PS 300

Volume 5

901194-200

PS 300 DOCUMENT SET

VOLUME 5

SYSTEM MANAGER REFERENCE

Copyright © 1984
EVANS & SUTHERLAND COMPUTER CORPORATION
P.O. Box 8700, 580 Arapeen Drive
Salt Lake City, Utah 84121

Volume 5

PS 300 SYSTEM MANAGER'S GUIDE

Supports DEC and ASCII host systems

Supports DEC VAX/VMS and DEC PDP11-RSX-11M
DEC DMR11-AE Driver Interface
DEC Parallel Interface

PREFACE

This manual is directed to users with a DEC/VAX or PDP-11 host system or ASCII systems that are functionally similar to the DEC systems.

The operation and communication subjects that are covered in this volume are directed towards a PS 300 System Manager. "System Manager" is defined as any programmer who works directly with the PS 300 and is responsible for system configuration, maintenance, and software or firmware installation.

Chapter 1 provides an overview of the hardware and hardware configuration of the PS 300 system. This information is provided to serve as reference for hardware particulars, to establish the hardware names, and to briefly describe what the PS 300 system offers as options and devices.

Chapter 2 describes the distributed media that accompanies all PS 300 systems. This media includes the PS 300 Graphics Firmware diskettes, other distributed system diskettes (diagnostic and demonstration), and the PS 300 host-resident software programs that are distributed on magtape. Included in the description are: the organization of the CONFIG.DAT file, steps used to create the SITE.DAT file, and the system management methods that are available to conform PS 300 firmware or software to meet specific site needs.

Chapter 3 describes the power-up procedures for the PS 300 System. Included are power-up steps for 1 and 2 drives, power-up confidence tests, and the definitions of the confidence test alphanumeric characters.

Chapter 4 describes the data flow between the PS 300 and the host processor. The initial sections of this chapter introduce some of the basic concepts of data communication, particularly those directly affecting the interface to be set up between the PS 300 Graphics System and the host computer. The standards for the EIA RS-232-C and the EIA RS-449 interfaces are provided. The chapter follows data from the host through the F:DEPACKET function in asynchronous, serial transmission.

Chapter 5 discusses internal data flow in the PS 300. This chapter includes block diagrams of data flow through the PS 300 system function network, a description of routing functions and routing bytes, the channels that data can be routed to, and the terminal emulator network.

Chapter 6 provides a description of PS 300 system functions, definitions of "constant" and "trigger" queues, and a list of data types for inter-function communication. The bulk of the chapter is a System Function Summary. Each system function is shown with a box diagram, acceptable data types for input and output ports, and a description of each function.

Chapter 7 covers several loosely linked topics: initial data structures, name suffixing, and system commands. Within these main topics are other important subtopics including the system Configure mode and its uses. The first section of the chapter describes a "runtime" system and the initial data structures that are built by the PS 300 firmware. The final sections discuss the Configure mode, name suffixing procedures, and system commands.

Chapter 8 discusses the PS 300 Terminal Emulator (TE) from several perspectives, including the ANSI modes and control sequences that are used to implement VT-100 terminal capabilities. Many of Digital Equipment Corporation private sequences and modes for the VT-100 are referred to in the discussion. The chapter also discusses the system functions that form the terminal emulator network and the three communication modes of operation of the keyboard.

Chapter 9 provides a description of the system error messages that a programmer might encounter during standard operation of the PS 300 Graphics System.

Chapter 10 provides a reference for the Utility Commands that are on the PS 300 Diagnostic Diskettes. The Utilities are used to back-up diskettes and for diskette file management.

Chapter 11 contains the installation instructions for the PS 300 host-resident software packages, the PSIOs and the GSRs. Installation instructions are provided for DEC VAX/VMS and DEC PDP-11/RSX-11M systems.

Chapter 12 describes how the PS 300 Interactive Devices hardware works. Each section contains the hardware functional description for a specific device, and information related to maintaining these devices.

Chapter 13 describes the data formats expected by PS 300 system functions, in particular, the command interpreter (CI). The information is made available to provide programmers whose systems are not supported by the PS 300 Graphics Support Routines with the information necessary to write their own.

CONTENTS

FIGURES

1. PS 300 SYSTEM OVERVIEW

The first chapter of this guide gives of an overview of the PS 300 hardware and hardware configuration. The material is presented to give the reader a better understanding of the machinery of the PS 300. It is recommended that any system manager requiring material of a more complete, or detailed nature about system hardware purchase the *PS 300 Maintenance Manual* and the *PS 300 Interactive Device Maintenance Manual*.

The PS 300 is a microprocessor-based, data-driven, interactive graphics display system that consists of a PS 300 Control Unit and monochrome and/or color displays. Calligraphic color and raster displays are available as PS 300 options. Circuitry in the PS 300 Control Unit defines, manipulates, and generates the images that appear on the displays. The control unit circuitry also communicates with a variety of host processors and with interactive devices. Interactive devices that may be used with the PS 300 include a data tablet, standard and IBM alphanumeric/function keyboards, function buttons, and control dials.

Figure 1-1 shows a standard system configured with one each of the interactive devices.

1.1 PS 300 CONTROL UNIT

PS 300 electronics are packaged in a freestanding control unit mounted on adjustable glides. The control unit is approximately 67 cm (26 inches) wide, 67 cm (26 inches) deep, and 77 cm (30 inches) high, and weighs 77 kg (170 lbs). Like all other PS 300 family products, the control unit cabinet is finished in off-white and black.

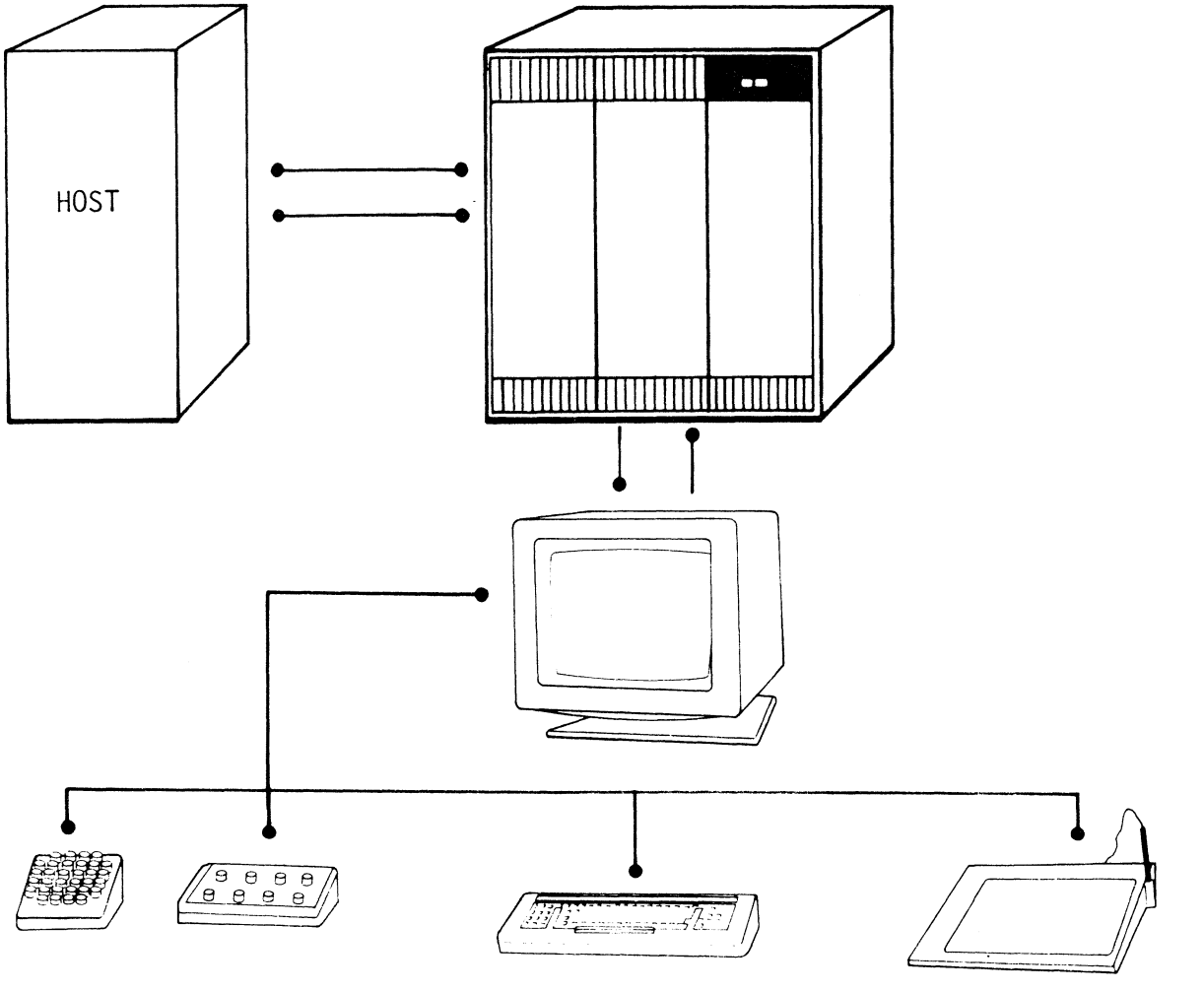


Figure 1-1. PS 300 and Interactive Devices

Major functional components of the control unit include:

- Communications Connector Panel (E&S #204020-100)
- Power Distribution Panel (E&S #204030-100)
- Qualidyne Power Supply (E&S #801844-005)
- Semi-conductor Circuits -5 Volt Power Supply (EC5S3000-K2)
- Card Bay and Power Supply Fans (E&S #801035-301)
- Floppy Disk Drive (E&S #204050-100)
- Optional Dual Floppy Disk Drive, (E&S #204430-200)
- Standard Eight Card Bay (E&S #204413-111)
- EMI Expansion Bay Option (E&S #204414-200)
- Bus Backplane and I/O Panel
- Display Connector Panel

Figure 1-2 is a cut-away view of the PS 300 Control Unit showing internal component placement.

The only external control on the PS 300 Control Unit is a 15-amp on/off circuit breaker switch, located at the top right of the front panel.

The PS 300 Control Unit has three panels on the front and three panels on the back that pop off to provide access to the power supply and to floppy disk drive(s), as well as to cabling and to connector and distribution panels. This also provides access for circuit card insertion.

The control unit is cooled by fans that pull intake air up through cleaning filters in the bottom of the control unit. The power supply has a built-in fan.

1.1.1 Communications and Power Distribution

A Communication Connector Panel and a Power Distribution Panel are located at the back of the PS 300 Control Unit.

- Communication Connector Panel

  The Communication Connector Panel (CCP) is located in the right rear of the control unit, above the Power Distribution Panel. Connectors are accessible from the rear of the unit.

Six ports are provided on the CCP. Ports 0 and 1 are equipped with 37-pin connectors for RS-449 communications; Ports 0 through 5 are equipped with 25-pin connectors for RS-232 communications. PS 300 interactive devices are not normally connected directly to this panel (a keyboard can be directly connected to the CCP for diagnostic purposes). Connection for these interactive devices is on the back of the PS 300 Display. One modular connecter is provided on Port 2 for diagnostic purposes.

Only one connector per port may be used at any one time.

- Power Distribution Panel

  PS 300 power is distributed through the Power Distribution Panel on the back of the control unit at the lower right.

Two power supplies supply power to the PS 300 system components, the main power supply (E&S #801844-005) and the -5 volt power supply (EC5S3000-K2).

