire surface module
R	Read surface module
S	Define center line management
T	Translate
U	Define texture cloud management
V	Set observer viewing position (option)
W	Write surface module
X	Exit
Y	Show buffer combinations
Z	Zero surface module

Each of these functions is discussed separately on the following pages.

6.1.1 EDIT SURFACE MODULE

SUR>> E

With the edit module function the user can declare the current module in memory to be active without destroying the data. This may by used after a program abort or a module write function to gain access to the surface module data. It is assumed by the system that the user is enabling access to valid surface module data.

6.1.2 READ SURFACE MODULE

SUR>> R

Previously generated surface modules are read into the modeling system with the read function. In response to the selection of the read function, the system prompts the user by typing:

FILE NAME=

The user reply depends on the type of device that the file is to be read from, as follows:

CS1: or CS2:	-Cassette tape drive (foreground mode only)
FD0: or FD1:	-Floppy disk drive (single-sided)
FT0: or FT1:	-Floppy disk drive (double-sided, top)
FB0: or FB1:	-Floppy disk drive (double-sided, bottom)

The device name is to be followed immediately by a file name, from one to six characters in length. If input is from cassette tape, the user is expected to have rewound the tape. If a file name is entered and the device name is omitted, FD0 is assumed. If an active module currently exists, the system will ask for verification of the read command by displaying VER> to which the user must respond with either "Y" to complete the read function or "N" to abort it.

The user may read either a texture (SP1/T) or non-texture (baseline SP1) model. If a non-texture model is read in, it is automatically converted to the texture format. No texture information is added to the model, but the model is compatible with the SP1/T realtime program. Faces are not given a default texture map number, contrast ratio, or lambda adjustor value. The 32 texture feature decode buffer entries contain an intensity value, intensity flags, and an indication that the face is non-textured.

If the faces of the SP1 model use more than 32 unique combinations of illumination function flags and intensities, any faces which do not access the first 32 unique combinations are deleted. If any faces are deleted, the message "FACES DELETED DURING CONVERSION = " is displayed, along with the number of faces deleted during the conversion.

The message "SP1 MODEL CONVERTED TO SP1T FORMAT" is displayed when the SP1 to SP1/T conversion is complete.

6.1.3 ZERO SURFACE MODULE

SUR>> Z

The zero function, which is the first command executed when defining a new surface module, clears all edge and face definitions and resets all the pointer structures. If an active module currently exists, the system asks for verification of the zero command by displaying VER>. The user must respond with either "Y" to complete the zero function or "N" to abort it.

6.1.4 EXIT

SUR>> X

After the user has completed surface module editing, he uses the exit function to select the editing mode of another type of module. If there is still an active module when he selects the exit function, the system responds with the message:

ACT MOD
VER>

The user replies with either a "Y" to exit or an "N" to abort the command.

6.1.5 INSERT MODULE DESCRIPTION

SUR>> H

Each module can carry a maximum of 30 characters of descriptive text. After the user selects the H function, the system replies with the prompt:

TEXT:

The user then enters or changes the text which is printed at the top of every complete module listing.

6.1.6 ENTRY POINT DEFINITION

SUR>> P

The hardware processes these groups together as the command list references them. This facilitates referencing groups of edges and faces in units smaller than a module. Each surface module has a minimum of one and a maximum of eight defined entry points for edges and faces. After the user has selected the entry point definition function, he receives the prompt:

ENTRY=

He responds with the name of the entry point to be defined, P1 through P8; he terminates the entry with either a tab or a carriage return. If a legal entry point was entered, the prompt ITEM #1 appears; the response to this prompt specifies the first edge or face. After the user has entered a legal edge or face name, the prompt ITEM #2 appears; it requests the second edge or face specification. The user responds with the second edge or face name.

Defining the first and second legal face name defines a group of edges or faces from the first through the second specification inclusively. The system requires that both the first and second item be either edges or faces. To delete an entry point, define the entry point with the first edge or face name higher than the second edge or face name.

For example: Entry = P3 Item #1= E5, Item #2= E3
would cause entry point P3 to be deleted.

Since entry points are entered independently and since they define a contiguous group of edges or faces, be careful not to overlap entry point definitions; this causes incorrect edge or face groupings.

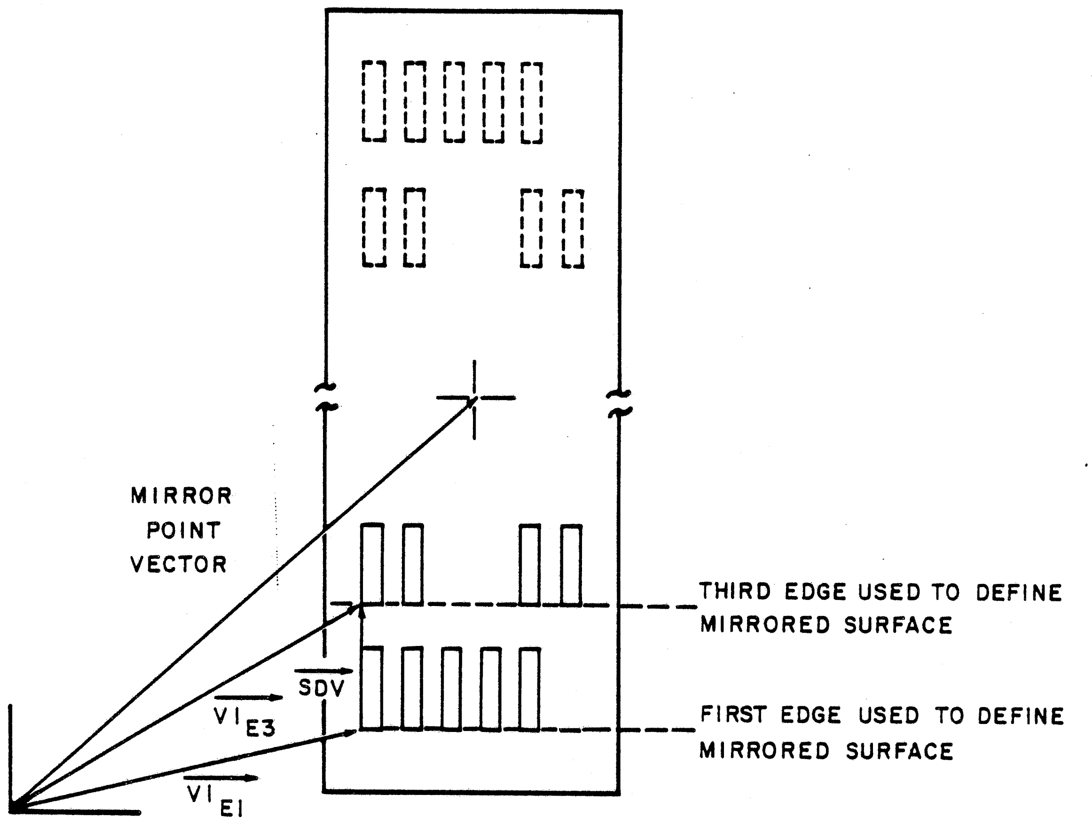
6.1.7 DEFINE MIRROR POINT VECTOR

SUR>> M

By defining a mirror point vector the user can manage symmetrical faces such as end markers and touchdown zones, and he does not have to model them on both sides of the runway.

In response to the MS= prompt, the user inputs either 0 or 1. Zero indicates that an existing mirror point is to be deleted; one indicates that a mirror point is to be defined or changed. If the mirror point is being defined or changed, the user is prompted with X= and Y=. The user specifies the X and Y coordinate values for the mirror point. The specified point represents the rotation origin about which the edges defined by entry point 1 will be rotated 180° by the realtime software when doing mirrored surface management.

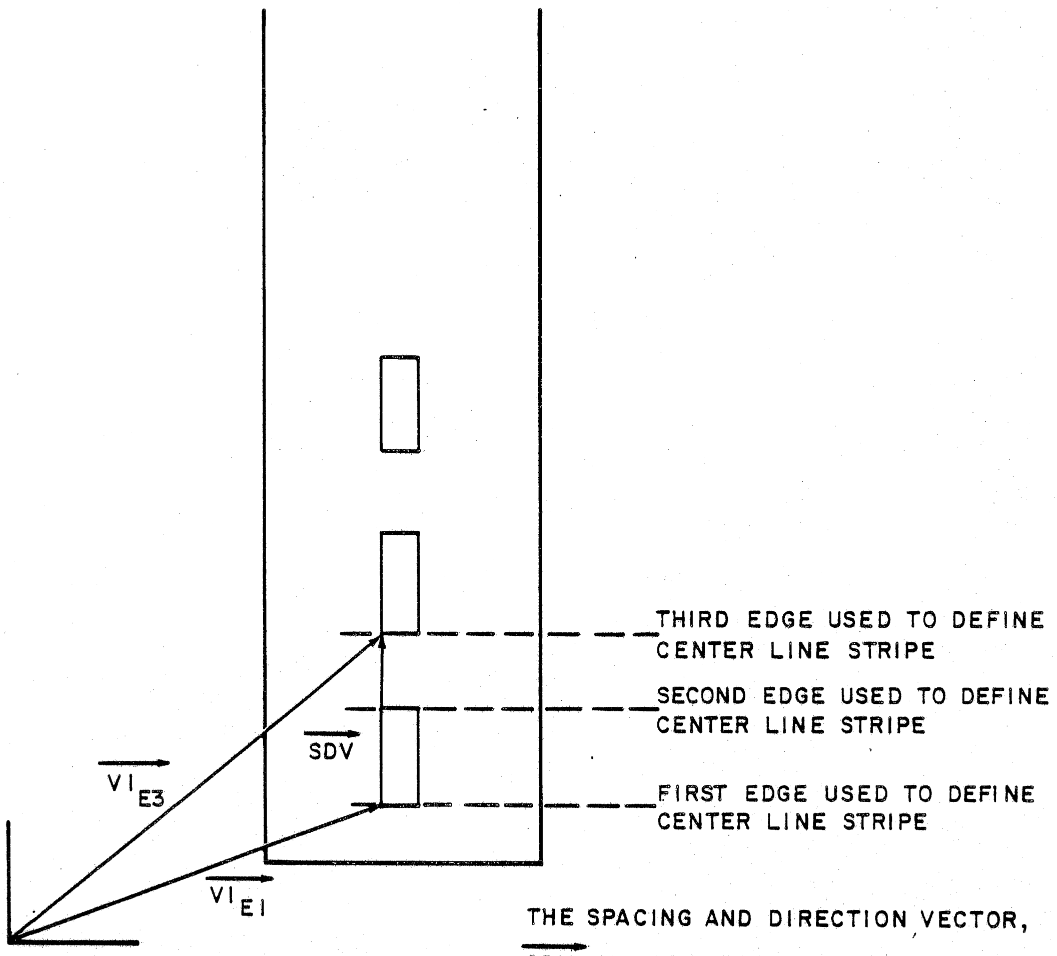
Figure 6-1 shows how some mirrored surface management requires that all mirrored surfaces must be horizontal surfaces and cannot be part of a dynamic coordinate system. Edge definitions used in defining the mirrored surfaces must also be defined as shown in Figure 6-1. NOTE: All edges being mirrored must run in the same direction.


MIRROR POINT VECTOR
$\overrightarrow{SDV}$
$\overrightarrow{VI}_{E3}$
$\overrightarrow{VI}_{E1}$
THIRD EDGE USED TO DEFINE MIRRORED SURFACE
FIRST EDGE USED TO DEFINE MIRRORED SURFACE

THE SPACING AND DIRECTION VECTOR, $\overrightarrow{SDV}$, IS COMPUTED FROM THE FIRST VERTEX VECTORS, $\overrightarrow{VI_{E1}}$ AND $\overrightarrow{VI_{E3}}$, USED TO DEFINE THE FIRST AND THIRD EDGES FOR THE MIRRORED SURFACES.

910057-0P1

FIGURE 6-1
MIRROR SURFACE AND MIRROR POINT VECTOR


THIRD EDGE USED TO DEFINE CENTER LINE STRIPE
SECOND EDGE USED TO DEFINE CENTER LINE STRIPE
FIRST EDGE USED TO DEFINE CENTER LINE STRIPE
$\overrightarrow{VI}_{E3}$
$\overrightarrow{SDV}$
$\overrightarrow{VI}_{E1}$

THE SPACING AND DIRECTION VECTOR, $\overrightarrow{SDV}$, IS COMPUTED FROM THE FIRST VERTEX VECTORS, $\overrightarrow{VI_{E1}}$ AND $\overrightarrow{VI_{E3}}$, USED TO DEFINE THE FIRST AND THIRD EDGES FOR THE STRIPE FACES.

910058-0P0

FIGURE 6-2
EDGE DEFINITION FOR CENTER LINE MANAGEMENT

6.1.8 DEFINE CENTER LINE MANAGEMENT

SUR>> S

By defining the center line management the user provides a center line strip for the entire length of runway; he does not have to model every segment of the stripe. The prompt MODEL STRIPES appears, and the user enters the number of segments that were modeled. The prompt REMAINING STRIPES= appears, and the user enters the number of remaining stripes. The "FLY" program uses this information to manipulate the edges grouped by entry point 2; the program does center line management when those edges/faces are within visual range. When the user is defining the edges to be used for center line face definitions, he must define them as shown in Figure 6-2. Defining them in this way provides the appropriate direction and spacing information for the "FLY" program. Edges and faces that are center line managed cannot be part of a dynamic coordinate system.

6.1.9 INSERT EDGE, FACE, INSTRUCTION, AND NODE

SUR>> I

With the insert function for surfaces the user can define edges, faces, command list instructions, and priority tree nodes for the active surface module. After the user selects the insert function, the prompt ITEM #= appears. The user selects the data type and number to be entered by responding with one of the following:

E1 thru E64 to define edges
F1 thru F64 to define faces
I1 thru I15 to define command list instructions
N1 thru N5 to define priority tree nodes

After the user specifies which data type is to be entered, the modeling system requests the input that is appropriate for that particular data entry. If the user is entering several items of the same data type that are in sequential order, typing a tab automatically selects the next item of that type following the first entry's completion.

6.1.9.1 INSERT EDGE DEFINITION

ITEM #= Ex

Respond to the ITEM #= prompt with E1 through E64 to define an edge. Edges are the bases for defining the faces for the system. An edge definition consists of some control flags and two vertices that define the edge location and direction. When inserting an edge definition the system prompts the user with the message FLG>. The user responds with Y if the flag values are to be changed and N if the current flag values are to be used. If the user chooses to set the flag values, the following flag prompts will be output.

*TRAN?	0 =	translate this edge as required during hardware processing (default)
	1 =	do not translate this edge during hardware processing (a wing tip, for example)
CHAN=	0 =	edge to be processed by all channels (default)
	1-8 =	edge to be processed only by the specified channel (a wing tip, for example)

After the flag values have been set, they remain at that value for any successive edge definitions as long as the user remains in the insert mode.

After he specifies the flags, the user inputs the vertices that define the edge position and direction. The prompts X=, Y=, and Z= appear for the first vertex; the same prompts appear for the second vertex. The positive direction of the edge is from the first vertex to the second vertex (see Figure 6-3).

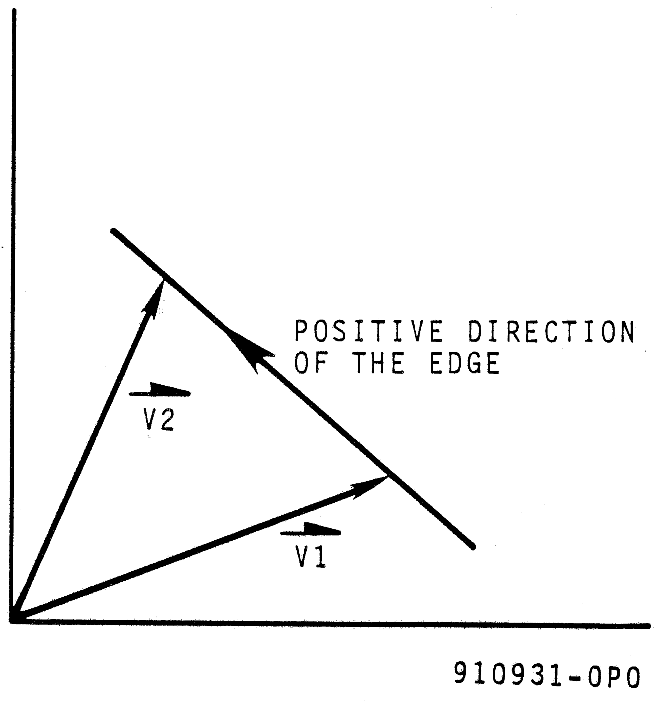


FIGURE 6-3
EDGE DEFINITION

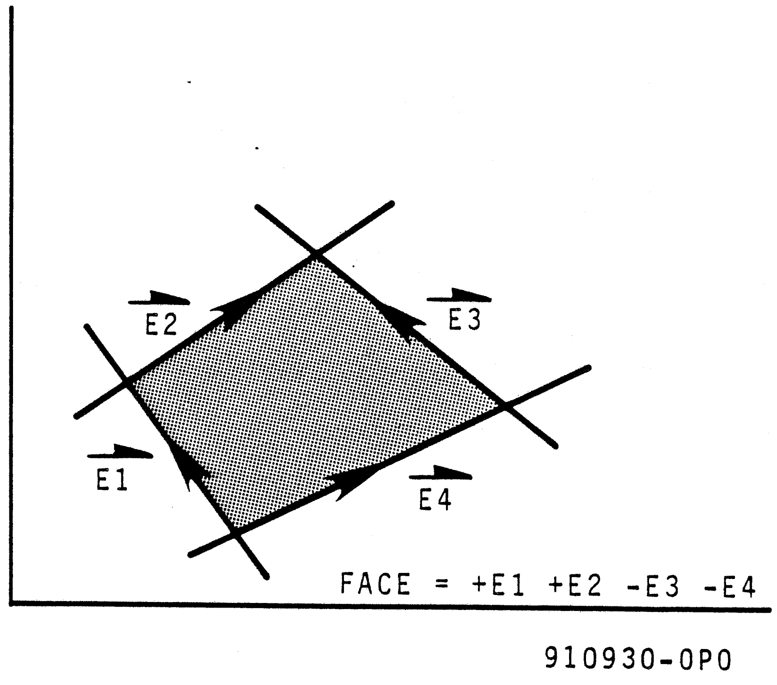


FIGURE 6-4
FACE DEFINITION

6.1.9.2 INSERT FACE DEFINITION

ITEM #=   Fx

Respond to the ITEM #= prompt with F1 through F64 to define a face. Faces are defined in the system by referencing four edge definitions in clockwise order. The sign of the edge name represents whether that edge is to be used in the positive or negative direction, relative to the definition of the particular edge. In addition to referencing the defining edges, the face definition also contains information about intensity, illumination, and special processing.

After the user selects a face for insertion, the prompt INT= appears; the response to this prompt defines face intensity. Valid responses are integers between 0 to 63, where 0=black and 63=white; these responses equal the 64 intensity levels for faces.

The prompt PRI= appears; the response to this prompt defines the default hidden priority for this face. Valid responses are integer values from 1 to 16. One = lowest priority (farthest away), and 16 = highest priority (closest). Realtime (dynamic) priority, as defined by the priority trees, takes precedence over any default (fixed) priority.

The prompt FLG asks whether any of the special processing flags are to be altered. By responding with a Y, the user can change any of the items listed in Table 6-1. After the flag values have been set, they remain at that value as long as the user remains in insert mode and responds with a N to the FLG prompt. The flags LLL, GND, 3D and OCC are illumination functions for the face. The TXT flag determines whether texture is to be applied to that face.

After the user has given the value, "N", to the 3D flag either explicitly by entering "N" or implicitly by entering the LLL and GND flag values, the program then allows the user to enter texture data. The prompt TXT? is displayed. If the user responds with a "Y", three additional prompts follow.

"CONTRAST RATIO=n <1=LO, 2=MED, 3=HI> n=", to which the user enters an integer between 1 and 3, where 1 = low, 2 = medium and 3 = high contrast, respectively. (If the face is a non-textured face, the contrast ratio is set to zero.)

"TEXTURE MAP #n <n=1,2,3,4> n=" to which the user enters an integer between 1 and 4, selecting the texture map to be applied to this face. The texture map numbers correspond to the texture map numbers in the group module.

"LAMBDA ADJUSTER = 1/n <n=1,2,4,8> n=" to which the user enters 1, 2, 4 or 8, denoting a lambda adjuster of 1/1, 1/2, 1/4 or 1/8, respectively. The lambda adjuster allows the map size (lambda) to be used at the same value specified in the group module, or reduced by a factor of 2, 4 or 8 for this face.

After 32 unique combinations of intensity, illumination and texture data have been used for a surface module, all faces inserted or changed within that module must access one of these 32 existing feature decode buffer entries. When all 32 feature decode buffer entries are used, and the face accesses an existing entry, the message:

```
"          #F33
ACCEPTED"
"FEATURE DECODE BUFFER FULL"
```

is displayed. When all 32 feature decode buffer entries are used, and the face entered does not match an existing entry, the message:

```
  "FEATURE DECODE BUFFER FULL"
"          #F33
NOT accepted"
```

is displayed.

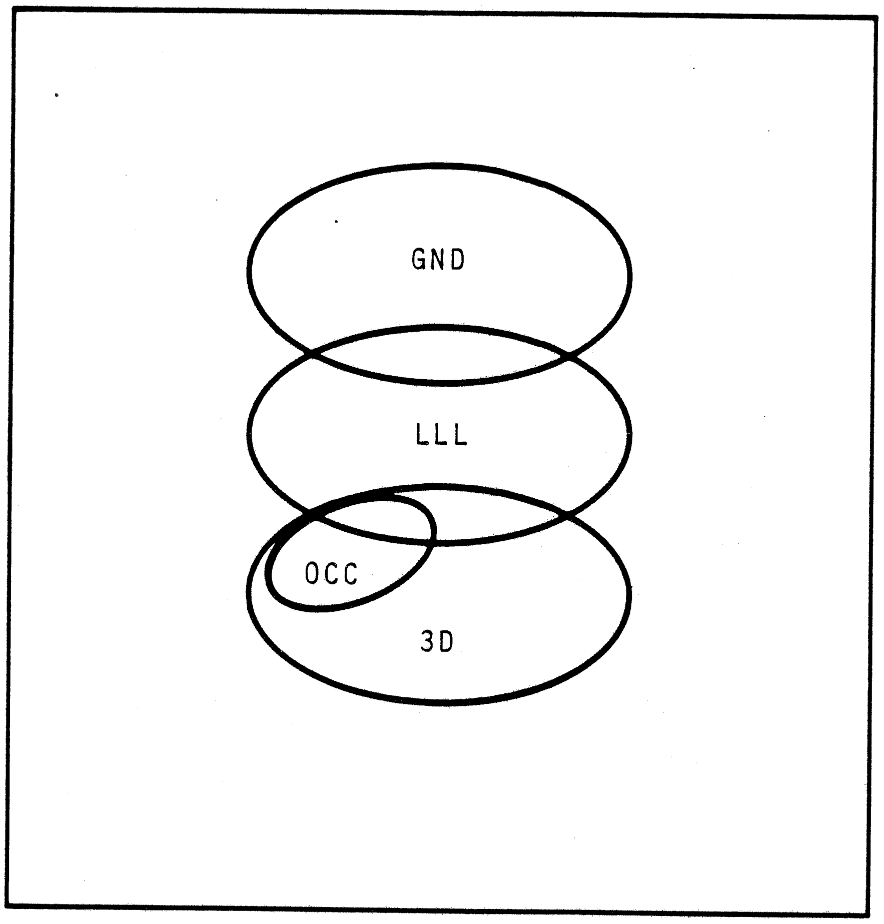
After specifying the flags, the user must input the edge references that define the face. The modeling system will output the prompts E1=, E2=, E3= and E4=; the user responds with the signed edge number for the four edges that define the face (see Figure 6-4). Make sure that the faces are defined to be planar. The modeling system does not check for planarity, and errors in operation of the data base occur if the faces are not planar. If the user wishes to define triangular faces rather than quadrilateral faces, then he simply repeats either the first or last edge reference.

NOTE

When inserting edges and faces the edges must be inserted first or an error will result.

TABLE 6-1
FACE FLAGS

PROMPT	RESPONSE	
LLL?	N =	This face will not be affected by landing lights (default).
	Y =	This face will be illuminated when in range of the landing lights.
GND?	N =	This face is not affected by ground illumination (default).
	Y =	This face is illuminated because it is a ground face.
3D?	N =	This face is not three-dimensional, and it is affected by ground illumination (default).
	Y =	This face is three-dimensional and will be affected by the 3D illumination functions.
OCC?	N =	This face does not occult lights; they will shine through (default).
	Y =	This face occults all lights, except for those lights where PRI=1.
*TRAN?	0 =	Translate this face as required during hardware processing (default).
	1 =	Do not translate this face during hardware processing (a wing tip, for example).
CHAN=	0 =	Face is processed by all channels (default)
	1-8 =	Face is processed only by the specific channel (a wing tip, for example).
TXT?	Y =	texture is applied to this face.
	N =	texture is not applied to this face.



910933-0P0

FIGURE 6-5
LEGAL COMBINATIONS OF ILLUMINATION FLAGS

6.1.9.3 INSERT COMMAND LIST INSTRUCTION

ITEM #=   Ix

Respond to the ITEM #= prompt with I1 through I15 to define a command list instruction. The surface module command lists tell the realtime system how to display the edge and face information contained in the module. For example, they indicate what faces are to be mirrored, and what edges are to be processed when the center line stripes are moved along the runway.

There are two command lists: one for edges and one for faces. Edges and faces are not referenced directly in their respective lists; instead, commands reference entry pointers (P1 through P8). Entry pointers are a means of naming collections of edges and faces. Only the edges and faces that appear on the screen must be referenced in the command lists. The order of the command lists must parallel the order of the edges and faces in the module. For example, suppose that entry point P6 references edges E5 through E10 and entry point P6 references E11 through E60. The command referencing P6 must precede the command referencing P4 in the command list.

Surface module command lists are usually very simple. They often consist of only two instructions: an EDGE or FACE command which references an entry point grouping all of the edges or faces and a LINK command which returns processing control to the governing environment command list. Both lists must be terminated with a LINK instruction.

A command is defined as follows:

The user specifies an instruction to be inserted by responding I1 through I15 to the prompt ITEM#=. The prompt EDG,FAC> appears; the response to this prompt, either E or F, indicates the command list to be used. E indicates edge command list and F indicates face command list. After the user selects the command list, the prompt INST= appears; this prompt requests the command to be inserted. User responses to this prompt are the command strings defined in Table 6-2. This table also provides a brief description of each command, any subsequent information requested for that command, and the appropriate responses to such subsequent requests.

The following two examples are surface module command lists that the user may use in a model.

Example #1 - Normal processing of edges and faces

```
Edge command list:     EDGE ENTRY=P1
                       LINK

Face command list:     FACE ENTRY=P1
                       LINK
```

Example #2 - Surface module containing mirrored surfaces, center line managed surfaces, and normal surfaces

```
Edge command list:     TMIR              mirrored surfaces
                       EDGE  ENTRY=P1      processing
                       MMIR
                       TVEC  MTX=T1      center line management -
                       TADD  OPT=01        management processing
                       EDGE  ENTRY=P2
                       TVEC  MTX=T1      normal edge
                       EDGE  ENTRY=P3      processing
                       LINK

Face command list:     FACE  ENTRY=P1
                       LINK
```

TABLE 6-2
SURFACE MODULE COMMAND LIST INSTRUCTIONS

COMMAND	ADDITIONAL PROMPT	RESPONSE	FUNCTION
PVEC			Loads the P-vector into the matrix multiplier. The P-vector defines the raster parameters for the display.
SCHN			Concatenates the surface channel matrix for each channel to the P-vector. The channel matrix defines the orientation for each specific channel.
MTX	MTX=	T1 thru T7	Concatenates the transformation matrix specified to the current transformation matrix.
MMIR			Signals end of mirrored surface processing.
TVEC	MTX=	T1 thru T7	Loads the specified T-vector.
TADD	OPT=	01,02,03	Adds the specified delta vector to the current T-vector. Currently 01 is for center line management and the other options are undefined. Center line edges must be on entry point P2.
TMIR			Signals start of mirrored surface processing. Edges must be on entry point P1.
EDGE	ENTRY=	P1 thru P8	Defines edges to be processed via specified entry point.
FACE	ENTRY=	P1 thru P8	Defines faces to be processed via specified entry point.
LINK			Returns to environment module command list.
TXTR			Loads the texture blocks.

6.1.9.4 INSERT PRIORITY TREE NODES

ITEM #= Nx

Respond to the ITEM #= prompt with N1 through N5 to define a priority tree node.

6.1.9.4.1 PRIORITY

SP1/T models are usually built to be viewed from several different perspectives. As such one often views a model from a position where a collection of faces hide (or occult) other faces or lights, i.e., a building is seen standing in front of another building or several street lights. In terms of the hardware, this means that two objects are mapped to the same pixel on the screen. However, only one of the objects can be displayed.