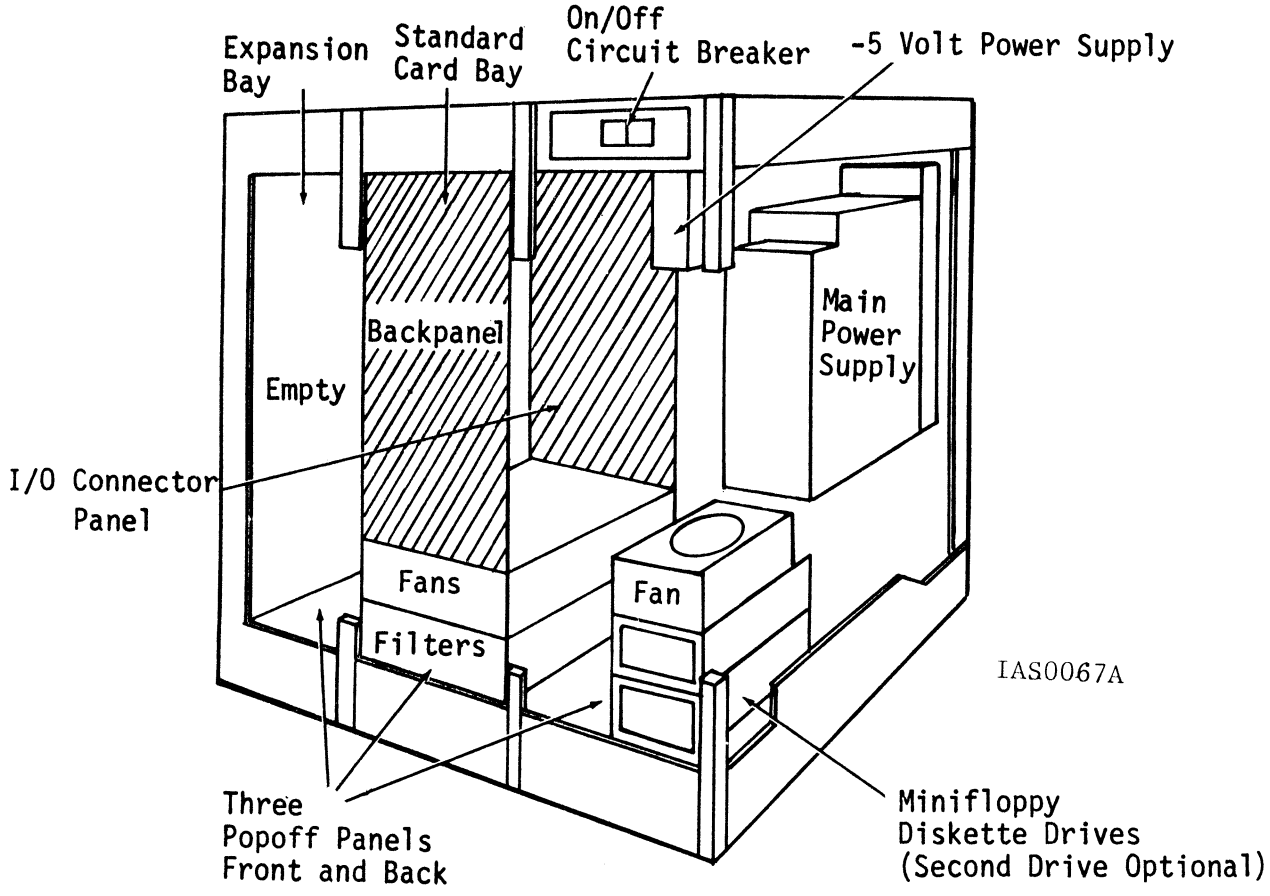


Figure 1-2. PS 300 Control Unit - Locator View

The Main Power Supply

The 750-watt Qualidyne system power supply has multiple outputs providing the following maximum values:

+5 volts, 112 amps
+12 volts, 13 amps
-12 volts, 1 amps
+15 volts, 1.25 amps
-15 volts, 0.6 amps

AC and DC power are distributed through the power distribution panel.

The power supply has a built-in cooling fan and is available in 110V and 220V models.

The Qualidyne power supply provides power for all cards and (via the power distribution panel) power for the peripherals, disk drives, LED lights, thermal sensors, the communications connector panel, and the data concentrator on the PS 300 displays.

To remove the power supply, free the ribbon cables from their cable clips on the communications connector panel, remove the communications connector panel, disconnect the four screws that connect the power supply to its mounting plate, remove the mounting plate and slide the power supply out. Reverse this procedure to re-install the power supply.

Minus 5 Volt Power Supply

The -5 volt power supply provides -5 volt power to the Line Generator Subsystem (LGS), Color Shadow Mask (CSM) color card, and Raster cards via the backplane.

1.1.2 Fans

The standard single chassis control unit contains five Pamotor 4800X fans (E&S #801035-301). One is mounted on the bottom of the Qualidyne power supply, one below the SCI power supply, and three in the bottom of the middle card chassis. Wiring should be kept clear of the fans.

1.1.3 Floppy Diskette Drive

The floppy diskette drive is placed in the power supply bay of the PS 300 Control Unit and is used to store and load the PS 300 system software, diagnostics, acceptance tests, microcode, and non-volatile data used to define the PS 300 system configuration. The PS 300 uses a double-density, single-sided 5-1/4 inch minifloppy diskette capable of storing 368,640 formatted data bytes on 80 tracks.

Up to two floppy diskette drives may be configured in a PS 300. The drive(s) are mounted inside the control unit as shown in Figure 1-2.

The floppy diskette drive is powered by the main power supply via the DC harness (E&S #204060-100) and the power distribution panel. It is connected to the Graphics Control Processor (GCP) via a ribbon cable (E&S #204053-006, non-EMI, and E&S #204054-006 A0, EMI). The cable terminates on both ends with a ground termination clamp. The clamp must be screwed to the Faraday cage on one end and to the floppy disk enclosure on the other end to maintain EMI integrity.

An optional dual-disk drive (the Dual Floppy Option, E&S #204430-200) is available for the PS 300.

1.1.4 Circuit Cards

All PS 300 circuit cards measure 38.4 cm by 41 cm (15 inches by 16 inches) and are designed with connector pins on two edges, as shown in Figure 1-3.

There are five standard circuit cards and three optional cards for the PS 300 systems (including the PS 320 and 330):

- Graphics Control Processor Card (GCP) (E&S #204111-100)
- Mass Memory Card (E&S #204120-100)
- Arithmetic Control Processor Card (E&S #204130-100)
- Pipeline System Card (E&S #204140-100)
- Line Generator Subsystem Card (E&S #204150-100)

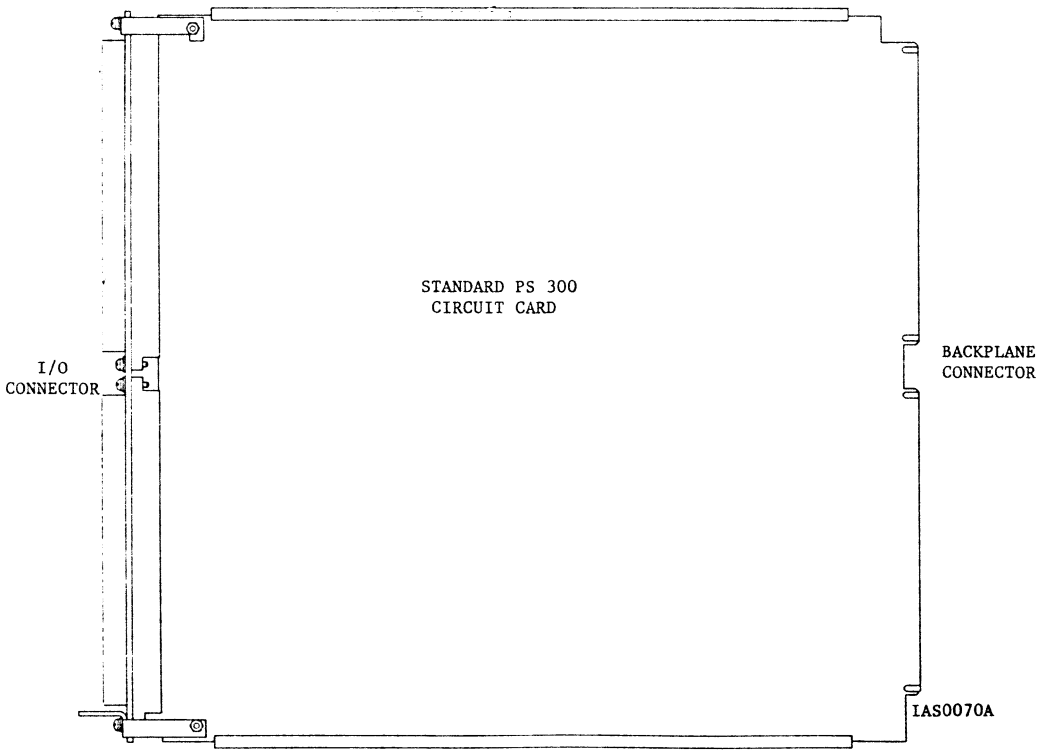


Figure 1-3. Circuit Card Connector Layout

There are four optional cards for PS 300 systems:

- General Purpose Interface Option (GPIO) (E&S #204170-100) which comes in three versions:

  –
  IBM 3278 Emulator GPIO Card (E&S #204171-100)

  –
  UNIBUS Parallel Interface GPIO Card (E&S #204172-100)

  –
  Wire Wrap GPIO Card (E&S #204173-101)

- Color Shadow Mask Calligraphic Display Color Card (CSM) (E&S #204180-101)

- Hardcopy (HCP) (E&S #204160-100)

- Raster Card (E&S #204310-170)

The PS 340 uses a raster display and the Raster card is available for PS 340 systems. The HCP, GPIO, and CSM are optional cards in PS 320, PS 330, and PS 340 systems.

1.1.5 Standard Card Bay

The standard control unit is divided into three bays: the first houses the power supply; the second or middle bay houses the standard eight-card rack made up of card guides, a backplane, three fans, a filter, and the I/O connector panels; the third bay is empty but can be used as an optional expansion card bay. The bay that holds circuit cards is shielded by a Faraday cage to prevent electrical interference to nearby equipment.

The middle bay of the control unit contains a card rack which holds up to eight standard PS 300 cards with non-ZIF connectors. Each card has an individual I/O panel. When the cards are inserted in the card rack, the individual I/O panels fit together to form the equivalent of the I/O panel of the older control units.

The circuit cards plug into the backplane which provides intercard bus communications and supplies power to the cards.

1.1.6 Card Insertion and Removal

Cards are inserted from the back of the control unit along card guides on the bottom of the bay. The component side of the card must face away from the power supply bay. Slide the card onto the bottom card guide and line it up with the notch in the top card guide. Push the card to the front of the control unit until it lines up with the bevel of the backplane connector. Use the insertion/extraction tool (E&S #504415-001) to ease the card into the backplane connectors.

CAUTION

Forcing the cards into the control unit can damage them. Also, be careful not to damage the fuses on the top right of the cards.

To remove a card, ease the card out of the backplane connector with the insertion/extraction tool. To avoid damaging the fuse, slide the card about half way out of the control unit. Tilt the back of the card up and draw the card completely out of the control unit.

Warped cards will not line up correctly with the backplane connectors. They tend to "bow out" in the middle of the card or even assume an "S" shape. To correct this, reach into the bay and bend, BUT DO NOT FORCE, the card until it lines up with the bevel of backplane connectors. Use the insertion/extraction tool to insert the card into the bevel of the backplane connectors. If the card cannot be seated because of warpage, return the card to Salt Lake City with an explanation.

1.1.7 Circuit Card Configuration

PS 300 circuit cards should be inserted in a specific order. Standard systems without options should have the cards inserted, left to right:

LGS
PLS
ACP
GCP(s)
Mass Memory(ies)

Systems with options can be configured in several ways and should be configured on case-by-case basis. Generally, PS 300 options should be placed in the control unit left-to-right as follows:

CSM
LGS
PLS
ACP
GCP(s)
Mass Memory(ies)
GPIO(s)
HCP
Raster

Note that there are a total of nine standard and optional PS 300 cards and only eight slots in a standard system.

1.1.8 Cabling the Cards

Cards are connected to each other, to displays, and to interactive devices in the following manner:

CARDS	CABLES
GCP:	1 to floppy disk drive(s)
	1 to Communication Connector Panel
	1 to real time clock generator on the Power Distribution Panel
ACP:	1 to PLS
PLS:	1 to LGS
LGS:	4 to Displays
	4 to Light Pens

1.1.9 PS 300 Expansion Bay

The third bay in the PS 300 Control Unit can be used as an optional expansion bay to allow the installation of one additional printed circuit card (or two cards, if one of the total cards is a CSM card). Two types of PS 300 Control Units can be upgraded: systems with FCC hardware (S/N 236 and above) and systems without FCC hardware (S/N 235 and lower). Each type of upgrade requires different hardware.

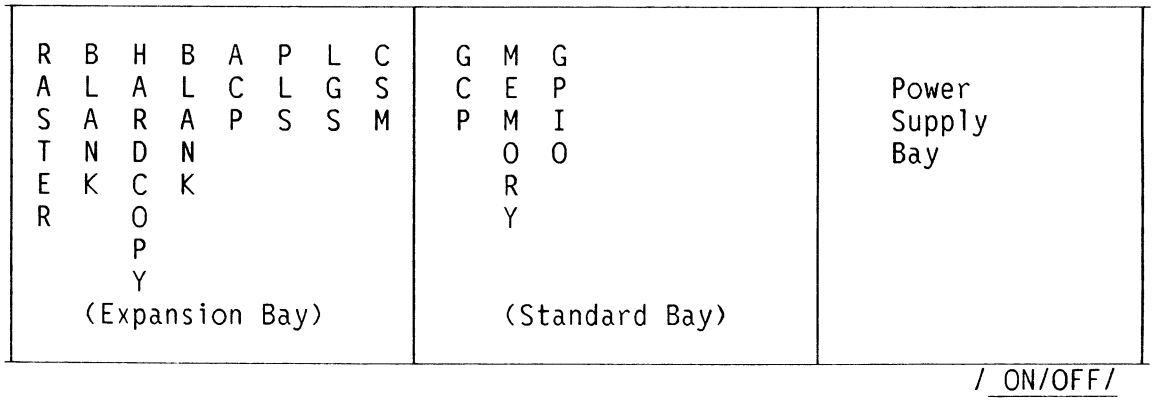


Figure 1-4. Standard Expansion Bay Card Configuration

In PS 300 systems with expansion bays, card placement follows several general rules (see Figure 1-4 for Standard Expansion Bay Card Configuration). The GCP, Mass Memory, and GPIO cards are installed in the center bay. The ACP, PLS, LGS, and option cards (Hardcopy, CSM Color Card, and Raster) are installed in the outside bay. The option cards are assigned specific positions. For example, if the system contains a Raster card but no CSM card or Hardcopy card, the CSM and Hardcopy card slots are left empty.