The realtime system needs to know which face or light has priority (see Figure 6-6). F1 has priority over F2 and, therefore, hides part of F2 on the screen. There are two ways of prioritizing faces: fixed priority and dynamic priority. Both are discussed following the figure.

6.1.9.4.2 FIXED PRIORITY

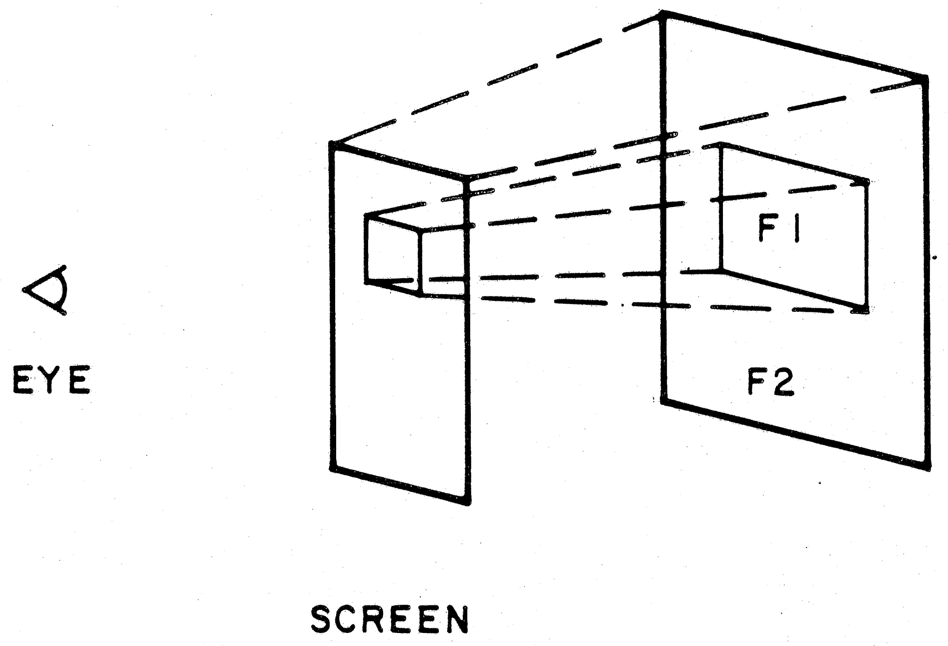
Often, one face is always viewed in front of another. When this happens, it is possible to 'fix' the priority of both faces; assign a higher default priority (one of the face definition flags) to the occulting face than the one assigned to the occulted face. For example, if F1 is always seen as standing before F2, F1 might be given priority 4, while F2 would be given priority 3.

There are 16 levels of priority. If two faces share the same default priority number, then the face with the lowest face number will appear in front (take precedence); this is known as list priority.

If the faces are not in the same module, then the face whose module is referenced earliest in the environment command list will have priority.

NOTE:

Large-scale use of Fixed Priority may cause problems during model tuning. Addition of faces late in the design may force priority levels of a large number of faces to be adjusted up or down. Use list priority or dynamic priority as much as possible.



910060-0P0

FIGURE 6-6
OCCULTING FACES

6.1.9.4.3 DYNAMIC PRIORITY

Dynamic priority changes as the viewpoint changes. It is modeled as a binary tree which is traversed every frame. The order in which the tree is traversed dictates the relative priority of the faces. A "plane test" can alter the traversal at any non-terminal node of a tree. For this, an imaginary plane dividing the subject into two parts is defined. The true side of the plane is defined to be the side from which the plane normal emerges. Normals are defined by the left-hand rule. Each node has two descendants: a 'true' son and a 'false' son. If the node does include a plane test, the realtime system tests for which side the viewer is on and traverses the opposite node first. For example, if the viewer is on the 'true' side of the plane then the 'false' son will be traversed first and its features will receive a higher priority and appear in front of the 'true' son features.

If there is no plane test then the 'false' son node is always traversed first, giving it a higher priority than the 'true' son. Thus, the 'false' son features always appears in front of those of the 'true' son.

A priority tree often spans environment and surface modules. The example of a jet shown in Figure 6-8 demonstrates this concept. SM1-SM4 refers to the surface modules, and P1-P3 refers to the dividing planes. In this example, the planes are shown on end with their respective normals indicated. The priority tree in Figure 6-8 would properly serve the model. Nodes N1-N3 are in the environment, while the remaining nodes are in the surface modules.

If the viewer were seated facing the CRT, as shown in Figure 6-8, then the tree would be traversed in the order shown in Figure 6-9, with the face clusters receiving the given priorities. A good exercise for the reader is to change the viewpoint and traverse the tree accordingly.

By defining priority tree nodes the user can modify the default priority order of the surfaces in a surface module. The priority tree in the environment module references entire surface modules and/or subtrees in particular surface modules.

After the user specifies the priority tree node to be defined, the prompt PLANE TEST>, appears; the user responds with an N if no separating plane is to be defined or a Y if a separating plane is to be defined. The user is defining a separating plane and is changing an existing node, the prompt NEW PLANE> appears. The user responds with a Y to define a new separating plane or an N if the existing one is to be used. To define a separating plane, three vertices are required to define a plane. The prompts X=, Y=, and Z= appear; input the three vertices in response to these prompts. They are repeated three times--one for each vertex.

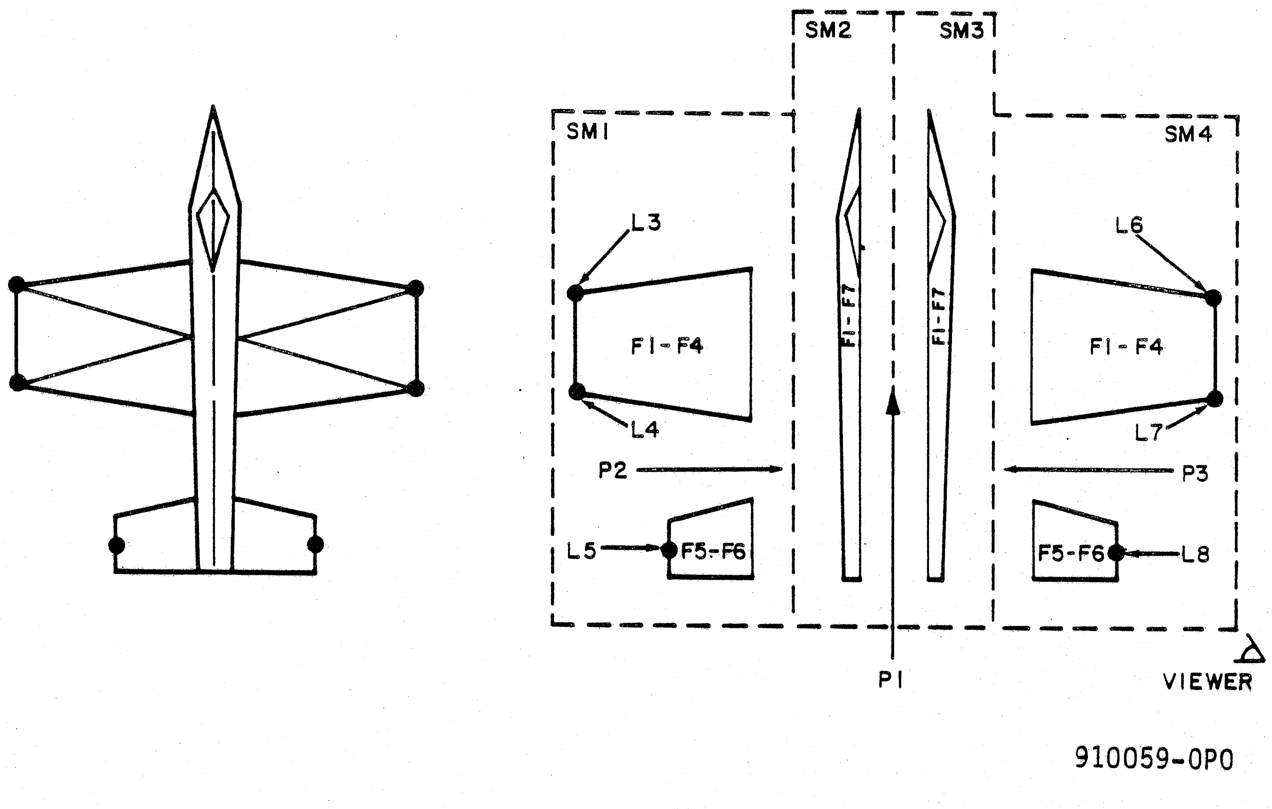
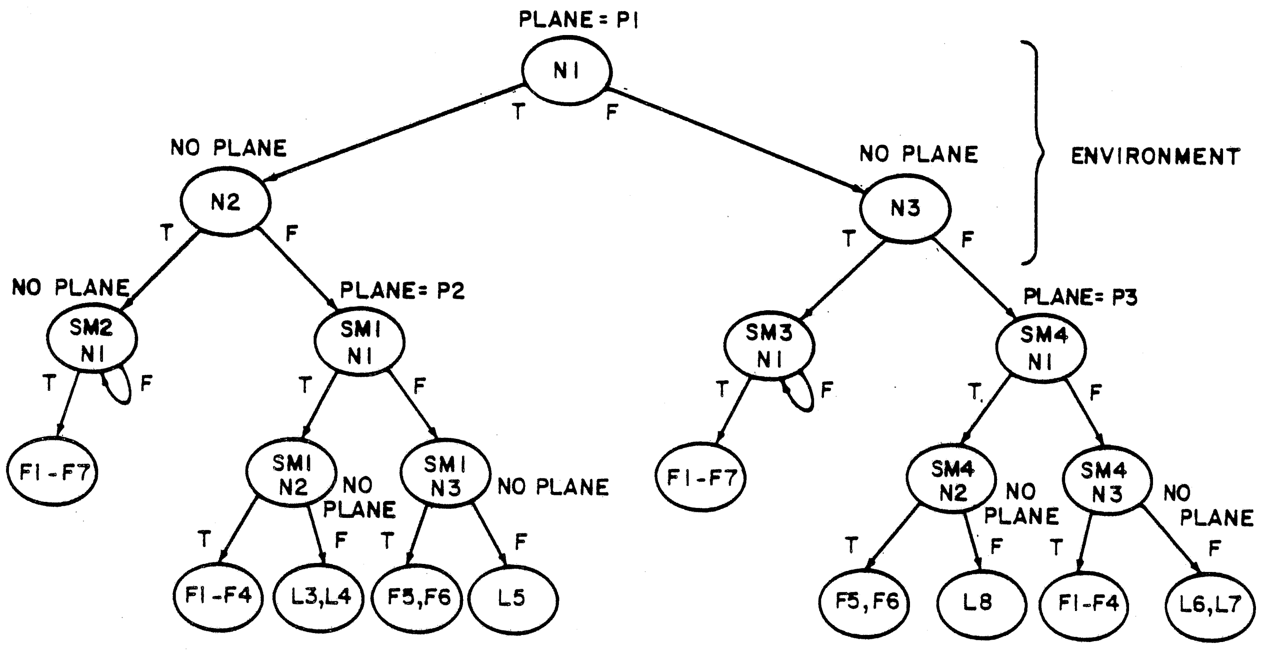


FIGURE 6-7
AIRCRAFT USED TO DEVELOP PRIORITY TREE EXAMPLE



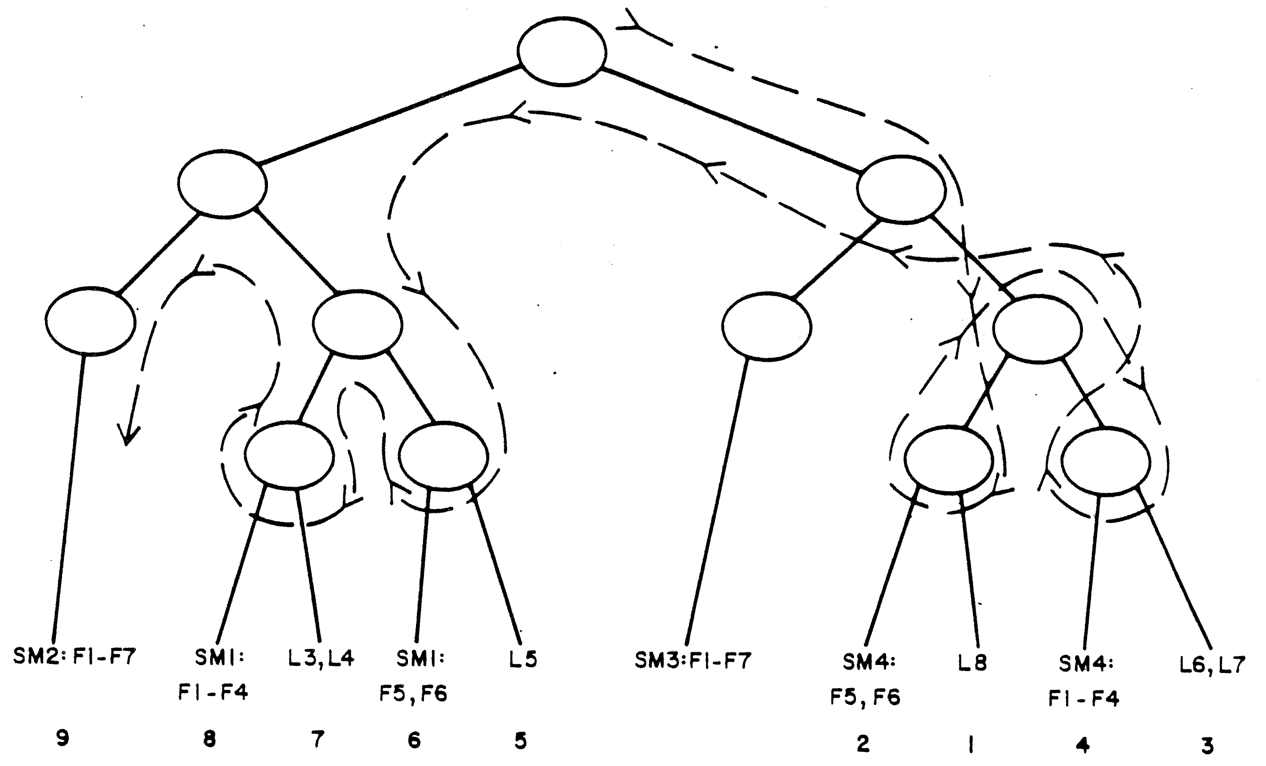


FIGURE 6-8
TRAVERSING A PRIORITY TREE

910061-0P0

The true and false son branches for this tree node is requested by the prompts TSON= and FSON=. User response to each of these prompts may be either another node number, N1 through N5, or a face number, F1 through F64. If the user specifies a face number, the prompt LAST FACE= is output to request the last face number in the cluster being defined for this branch of the priority tree. Note the two different formats in the example below:

```
#N1       PLANE TEST>      N
    FTR=  N1   (determined by the system)
    TSON= N1
    FSON= F1     LAST FACE=F50
```

6.1.10 CHANGE EDGE, FACE, INSTRUCTION, AND NODE

```
SUR>>    C
```

With the change function, the user can modify the definition of an existing edge, face, command list instruction or priority tree node in the active surface module. After the user selects the change function, the prompt ITEM #1= appears. At this point the user proceeds as if he were performing an insert on the specified item. For the parts of the definition that remain unchanged, answer the prompt with a tab. If a particular item is to be changed, enter the new value. After the user changes an item which is clear in all other aspects, then he can skip the remaining change prompts by typing the escape or alt-mode key.

If the user specifies a nonexistent item, the error message NO ITEM appears. The system then returns to the prompt SUR>> which allows the user to select another function.

NOTE:

After doing a change to the coordinates of an edge the user must translate the entire module to X=∅, Y=∅, and Z=∅ to reset the pointers in the module.

6.1.11 KILL EDGE, FACE, INSTRUCTION, AND NODE

SUR>>   K

With the kill function, the user can delete items from the active surface module. After the user selects the kill function, the prompts ITEM #1= and ITEM #2= appears. They request that the user enter the first and last item of the same type to be deleted. If the item number for the first item is greater than the second item, nothing is deleted and the command is ignored. Appropriate responses to the prompts are:

E1 thru E64 for edges
F1 thru F64 for faces
I1 thru I15 for command list instruction
N1 thru N5 for priority tree nodes

NOTE:

When deleting Edges and Faces delete faces first or the warning message "JEOPARDY" followed by a list of faces using the edge will be printed.

6.1.12 LIST ENTIRE SURFACE MODULE

SUR>>   Q

The Q function creates a complete listing of the module, including file name, descriptive text, statistics, priority nodes, command lists, entry pointers, edges, and faces.

6.1.13 LIST EDGE, FACE, INSTRUCTION AND NODE

SUR>>   L

With the list function the user can list edge definition, face definitions, command list instructions, and priority tree nodes in the active surface module. The prompts ITEM #1= and ITEM #2= appears; these request that the user specify the first and last item of the same type to be listed.

E1 thru E64 for edges
F1 thru F64 for faces
I1 thru I15 for command list instructions
N1 thru N5 for priority tree nodes

6.1.14 WRITE SURFACE MODULE

SUR>> W

After the user defines a surface module, it is written on cassette tape or floppy disk by invoking the write function. Following the selection of the write function, the following message appears.

FILE NAME=

The user reply depends on the type of device that the file is to be read from, as follows:

CS1: or CS2:	-Cassette tape drive (foreground mode only)
FD0: or FD1:	-Floppy disk drive (single-sided)
FT0: or FT1:	-Floppy disk drive (double-sided, top)
FB0: or FB1:	-Floppy disk drive (double-sided, bottom)

The device name is to be followed immediately by a file name, from one to six characters in length. If output is to cassette tape, the user is expected to have rewound the tape to load point.

If multiple output files are to be put onto a cassette tape, they must be written sequentially without rewinding the tape. To add modules to the end of a cassette tape that already contains modules, the user must first read past the existing modules to position the tape before outputting a new module to the tape.

When the output is to be floppy disk, the file name can be specified without the device name and the system will assume FD0: as the output device.

If a standard SP1 surface module is being updated to SP1/TEXTURE format, and the filename to be written is identical to the SP1 file originally read, the program displays the message "FILE SIZE TOO SMALL--USE DIFFERENT FILE NAME." The user specifies a different file name. This prevents writing the new larger 17-sector SP1/TEXTURE surface module file over the standard size 16-sector file, and possibly destroying data.

6.1.15 BLANK SCREEN

SUR>> B

The blank screen function erases the screen of the Tektronix display. If the system does not include the Tektronix hardware, the command is ignored.

6.1.16 DISPLAY FACES/EDGES

SUR>> D

With the display function the user can specify the faces or edges in the active surface module that are to be displayed. This applies only to systems containing the Tektronix display. After the function has been selected, the modeling system requests the items to be displayed by outputting the message ITEM #1= and ITEM #2=. The user responds to these prompts with the name of the first and last face or edge to be displayed. If the system does not contain a Tektronix display, this command is ignored. The view position must be set first.

6.1.17 VIEW POSITION

SUR>> V

The set view position function defines the viewing window for displaying faces or edges on the Tektronix display. To define the viewing window, the user responds to the prompt X= and Y=; the response determines the X and Y ground position. He then responds to the prompt SPAN= to indicate the ground span to be covered by the long axis of the display. If faces or edges have previously been displayed from the active module, the screen is erased and the faces or edges displayed using the new viewing window. For systems without a Tektronix display, this command is ignored.

6.1.18 FIND FACE

SUR>> F

With the find function, the user can locate faces or edges on the screen. This applies only to systems that have a Tektronix display. To use the find function, the user positions the crosshairs displayed on the screen over the face in question. The prompt POS CROSS> appears and the user enters any printing character except an X. The modeling system then prints out the position of the crosshair and compare the crosshair position to all of the displayed faces in the active surface module and output, in listing format, any faces that surround the crosshair position. This process may be repeated until the character X is entered in response to POS CROSS> and the system then returns to the SUR>> prompt. For systems that do not have a Tektronix display, this command is ignored.

6.1.19 ANGLE ROTATION FOR EDGES AND FACES

SUR>> A

By selecting the angle rotation function the user can rotate all of the defined edges and faces in the active surface modules by a specified angle. The prompt ANG= appears; the user enters the rotation angle in degrees.

6.1.20 TRANSLATION FOR EDGES AND FACES

```
SUR>>    T
```

With the translate function the user can translate all of the defined edges in the active surface module. The prompts X=, Y=, and Z= appear; the user's response to these prompts defines the translation vector.

NOTE:

Generally the user will want to use the rotate function first and then the translate function since both are relative to model origin.

6.1.21 PRINT STATISTICS

```
SUR>>    N
```

When the user selects the print statistics functions, statistical information concerning the edges and faces defined in the active surface module is output. Information is printed out as shown in the following example:

```
SUR>> N

EDGES = 64  FACES = 42
FEATURE DECODE BUFFER ENTRIES = 32

SUR >>
```

6.1.22 DISPLAY FEATURE DECODE BUFFER ENTRIES

```
SUR>>    Y
```

By entering a Y the user can examine the existing combinations of intensity and texture information in the texture feature decode buffer. For a module with three feature decode buffer entries, the following listing might be displayed:

```
FEATURE DECODE BUFFER ENTRIES:

#1  INT = 35  LLL=N  GND=Y  3D=N  OCC=N  TXT=Y  MAP#=2  LADJ=1/2  CR=3
#2  INT = 22  LLL=N  GND=N  3D=Y  OCC=Y  TXT=N
#3  INT = 22  LLL=Y  GND=Y  3D=N  OCC=N  TXT=Y  MAP#=1  LADJ=1/8  CR=3
```

6.1.23 DEFINE TEXTURE CLOUD MANAGEMENT

```
SUR>>   U
```

Alternate horizon intensity and texture values allow texture cloud management (TCM) in the realtime program. When TCM is active and the aircraft is below the clouds, the sky horizon plane uses alternate intensity and texture values to simulate a textured cloud bottom. When TCM is active and the aircraft is above the clouds, the ground horizon plane uses its alternate intensity and texture values to simulate the top of the clouds.

To define the alternate intensity and texture values that are used for texture cloud management, the sky horizon plane must already be defined as the first face under entry point P8, and the ground horizon plane must already be defined as the second face under entry point P8. This may be accomplished using the SUR>>I (insert) and SUR>>P (define entry point) commands before the SUR>>U command. Inserting cloud texture management data may also require one or two additional feature decode buffer entries. If the feature decode buffer is full, then neither of the alternate values is accepted. When the alternate values are not accepted, then the clouds are not textured.

If the sky horizon plane and ground horizon plane have already been defined under entry point P8, the prompt 'TEXTURE CLOUD MANAGEMENT? <Y,N>=' is displayed. A response of 'N' deletes any existing cloud texture management data and the user is returned to the surface module prompt. If the user responds with 'Y', the program prompts for cloud bottom and cloud top intensities, texture map numbers, lambda adjusters, and contrast ratios. For both cloud bottom and cloud top data, the program displays the prompts:

"INTENSITY=n <0=black 63=white> n=", to which the user should enter an integer between 0 and 63, where 0=black and 63=white.

"TEXTURE MAP #n <n=1,2,3,4> n=" to which the user should enter an integer between 1 and 4, selecting the texture map to be applied to this face. The texture map numbers correspond to the texture map numbers in the group module.

"CONTRAST RATIO=n <1=LO, 2=MED, 3=HI> n=", to which the user should enter an integer between 1 and 3, where 1 = low, 2 = medium and 3 = high contrast, respectively.

"LAMBDA ADJUSTER = 1/n <n=1,2,4,8> n=" to which the user should enter 1, 2, 4 or 8, denoting a lambda adjuster of 1/1, 1/2, 1/4 or 1/8, respectively. The lambda adjuster allows the map size (lambda) to be used at the same value specified in the group module, or reduced by a factor of 2, 4 or 8 for this face.

If the feature decode buffer is full, the program displays the message:

```
        "FEATURE DECODE BUFFER FULL"
        "Texture cloud management data NOT accepted"
```

If the sky horizon plane and ground horizon plane are not defined under entry point P8, the program displays the message:

"Face under P8 is NOT a horizon"

If entry point P8 is not defined, the program displays the message:

"NO ITEM"

6.1.24 JOIN SURFACE MODULES

```
SUR>>  J
```

The join command merges the specified module file with the current contents of the surface module in memory. The prompt FILENAME: requires a file name to be entered.

Only edge and face data is retained during the merge operation. Mirror point vector data, center line management data, entry points, priority trees, texture cloud management data, and command list instructions are removed from the module in memory as they are merged.

The join operation occurs only if:

1) The sum of the number of edges in memory and the number of edges in the file does not exceed the maximum number of edges allowed, 64.

2) The sum of the number of faces in memory and the number of faces in the file does not exceed the maximum number of faces allowed, 64.

3) The sum of the number of feature decode buffer entries used by the module in memory and the number of feature decode buffer entries used by the file does not exceed the maximum number of feature decode buffer entries, 32.

If all three conditions are met, the modules are joined, the new statistics for the module are displayed, and the SUR>> prompt is displayed.

If condition (1) or (2) above is not met, the corresponding error message is displayed, the module in memory is unchanged, and the SUR>> prompt is displayed.

For condition (3), if not enough feature decode buffer entries are available, the program will first attempt to make more buffer entries available by deleting texture cloud management (TCM) definitions. If sufficient buffer space is available after deleting the TCM data, the modules are joined, the new module statistics displayed, and the SUR>> prompt is displayed. If buffer space is not available, the modules are not joined, with the texture cloud management definition removed from the module in memory.

SECTION 7

ENVIRONMENT MODULE FUNCTIONS

7.1 INTRODUCTION

The environment module provides all of the information required to integrate the light and surface modules that define a model. An environment module contains a module list which defines the required light and surface modules, a command list to control processing, a priority list, a command list to control order, and several other lists associated with the global definitions of the model.

Environment modules are defined and manipulated with the following functions. To select a function, type the appropriate character in response to the prompt ENV>>.

A	Rotate Environment
B	ADBM boundary
C	Change module, submodel, instruction, node, runway offsets, routed DCS, converging DCS, Way segments, and LAT/LON references
D	Dynamic coordinate system map
E	Edit environment module
H	Insert module description
I	Insert modules, submodel, instruction, node, runway offsets, routed DCS, converging DCS, Way segments, and LAT/LON references
K	Kill modules, submodel, instruction, node, runway offsets routed DCS, converging DCS, Way segments, and LAT/LON references
L	List modules, submodel, instruction, node, runway offsets routed DCS, converging DCS, Way segments, and LAT/LON references
Q	List entire environment module
R	Read environment module
T	Translate environment module
W	Write module
X	Edit to module selection
Z	Zero environment module

Each of these functions is discussed separately on the following pages.

7.1.1 EDIT ENVIRONMENT MODULE

ENV>> E

By selecting the edit module function the user can declare the current module in memory to be active without destroying the data. This function is used after a program abort or module write function. With it the user can gain access to the current environment module data. The system assumes that the user is enabling access to a valid environment module data.

7.1.2 READ ENVIRONMENT MODULE

ENV>> R

Previously generated environment modules are read into the modeling system using the read function. When the user selects the read function, the following prompt appears:

FILE NAME=

The user reply depends on the type of device that the file is to be read from, as follows:

CS1: or CS2:	-Cassette tape drive (foreground mode only)
FDO: or FD1:	-Floppy disk drive (single-sided)
FTO: or FT1:	-Floppy disk drive (double-sided, top)
FBO: or FB1:	-Floppy disk drive (double-sided, bottom)

The device name is to be followed immediately by a file name, from one to six characters in length. If input is from cassette tape, the user is expected to have rewound the tape. If a file name is entered and the device name is omitted, FDO: is assumed. If an active module currently exists, the system asks for verification of the read command by displaying VER>, to which the user must respond with either "Y" to complete the read command or "N" to abort it.

7.1.3 ZERO ENVIRONMENT MODULE

ENV>> Z

The zero function, which is the first command executed when defining a new environment module, clears all of the data in the environment module and resets all the pointer structures. If an active module currently exists, the system asks for verification of the zero command by displaying VER>; the user must respond with either "Y" to complete the zero command or "N" to abort it.