To install cards in the center bay of the PS 300 Control Unit, work from the partition between the center and outside bays. Install all GCP(s), all Mass Memory cards, and then all GPIO(s). These cards must be installed with no empty slots between them.

1.1.10 I/O Panel and Bus Backplane

Circuit cards connect to the bus backplane. The I/O panel is formed by sections of the panel attached to individual cards. Each circuit card is keyed so that it can only be locked into place if it is inserted right-side up and facing the right direction.

The middle bay of the control unit contains a card rack that holds up to eight standard PS 300 cards with non-ZIF connectors. Each card has an individual I/O panel. When the cards are inserted into the card rack, these I/O panels fit together to form the equivalent of the I/O backplane of the older models. The I/O panel expands the number of pins available for card I/O and provides for configuring communications between cards that cannot be done on the backplane.

The bus backplane is located on the front side of the card bays. It consists of a four-layer printed circuit board. The common bus and several smaller buses run the entire length of the board. The bus backplane provides for intercard bus communication and supplies power to the cards.

1.1.11 Display Connector Panel

The display connector panel is located on the bottom left at the back of the control unit. All displays connect to four plugs (0-3) on the display connector panel.

The display connector panel connects DC power and deflection and intensification signals to the display.

1.2 PS 300 DISPLAYS

There are four different types of PS 300 displays:

- PS 300 Monochrome Display
- PS 300 26-inch Color Shadow Mask Calligraphic Display
- PS 300 19-inch Color Shadow Mask Calligraphic Display
- PS 340 Static Raster Display

These displays are described below.

1.2.1 The PS 300 Monochrome Display

The PS 300 Monochrome Display Unit (E&S #204090-112) is approximately 54 cm (21 inches) wide, 56 cm (22 inches) high, 61 cm (24 inches) deep, and weighs 54 kg (120 lbs.). The visible screen area is 12 x 14.5 inches. The usable display area is 10.5 x 10.5 inches. Deflection and intensification circuits in the PS 300 Monochrome Display use analog signals from the PS 300 Line Generator Subsystem (LGS) to draw lines on the CRT.

The PS 300 Monochrome Display has a tilt/rotate base that allows the display to tilt from -8 degrees to +8 degrees from vertical and to rotate a full 360 degrees around the base. Vertical tilt of the display head is controlled by a lock located below the display control panel. When this lock is released, the display head can be moved to the desired position. The vertical tilt is maintained as the display is rotated.

The on/off switch and an intensity control dial are located on the front of the display. The AC power connection for PS 300 Displays should be plugged into a wall outlet in the same manner as the PS 300 Control Unit. Both the outlet for the display unit and for the control unit must be on the same circuit to ensure that power is in phase and that no dangerous voltage potentials exist between the control and display units.

A Data Concentrator Connector Panel is configured on the back of the monochrome display. The Data Concentrator Panel allows as many as six interactive devices to be multiplexed onto one port of the Communication Connector Panel in the PS 300 Control Unit. Ports A through F on the Data Concentrator Connector Panel provide connections for as many as six interactive devices. (A RS-232 25-pin connector and a modular-type connector are provided for each port.)

An additional internal port on the Data Concentrator communicates with one of Ports 2 through 5 of the Communications Connector Panel in the control unit. For example, the Data Concentrator Panel on Display 0 communicates to Communication Panel Port 5, Display 1 communicates with Port 4, and so on. If a port on the Communications Panel is connected to a Data Concentrator, the connector on the Communications Panel may not be used.

1.2.2 PS 300 26-inch Color Shadow Mask Calligraphic Display

PS 300 26-inch Color Shadow Mask (CSM) Calligraphic Display Unit (E&S #201200-103) is functionally identical to the PS2/MPS CSM display and is packaged in the black PS2/MPS-style cabinet. It has a 13 x 13 inch usable screen area. It receives analog signals from the CSM color card. RGB video signals are generated on the CSM color card. Deflection and unblank signals originate on the LGS card and are sent to the display via the CSM color card.

The 26-inch CSM display requires a 220-volt AC input supply.

For a detailed theory of operations refer to the *CSM Display System Maintenance Manual.* For tuning and alignment procedures, refer to the *PS 300 CSM Display Installation Manual.*

1.2.3 PS 300 SM19 19-inch Color Shadow Mask Calligraphic Display

PS 300 19-inch SM19 Color Shadow Mask Calligraphic Display (E&S #204455-TAB) physically matches the PS 300 monochrome display packaging. It uses a high-resolution (0.31 mm dot trio pitch) in-line gun shadow mask cathode ray tube (CRT). The display has a usable screen area of 10.5 by 10.5 inches. The 19-inch CSM is an XY magnetic deflection display and can generate colored wire frame images. Its drawing speed runs at half the speed of the monochrome display.

Like the 26-inch CSM display, the 19-inch CSM display receives analog signals from the CSM color card. RGB video signals are generated on the CSM color card. Deflection and unblank signals originate on the LGS card and are sent to the display via the CSM color card.

Refer to *The PS 300 SM19 Color Display Tuning and Alignment Manual* for a theory of operations and tuning procedures for the SM19.

1.2.4 PS 340 Raster Display

The PS 340 Raster Display (E&S #204315-TAB) is a 640 x 480 pixel, RGB, high resolution static raster display. The Raster display matches the PS 300 monochrome display in appearance. It has a 3 x 4 aspect ratio and has a 9 inch x 12 inch usable screen area. It is capable of displaying solid images of PS 300 polygonally defined objects. It receives RGB video signals and composite sync signals from the PS 340 Raster card. These signals are generated from an image buffer memory on the Raster card. It can display 256^3 colors.

For alignment and tuning procedures, refer to the *Raster Option Installation Manual.*

1.3 FUNCTIONAL OVERVIEW

The PS 300 consists of several general-purpose and special-purpose processors and subsystems that are interfaced over a common data bus in conjunction with a mass memory system. Figure 1-5 is a simplified block diagram of the PS 300 system.

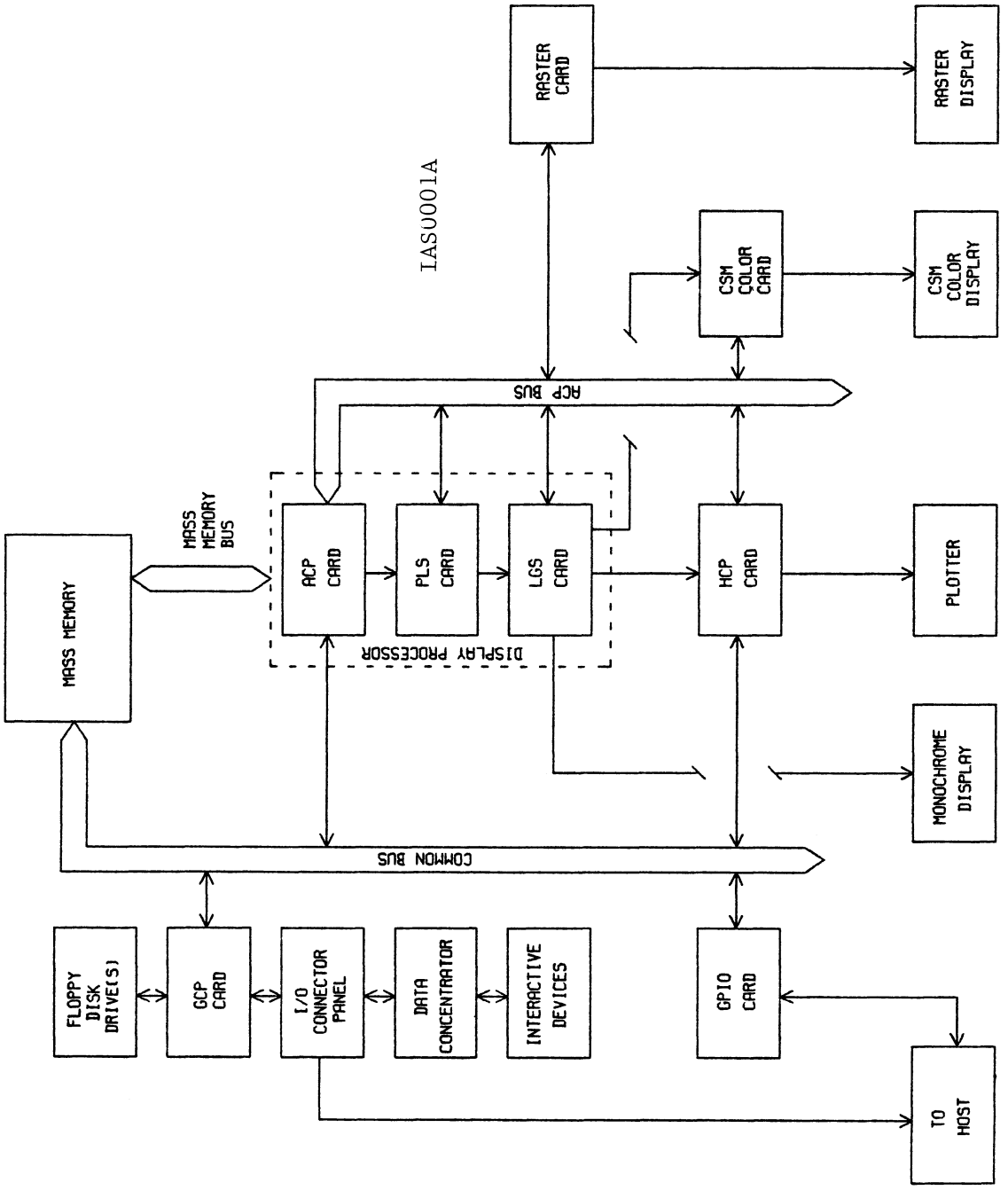


Figure 1-5. PS 300 - Simplified Functional Diagram

As the diagram shows, the PS 300 consists of the Common Bus, an ACP Bus, a Graphics Control Processor, one or more Mass Memory cards, and a Display Processor that includes the Arithmetic Control Processor, the Pipeline Subsystem, and the Line Generator Subsystem.

As shown in Figure 1-5, the Graphics Control Processor communicates with Mass Memory over the common bus and with the Display Processor through Mass Memory. The path shown between the common bus and the Display Processor is a maintenance path only.

Major PS 300 component functions are described below.

- Common Bus

  The common bus provides a 16-bit, high-speed, asynchronous communications path for PS 300 components. The common bus is used mainly for communications between the Graphics Control Processor and Mass Memory in single processor systems. Maintenance functions between the GCP and the Display Processor also take place on the common bus.

- Graphics Control Processor

  The Graphics Control Processor (GCP) is a general-purpose microprocessor that manages display data structures residing in Mass Memory and initiates the display of data by the Display Processor. Host communication, interactive devices, and the system floppy diskette drive(s) are interfaced to controllers on the GCP.

- Mass Memory

  The general-purpose, dual-ported Mass Memory contains the display data structures that are stored and manipulated by the GCP and accessed for display by the Display Processor. Mass Memory consists of one or more Mass Memory Cards, each of which contains 1024K bytes of error-corrected MOS memory. This memory may be used as program memory by the GCP in addition to holding the display data structures.

- Display Processor

  The Display Processor includes an Arithmetic Control Processor, a Pipeline Subsystem, and a Line Generator Subsystem. The Display Processor traverses display data structures in Mass Memory and processes that data for display. A traversal of the display data structures occurs each refresh cycle under control of the Arithmetic Control Processor (ACP). The Display Processor transforms the data to be displayed; performs clipping, perspective projection, and viewport mapping; and finally converts digital display data to their analog equivalents to drive deflection and intensification circuitry in the CRT. The major functional elements of the Display Processor are described in the following paragraphs.

  - Arithmetic Control Processor

    The Arithmetic Control Processor (ACP) contains high-speed multipliers, special control logic, and a general-purpose, bit-slice microprocessor that interfaces with Mass Memory by means of a high-bandwidth, 32-bit memory-access port. The ACP traverses linked Set-Operate-Data (SOD) structures in Mass Memory. These SOD structures contain commands that indicate functions the ACP is to perform for each SOD block. For example, these commands can be used to modify the state of the ACP and to process 3-dimensional data. The state of the ACP is considered to be those values that are context dependent, such as transformation matrix contents, viewport boundaries, and color.