7.1.4 EXIT

<u>ENV>></u> X

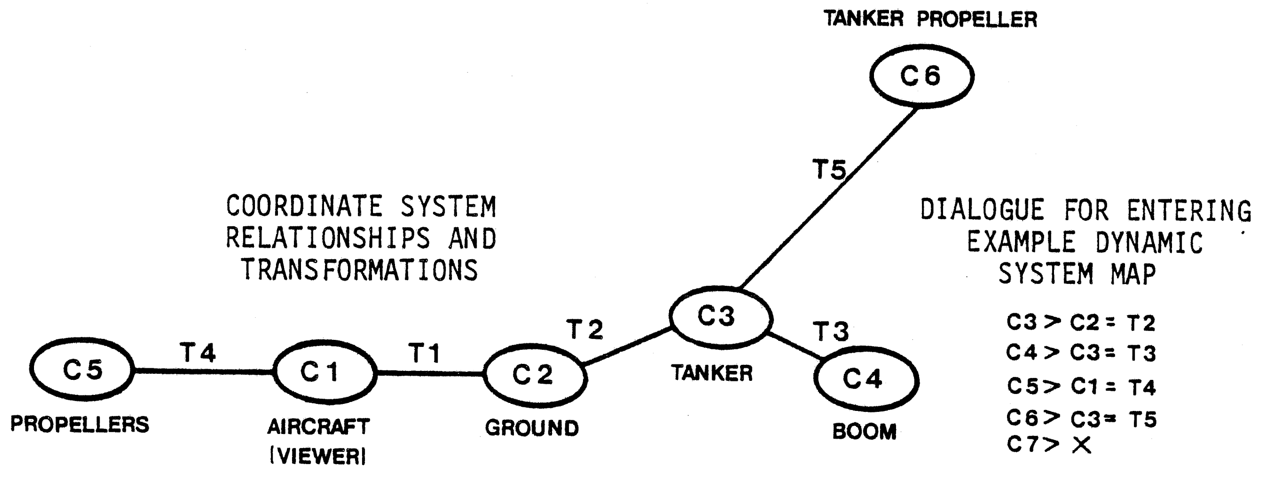
When the user has completed editing the environment module, he uses the exit function to select the editing mode for another type of module. If an active module still exists when the exit function is selected, the system responds with the message <u>ACT MOD VER></u>; the user must reply with either a "Y" to exit or a "N" to abort the command.

7.1.5 DEFINE DYNAMIC COORDINATE SYSTEM MAP

<u>ENV>></u> D

The dynamic coordinate system map defines the relationship between different coordinate systems and the required transformations. To make an object move independently of the viewer aircraft, the user must define all lights and surfaces in the object relative to a different coordinate system. By defining a translation vector and rotational matrix in the command list the user can move this coordinate system dynamically with respect to the first.

Because there may be more than just the predefined coordinate systems C1 and C2 (aircraft and ground), the user must map the relation between them. For example, consider the case of a refueling tanker with a movable boom docking with an aircraft with visible moving propellers. The following map must be entered as shown in Figure 7-1:



910062-0P1

FIGURE 7-1
DCS MAP EXAMPLE

The map says that the translation vector must be formed and the rotational matrices concatenated so that: 1) the propellers move relative to the aircraft (viewer), 2) the ground moves relative to the aircraft, 3) the tanker moves relative to the ground, and 4) the boom and propeller move relative to the tanker. The character X indicates the end of the map. It returns the user to the command selection prompt.

Coordinate system C1 is predefined to be the aircraft and coordinate system, and C2 is predefined to be the ground; they are associated with transformation T1. The system assumes this relationship; the user does not need to enter it.

7.1.6 INSERT MODULE DESCRIPTION

```
    ENV>>    H
```

Each module can carry a maximum of 29 characters of descriptive text. In response to the selection of the H function, the system offers the prompt:

```
            TEXT:
```

after which the user may enter or change the text which is printed at the top of every complete module listing.

7.1.7 INSERT ITEM

```
    ENV>>    I
```

With the insert function for environments the user can define module lists, runway offsets, command list instructions, and priority tree nodes for the active environment module. After the user selects the insert function, the prompt ITEM #= appears.

The user selects the data type and number to be entered by responding with one of the following:

M1	thru M11	to define	Modules
R1	thru R4	to define	Runway Offsets
N1	thru N6	to define	Priority Tree Nodes
I1	thru I64	to define	Command List Instructions
L1	thru L3	to define	Routed DCS
P1	thru P3	to define	Tracking DCS
W1	thru W11	to define	Way-Segments
G1	thru G4	to define	Lat-Lon references

7.1.7.1 INSERT COMMAND LIST INSTRUCTIONS

ITEM #= Ix

Respond to the ITEM #= prompt with I1 through I64 to define a command list instruction.

The environment command list directs the processing of all of the lights and surfaces in a model. It has three sections which must appear in the following order: 1) faces, 2) edges, 3) lights.

The face section contains the LINK-face commands to all of the surface module face command lists. A TVEC instruction specifies the proper translation transformation defined in the dynamic coordinate map (T1-T6). A TVEC instruction must precede each collection of LINKs to modules with the same coordinate system.

A PVEC command followed by SCHN, GCS, and TXTR commands initiates the edge section. These four commands set up the viewer perspective for the edges that follow. For every LINK-face in the preceding section, there must be a LINK-edge in the current section. Moreover, each collection with the same dynamic coordinate system must be preceded by the proper MTX and TVEC commands. The edge section is always concluded by a DRAW instruction which sends down raster parameter information followed by a STRT instruction which starts drawing the raster.

The light section contains all of the LITE commands (analogous to the EDGE and FACE commands) which specify which lights are to be displayed. The section is initiated by the LCHN command. If a collection of lights is marked for special handling (for example, it might be a star fog function), then the instruction directing that handling (SFOG in this case) must precede the particular LITE call in the list.

The environment command list must be concluded with an END instruction. After the user specifies the instruction to be inserted by responding to the prompt ITEM #= with I1 through I64, the prompt INST= appears. This prompt requests the command to be inserted. User responses to this prompt are the command strings defined in Table 7-1. This table also provides a brief description of each command, any subsequent information requested for that command, and the appropriate responses to such subsequent requests. The listing example for command lists in Figure 7-2 below shows a typical command list.

```
#I1   GCS :   OPT=01
#I2   LINK:   MOD=M1    TYPE= F
#13   LINK:   MOD=M5    TYPE= F
#I4   LINK:   MOD=M6    TYPE= F
#I5   LINK:   MOD=M7    TYPE= F
#I6   PVEC:
#I7   SCHN:
#I8   GCS :   OPT=01
#I10  TXTR
#I11  LINK:   MOD=M1    TYPE= E
#I12  LINK:   MOD=M5    TYPE= E
#I13  LINK:   MOD=M6    TYPE= E
#I14  LINK:   MOD=M7    TYPE= E
#I15  DRAW:
#I16  STRT:
#I17  LCHN:
#I18  GCS : . OPT=01
#I19  FOG :   MOD=M8    ENTRY=P1
#I20  LITE:   MOD=M8    ENTRY=P1
#I21  FOG :   MOD=M9    ENTRY=P1
#I22  LITE:   MOD=M9    ENTRY=P1
#I23  FOG :   MOD=M10   ENTRY=P1
#I24  LITE:   MOD=M10   ENTRY=P1
#I25  FOG :   MOD=M11   ENTRY=P1
#I26  LITE:   MOD=M11   ENTRY=P1
#I27  SFOG:
#I28  STAR:
#I29  END :
```

FIGURE 7-2
COMMAND LIST PRINTOUT SAMPLE

TABLE 7-1 (PART A)
ENVIRONMENT MODULE COMMAND LIST INSTRUCTIONS

COMMAND	ADDITIONAL PROMPT	RESPONSE	FUNCTION
PVEC			Loads the P-vector into the matrix multiplier. The P-vector defines the raster parameters for the display.
SCHN			Concatenates the surface channel matrix for each channel to the P-vector. The matrix defines the orientation for each specific channel.
LCHN			Loads the light channel matrix.
MTX	MTX=	T1 thru T7	Concatenates the transformation matrix specified to the current transformation matrix. MTX T1 is equivalent to the GCS 01 command and includes both an aircraft matrix and a T-vector.
TVEC	MTX=	T1 thru T7	Loads the specified T-vector.
STAR			Includes the star data base lights, which are contained in a file in the "FLY" program.
SFOG			Invokes special star fog which fogs out all stars at the same time.
TXTR			Loads the texture blocks.

TABLE 7-1 (PART B)
ENVIRONMENT MODULE COMMAND LIST INSTRUCTIONS

COMMAND	ADDITIONAL PROMPT	RESPONSE	FUNCTION
DRAW			Loads the raster control information into the hardware. This command follows the references to edge and face data but is not required if no faces are to be processed.
GCS	OPT=	01,02,03,04	Corresponds to the ground coordinate system being used. It replaces a TVEC and MTX for ground coordinate systems.
STRT			Starts the display processor running. This command follows the DRAW command but precedes light string processing.
LINK	MOD= EDGE,FAC>	M1 thru M11 E,F	Processes the command list in the referenced surface module. Note that both the edge and face command lists in the surface module must be referenced in the environment module command list. Note that both the edge and face command lists in the surface module must be referenced in the environment module command list.
FOG	MOD= ENTRY=	M1 thru M11 P1 thru P8	Does fog processing on the light strings grouped by the specified entry point in the referenced light module.
LITE	MOD= ENTRY=	M1 thru M11 P1 thru P8	Does fog processing on the light strings grouped by the specified entry point in the referenced light module.
END			Defines the end of the command list.

7.1.7.2 INSERT MODULE LIST

ITEM #= Mx

Respond to the ITEM #= prompt with M1 through M11 to define the module to be inserted in the environment module list.

The module list in the environment module provides the association between the environment module, surface modules, and light modules that make up the SP1/T model. After the user specifies the module to be defined, the prompt LIT,SUR>> appears. These prompts request the type of module being defined. For light modules, the user replies with an L. For surface modules, the user replies with an S. After the user specifies the module type, the prompt FILE NAME= appears. The user responds with the name of the module being referenced. The user inputs the appropriate one-to-six character file name.

For submodel switching, module list entries M1 through M4 are assumed to reference the four switchable submodels. Therefore, if a model does not require submodel switching, M1 through M4 are defined to be the same surface module. By doing this, system crashes are prevented if a submodel switch is invoked accidentally.

All surface modules must have the same number of faces. If they have a different number of faces, then the larger modules may be truncated or the smaller modules may be filled with garbage by the FLY program. Smaller modules can be padded with dummy faces to equal the number of faces in the largest module. A dummy face can be modeled by specifying the same edge four times and alternating the sign of that edge. For example: E12, -E12, E12, -E12. Edge number used is not important except that it is present in the surface module.

7.1.7.3 INSERT RUNWAY OFFSET

ITEM #= R

Respond to the ITEM #= prompt with R1 through R4 to define the runway offsets.

The runway offset information provides location and heading for submodel swapping and center line management. After the user specifies the runway offset to be defined, the prompts X=, Y=, and Z= appear. The user responds with the center location of the runway and the heading. See Figure 7-10 for a sample printout.

NOTE

The X, Y position coordinates control switching for the switchable modules, while HDG= controls direction of increment of the centerline.

7.1.7.4 INSERT PRIORITY TREE NODE

ITEM #= Nx

Respond to the ITEM #= prompt with N1 through N6 to define a priority tree node.

By defining priority tree nodes the user can modify the default priority order for the surfaces that are contained in the model. The priority tree in the environment module references entire surface modules and/or subtrees in particular surface modules. After the user specifies the priority tree node to be defined, the prompt PLANE TEST> appears; the user responds with a N if no separating plane is defined for this node, or a Y if a separating plane is defined. If the user is defining a separating plane for this node, then the prompt NEW PLANE> appears. The user responds with a Y to define a new separating plane or a N if the existing plane is to be used. When defining a separating plane, the system requests three vertices to define the plane. These vertices are input in response to the prompts X=, Y=, and Z= which is repeated three times, once for each vertex. Again use the left hand rule to determine which side of the plane is the true son side (side where thumb points).

If the user is defining priority tree node N1, where all priority tree processing begins, the prompt CS= appears. The response to this prompt indicates the initial coordinate specification. The responses are C1 through C8.

The system then requests the user to specify the true son and false son branches for this tree node by the prompts TSON= and FSON=. User responses to each of these prompts are either another node name, N1 through N6, or a submodel option name, S1 through S4. The options named reference the seven surface modules in the following manner:

		M1
S1	=	M2
		M3
		M4
S2	=	M5
S3	=	M6
S4	=	M7

The first four surface modules are switchable submodels. The proper submodel is selected by the fly program depending on the last character of the file name, as follows:

R - RUNWAY MODULE. The aircraft must be below 200 feet and within 250 feet of the runway origin. When the aircraft is on the ground, then it must be within 125 feet of the runway origin.

T - TAXIWAY MODULE. The aircraft must be below 200 feet, and not within the box made by the runway module.

Z - TERRAIN MODULE. The aircraft must be above 200 feet, and not in the box made by the runway module. Module M3 is used when the aircraft is on the positive (+) side of the runway origin, and M4 is used when the aircraft is on the negative (-) side.

If the user specifies a submodel option name, the prompt <u>NODE=</u> appears. By responding to this prompt the user indicates the node number in the surface module where priority processing is to continue in the reference surface module.

For both the true and false son branches of the tree, the user also specifies the coordinate system in which faces are defined for that branch. The prompt <u>CS=</u> requests the coordinate system; the user replies with C1 through C8. A priority tree node printout sample is given in Figure 7-4.

7.1.7.5 INSERT ROUTED DCS BLOCK

<u>ITEM #=</u> Lx

Respond to the ITEM #= prompt with L1 through L3 to define a Routed DCS item.

By defining Routed DCS blocks the user can model the path of a previously defined model. Using the DCS blocks he can make the model move along the path relative to the coordinate system as defined in the Dynamic Coordinate System map.

After the user specifies the Routed DCS block to be inserted, the prompt <u>COORDINATE SYS =</u> appears; the user response to this prompt with C3 through C7 depending on the DCS Map Definition. After this, the user models a path which is determined by pointing to specific modeled Way-Segments. The message <u>SEGMENT POINTERS:</u> appears and is followed by a maximum of 10 prompts--<u>S0 ></u> through <u>S9 ></u>. The user responds to these prompts with W1 through W11, which are defined by Way-Segment insertion, or X; W1 through W11 indicate the modeled Way-Segments, and X indicates that no more Segment Pointers are required. The final prompt, <u>RECYCLE=</u>, requires a 'Y' or 'N' response. A 'Y' input signifies that the routed traffic resets and starts at the first segment again after reaching the end of the last segment.

A sample model of Routed DCS is given in Figure 7-5.

7.1.7.6 INSERT WAY-SEGMENT DEFINITION

<u>ITEM #=</u> Wx

Respond to the <u>ITEM #=</u> prompt with W1 through W11 to define a Way-Segment.

By defining a Way-Segment, in conjunction with a Routed DCS block, the user can define a path beginning at a start point in an environment and ending at an end point. Within that path, he can make the moving object change velocity, heading, pitch and roll.

ATTITUDE refers to heading, pitch, and roll. POSITION refers to X, Y, and Z (in respect to a change in the position, speed).

Following the <u>START</u>: message, the modeler is asked END-POINT SAME AS START?, to which he replies Y if the object is not to move positionally or N if it is to move.

NOTE

A helicopter rotor blade is modeled with the end equaling the start position, with only the attitude changing, (in particular, the heading).

Next, the modeler is asked to define the starting position of the path by responding to the prompts <u>X=</u>, <u>Y=</u>, and <u>Z=</u>. If end does not equal start, the prompt <u>SPEED=</u> appears; the response to this prompt is the starting speed, expressed as a positive integer. If end does equal start, the prompt <u>MOTION TIME=</u> appears; the response to this prompt indicates the time in seconds at which the object is traveling in its attitude at the position that has already been entered. The amount entered is a positive real number less than 1090.0.

Next, the prompts for the start attitude appear; they are <u>HDG=</u>, <u>PITCH=</u>, and <u>ROLL=</u>. If end does not equal start, then the user enters the end position. If end does equal start, then the prompts <u>END:</u> and <u>DELTA:</u> appear; the response to these prompts indicates the delta heading, pitch, and roll. The delta changes are the angular changes in degrees per frame. As an example, if he wants the rotor blade to rotate at 6 times a second, he enters a delta heading of 72 degrees. That corresponds to 72 degrees per frame @30 frames per second.

If an overflow error occurs, the appropriate message is displayed. An overflow error indicates that the acceleration or velocity value is too high.

If a change is made to the way-segment, two additional questions are asked: <u>CHANGE POSITION?</u> and <u>CHANGE ATTITUDE?</u>. If the response to either question is yes, only the appropriate information is required for the change.

If the modeler inserts consecutive way-segments, he can tab over the start information. The beginning information of the second way-segment then corresponds to the end information of the last segment listed or to the end information of the last changed or inserted segment. If the modeler's last operation was a change on segment W2, and he then inserts segment W5, and tabs over the start position and attitude, the values correspond to the end position and attitude of segment W2. This helps the modeler to create contiguous segments by changing the segment which precedes the next segment in the path with N and N responses to the change questions, and tabbing over the start position and attitude.

If the modeler sets the recycle bit on the routed path L1 thru L3 and the end position and/or attitude of the end segment do not match up very closely with the starting segment's start position, a jump in the position and/or attitude may be noticeable.

A sample model of Way-Segments is given in Figure 7-6.

TABLE 7-2
HEADING (ANGLE OF IMPACT)

PROPORTIONALITY CONSTANT K (SPEED)

	1/8	1/4	1/2	3/4	1	1.25	1.5	2	2.5	3	4
0.	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5.	49.2	25.4	15.0	11.7	10.0	9.0	8.3	7.5	7.0	6.7	6.2
10.	****	54.0	30.3	23.4	20.0	18.0	16.6	15.0	14.0	13.3	12.5
15.	****	****	46.2	35.2	30.0	26.9	24.9	22.4	20.9	19.9	18.7
20.	****	****	63.2	47.1	40.0	35.9	33.2	29.8	27.9	26.5	24.9
25.	****	****	82.7	59.3	50.0	44.8	41.4	37.2	34.7	33.1	31.1
30.	****	****	119.6	71.8	60.0	53.6	49.5	44.5	41.5	39.6	37.2
35.	****	****	****	84.9	70.0	62.3	57.5	51.7	48.3	46.0	43.2
40.	****	****	****	99.0	80.0	70.9	65.4	58.7	54.9	52.4	49.2
45.	****	****	****	115.5	90.0	79.5	73.1	65.7	61.4	58.6	55.2
50.	****	****	****	****	100.0	87.8	80.7	72.5	67.8	64.8	61.0
55.	****	****	****	****	110.0	95.9	88.1	79.2	74.1	70.8	66.8
60.	****	****	****	****	120.0	103.8	95.3	85.7	80.3	76.8	72.5
65.	****	****	****	****	130.0	111.5	102.2	91.9	86.3	82.6	78.1
70.	****	****	****	****	140.0	118.7	108.8	98.0	92.1	88.3	83.6
75.	****	****	****	****	150.0	125.6	115.1	103.9	97.7	93.8	89.0
80.	****	****	****	****	160.0	132.0	121.0	109.5	103.2	99.2	94.2
85.	****	****	****	****	170.0	137.8	126.6	114.9	108.5	104.4	99.4
90.	****	****	****	****	180.0	143.1	131.8	120.0	113.6	109.5	104.5

7.1.7.7 INSERT CONVERGING DCS BLOCK

ITEM #= P1

Respond to the ITEM #= prompt with a P1 through P3 to define a Converging DCS block.

By defining Converging DCS Blocks the user can model various parameters relative to a Dynamic Coordinate System previously defined in the DCS Map. With this the user can track the aircraft position relative to the aircraft.

After the user specifies which Converging DCS block is to be inserted, he is prompted with IMPACT TIME= and VEER TIME=. IMPACT TIME is the time in seconds, from the point in time in which the tracking model is introduced until it makes contact with the aircraft. VEER TIME, also modeled in seconds, is impact time until the converging model stops tracking.

Next, the message APPROACH: appears; and the prompts HDG=, SLOPE=, and ROLL=, appear. They are all modeled in degrees and they define the tracking model's position relative to the aircraft. Next, the program prompts the user with SPEED=, which is a proportion of the speed of the viewer.

The final prompt, COORDINATE SYS=, requires an input of C3 through C6, depending on the tracking model's definition in the Dynamic Coordinate System Map.

A sample model of Converging DCS blocks is given in Figure 7-7.

NOTE

The approach heading represents the angle of impact (see Figure 7-8). The angle of view can be looked up in Table 7-2. To find the impact angle knowing the view angle, find the row corresponding to the view angle. Find the column corresponding to the speed (the proportion of the viewer's speed) to be entered. At the point of intersection of the speed column and view angle row is the impact angle.

NOTE

Pitch = Slope

7.1.7.8 INSERT LAT/LON REFERENCE DEFINITION

ITEM #= Gx

Respond to the ITEM #= prompt with G1 through G4 to define a Lat/Lon reference.

The user can define a maximum of four different latitude/longitude references. The latitude is requested first; the prompt N or S: appears. The response to this prompt is 'N', meaning north latitude, or 'S', meaning south latitude. Following this response are the prompts for degrees, minutes and seconds. When the prompt DEG= appears, the user responds with an integer value between 0 and 90. If the response is 90, then there is no prompt for minutes and seconds. When the prompt MIN= appears, the user responds with an integer value between 0 and 59. Finally, when the prompt SEC= appears, he responds with a real number between 0 and 59.9999.

Longitude, whose prompts are similar to latitude, appears next. The prompt E or W: requires a response of 'E or 'W'. When the DEG= prompt appears, the user response is an integer value between 0 and 180. When the response is 180, there are no prompts for minutes and seconds. Otherwise, respond to MIN= and SEC= using the same instructions given under latitude.

Next, the user is prompted with ANGLE=, to which he enters the correct orientation angle. See Figure 7-9 for a sample printout.

NOTE

If more than one latitude/longitude reference is defined in an environment module, then no DCS's may be defined.

7.1.8 CHANGE ITEM

ENV>> C

With the change function the user can modify the definition of an existing module, runway offset, command list instruction, priority tree node, routed DCS, Way-Segment, or tracking DCS in the active environment module. After the user selects the change function, the prompt ITEM #= appear. At this point the user proceeds as if he were performing an insert. For the part of the definition that remains unchanged, answer the prompt with a tab. When a particular entry is to be changed, enter the new value. After the user has changed a part of an item, remaining prompts can be skipped by typing the ESCape or Alt-Mode key.

7.1.9 KILL ITEM

ENV>>   K

With the kill function the user can delete items from the active environment module. After the user selects the kill function, the system requests the user to specify the first and last item of the same type to be deleted in response to the prompts ITEM #1= and ITEM #2=. If the item number for the first item is greater than the second item, nothing is deleted and the command is ignored. Appropriate responses to the prompts are:

M1 thru M11 for Module Names
R1 thru R4 for Runway Offsets
I1 thru I64 for Command List Instructions
N1 thru N6 for Priority Tree Nodes
L1 thru L3 for Routed DCS
W1 thru W11 for Way-Segments
P1 thru P3 for Tracking DCS
G1 thru G4 for Lat/Lon References

7.1.10 LIST ITEM

ENV>>   L

With the list function the user can list module names, runway offsets, command list instructions, priority tree nodes, etc. in the active environment module. The user is asked to specify the first and last item of the same type to be listed in response to the prompts ITEM #1= and ITEM #2=. Appropriate responses to the prompts are:

Instructions

M1 thru M11 for Module Names
R1 thru R4 for Runway Offsets
I1 thru I64 for Command List
N1 thru N6 for Priority Nodes
L1 thru L3 for Routed DCS
W1 thru W11 for Way-Segments
P1 thru P3 for Tracking DCS
G1 thru G4 for Lat/Lon References

The following examples show the listing format for each data item:

```
#M1      TYPE= S      FILE NAME=FDO:HKGS1R
#M2      TYPE= S      FILE NAME=FDO:HKGS1R
#M3      TYPE= S      FILE NAME=FDO:HKGS1R
#M4      TYPE= S      FILE NAME=FDO:HKGS4Z
#M5      TYPE= S      FILE NAME=FDO:HKGS5T
#M6      TYPE= S      FILE NAME=FDO:HKGS6Z
#M7      TYPE= S      FILE NAME=FDO:HKGS3Z
#M8      TYPE= L      FILE NAME=FDO:HKGL1A
#M9      TYPE= L      FILE NAME=FDO:HKGL2A
#M10     TYPE= L      FILE NAME=FDO:HKGL3A
#M11     TYPE= L      FILE NAME=FDO:HKGL4A
```

FIGURE 7-3
MODULE LIST SAMPLE PRINTOUT

```
#N1    PLANE TEST >  N
   FTR=  N1  CS= C2
   TSON= N2  CS= C2
   FSON= S1  NODE= N2  CS= C2

#N2    PLANE TEST >  N
   FTR=  N1
   TSON= N5  CS= C2
   FSON= N3  CS= C2

#N3    PLANE TEST >  Y
   0.0000  X+  0.9960  Y+  0.0000  Z = 100.9960
   FTR=  N2
   TSON= N4  CS= C2
   FSON= S1  NODE= N1  CS= C2

#N4    PLANE TEST >  Y
   -1.0000 X+  0.0000  Y+  0.0000  Z = -399.9648
   FTR=  N3
   TSON= S2  NODE= N1  CS= C2
   FSON= S4  NODE= N1  CS= C2

#N5    PLANE TEST >  N
   FTR=  N2
   TSON= S1  NODE= N3  CS= C2
   FSON= S4  NODE= N2  CS= C2
```

FIGURE 7-4
PRIORITY NODES PRINTOUT SAMPLE

```
#L1  COORDINATE SYS.=  C3
   RECYCLE = N
   SEGMENT POINTERS:
         S0    >  W1
         S1    >  W2
         S2    >  W3
         S3    >  W4
#L2  COORDINATE SYS.=  C3
   RECYCLE = Y
   SEGMENT POINTERS:
         S0    >  W3
         S1    >  W4
         S2    >  W2
         S3    >  W1
```

FIGURE 7-5
ROUTED DCS SAMPLE

```
         #W1  MOTION TIME = 1000.0000
            START:                  END:
          X=  125.0000                 983.9609
          Y=  250.0000                 250.0000
          Z=  500.0000                 500.0000
 SPEED = 0                              0
   HDG=0.0000                         0.0000
 PITCH = 0.0000                         0.0000
 ROLL =  0.0000                         0.0000
 DELTA:    HDG=22.5000         PITCH = 45.0000    ROLL = 90.0000

   #W2  MOTION TIME = 9.0664
      START:                  END:
    X=  0.0000                   999.8125
    Y=  0.0000                   0.0000
    Z=  0.0000                   0.0000
SPEED = 100                       50
  HDG=0.0000                    0.0000
PITCH = 0.0000                    0.0000
ROLL =  0.0000                    0.0000
DELTA:    HDG=0.0000    PITCH = 0.0000    ROLL =  0.0000

   #W3  MOTION TIME = 18.2304
      START:                  END:
    X=  999.8125                 1999.7968
    Y=  0.0000                   0.0000
    Z=  0.0000                   0.0000
SPEED = 49                        24
  HDG=0.0000                    0.0000
PITCH = 0.0000                    9.0078
ROLL =  0.0000                    0.0000
DELTA:    HDG=0.0000    PITCH = 0.0117    ROLL = 0.0000
```

FIGURE 7-6
WAY-SEGMENTS SAMPLE

```
#P1  COORDINATE SYS. =  C3
   IMPACT TIME = 20 VEER TIME = 6
   APPROACH:
         HDG=14.9960  SLOPE = 350.0039   ROLL =  0.0000
         SPEED = 1.2500
```

FIGURE 7-7
TRACKING DCS SAMPLE

```
         X=   0.0000             Y=   0.0000
         DX=  100000.0000        DY=  100000.0000
         ANGLE=23.4453
```

FIGURE 7-8
DBM BOUNDING BOX SAMPLE PRINTOUT

```
         LAT/LON REFERENCES

         #GCS1
LATITUDE:      N
 DEG= 90      MIN= 0      SEC= 0.0000
LONGITUDE:     W
 DEG= 45      MIN= 45     SEC= 45.4570
ELEV=    5280.0000           ANGLE=270.0000

         #GCS2
LATITUDE:      S
 DEG= 22      MIN= 30     SEC= 0.0000
LONGITUDE:     E
 DEG= 105     MIN= 37     SEC= 26.0000
ELEV=     1234.5664        ANGLE=0.0000
```

FIGURE 7-9
LAT/LON REFERENCES SAMPLE PRINTOUT

```
         RUNWAYS
#R1           X=          Y=          HDG=
#R2           X=          Y=          HDG=
#R3           X=          Y=          HDG=
#R4           X=          Y=          HDG=
```