    In traversing the SOD structures, the ACP performs matrix operations required to transform data points and passes transformed data coordinates to the Pipeline Subsystem.

  - Pipeline Subsystem

    The Pipeline Subsystem (PLS) accepts transformed data coordinates from the ACP and performs spatial translation, clipping, perspective division, and viewport mapping functions on the data. The Pipeline Subsystem processes data asynchronously in relation to the processing performed by the ACP. Data from the Pipeline Subsystem are output to the Line Generator Subsystem.

  - Line Generator Subsystem

    The Line Generator Subsystem (LGS) accepts screen coordinate and intensity data from the Pipeline Subsystem. From these data, the LGS calculates the speed at which the vector is to be drawn, determines the minimum amount of settle time necessary for CRT deflection and intensification circuits, blanks very short vectors, determines which vectors fall within pick boundaries established by the ACP, and converts digital coordinate and intensity data to analog form for use in the CRT intensification and deflection circuitry.

- PS 300 Options

  A number of PS 300 Options are available, including a Hardcopy (HCP) card, a General Purpose Interface Option (GPIO) card, a Color Shadow Mask (CSM) Calligraphic Color card, and a Raster card. PS 300 Options are described in detail in the ***PS 300 Options Maintenance Manual.***

  - The Hardcopy Card

    The Hardcopy card (E&S #204160-100) allows a displayed screen image to be copied to a Versatec V80 or Hewlett-Packard 2871G graphics plotter. The Hardcopy card accepts vector representations of data from the Display Processor's Line Generator Subsystem. It converts these data to raster format and sends them to the plotter through an 8-bit parallel interface under microprocessor control.

  - The General Purpose Interface Option

    The GPIO card (E&S #204170-100) allows the PS 300 to interface with various host computers by emulating devices with which the host computer is familiar. The GPIO card has been used to implement high-speed interfaces including the DEC UNIBUS Parallel Interface and the IBM 3278 Emulator interface.

  - The Color Shadow Mask (CSM) Calligraphic Color Card

    The Color Shadow Mask (CSM) Calligraphic Color card (E&S #204180-101) provides an interface between an E&S CSM display and a PS 300 Control Unit. It gives the PS 300 the ability to display multi-colored objects using 128 intensities and 120 hues at 16 saturations on a 26-inch or 19-inch monitor.

  - The Raster Card

    The Raster card (E&S #204310-170) resides on the ACP Bus and is controlled exclusively by the Arithmetic Control Processor (ACP) card. The Raster card has a 640 x 512 x 24 bit (960K byte) pixel memory that stores raster images for display on a PS 300 raster monitor. Pixel memory has two ports; one to the ACPBUS for communication with the ACP card and one that outputs an image to the display. The field rate is 60 Hz with a frame rate of 30 Hz (30 Hz interlaced display). The Raster card is designed to produce signals that are compatible with National Television System Committee (NTSC) standards. (To produce an actual NTSC signal, an encoder is required.)

1.4 INTERACTIVE DEVICES

PS 300 Interactive Devices are described in detail in the ***PS 300 Interactive Devices Maintenance Manual.*** Interactive devices that may be used with the PS 300 include:

- Keyboards

  The standard PS 300 keyboard (E&S #204201-100, with LEDs, E&S #204201-101, without LEDs) includes all standard alphanumeric keys, a separate numeric keyblock, typical typewriter control keys (SHIFT, TAB, etc.), symbols, and a set of 12 function keys that may be used to perform graphics interactive functions, an integral microprocessor that provides local intelligence in the keyboard, and an interface to the GCP serial communications port.

  The keyboard may optionally include an integrated set of LEDs on which capital letters, numerals, symbols, and indications of functions (such as 'space') may be displayed. Functions such as carriage return, line feed, or form feed can not be handled. The LEDs may be programmed to display prompts to aid the operator or labels that describe the operations being displayed (i.e., ROTATE IN X AXIS).

  An optional IBM Keyboard (E&S #204201-102, with LEDs, and E&S #204201-103, without LEDs) is available for the PS 300.

- Data Tablet

  The PS 300 Data Tablet (E&S #204220-100) consists of an 11-inch by 11-inch tablet, a stylus, a controller, and an interface to the GCP serial communications port. The data tablet is usually used as an interactive pointing and positioning device to control the cursor that is displayed on the CRT. The tablet surface may be overlaid with a map, a chart, or a menu that is touched with the stylus causing the coordinate position on the tablet to be converted to its digital equivalent for use by the system.

- Control Dials

  The PS 300 Control Dials Unit (E&S #204211-100) consists of two rows of four dials mounted in a cabinet similar in appearance to the keyboard packaging. The function of each control dial is programmable. Above each control dial is an optional 8-digit LED display that may be used to display the function of the dial or other variable data associated with the use of the dial.

  Power and communications for the control dials are provided through a single cable connection.

- Function Buttons

  The PS 300 Function Buttons (E&S #204230-100) provide 32 programmable function buttons in addition to the twelve function buttons on the PS 300 Keyboard. The function buttons are arranged with one row of four buttons, four rows of six buttons, and a final row of four buttons. The buttons are numbered from left to right, beginning at the top row with button 0. Pressing a function button results in a user-specified action. The buttons are usually programmed to rotate, scale, and translate objects with separate buttons used for the X, Y, and Z axes.

System communication control circuitry for the interactive devices is located on the Graphics Control Processor card. This circuitry provides as many as six communications ports used to interface to the interactive devices and to the host processor. Peripheral devices are connected to the system through the Data Concentrator Connector panels A - F on the back of the display.

1.5 DATA CONCENTRATOR

The Data Concentrator (DC, E&S #204081-100) is located on the backpanel of the PS 300 monochrome or 19-inch CSM display or in the 26-inch CSM power supply. The Data Concentrator communicates directly with one of the Enhanced Programmable Communications Interfaces (EPCI) of the GCP. The Data Concentrator is the control center for information being received from and relayed to any interactive devices used with the PS 300. As the Data Concentrator multiplexes information received from these devices, the PS 300 Display Station can support these devices while using only one system port on the GCP. The Data Concentrator is capable of supporting up to six interactive devices. These devices are supported through six sets of Data Concentrator connectors located on the backpanel of the Display unit.

1.5.1 Data Concentrator Ports

Port definitions are established by configuration parameters contained in the PS 300 Graphics Firmware, and must be used according to these parameters. Each port has a 25-pin D type connector that provides minimal RS-232 support. An 8-pin modular connector that provides E&S modified RS-449 support is located below the 25-pin receptacle. The standard configuration for the interactive devices is:

- Port A supports the Keyboard
- Port B supports the Control Dials
- Ports C supports the 32 Function Buttons
- Ports D and E are unused at this time
- Port F supports the Data Tablet

Only one connector per port may be used at any given time.

The backpanel also includes a 9-pin D-type connector that supplies power to the Data Tablet.

1.5.2 Operating Modes

The Data Concentrator has four modes of operation:

1. Debug Mode is entered by typing in CTRL D on the Keyboard or terminal connected to Port A while the DC is in transparent mode, or by momentarily moving the #1 maintenance switch to the "up" reset position when the #2 maintenance switch in the "up" position. Debug mode allows the user to communicate with the Data Concentrator as a stand-alone unit.

2. Confidence Test Mode is entered when the DC is first powered up. In this mode, the GCP directs the DC to run its series of confidence test routines.

3. Transparent Mode is entered by the DC upon power-up. In transparent mode, direct communication is established between the GCP and DC Port A at 300 baud. Communication with the interactive device connected to Port A occurs at either 300 or 2400 baud, depending on the position of the baud rate maintenance switch.

4. Multiplexed Mode is the normal operating mode for the DC. Multiplexing of Ports A through F occurs as needed to maintain communication between the GCP and the interactive devices. Communication between the GCP and the DC occurs at 19.2K baud. Baud rates used by the interactive devices and the DC are set up by special codes transmitted from the GCP.

1.5.3 Maintenance Switches

The upper-right corner of the Data Concentrator panel contains a recessed opening that provides access to three switches. These switches are for maintenance purposes.

Switch 1 — Moving Switch 1 momentarily to the UP position and then moving is back down gives a "hard" reset to the microprocessor-controlled DC circuitry. This is used to initialize only the DC, when an entire system reset is not necessary.

Switch 2 — Switch 2 is used to select the DC Maintenance Firmware. By placing this switch in the UP position and resetting the DC circuitry using Switch 1, the Maintenance Firmware will be brought up.

Switch 3 — This switch is used to set the baud rate for Port A. Placing the switch in the UP position while in transparent or maintenance mode selects a baud rate of 300. This rate is used for slower terminals, such as the Silent 700. The normal transmission rate for Port A is 2400.

During normal system operation, all three MAINT switches are in the DOWN position.

1.5.4 Data Concentrator Protocol Description

Each DC operates as a slave to the GCP, in that the GCP defines all activities and operational characteristics for each DC. The following subsections describe the DC's interaction with the GCP and with the interactive devices, detailing the methods used to initialize, multiplex, and control GCP-DC communications.

Figure 1-6 illustrates system communication initialization.

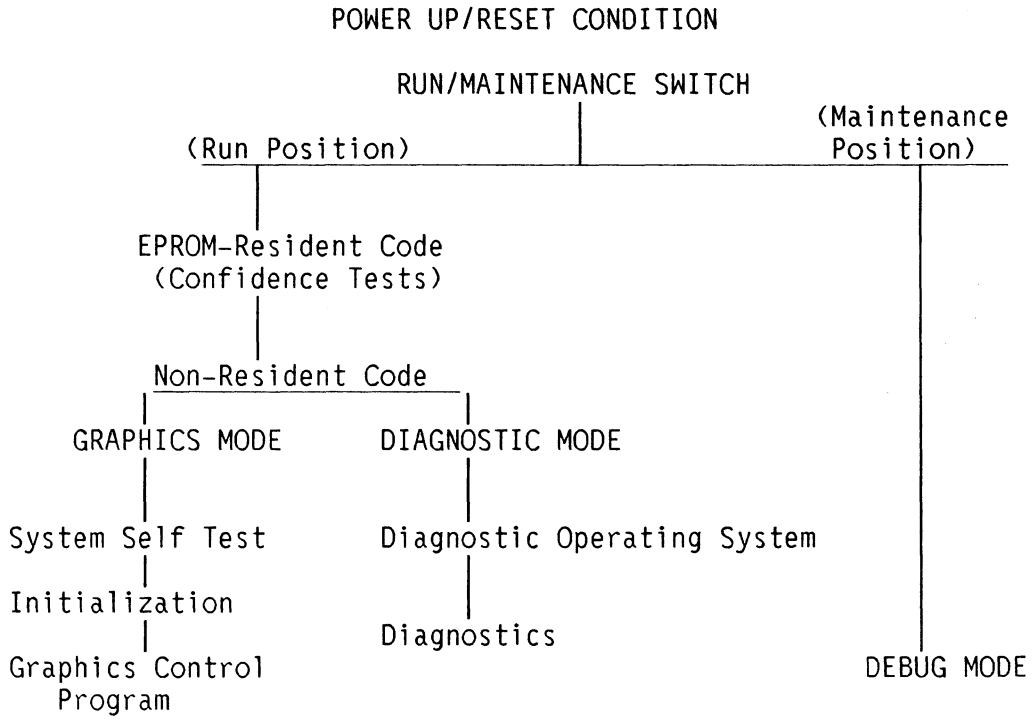


Figure 1-6. System Communication Initialization

1.5.5 Powerup

When the PS 300 Control Unit is switched on, all system logic is powered up, including the Communications Connector Panel, all Data Concentrators, and all connected interactive devices. Every microsystem receives a "hard" reset when the control unit is powered up.

The DC starts in a transparent mode that permits direct communication between the GCP and Port A of the DC. All other DC ports are disabled at this time.

The GCP confidence test results are transmitted at 300 baud directly to all DCs. The DC, operating in a transparent mode, retransmits the test results at 2400 baud through Port A to the Keyboard. Once the confidence tests are successfully completed, system control is transferred from the EPROM to the code loaded from the floppy disk. During the execution of this non-resident code, the GCPs send a special code to the DC, indicating that it is ready to initialize communication. Special codes and ASCII characters are used by both the GCP and the DC to initialize and control communication between the GCP and each interactive device.

1.5.6 Data Concentrator Maintenance Firmware

The following section describes the Debug mode and Debug commands available in the Data Concentrator maintenance mode of operation.

The DC communicates bidirectionally with the GCP through the DC/GCP port. When the PS 300 Graphics System is powered up, the GCP enters either Debug or Run mode, depending on the position of the Maintenance Switch in the Control Unit. The DC will begin running in transparent mode. In transparent mode, the DC channels data bidirectionally at 300 baud from the GCP to Port A of the DC panel. Any of the following conditions will cause the DC to enter the transparent mode of operation:

- PS 300 power-up or reset
- GCP break transmitted to the DC
- GCP transmits data that is misinterpreted as a break
- DC Watchdog Timer times out

1.5.7 Confidence Test Reporting

Upon power-up the PS 300 attempts to execute a series of confidence tests contained in the EPROM-resident code on the GCP card. These tests determine whether or not the hardware on the GCP card is capable of running a program loaded from a floppy disk.