FIGURE 7-10
RUNWAY OFFSET SAMPLE PRINTOUT

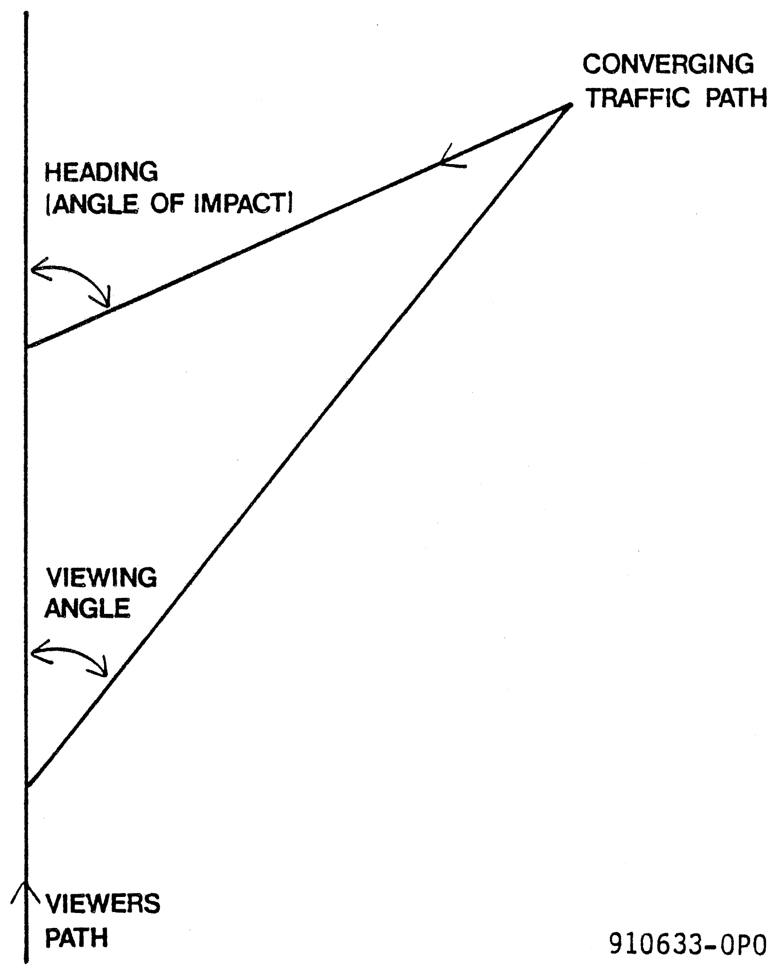


FIGURE 7-11
CONVERGING TRAFFIC PATH

7.1.11 WRITE ENVIRONMENT MODULE

ENV>> W

After an environment module is defined, it is written on cassette tape or floppy disk by invoking the write function. After the user selects the write function, the system requests the output file name by typing:

FILE NAME=

The user reply depends on the type of device that the file is to be read from, as follows:

CS1: or CS2:	-Cassette tape drive (foreground mode only)
FD0: or FD1:	-Floppy disk drive (single-sided)
FT0: or FT1:	-Floppy disk drive (double-sided, top)
FB0: or FB1:	-Floppy disk drive (double-sided, bottom)

The device name is to be followed immediately by a file name, which is from one to six characters in length.

If multiple output files are to be put onto cassette tape, they must be written sequentially without rewinding the tape. To add modules to the end of a cassette tape that already contains modules, the user must first read past the existing modules to position the tape before outputting a new module to the tape.

When the output is to be to floppy disk, the name can be specified without the device name and the system assumes FD0: as the output device.

7.1.12 LIST ENTIRE ENVIRONMENT MODULE

ENV>> Q

The Q function creates a complete listing of the environment module.

7.1.13 DEFINE DATA BASE MANAGEMENT BOUNDING BOX

ENV>> B

The DBM bounding box defines the environment's location, size, and orientation. The location is requested by the prompts X= and Y=. The size is requested by DX= and DY=. The orientation is requested by ANGLE=.

7.1.14 ANGLE ROTATION FOR AN ENVIRONMENT

```
ENV>> A
```

Selection of the angle rotation function allows the user to rotate all of the defined items in the active environment module by a specified angle. The user is prompted by ANG=, to which he enters a rotation angle in degrees.

7.1.15 TRANSLATION FOR AN ENVIRONMENT

```
ENV>> T
```

Selection of the translation function allows the user to translate all of the defined items in the active environment module. The user defines the translation vector in response to the prompts X=, Y=, Z=.

NOTE

The LAT/LON information in G1-G4 is not rotated or translated by the above commands.

BLANK

SECTION 8

GROUP MODULE FUNCTIONS

8.1 INTRODUCTION

The group module file provides all of the information that the real-time fly program requires to associate model numbers from the host computer with active files from the disk. In addition, information required for data base management is defined for each model. When the real-time program is started, it requests the name of the group module that defines the models which are available for use. Using the information extracted from the group module, the real-time program checks for the existence of all the specified files, and sets up all of the appropriate internal tables required for operation.

The group module also contains information on how the texture patterns are to be modified and displayed on the faces in the model:

1. Texture map names. Four filenames are specified. (Other programs generate the texture maps and store them on floppy disks.) The group module contains the names of these files containing the texture maps, and the maps are moved from floppy disk to the texture memory by the real-time program.

2. Lambdas. Two lambdas or map sizes are specified and used for all maps. The ground lambda is applied to the maps when they appear on the ground, and the sky lambda is applied to the maps when they appear in the sky. The lambda value denotes the length of a side of the square texture map. The same map pattern is repeated to completely cover the face to which it is applied. Lambdas have a range of 64 feet to 65,536 feet. A lambda of 64 feet would have the effect of making the map pattern much smaller than it would be with a lambda of 10,000 feet. A lambda of 64 feet also means the map will be repeated every 64 feet.

3. Movement vectors. Two movement vectors can be specified--one is for maps on the ground and one is for maps in the sky. All four maps use these two movement vectors. If the vectors are other than zero, the map moves everywhere it is displayed. The range of the movement vector components is from -128 to 127.99 feet per frame.

4. Heights. Two heights can be specified--one for ground planes and one for sky planes. The same heights are used for all four maps. The range of the heights is from -32,768 to 32,767 feet for ground planes, 0 to 32,767 feet for sky planes.

5. Cloud management delta. The cloud management delta is used when texture cloud management is defined in the surface module and when clouds are active; it varies the height at which the textured cloud plane are displayed. The range of the delta is from -128 to 127.99 feet.

The group module is defined and manipulated with the following functions. To select a function, type the appropriate character in response to the GRP>> prompt:

C	Change group, lambda, height, movement vector, texture map name or cloud delta definition.
E	Enable editing of the group module.
H	Insert header information describing the group module.
I	Insert group, lambda, height, movement vector, texture map name or cloud delta definition.
K	Kill group, lambda, height, movement vector, texture map name or cloud delta definition.
L	List group, lambda, height, movement vector, texture map name or cloud delta definition.
Q	List entire group module, including texture information.
R	Read a group module.
W	Write the group module.
X	Exit group module to module selection prompt.
Y	List all texture information for this group module.
Z	Zero the group module.

8.1.1 EDIT GROUP MODULE

<u>GRP>></u> E

With the edit module function, the user can declare the current module in memory to be active without destroying the data. It is used after a program abort or a module write function to gain access to the group module data. The system assumes that the user is enabling access to valid group module data.

8.1.2 READ GROUP MODULE

<u>GRP>></u> R

Previously generated group modules are read into the modeling system using the read function. In response to the selection of the read function, the system will prompt the user by typing:

<u>FILE NAME=</u>

The user reply depends on the type of device that the file is to be read from, as follows:

CS1: or CS2: - Cassette tape drive (foreground mode only)

FD0: or FD1: - Floppy disk drive (single-sided)

FT0: or FT1: - Floppy disk drive (double-sided, top)

FB0: or FB1: - Floppy disk drive (double-sided, bottom)

The device name is to be followed immediately by a file name, from one to six characters in length. If input is from cassette tape, the user is expected to have rewound the tape. If a file name is entered and the device name is omitted, FD0: is assumed.

If an active module currently exists, the system asks for verification of the read command by displaying <u>VER></u>. The user must respond with either a "Y" to complete the read operation or "N" to abort it.

To convert an SP1 module to SP1/TEXTURE, the modeler uses the R command, specifying the name of the SP1 group module to be converted. The program displays the messages "CKS ERR" and "CONVERTED FROM SP1 to SP1/TEXTURE" as group, module, filename and search number data and the text header are read into memory, and all texture data items are set to zero. The insert command should be used to define the lambdas, movement vectors, delta, heights, and texture map names.

Because the SP1/TEXTURE group is one sector larger than the standard SP1 group module, the user specifies a different filename when the module is written to the output device. This prevents the possibility of destroying model data.

8.1.3 ZERO GROUP MODULE

```
GRP>>    Z
```

The zero function, which is the first command executed when defining a new group module, performs the following functions. If an active module currently exists, the system will ask for verification of the zero command by typing VER>; the user must respond with either "Y" to complete the zero command or "N" to abort it.

8.1.4 EXIT

```
GRP>>    X
```

After the user has completed group module editing, he uses the exit function to select the editing mode for another type of module. If an active module still exists when the exit function is selected, the system will respond with the message:

```
ACT MOD
VER>
```

The user must reply with either a "Y" to exit or a "N" to abort the command.

8.1.5 INSERT DEFINITION

```
GRP>>    I
```

After the user selects the insert function, the prompt ITEM#= will be output. The user specifies one of the following data types and numbers:

D1 -- Cloud management delta
G0 through G255 -- Group definition
H1 -- Texture map height
L1 -- Lambda
T1 through T4 -- Texture map file names
X1 -- X component, movement vector
Y1 -- Y component, movement vector

```
ITEM#= D1
```

The user should enter the value of the cloud management delta, between -128 and 127.99 feet.

ITEM#- G0 through G255 - group name

After the user selects the ITEM#= G0 thru G255, the prompt GRP #= appears. This prompt requests which group number is to be defined in the active group module. A response of G0 through G255 specifies the group number.

The prompt MODULE #= requests the module number to be defined; a response of M1 through M256 indicates the module number. Module numbers associate an order with the environment modules to be specified as belonging to this group. This order is used in considering various environment choices during data base management. For systems that do not have data base management, M1 is used regardless of the number of environment modules defined under a particular group.

After a module number has been selected, the prompt FILE NAME= requests the environment module name to be referenced. The reply must be a one-to six-character file name. Search #: prompts the user to provide the number of the first module, which might be encountered when crossing the ADBM boundary. See Figure 8-1 for an example. The search # for each environment is set to the lowest environment number which may be entered from the environment. For example in Figure 8-1 when the eyepoint is in ENV #11 the lowest numbered neighboring environment is M6. Therefore, from ENV#11 the search # should be 6.

ITEM#= H1

To the prompt GND/SKY?, the user types a G or an S, to specify whether this is the height for the ground plane or the height for the sky plane. To the prompt HEIGHT =, the user should enter -32,768 to 32,767 feet for ground planes, 0 to 32,767 feet for sky planes.

ITEM#= L1

To the prompt GND/SKY?, the user types a G or an S, to specify whether this is the lambda for the ground plane or the lambda for the sky plane. To the prompt LAMBDA =, the user should enter the value between 64 and 65,536 feet.

ITEM#= T1 through T4

The user enters a maximum of six characters representing the filename of the texture map. (Texture maps are not defined within the build program, they are generated and stored on disk using another program.) The numbers in the item names T1, T2, T3 and T4 correspond to the numbers called out by surfaces in the surface modules.

ITEM#= X1

To the prompt GND/SKY?, the user types a G or an S, to specify whether this is the x-vector for the ground plane or the x-vector for the sky plane. To the prompt X-VECTOR =, the user should enter a value between -128 and 127.99 feet per frame.

ITEM#= Y1

To the prompt GND/SKY?, the user types a G or an S, to specify whether this is the y-vector for the ground plane or the y-vector for the sky plane. To the prompt Y-VECTOR =, the user should enter a value between -128 and 127.99 feet per frame.

NOTE

Any X, Y, H, D or T entry not specified defaults to zero.

M1. S#=2	M2. S#=1	M3. S#=2	M4. S#=3
M5. S#=1	M6. S#=1	M7. S#=2	M8. S#=3
M9. S#=5	M10. S#=5	M11. S#=6	M12. S#=7
M13. S#=9	M14. S#=9	M15. S#=10	M16. S#=11
M17. S#=13	M18. S#=13	M19. S#=14	M20. S#=15

FIGURE 8-1
MODULES AND THEIR SEARCH NUMBERS

8.1.6 CHANGE DEFINITION

GRP>> C

With the change function the user can modify the definition of an existing item. After the user selects the change function, the prompt ITEM#= appears.

The modeler should respond with an item name:

D1	Texture cloud management delta
GO..G255	Group name
H1	Height
L1	Lambda
T1..T4	Texture map name
X1	X movement vector
Y1	Y movement vector

At this point, the user proceeds as if he were using the insert function. If there is no change to the particular entry, use a tab. If an entry is to be changed, enter the new value. If the item is not yet defined, the message "NO ITEM" is displayed.

8.1.7 KILL DEFINITION

GRP>> K

The kill function allows the user to delete items from the active group module. The program displays the prompt "ITEM#1 =". The user should responds with an item name:

D1	Texture cloud management delta
GO thru 255	Group name
H1	Height
L1	Lambda
T1 thru T4	Texture map name
X1	X movement vector
Y1	Y movement vector

If the user responds with D1, the delta is deleted.

If the user responds with H1, L1, X1, or Y1, the program then prompts GND/SKY? to which the user responds with G for the ground plane item or S for the sky plane item. The item indicated is then deleted.

If the user responds with ITEM#1 = T1 thru T4, the prompt ITEM#2= appears. The user responds with another texture map name data item, T1 thru T4. The texture map names specified are then deleted.

If the user selects the ITEM#1 = GO thru G255, the prompt GRP# appears to request the group number. After the user has entered the group number, the MOD #1= and MOD #2= appear. These prompts request the number of the first and last modules to be deleted.

8.1.8 LIST DEFINITION

```
GRP>>    L
```

The list function allows the user to list definitions in the active group module.

The program prompts for the item(s) to be listed:

```
ITEM #1=
```

The modeler responds with an item name:

D1	Texture cloud management delta
GO thru 255	Group name
H1	Height
L1	Lambda
T1 thru T4	Texture map name
X1	X movement vector
Y1	Y movement vector

If the modeler specifies GO thru G255, the program will prompt for the second group name:

```
GROUP #2=
```

The modeler responds with the last group number to be listed. The program lists all groups in the range between the first group number specified and the second group number specified. Figure 8-2 shows the listing format for group definitions:

```
#GO     MOD CNT=3
   # M1     FILENAME =  :ENV001   SEARCH # = 2

   # M2     FILENAME =  :ENV002   SEARCH # = 1

   # M3     FILENAME =  :ENV003   SEARCH # = 2

#G1     MOD CNT=1
   # M1     FILENAME =  :ENV101   SEARCH # = 1
```

FIGURE 8-2
GROUP MODULE SAMPLE PRINTOUT

If the modeler selects H1, L1, X1, or Y1, the program will prompt for the ground or sky:

```
GND OR SKY?
```

The modeler should enter the letter 'G' to select the ground, or 'S'; to select the sky.

If the modeler specifies the item D1, the program lists the texture cloud management delta.

If the modeler specifies the item T1 thru T4, the program prompts for the second item in the range to be listed:

```
ITEM #2 =
```

The modeler then selects a second T1 thru T4 item to complete the range. The following example shows a listing for an entry of ITEM #1 = T1 and ITEM #2 = T4:

```
#T1 = TXMAP1
#T2 = TXMAP2
#T3 = TXMAP3
#T4 = TXMAP4
```

NOTE

Lambdas which have not been inserted print out as ******, alerting the user that this item must be defined or the model will not work properly.

8.1.9 WRITE GROUP MODULE

```
GRP>>   W
```

After a group module is defined, it is written on cassette tape or floppy disk by invoking the write function. After the user selects the write function, the system requests the output file name by typing:

```
FILE NAME=
```

8.1.9 WRITE GROUP MODULE

GRP>> W

After a group module is defined, it is written on cassette tape or floppy disk by invoking the write function. After the user selects the write function, the system requests the output file name by typing:

FILE NAME=

The user reply depends on the type of device that the file is to be read from, as follows:

CS1: or CS2: - Cassette tape drive (foreground mode only)

FD0: or FD1: - Floppy disk drive (single-sided)

FT0: or FT1: - Floppy disk drive (double-sided, top)

FB0: or FB1: - Floppy disk drive (double-sided, bottom)

The device name is to be followed immediately by a file name, from one to six characters in length.

If multiple output files are to be put onto a cassette tape, they must be written sequentially without rewinding the tape. To add modules to the end of a cassette tape that already contains modules, the user must first read past the existing modules to position the tape before outputting a new module to the tape.

When the output is to be to floppy disk, the file name can be specified with the device name and the system will assume FD0: as the output device.

If the user is updating a standard SP1 file to SP1/TEXTURE format, a different filename must be specified than the file of the SP1 file originally read. This prevents the possibility of destroying model data.

8.1.10 LIST ENTIRE GROUP MODULE

GRP>> Q

The Q function gives a complete listing of the group module. Refer to Figure 8-2 for a sample listing.

8.1.11 INSERT MODULE DESCRIPTION

```
GRP>> H
```

This function is used to give a description of the group module. Up to 30 characters can be used for this. The prompt is TEXT:. At this point the user may wish to enter or change the text, which is printed at the top of every complete module listing.

8.1.12 LIST TEXTURE INFORMATION FOR GROUP MODULE

```
GRP>> Y
```

The Y command lists all texture-related information for the module: ground and sky lambdas, ground and sky heights, ground and sky x and y movement vectors, the cloud delta, and all four texture map names. The Y command is like the Q command, as it allows the user to inspect several different data items without specifying the individual item names. The Y command differs from the Q command in listing neither the group module text header nor the group and module data.

BLANK

SECTION 9

MODEL MERGE PROGRAM MERXxx

9.1 INTRODUCTION

The model merge program, MERXxx, allows the user to combine two light or surface modules into a single module. Merging of two modules results in the data in the second module specified being concatenated to the data in the first module. All of the control information such as entry points, command lists, and priority tree nodes are cleared and must be defined using the build program after the merge is complete.

9.2 EXECUTION

The prompt MER>> is output to the terminal signifying that the program is ready to execute. There are four options available in this program.

Z - Zero the input module
M - Merge a module with the current module
W - Write module out to a file
X - Exit the merge program

Each of these functions is described below.

9.2.1 ZERO COMMAND

```
MER>>   Z
```

The Zero command allows the user to start from scratch. When first entering the program this should be the first command executed. After merged modules have been written out to a file, the Zero command is used to start again from scratch.

9.2.2 MERGE COMMAND

```
MER>>   M
```

The Merge command merges a specified module file with the current contents of the merge module. The prompt FILENAME: requires a file name to be entered (the drive is optional, default is FDO:). If the previous command entered was a Zero command, then only the module specified will remain in the merge module after the function is complete. If other modules were merged previously, then the contents of the merge module will include the items in the module file specified when the function is complete. The statistics for the module will be output at the end of the function.

9.2.3 WRITE COMMAND

```
MER>>   W
```

The Write command outputs the contents of the merge module to the file specified by the user in response to the prompt FILENAME:. If the file already exists then the user will be asked to verify the function by responding to the prompt VER> with a Y or N.

9.2.4 EXIT COMMAND

```
MER>>   X
```

The Exit command simply exits the program.

Error messages for the merge program are described in Appendix A.

APPENDIX A

ERROR MESSAGES

A.1 INTRODUCTION

This appendix lists the error messages and interpretation for the SP1/T modeling system.

A.2 GENERAL ERROR MESSAGES - BUILD

SORRY	The user has requested a function that the modeling system software cannot provide.
?	The response is not an appropriate answer. Reenter the requested data.
BAD NAME	The name specified by the user is not legal in the context it has been used. Enter a new name.
MOD NOT RESIDENT	Editing functions are being issued but there is no active module. Issue a zero or read command.
INCORRECT MOD TYPE	The module type read by the modeling system is not correct. Read the module under the appropriate set of editing functions.

NO ITEM	An item was specified that does not exist. Specify a new item name.
NO ROOM	No more room remains in the active module to store information of the type being specified.
JEOPARDY:	This message is followed by a list of face names whose definitions are not necessarily correct because one or more of the edges referenced have been deleted.
VAL OVF	Valve overflow Command disk not in drive #1
STK OVF	Stack overflow
FEATURE DECODE BUFFER FULL TEXTURE CLOUD MANAGEMENT DATA NOT ACCEPTED	Texture cloud management data NOT accepted.
FILE SIZE TOO SMALL--USE A DIFFERENT FILE NAME	Surface and group module must be renamed during conversion from SP1 to SP1/T.
FEATURE DECODE BUFFER FULL-- FACE NOT ACCEPTED	Any face inserted after the feature decode buffer is full must access one of the existing 32 entries.
FACE UNDER P8 IS NOT A HORIZON	Texture cloud management not accepted.

A.3 GENERAL ERROR MESSAGES -- MERGE

TOO MANY FLASHING LIGHTS	The number of flashing light strings contained in the two modules being merged exceeds 6.
TOO MANY STROBES	The number of strobe light strings contained in the two modules being merged exceeds 6.
TOO MANY ROTATING LIGHTS	The number of rotating light strings contained in the two modules being merged exceeds 4.
TOO MANY VLA	The number of VASI light strings contained in the two modules being merged exceeds 2.

TOO MANY STRINGS	The total number of light strings contained in the two modules being merged exceeds 110.
TOO MANY FEATURE DECODE BUFFER ENTRIES	The total number of buffer entries contained in the two modules being merged exceeds 32.
TOO MANY FACES	The number of faces contained in the two modules being merged exceeds 64.
TOO MANY EDGES	The number of edges contained in the two modules being merged exceeds 64.
TOO MANY SYMM. SURFACES	Symmetrical (mirrored) surfaces defined in both modules being merged.
TOO MANY RUNWAYS	Center line stripe management is defined in both modules being merged.
VLA DELETE ERROR!	A VLA light string pointed to by a VLA definition was deleted.
OLD VASI MODULE, UPDATED TO VLA!	The module read in was not a VLA module and an update occurred to make the module compatible.

A.4 I/O ERROR MESSAGES -- BUILD AND MERGE

DUP FILE VER>	The file name specified on a write command already exists on the diskette. If the user responds with a Y, the file is overwritten; if the user responds with an N, the command is aborted.
CKS ERR	A checksum error was detected during an I/O operation.
I/O ERROR u	An error was detected during an I/O operation. The error code values are shown below.

ERROR CODE	MEANING
1	File not opened
2	File not defined
3	End of file
4	No memory available
5	Hardware error
6	Disk full
7	Illegal opcode
8	Directory full
9	Duplicate file name
10	Bad logical unit
11	Buffer outside user area
12	Bad device name
13	Bad type of extent
14	Logical device tables full
15	No disk or door open
16	Disk select error

APPENDIX B

LISTING OF COMMAND FILE

B.1 INTRODUCTION

This appendix contains a listing of the @EDI command file. This file is used when editing, assembling, re-editing, and re-assembling TI source code. All command file commands are used in this file except the $S command.

```
//ECM.      @EDI 22-APR-80
$U
$B
$M //ASS,4,UNL.
$M //EDI,DMY,,DMY,,DMY.
Q
EDIT FILE
LIST DEVICE=
$Q <<6 3                     <<6 LIST DEVICE
$E
$M //ASS,6,<<6.
$F $-4
DISK DRIVE=
$Q <<5 3                     <<5 SOURCE DRIVE
$E
$M //ASS,32,<<5.
```

```
0017  $F $-4
0018  OPTION=
0019  $Q <<0 1
0020  $E
0021  $T <<0 C $+14               CATALOG.           <<0 OPTION
0022  $T <<0 F $+18               FREE DISK SPACE.   <<1 FILE NAME
0023  $T <<0 M $+21               MAKE FILE.         <<2 BACK UP
0024  $T <<0 R $+26               RECOVER FILE.      <<3 ASM[Y/N]=
0025  $T <<0 E $+32               EDIT FILE.         <<4 ERR[Y/N]=
0026  $T <<0 Z $+53               ZAP :EF FILE.      <<9 FREE
0027  OPTIONS:
0028    C - CATALOG
0029    E - EDIT FILE WITH BACK UP
0030    F - FREE DISK SPACE
0031    M - MAKE A FILE
0032    R - RECOVER BACK UP FILE
0033    Z - ZAP EXISTING FILE
0034  $G $-16                     BACK TO OPTION Q.
0035  $M //ASS,6,<<6.             ***CATALOG***
0036  $M //EXE,FDO,CATLOG.
0037  $M //REW,6.
0038  $M //REW,6.
0039  $G $-21
0040  $M //ASS,6,KEY.             ***FREE SPACE***
0041  $M //EXE,FDO,CATX01.
0042  $W OUT OF 2430 SECTORS
0043  $G $-25
0044  $J $-67 $+55                ***MAKE FILE***, QUESTIONS
0045  $M //ASS,4,UNL.
0046  $M //EDI,<<5,<<1,FDO,HEDDER,DMY.
0047  Q
0048  $M //ASS,4,KEY.
0049  $G $+10                     GOTO EDIT A FILE
0050  RECOVER FILE, ARE YOU SURE[Y/N]=
0051  $Q <<9 1                    ***RECOVER A FILE***
0052  $G $-34                     BACK TO OPTIONS
0053  $T -<<9 Y $-35              NO, BACK TO OPTIONS
0054  $J $-57 $+45                QUESTIONS
0055  $M //DEL<<5,<<1.
0056  $G $+5                      GOTO EDIT WITH BACK UP
0057  $J $-54 $+42                ***EDIT FILE***, QUESTIONS
0058  $M //DEL,<<5,<<2.           DELETE BACK FILE
0059  $M //REN,<<5,<<1,>>2.       RENAME FILE
0060  $F $-42                     ERROR, BACK TO OPTION
0061  $T -<<3 Y $+5               ASSEMBLE?
0062  $T >>4 Y $+2                ERRORS ONLY?
0063  $J $-54 $+50                LIST ALL
0064  $J $-63 $+58                ERRORS ONLY
0065  $G $+2
0066  $J $-71 $+65                EDI LIST
0067  $M //EDI,<<5,<<1,<<5,<<2,KEY.
```

```
$F $-50                       ERROR, BACK TO OPTION
$J $-98 $+82                  LIST/ASSEMBLE
RE-EDIT[Y/N]=
$Q <<9 1
$G $+6                        NO
$T -<<9 Y $+5                 NO
$M //DEL,<<5,<<2.             DELETE BACK FILE
$M //REN,<<5,<<1,<<2.         RENAME
$F $-58                       ERROR, BACK TO OPTION
$J $-72 $+54                  EDI LIST
$J $-132 $+61                 FINAL ASM
ZAP FILE, ARE YOUR SURE![Y/N]=
$Q <<9 1                      ***ZAP FILE***
$G $-63                       BACK TO OPTION
$T -<<9 Y $-64                BACK TO OPTION
$J 1$-28 $+19                 QUESTION+1
$T -<<3 Y $+5                 ASSEMBLE?
$T <<4 Y $+2                  ERRORS ONLY?
$J $-31 $+27                  LIST ALL
$J $-40 $+35                  ERRORS ONLY
$G $+2
$J $-48 $+42                  EDI LIST
$M //EDI,<<5:EF,<<1,<<5,<<1,KEY.
$F $-73                       ERROR, BACK TO OPTION
$J $-75 $+59                  LIST/ASSEMBLE
RE-ZAP[Y/N]=
$Q <<9 1
$G $+3
$T -<<9 Y $+2
$J $-48 $+34                  EDI LIST
$J $-132 $+41                 FINAL ASM
BACK UP=
$Q <<2                        %%%QUESTIONS%%%
$G $-83
$W FILE NAME=
$Q <<1
$G $-86
ASSEMBLE FILE[Y/N]=
$Q <<3 1
$G $+5
$T -<<3 Y $+4
$W ERRORS ONLY[Y/N]=
$Q <<4 1
$G $+1
$R
$V 0
$V 5                          %%%LIST ALL%%%
D9999
L
F2-2F/ UNL//*/
ST8,15,25
```

```
T
$V
$R
$V 0
$V 5                        %%%ERRORS ONLY%%%
D9999
L
F2-2F/*// UNL/
ST8,15,25
T
$V
$R
$V 0
$V 4                        %%%EDI LIST%%%
D9999
L
ST8,15,25
T
$V
$R
$T -<<3 Y $+11              %%%FINAL ASM%%%, ASSEMBLE?
$T -<<4 Y  $+10             ERRORS ONLY?
$M //ASS,4,UNL.             FULL LIST
$M //EDI,<<5:EF,<<1,<<5,<<1,KEY.
D2
F2-2F/ UNL//*/
Q
$M //ASS,4,KEY.
$M //ASM,DMY,,<<5,<<1,<<6.
$M //REW,6.
$M //REW,6.
$R                          BACK TO OPTIONS
$T <<3 >9B $+17             ESCAPE LIST %%%LIST/ASSEMBLE%%%
$T <<3 Y $+9                ASSEMBLE?
$M //ASS,4,UNL.
$M //EDI,DMY,,<<5,<<1,<<6.
D9999
T
P9999
Q
$M //ASS,4,KEY.
$G $+6
$M //ASS,6,KEY.
$T <<4 Y $+2
$M //ASS,6,<<6.
$M //ASM,DMY,,<<5,<<1.
$T <<4 Y $+3
$M //REW,6.
$M //REW,6.
$R
$U
```