The Data Concentrator should be suspect when all of the following conditions apply:

- A debug terminal is connected to Port A of the Data Concentrator, with Switch 3 set for the appropriate baud rate.
- The confidence tests fail to complete and no valid characters are output on the PS 300 Keyboard LEDs or on the display of the debug terminal.
- The debug terminal, when connected to the GCP, displays valid characters when the confidence tests are run.

CONTENTS

2. DISTRIBUTED MEDIA: PS 300 FIRMWARE AND HOST RESIDENT SOFTWARE

This chapter describes the distributed media that accompanies all PS 300 systems. This media includes the PS 300 Graphics Firmware diskettes, other distributed system diskettes (diagnostic and demonstration), and the PS 300 host-resident software programs that are distributed on magtape. Included in the description are: the organization of the CONFIG.DAT file, steps used to create the SITE.DAT file, and the system management methods that are available to conform PS 300 firmware or software to meet specific site needs.

2.1 THE PS 300 GRAPHICS FIRMWARE

The PS 300 Graphics System Firmware is distributed on 5-inch floppy diskettes. All versions of the system firmware contain files that perform the described functions.

- Microcode defines the functions of the Display Processor.
- Runtime code defines the PS 300 command language and functions. This is made up of various files and includes:
  - THULE.DAT file contains code that is loaded into mass memory. It includes code for the hardcopy option.
- CONFIG.DAT file links the microcode and the runtime code into a coherent system function network.

- SITE.DAT file allows users to enter PS 300 commands into a readable file. This is not a required file. It is used by sites that want to change the default PS 300 system interface values during initialization.

The PS 300 supports ASCII transmission over a 9600 baud line and several high speed serial and parallel interfaces. These interfaces all require different system level code to support data transmission over the host line. The PS 300 system is also defined by the configuration of the system. The following matrix shows the separate firmware packages that are distributed to support the interface packages and the PS 300 configurations.

	PS 300 Configuration		
Interface	*PS 320*	*PS 330*	*PS 340*
ASCII asynchronous	*X*	*X*	*X*
DMR-11-AE serial		*X*	*X*
DEC Parallel	*X*	*X*	*X*

The configuration of the firmware shipped with each system depends on the variables shown above. The firmware shipped with the PS 320 and PS 330 systems contains the code to support all options available with those systems. The PS 340 firmware contains additional user commands for surface and solid rendering, as well as the support code for the PS 340 Raster Display.

2.1.1 The Runtime Code

The runtime code contains all definitions for system level commands and functions, as well as the definitions for user-accessible commands and functions. These definitions are loaded into the GCP local memory.

During the loading of the PS 300 firmware, all system functions and initial functions are "instanced" by the system. "Instancing" is the process of creating a unique case of the function. This case includes all the information necessary to identify the specific case, including an index pointing to the Pascal-callable procedure that performs the required function, the input source(s), the output destination(s), the priority value assigned for scheduling purposes, and the current status of the function.

User-accessible functions and commands are documented in the ***Command Summary*** and the ***Function Summary***. System functions are documented in Chapter 6 of this guide.

2.1.2 The CONFIG.DAT File

The CONFIG.DAT file contains the instances and connections to link the system functions into a system network. This file is written in the PS 300 command language and is similar to any function network built to manipulate graphical data. The design of the configuration file allows for flexibility in changing the initial environment; the change to the file is loaded and read from the disk, rather than being a change to a program. This means that changing the design features of a system does not require recompiling and relinking code and it reduces the size of executable code in the system.

The system network constructed by the CONFIG.DAT file is shown in block diagram form in Chapter 5, "Local Data Flow and System Network." The CONFIG.DAT file is also used in PS 320 systems to allocate memory and system accessibility for dual users.

The CONFIG.DAT file is interpreted by an instance of the command interpreter. While reading this file, the command interpreter is in a privileged mode of operation called the CONFIGURE MODE. The command interpreter must be in the CONFIGURE MODE to configure any system functions.

One of the final commands in the CONFIG.DAT file is a command that will attempt to read the SITE.DAT file, which may or may not reside on the firmware diskette, depending on whether the system manager has chosen to write a SITE.DAT on the diskette. This file will normally be created by the system manager and will contain commands to tailor PS 300 system default parameters to the requirements of that particular site.

The command interpreter will read the SITE.DAT file and exit the CONFIGURE mode. If no SITE.DAT exists on the diskette, the command interpreter will execute the final commands in the CONFIG.DAT and then exit the configure mode.

More information on CONFIGURATION MODE is given in Chapter 7, "Initial Data Structures, Name Suffixing, and System Commands."

2.1.3 The SITE.DAT File

The final file on the PS 300 Graphics Firmware is the SITE.DAT file. This file is not required, but when used enables the user to change default features for the PS 300 system in a bootable file. The file is assumed to contain a stream of ASCII commands. Because of limited disk space, the file can only contain a small number of commands.

The SITE.DAT file allows users to store information across power-up sequences.

The following section of this chapter give the steps to be followed to create and download the SITE.DAT. The actual values and parameters that may be set or changed using the SITE.DAT are described in the pertinent chapters.

Chapter 4, "PS 300/Host Communications and System Networking", describes the port values that may be changed.

Chapter 5, "Local Data Flow", describes certain data packet and routing characters that can be changed.

Chapter 8, "Terminal Emulator Modes and Functions", describes the keyboard and display features that can be changed or set.

Including the SITE.DAT File

Users may want to include a SITE.DAT file when at least one of the following conditions exist:

- Dual-line application of the PS 300 Terminal Emulator is desired.
- Host - PS 300 communication default values need to be changed.
- Terminal Emulator features need to be changed.
- Specific PS 300 features need to be implemented that are not part of the standard firmware package.

Creating and Downloading the SITE.DAT File

1. Make a copy of the PS 300 Graphics Firmware using a scratch diskette and following the procedures in Chapter 10.
2. Create a file on the host called SITE.DAT that contains PS 300 commands. These commands can be used to modify the default values of the Graphics Control Processor ports on the PS 300 as well as any data communication protocol procedures so that the new values reflect site-specific values.
3. The file created must begin with the demuxing character and routing byte:

   ↑\: (↑ represents the CONTROL value of the \ key)

   that will route the file to the PS 300 Firmware diskette.

4. The file must then include the demuxing character and routing byte:

   ↑\;

   followed by the command CLOSE SITE; that will close the file.

5. The file must end with the routing bytes to the TE:

   ↑\>

   The last command in the SITE.DAT file should cause some action on the PS 300 Display or attached terminal to signal the operator that the SITE.DAT file has been successfully loaded (e.g., "SYSTEM READY").

   Because of limited disk space, make the file as compressed as possible.

6. Mount the backup copy of the Firmware diskette and boot the system.

7. Copy the host-resident SITE.DAT file to the PS 300 diskette using the host copy utilities (e.g., COPY on VAX/VMS, PIP on PDP-11/RSX-11.) Refer to the manuals on the host system utilities for help in copying files from the host. Be sure the Write Protect tab is removed from the diskette. If the tab is in place, and an attempt is made to write to the diskette, an error message will be displayed.

All commands in the SITE.DAT file must be terminated with a semicolon.

Name suffixing conventions must be used when system functions are referenced in the SITE.DAT file. Chapter 7 of this guide, "Initial Data Structures, Name Suffixing, and System Commands", discusses the naming suffixes and how they should be used.

2.2 OTHER DISKETTES SHIPPED WITH THE SYSTEM

The PS 300 system is shipped with several other diskettes. These include:

- A back-up copy of the Graphics Firmware (only for systems with single diskette firmware)
- The PS 300 Customer Acceptance Test
- One formatted blank diskette
- The PS 300 Demonstration Package diskettes
- The PS 300 Diagnostics Diskettes

The instructions for running the Customer Acceptance Test is in the ***PS 300 Site Preparation and Customer Support Guide.*** Documentation for running the PS 300 Demonstration Package is in Volume 1 of this document set.

The four diskettes that contain the PS 300 Diagnostics that are shipped with the system are normally used by E&S to test the functionality of the hardware components of the system. However, the Utility Commands that are used to copy diskettes, and to delete files on diskettes are on the PS 300 Diagnostics diskettes. Refer to Chapter 10 of this guide for the definition and use of the diagnostic utilities.

2.3 HOST RESIDENT SOFTWARE

Along with the system diskettes, the E&S distributes files on magtape that will be loaded onto the host system. These files contain various applications and utilities that are used for host/PS 300 communication.

The magtape can contain the following files:

- A README file that describes the files on the magtape.
- Two host/PS communication tests, COMTST1.FTN and COMTST2.FTN.
- The PS 300 Host-Resident I/O Subroutines.
- The PS 300 Graphics Support Routines.
- A SITE.DAT file for setting up dual-line configuration of the terminal emulator.
- Files used to build the Driver for the DEC/DMR11-AE interface or the DEC Parallel interface.
- Customer Acceptance Test for the PS 300 DEC/DMR11-AE interface or the DEC Parallel interface.
- The E&S supplied files that are used in developing and transporting User-Written Functions.

2.3.1 Host Communications Tests

The two host/PS communication tests, COMTST1.FTN and COMTST2.FTN, test the communication link between the host and the PS 300. The tests, and their descriptions are documented in the ***PS 300 Site Preparation and Customer Support Guide.*** Both are data recirculation tests that involve reading from and writing to the PS 300.

2.3.2 The PS 300 Host-Resident I/O Subroutines

The PS 300 Host Resident I/O Subroutines (PSIOs) are distributed on magtape by E&S as callable FORTRAN subroutines. It is the responsibility of the user to load, compile, and link these subroutines with an application program. The Subroutines are supported in the following environments:

- DEC VAX/VMS
- DEC PDP11-RSX-11

The definitions and parameters for the PSIO Subroutines are in the Volume 3B of this document set. Installation instructions are provided in Chapter 11 of this guide.

2.3.3 The Graphics Support Routines

The PS 300 Graphics Support Routines (GSRs) are a set of software routines developed and supported by Evans & Sutherland that allow for faster graphics transactions on the PS 300 System.

The GSRs are distributed as either callable FORTRAN subroutines or Pascal procedures. It is the responsibility of the user to load, compile, and link the routines with their application program. Installation instructions for the GSRs are provided in Chapter 11 of this guide.

The source code for the GSRs contains all files necessary for the customer to compile and link the GSRs, once the tape is loaded on the host. The file names and descriptions are included in the installation instructions.

The GSRs are supported in the DEC VAX/VMS environment.

The documentation for the GSRs is provided in Volume 3B of this documentation set. For easy user reference, documentation has been provided for each language: there is a GSR DEC/VAX Pascal section, and a DEC/VAX FORTRAN section.

Minimum PS 300 System and Host Requirements

- Standard PS 300 system configuration with 2K ACP
- PS 300 Graphics Firmware P5.V03 or higher
- Standard RS-232-C, DMR11-AE, or Parallel Interface
- VMS Version 3.2 or higher
- FORTRAN-77 compiler and/or VAX Pascal V.2 compiler

Software Update Policy

All software distributed by E&S are the latest released versions. As the GSRs need to be relinked with user application programs at the time of a new, or updated release, a strong attempt has been made to make the GSRs as "fixed" as possible. Modifications made to the low-level routines that set up the communication line between the host and the PS 300 that might be made should not affect the routines and/or parameters that are used by the programmer.

2.3.4 Copy of the SITE.DAT (DUALINE.DAT)

A version of the SITE.DAT, DUALINE.DAT, is included on the magtape. The description of the contents of the file appears in the A1 Firmware Release Notes. Should your site require the configuration included in the file, the file can be downloaded from the host onto the PS 300 Firmware Diskette.

2.3.5 Driver Files and CAT Files for the DMR11 or the Parallel Interface

These files and the instructions for building the drivers or running the Customer Acceptance Tests for the interfaces are described in the Installation Manuals for each interface.

CONTENTS

FIGURES

TABLES

3. POWERING UP THE SYSTEM

This chapter describes how to power up and initialize a single-display PS 300. It also contains the definitions of the power-up confidence tests that are run when the system is initialized. It is assumed that the hardware is properly installed and that the system has passed the Customer Acceptance Test.

3.1 POWERING UP THE PS 300

The following stages are involved in powering up the PS 300 Graphics Firmware:

1. The PS 300 Graphics Firmware diskette (or other system diskette) must be loaded in the disk drive.

2. The Control Unit and the Display Station must be turned on.

3. The PS 300 must successfully complete the confidence tests.

4. The PS 300 Graphics Firmware (or other system software) must be successfully loaded.

Before proceeding with power up, make sure:

1. The Control Unit AC power switch is in the "off" position.

2. Both the Control Unit and the Display Station have their AC power cords connected to active wall outlet(s) of adequate capacity (i.e., one 30-amp outlet or two 15-amp outlets).

3. The PS 300 Keyboard with the Terminal Emulator, or an auxiliary ASCII terminal, is available for entering commands to the PS 300.

4. The Display Station cable is connected to the jack labeled "Display 0" at the rear of the Control Unit.

5. The interactive devices are terminated in their proper connectors on the Display Station's Data Concentrator.

6. Provision has been made for monitoring the progress of the confidence tests and other power-up initialization sequences. The test results are displayable on the optional LED arrays of both the Keyboard and Control Dials Unit, and may also be viewed by connecting a 300-baud terminal to Port 3 on the Communications Connector Panel.

3.1.1 Loading the PS 300 Diskettes

The section explains how to load the PS 300 diskettes in the disk drive and contains storage and handling tips.

The floppy diskette drive is located at the front of the Control Unit in the rightmost bay. A diskette is loaded by lifting up the latch on the drive and inserting the diskette with the manufacturer's label (or an E&S label) facing up and the write-protect slot (covered with a foil tape on PS 300 diskettes) facing to the left. The latch should be pushed back down when the diskette is in place.

On some drives, there is a switch in the upper right-hand corner of the drive. This switch is used to lock the drive latch; a red dot is visible when the the latch is moved to the left, signifying that the latch is in a locked position.

CAUTION

Should the activity light remain lit continuously, the diskette drive should not be used. Contact your field service representative so that adjustments can be made to prevent damage to the drive or the diskette.

Keep each diskette in its protective envelope when it is not in use, and store them in an area that is isolated from magnetic or electrical fields. The diskettes should be stored in an environment that is similar in temperature and humidity to the one in which it will be used.

Both the Diagnostics and Graphics Firmware diskettes should be copied as soon as the system is received. Steps for copying the diskettes are given in Chapter 10 of this guide.

3.1.2 Power-Up

Once the Graphics Firmware diskette has been properly inserted into the drive, turn on the PS 300 Control Unit and the Display Station.

The Display Station's switch is located on the lower right-hand edge of the Display, and the Control Unit's power switch is located on the top right corner of the cabinet front. Both switches have activity lights that indicate a power-on condition.

3.1.3 Confidence Tests

Before the Firmware is loaded into the system, a series of hardware confidence tests is performed. The status of this testing is indicated as a series of alphabetic characters, with each character reporting the successful completion of a different portion of the test. The start-up confidence tests are located in the ROM in the Graphics Control Processor (GCP). They are not booted with the PS 300 Graphics Firmware. All letters of the alphabet A through O should appear on the LED displays (or on the 300-baud terminal) as the confidence test steps are completed and the firmware is loaded. (Further characters will appear after the letter O, but these are not related to the confidence test phase of power-up.)

The letters appear at varying time intervals because some tests take longer than others. The audible alarm sounds upon successful completion of the confidence tests.

If the system goes longer than 50 seconds without issuing the audible alarm, a confidence test step has failed, and the power-on sequence must be repeated. The last letter appearing on the LEDs or terminal should give a clue as to what went wrong. For instance, if the last character appearing is L, the diskette is probably not mounted in the drive.

Failures after the confidence tests complete (after O is displayed) are almost always related to problems reading the diskette files.

3.1.4 Graphics Firmware

Following successful completion of the confidence tests, data from the Graphics Firmware diskette load into the system.

CAUTION

Do not transmit to the GCP from the host, keyboard, or any interactive devices while initialization is taking place.

As the various initialization files are loaded and executed, alphanumeric characters are output as with the confidence tests. The following sequence of characters signifies the successful completion of each self-test and initialization step.

P 0 1 2 3 4 5 Q R S T

Once these steps are complete, the assembly date and initialization date for the ACP's code are output. The LEDs (or terminal) should display the following:

PS 300: SYSTEM Vxxx A: (date) I: (date)
(version number and dates will vary)

The PS 300 software then reads in the CONFIG.DAT file from the diskette. When the PS 300 has processed all commands in the configuration file, the SITE.DAT file is loaded. The SITE.DAT file is optional and usually contains a final command that may cause the system to display a message or symbol on the screen. This message or symbol notifies the operator that the system has successfully loaded the SITE.DAT. (Refer to Chapter 2 for information on the SITE.DAT file.) If no SITE.DAT file is found, the message:

SITE.DAT NOT FOUND

is displayed on the screen.

The system has been successfully powered up when:

1. The final command of the SITE.DAT is actuated.
2. The LEDs on the keyboard and control dials are blanked.
3. The Graphics Firmware version (or version of other system software) is displayed at the middle of the Display Screen.

If the loading process is not completed in three to five minutes, some step has failed. A retry sometimes succeeds, especially if the diskette is in marginal condition.

CAUTION

Do not transmit to the GCP from the host, keyboard, or any interactive devices while initialization is taking place.

As the various initialization files are loaded and executed, alphanumeric characters are output as with the confidence tests. The following sequence of characters signifies the successful completion of each self-test and initialization step.

P 0 1 2 3 4 5 Q R S T

Once these steps are complete, the assembly date and initialization date for the ACP's code are output. The LEDs (or terminal) should display the following:

PS 300: SYSTEM Vxxx A: (date) I: (date)
(version number and dates will vary)

The PS 300 software then reads in the CONFIG.DAT file from the diskette. When the PS 300 has processed all commands in the configuration file, the SITE.DAT file is loaded. The SITE.DAT file is optional and usually contains a final command that may cause the system to display a message or symbol on the screen. This message or symbol notifies the operator that the system has successfully loaded the SITE.DAT. (Refer to Chapter 2 for information on the SITE.DAT file.) If no SITE.DAT file is found, the message:

SITE.DAT NOT FOUND

is displayed on the screen.

The system has been successfully powered up when:

1. The final command of the SITE.DAT is actuated.
2. The LEDs on the keyboard and control dials are blanked.
3. The Graphics Firmware version (or version of other system software) is displayed at the middle of the Display Screen.

If the loading process is not completed in three to five minutes, some step has failed. A retry sometimes succeeds, especially if the diskette is in marginal condition.

3.1.5 Loading from Two Diskettes

To load versions of the firmware that require two diskettes, the diskette labeled "A" should be installed in the Drive labeled "0", and the diskette labeled "B" should be installed in the Drive labeled "1" before powering on the Control Unit. If two diskettes are to be loaded from one drive, the system will prompt on the LEDs or debug terminal with:

INSERT ANOTHER DISK AND HIT RETURN

and these steps should be followed:

1. Remove Diskette "A" from Drive 0. The drive light will remain ON.
2. Insert Diskette "B" into Drive 0. Close the latch.
3. Press the RETURN key on the PS 300 Keyboard.

The second diskette will now load from Drive 0, and the power-up confidence tests should complete normally. The LEDs will clear, and the system version will appear in the middle of the Display screen.

3.1.6 Soft Reboot

REBOOT causes the PS 300 to reboot just as if it had been powered up (starts the confidence tests at "A", etc.).

The REBOOT command can appear anywhere; it can occur within BEGIN...END and BEGIN_STRUCTURE...END_STRUCTURE as well as without. It may be named or not. It can not be within a quote or comment because the parser will not find it.)

The command

REBOOT password;

reboots the PS 300 as if from power-up. If no password has been set up, then any character string will do. Otherwise, entering an incorrect password will give an error message.

No GSR procedure will be created for this command. If this command needs to be executed within a program, it can be sent as an ASCII command.

3.1.7 Trouble-Shooting Tips

If the booting process is not complete after three to five minutes, something is wrong. There are several things that should be checked before trying to boot the system again.

1. If after installing the diskette and turning on the Control Unit and Display, the system goes longer than 50 seconds without beeping, a confidence test has failed and the booting process should be repeated. Start over by turning off the Control Unit and taking out the diskette. Then reinsert the diskette, close the drive, and turn the Control Unit back on.

2. Make sure the Display is turned on.

3. Check to make sure that the diskette is properly inserted and that the latch is closed.

4. Check all power and cabling connections. Make sure that the host and debug terminal lines are in the correct ports for the system configuration being booted.

5. Check to make sure that you have the proper diskette in the drive.

6. If the confidence tests do not reach the letter "N" in the alphanumeric sequence, the system will not boot because of a hardware failure. Call your E&S Field Engineer.

7. If the confidence tests run successfully through "N", but the LEDs do not clear and the diskette name and version number do not appear on the Display, the problem is related to trouble in reading the diskette files. If you have another copy of the diskette you are trying to load, attempt to reboot using that diskette.

3.2 POWER-UP CONFIDENCE TEST DEFINITIONS

This section is provided as a reference for the definitions of the characters that are displayed on the PS 300 Keyboard LED's, or on a separate debug terminal connected to Port 3 of the Control Connector Panel, or Port C of the data concentrator during a power-up sequence.

The characters are displayed as successive confidence tests are successfully completed. These definitions may help a site manager in diagnosing minor problems that may occur during a power-up sequence. They can also be used to aid in reporting specific problems to E&S Field Support personnel.

The confidence testing run on the peripheral devices is described in the Interactive Device section of this Volume.

The following diagram illustrates the power-up sequence of the PS 300 Graphics System.

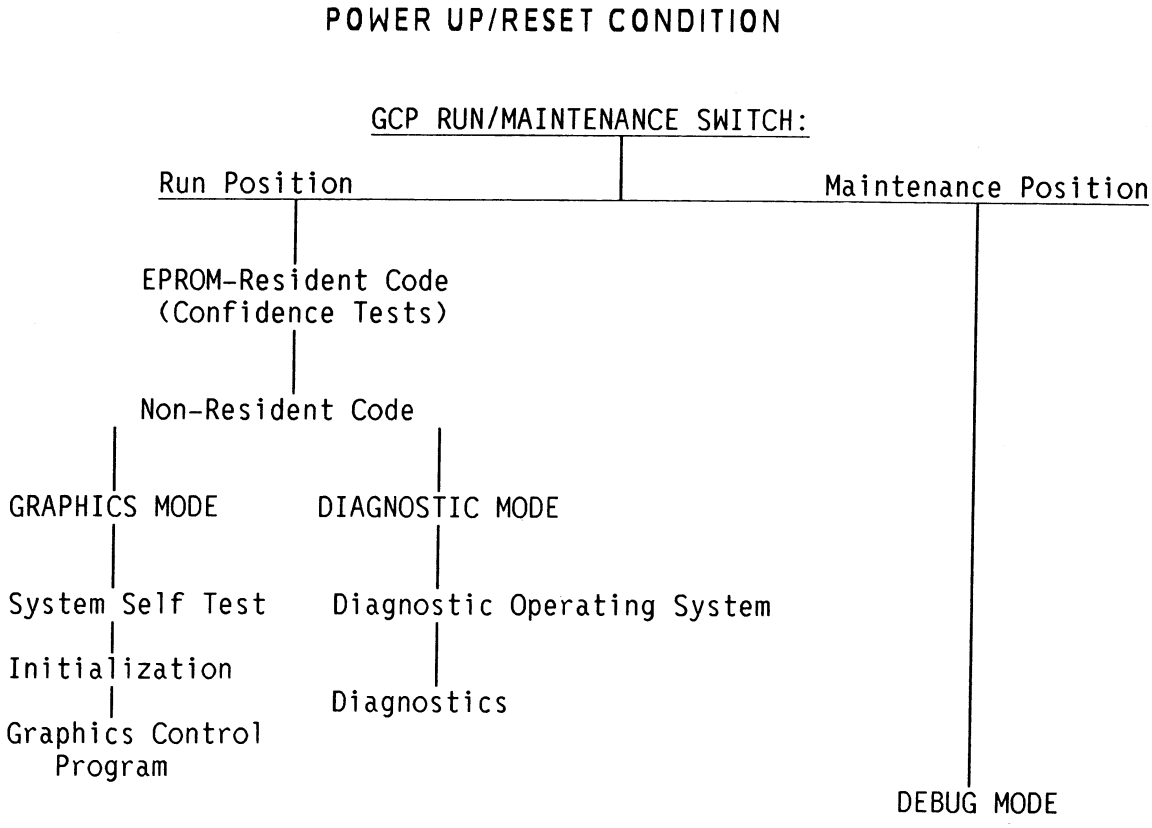


Figure 3-1. Functional Flow Chart

3.2.1 Confidence Tests

The confidence tests are initiated when the GCP receives a Reset signal while in RUN mode. This Reset may be generated either by a power-up condition or by pressing the Reset button on the GCP card. The confidence tests evaluate a minimal hardware configuration and then read in an executable program from the PS 300 floppy diskette. Because some components must work for the confidence tests to run, several components are assumed operational at this point: the MC68000 microprocessor, the two EPROM's, a serial port, and the logic between these components.

The confidence tests perform the following operations:

- Initialize serial output ports.
- Perform a quick check of the MC68000 microprocessor.
- Perform a checksum operation to verify EPROM.
- Verify memory refresh.
- Verify error detection and correction.
- Test local memory.
- Test the floppy diskette.
- Load the boot file from the diskette and begin executing.

Depending upon which diskette has been loaded, the non-resident code that is read in may continue with the self-testing, as the Graphics Control Program does, or the non-resident code may be an operating system for diagnostics. Whenever a confidence test fails, Debug mode is entered automatically.

Whenever the program designated to be read in cannot be located on the diskette, the Debug mode is entered automatically on the debug terminal connected to Port 3. If there is no debug terminal, the user will not see that the debug mode has been entered.

The following section details each of the confidence test steps.

3.2.2 Confidence Test Steps

When the entire test is completed, a BEL character sounded from the PS 300 Keyboard provides an audio indication that the system is functioning correctly. Control is then passed to a program loaded in from the floppy diskette.

Table 3-1. Confidence Test Steps

STEP	DESCRIPTION	OUTPUT
1	A reset signal, generated either by powering up the system or by pressing the reset button, causes the processor to enter supervisor state, to mask all interrupts, and to load the supervisor stack pointer and the program counter from locations X'000000' and X'000004', respectively. Hardware causes these two locations to actually be fetched from locations X'040000' and X'040004' in EPROM. Once the program counter has been fetched, program execution begins at the location specified by the value in location X'000004'.	
2	Immediately after reset, the confidence test initializes serial ports 2 through 5 to use an internal clock to generate a baud rate of 300. The character is set to be 8 bits in length, to have one stop bit and no parity. The letter "A" is sent to each of the four ports to indicate that the processor is indeed running.	A
3	This step turns on the motor(s) on the floppy diskette drive(s) (without selecting a drive) to get diskette rotation up to speed before the diskette is needed. The confidence test continues executing during spinup time.	
4	This step checks the internal registers of the MC68000 CPU and verifies that the ALU inside the processor functions correctly. All registers are checked to verify that each bit can hold both a 0 and a 1 and that no bit affects any other bit. The add, subtract, multiply, and divide instructions are then tested, and all status bits are checked. Upon successful completion of step 4, the stack pointer (register A7) is set at X'10000' (minimum amount of memory to be used in a system). The character "B" is output to indicate successful completion of this step.	B
5	This step performs a checksum calculation on all of the EPROM program code by setting a register to 0 and summing all 16-bit words in the EPROM. A valid checksum equals X'FFFF'. The character "C" is output to indicate successful completion.	C

Table 3-1. Confidence Test Steps - Continued

STEP	DESCRIPTION	OUTPUT
6	This step verifies that refresh is occurring for the dyhamic memory chips of local memory by determining that the refresh zero bit changes to one at least once every 4 milliseconds and that it remains a zero otherwise. The character "D" is output to indicate successful completion of this step.	D
7	This step toggles the external refresh clock line enough to guarantee that all memory has been refreshed at least 8 cycles. (This meets the chip manufacturer's specification that memory must be refreshed 8 cycles before using it.)	
8	This step tests the memory area that holds the exception vectors. (If an exception vector cannot remain correct in memory, any test which may cause an exception is unreliable.) This must be done before the local memory error correction circuitry is tested. The character "E" is output to indicate successful completion.	E
9	Step 9 verifies local memory error-correction-bit generation circuitry by determining that the proper code is generated for each of the sixteen bits of a word. The character "F" is output to indicate successful completion.	F
10	This step verifies that a single-bit error is corrected properly for every bit position of the word and that each bit can be corrected both to a one and a zero. The character "G" is output to indicate successful completion of this step.	G

Table 3-1. Confidence Test Steps - Continued

STEP	DESCRIPTION	OUTPUT
11	This step verifies that a double-bit error is detected properly. It also verifies that a double-bit error indication is not caused by a memory location with no error or by a single-bit error. When testing is completed, memory is initialized so that single-bit errors are corrected, but double-bit errors are ignored. (Double-bit errors must be ignored until step 16 is completed because the level-7 interrupt vector used by the double-bit error will be invalid during testing.) The character "H" is output to indicate successful completion of this step.	H
12	This step verifies that a bus error occurs if the processor receives no DTACK signal by writing to read-only location X'000000'. If a bus error is not generated, the system hangs. The character "I" is output to indicate successful completion of this step.	I
13	This step determines the size of local memory by attempting to access memory at different boundaries and then checking whether a bus error occurs. Only legal memory sizes are checked: 32K, 64K, 128K, and 256K. Once the size has been determined, the stack pointer (register A7) is set to point to the upper boundary. The character "J" is output to indicate successful completion of this step.	J

Due to the following memory test, a pause of approximately 19 seconds (for a 256K system) occurs at this point of testing.

14	This step tests all of local memory including check bits by using the "stripe" test. Initially, the value X'5555' is stored in each memory location. Starting at the lower memory boundary, each location is tested to ensure that all original values are unchanged. Each memory location is then complemented to X'AAAA' and rechecked. All memory locations are then complemented again to X'5555', and the entire testing procedure is repeated. This time, however, the memory test starts from the upper memory boundary.	

Table 3-1. Confidence Test Steps - Continued

STEP	DESCRIPTION	OUTPUT
	This test detects both data and address line errors, as well as individual memory chip errors. After testing is completed, error correction is enabled and the entire local memory area is initialized to zeroes, putting all of local memory into a known state. The character "K" is then output to indicate successful completion of this step.	K
15	Programs loaded from the diskette do not work if interrupts cannot be generated properly. This step verifies that the serial ports are capable of causing an interrupt. If the wrong interrupt vector is used for this interrupt, the system may hang. After the serial port interrupt occurs, the character "L" is output to indicate successful completion of this step.	L
16	This step initializes all exception vectors to point to routines contained in the Debug EPROM. All vectors except the interrupt vectors are unique and indicate the type of exception when they occur.	
17	This step initializes local memory to correct single-bit errors and to detect double-bit errors.	
18	This step initializes the floppy diskette controller chip and selects drive 0. It then verifies that index pulses from the diskette drive are being received and that the read/write head has moved out to track 0. The character "M" is output to indicate successful completion of this step.	M
19	This step reads the header block from the diskette and determines if a valid boot file exists on the diskette. The character "N" is output to indicate successful completion of this step.	N

Table 3-1. Confidence Test Steps - Continued

STEP	DESCRIPTION	OUTPUT
20	Once the boot file has been found, it is loaded into memory. If the start address is 0, the file is loaded at the lowest possible point in memory. Otherwise the file is loaded at the specified address. (Since most boot files should be completely position-independent, it is best to load them at the lowest possible memory location to provide the largest possible amount of stack and heap space in memory.) Once the file has been successfully loaded, the character "0" is output.	0

Up to this point, duration of the confidence test should not exceed one minute.

21	At this point, the system has been fully tested, and the program loaded in from the diskette is ready to run. A BEL character is output to provide an audible indication that the EPROM-based confidence test has completed successfully.	
22	Execution now begins at the first word of the file loaded in from the diskette.	

Graphics Mode

After the confidence test has been completed successfully, self-testing continues under control of the program code that was loaded in from the diskette. This code tests and/or initializes all PS 300 components not tested or initialized by the confidence test (i.e., all serial ports, the clock timer, mass memory). If the system fails to initialize these properly, the "runtime debug" mode is entered and the system prompts the user to see if a restart is desired. If not, debug mode is entered. Once the entire system has been tested and initialized, the Graphics Control Program is loaded, the PS 300 enters the normal mode of operation.

Table 3-2. Confidence Test Steps - Runtime Firmware

STEP	DESCRIPTION	OUTPUT
If you are booting the runtime firmware, the following steps also apply.		
23	Initialize and test serial ports. After mass memory has been tested, "P" indicates there is no bad memory; "*" indicates there is.	P or *
24	These number appear after auxiliary files have been successfully read. 0 - Character font file 1 - parser dictionary file 2 - parser code file 3 - generic function dictionary file 4 - generic function table file 5 - file containing errors and messages A question mark appears instead of a number if a file is not read successfully and the system will not continue booting.	0 - 5
25	A "Q" appears after the files have been read.	Q
26	This character appears after the initial display data structures have been created.	S
27	"T" appears after the initial function network has been created and the clock has been initialized and started.	T
Finally, the system number and ACP microcode dates are displayed.		

CONTENTS

TABLES

4. PS 300/HOST COMMUNICATIONS

This chapter describes the data flow between the PS 300 and the host processor. The initial sections of this chapter introduce some of the basic concepts of data communication, particularly those directly affecting the interface to be set up between the PS 300 Graphics System and the host computer. The standards for the EIA RS-232-C and the EIA RS-449 interfaces are provided. Pin connector definitions and switch settings for the DMR-11AE interface are listed in the *PS 300 DEC/DMR11-AE Interface Installation Manual* in the final section of this Volume. Switch settings for the parallel interface are found in the *PS 300/Unibus Parallel Interface Installation Manual* in the final section of this Volume.

Complete definitions of the system functions referred to in this chapter are found in Chapter 6 of this guide, "System Functions."

4.1 HOST/PS 300 INTERFACE

One of the most important considerations in setting up the configuration characteristics of a PS 300 Graphics System is the interface between the host computer system and the PS 300.

The standard data communication interface to the PS 300 is an asynchronous serial line, RS-232-C or RS-449. The terms "asynchronous" and "serial" refer to two important communication characteristics. Binary data may be transferred between electronic devices in "serial", over a single line, or in "parallel", over several lines at once, by changes in current or voltage. In serial transmission, the bits that represent a character are sent down a single wire, one after the other. (In most data communications applications, serial transmission is preferable to parallel transmission, since fewer wires must be run but parallel transmission is faster as more data can be sent across the line at once.) These serial signals are converted to parallel form at the reception end by shift registers.

Data transfers may be of a "synchronous" nature, where the exact bit framing of each byte of information is coordinated for the entire message by the transmission of two or more synchronization characters at the beginning of the message. All characters that follow these characters occur within a specific time frame called a "character time". Or, data transfers may be of an "asynchronous" nature, where each character is self-defined by the use of a start bit and one or more stop bits. The start and stop bits occur before and after the byte of data. For this reason, this mode of transmission is referred to as "Start/Stop Transmission." In this mode, the arrival time of each character is random. Each end of the transmission line must know what the transmission rate is to sample the line at correct intervals following the receipt of a start bit.

Under PS 300 Graphic System protocol, the RS-232-C and RS-449 standard interface send data signals over a single, serial line using asynchronous transmission. The PS 300 also may be interfaced to a DEC PDP11 or a DEC VAX host with a DEC DMR11-AE and driver over an synchronous serial line or to a DEC VAX over a asynchronous parallel line.

The RS-232-C is a standard of interface communication set by the Electronic Industries Association (EIA). There are newer standards, such as the RS-449, but the RS-232-C is still the standard to which most interfaces are designed. The standard for both RS-232-C and RS-449 contain:

- The electrical signal characteristics.
- The interface mechanical characteristics.
- A functional description of the interchange circuits.
- A list of standard subsets of specific interchange circuits for specific groups of communication system applications.

It is important when reviewing specifications for computer/system interfaces to understand what the various interface leads do, and which are essential for proper interface between the PS 300 Graphics System and the host computer.

4.1.1 RS-232-C Specifications

The physical connection between the PS 300 and the host is made through plug-in, 25-pin connectors (Cannon or Cinch DB Series). These connectors are keyed for 13 pins on the top row, and 12 pins on the bottom row. The PS 300 Ports on the Communication Connector panel provide the male element for the interface. The pin assignments and signal definitions supported by the PS 300 Graphics System are given in Table 4-1.

RS-232-C standard states that the cable run distance between the data communications equipment should be no longer than 50 feet. However, longer cabling distances have been successfully used.

Table 4-1. RS-232-C Connector Pin Definitions

PIN NO	EIA LABEL	ABBREV. NAME	SIGNAL NAME	DIRECTION
1	AA	GND	Protective ground	N/A
2	BA	TXD	Transmit data	To Data Communicatior Equipment (DCE)
3	BB	RXD	Receive data	From DCE
4	CA	RTS	Request to send	To DCE
5	CB	CTS	Clear to send	From DCE
6	CC	DSR	Data set ready	From DCE
7	AB	GND	Signal ground	N/A
8	CF	DCD	Data carrier detect	From DCE
15	DB	TXCA	Transmit clock	From DCE
17	DD	RXC	Receive clock	From DCE
20	CD	DTR	Data terminal ready	To DCE
24	DA	TXCB	External transmit clock	To DCE

For the PS 300 EIA-RS-232-C communication ports, a Control-ON (logical 0), or "SPACE" condition exists if the voltage present is greater than +5 volts and less than +25 volts with respect to signal ground. A Control-OFF (logical 1), or "MARK" condition exists if the voltage present is less than -5 volts and greater than -25 volts with respect to signal ground. This assumes that the PS 300 signal ground and the communication data device signal ground are at the same potential.

Signal Definitions

The following are definitions of the RS-232-C signals shown in Table 4-1.

- AA, AB (Protective Ground and Signal Ground) - These two grounds are electrically identical. They connect to a power ground. No direct frame grounding occurs at the connector. Strict EIA RS-232-C standard definitions are not directly applicable.

- BA (Transmit Data) - Data from the PS 300 are transmitted on this line. The signal is generated by the PS 300 processor.

- BB (Receive Data) - Data are sent to the PS 300 on this line. The signal is passed to the PS 300 via the data communications equipment.

- CA (Request to Send) - This signal is generated by the PS 300 processor. The output may be programmed to conform with EIA-RS-232-C protocol. Generally, an "ON" CA (request to send) signal indicates the PS 300 processor is ready to transmit information.

- CB (Clear to Send) - This signal may be generated by data communication equipment. An OFF condition will terminate data transmission. An ON condition allows data transmission to resume. If no connection is made, an internal pull-up resistor will assert this line to an ON condition (+12V) for non-standard RS-232-C communication.

- CC (Data Set Ready) - This signal may be generated by data communication equipment. The function of this signal is controlled by software within the PS 300 processor. Usually, an 'ON' CC (data set ready) is sent by data communication equipment to indicate that data communications equipment is ready to transmit.

- CF (Data Carrier Detect) - This signal may be generated by a data communication equipment. ON assertion of this signal allows BB (receive data) to be accepted by the PS 300 processor. If no connection is made, this line will be pulled to an ON condition (+12V) to allow non-standard EIA-RS-232-C communication. To disable the BB (receive data) communication, an OFF condition must exist. Definition of this pin is software controlled for Port 0.

- CD (Data Terminal Ready) - This signal is generated by the PS 300 processor and is under software control. When asserted to an ON level, CD indicates that the PS 300 processor is ready to communicate.

- DA TXCB (Transmit Clock B) - This signal is generated by the PS 300 processor. DA provides a timing clock to indicate the center of each element of data. This timing clock can either be equal to the transmitted data frequency, or equal to 16 times the data frequency. DA TXCB is under software control . Port 0 of the PS 300 processor does not directly generate this signal. It relies on DB (transmit clock A), or Port 1 to generate this clock.

- DB TXCA (Transmit Clock BA) – This input signal is generated by external transmitting data communications equipment. This clocking signal input can control the rate at which the PS 300 processor transmits data out. The ability to use this clock input is software controlled.

- DD RXC (Receive Clock) – This input signal is generated by external transmitting data communications equipment. This clock determines the rate at which the PS 300 processor receives data. The ability to use this clock is software controlled.

4.1.2 RS-449 Specifications

RS-449 is a general purpose 37-position interface utilizing a 37-pin connector. In conjunction with RS-422 (balanced) or RS-423 (unbalanced), RS-449 incorporates new electrical characteristics that permit higher data signalling rates, allow greater distance between equipment, and reduce outside noise interference.

Of the two standards for electrical characteristics (RS-422 and RS-423), RS-422 allows the greatest distance, while maintaining high data transmission speed between the host and the PS 300. The balanced voltage-circuit maintains two digital signals (A and B) as opposed to the single-digital signal used in RS-232-C. The "A" terminal of the generator is negative, with respect to the "B" terminal for a MARK, or OFF condition. The "A" terminal of the generator is positive, with respect to the "B" terminal for a SPACE or ON condition. As one signal in RS-422 goes to a SPACE state (ON condition), the corresponding signal goes to a MARK (OFF condition). The data is read by the receiver by comparing both signals. This significantly reduces the effect of outside noise on the line.

RS-449 and RS-232-C are highly compatible, but RS-232-C restrictions apply when it used in conjunction with RS-449. Adapter connectors are available to modify the 37-pin RS-449 connector to the 25-pin RS-232-C connector through major computer product supply centers.

The connector pin definitions for EIA-RS-449 interface supported by Ports 0 and 1 of the PS 300 Communication Connector panel are listed in the following table.

Table 4-2. RS-449 Connector Pin Definitions

PIN NO	TERMINAL	EIA LABEL	E&S LABEL	SIGNAL NAME	DIRECTION
1	A	Shield	GND	Signal Ground	N/A
4	A	SD	SDA	Send Data	To Data Communication Equipment (DCE)
5	A	ST	STA	Send Timing	From DCE
6	A	RD	RDA	Receive Data	From DCE
7	A	RS	RSA	Request to Send	To DCE
8	A	RT	RTA	Receive Timing	From DCE
9	A	CS	CSA	Clear to Send	From DCE
11	A	DM	DMA	Data Mode	From DCE
12	A	TR	TRA	Terminal Ready	To DCE
13	A	RR	RRA	Receiver Ready	From DCE
17	A	TT	TTA	Terminal Timing	To DCE
19		SG	GND	Signal Ground	N/A
20		RC	GND	Receive Common	N/A
22	B	SD	SDB	Send Data	To DCE
23	B	ST	STB	Send Timing	From DCE
24	B	RD	RDB	Receive Data	From DCE
25	B	RS	RSB	Request to Send	To DCE
26	B	RT	RTB	Receive Timing	From DCE
27	B	CS	CSB	Clear to Send	From DCE
29	B	DM	DMB	Data Mode	From DCE
30	B	TR	TRB	Terminal Ready	To DCE
31	B	RR	RRB	Receiver Ready	From DCE
35	B	TT	TTB	Terminal Timing	To DCE
37		SC	GND	Send Common	N/A

RS-449 Ports 0 and 1 on the PS 300 processor are capable of operating as RS-422 balanced communications, or as RS-423 unbalanced operations. For RS-422 balanced or RS-423 unbalanced operation, a MARK or OFF state exists when A terminals are negative with respect to B terminals. An ON, or SPACE condition exists when A terminals are positive with respect to B terminals.

The actual receivers used on the PS 300 have differential inputs. Their positive (+) inputs correspond to A terminals, and their negative (-) inputs correspond to B terminals. These receivers will accept a wide voltage range from +25 volts to -25 volts, and meet RS-449 specifications.

The drivers used by the PS 300 have differential outputs with positive (+) corresponding to A terminals and negative (-) corresponding to B terminals. The output voltages range from +0.5 volts to +2.5 volts minimum. These meet RS-449 specifications.

Connector pins 4 through 17, correspond to A terminals, and an ON condition. Pins 22 through 35 correspond to B terminals.

RS-449 Signal Definitions

The following are definitions of the RS-449 connector pin signals shown in Table 4-2.

- Shield, SG, SC, RC (Receive Common) - These four signals are electrically identical. They connect the PS 300 processor ground with the data communication equipment ground.

- SD (Send Data) - This signal is generated by the PS 300 processor as serial data. The following signals, when connected, need to be "ON" for transmission of serial data:

  - RS - Request to send
  - CS - Clear to send
  - DM - Data mode
  - TR - Terminal ready

Refer to the signal definitions of RS, CS, TR, and DM. If no external connection is made to CS and DM, internal pull-ups will allow SD to function.

- RD (Receive Data) - This signal is generated by data communications equipment as serial data. Receiver circuits are disabled if RR (receiver ready) is connected and in an OFF condition. If no connection is made, internal pull-ups will enable RR and allow reception of data, RD.

- RS (Request to Send) - This signal is generated by the PS 300 processor and is under software control. The RS (request to send) signal when ON indicates information is ready to send.

- RT (Receiver Timing) - This signal is generated by data communication equipment. The clock used is the rate at which the PS 300 processor receives data, signal RD. The use of this clock input is software controlled.

- CS (Clear to Send) - This signal may be generated by data communication equipment. An OFF state will terminate data transmission. An ON state allows data transmission to resume. If no connection is made, internal pull-up resistors will assert an ON state.

- DM (Data Mode) - This signal may be generated by data communications equipment. The function of this signal is controlled by software within the PS 300 processor. Usually, an ON DM signal indicates data communication equipment is ready to send.

- TR (Terminal Ready) – This signal is generated by the PS 300 processor and is under software control. When asserted to an ON condition, TR indicates that the PS 300 processor is ready to communicate, and that the data communication equipment should prepare for data reception.

- TT (Terminal Timing) – This signal is generated by the PS 300 processor and is under software control. TT provides the data communication equipment with transmitted data element timing that corresponds to the SD signal of the PS 300 processor. Port 0 of the PS 300 processor does not directly generate this signal and relies on ST or Port 1 to generate the clock.

- ST (Send Timing) – This signal is generated by external data communication equipment. The input clock can control the rate that the PS 300 transmits data (the SD signal) to the data communication equipment generating ST. The use of this input is software controlled.

4.1.3 RS-232-C and RS-449 Cabling, Connectors and Pins

The user of the PS 300 Graphics System is the supplier of all cabling and connectors to be used in the interface between the PS 300 and the host system. Several factors need to be considered when deciding what standard of interface is to be used. The two basic considerations are:

- Distance between the host and the PS 300

- Level of outside electrical interference

RS-449 was designed to gracefully replace RS-232-C as the standard of data communication equipment interface, and it supports the highest line integrity standards. RS-449, in conjunction with balanced RS-422, should be considered when the distance or noise level dictates that such an interface is necessary.

A null-modem cable configuration may be necessary to correctly connect the pin signals on either RS-232-C or RS-449 interface.

Cables and the 37-pin connectors for RS-449, and cables and the 25-pin connectors for RS-232-C are available through most major computer product supply centers.

The cables running from the host to the PS 300 processor should terminate with a female connector, as the PS 300 data communication ports house male elements.

The decision to use shielded or unshielded cable is left to the user. Shielded cable is highly recommended in noisy environments, but typically it has a higher capacitance per foot than unshielded cable, which may reduce the operating speed.

4.2 ASYNCHRONOUS-SERIAL FULL-DUPLEX DATA CHARACTERISTICS FOR THE PS 300

This section describes the serial I/O parameters the PS 300 Graphics System has defined for each port. The defaults (values assigned to each port when the system is powered on in standard configuration) for the data characteristics are listed in this section. For information on how these values can be configured in a bootable file on the PS 300 Graphics Firmware diskette, refer to the final section in this chapter. The following information applies to PS 300 Graphics Systems asynchronous transmission:

- The baud rates available on Ports 1 through 4 on the PS 300 are: 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 9600, and 19200. Port 5 runs at 19200.

- The PS 300 may be configured for 5, 6, 7, or 8 bits per character, although the host port must pass all characters of the 7-bit ASCII character set (i.e., 7 or 8 bits per character).

- Only one start bit will be accepted (and generated) by the PS 300.

- The PS 300 will accept (and generate) 1 or 2 stop bits.

- The PS 300 and the host can communicate using an X_ON - X_OFF protocol. In this protocol, control sequences are generated that tell the sender (either the PS 300 or the host) when to start (X_ON), or stop (X_OFF) data transmission. These control sequence values default to Control-S (DEC 17 character) for X_ON, and Control-Q (DEC 19 character) for X_OFF. Under X_ON - X_OFF, bit stripping is controlled by the /MASK_TO_7 BITS option.

  Additionally, there are available values for data characteristics that are unique to the X_ON X_OFF protocol. These values and their definitions are shown in the SETUP INTERFACE command section in Chapter 6 of this manual.

- The PS 300 will run with even, odd, or no parity. Parity is a character checking device that operates by adding non-information bits to data, making the total number of ones in each grouping of bits either odd, for odd parity, or even for even parity. This permits error detection for an odd number of incorrect bits in each group.

- Each port may be configured to cause a trap to the PS 300 Debugger in the event a break is detected on that port.

- The PS 300 may be set to hold a maximum number of 127 buffers to hold data transmitted from the host. The default is eight buffers. Each buffer may be set to a maximum of 32,767 bytes, with the default at 48 bytes per buffer. This option allows the user to specify the amount of memory space to be allocated for data reception from the host.

The user may specify the number free input buffers below which the host will be sent an X_OFF to suspend transmission. The number of free buffers above which the host will be sent an X_ON to resume transmission may also be specified.

4.2.1 Synchronous and Asynchronous Port Defaults

The defaults for Ports 0 through 5 are:

Port 0 - Supports the DEC/DMR11-AE Synchronous Interface option. Refer to the **PS 300 DEC/DMR11-AE Interface Installation Manual** for more information on this port and option.

Port 1 - 9,600 baud, 8 bits per character, 1 stop bit, no parity, no_mask_to_7_bits, transparent mode. Sends all X_ON and X_OFF protocol characters, ignores incoming X_ON - X_OFF (no_hear_XON), 8 48-byte buffers with 0 STOP buffers and 1 GO buffer, and debug break disabled.

Port 2 - 9,600 baud, 8 bits per character, 1 stop bit, no parity, non-transparent mode that sends and receives all X_ON and X_OFF protocol characters, 8 48-byte buffers with 1 STOP buffer and 2 GO buffers, and debug break disabled.

For the PS 320, Port 2 is the host port and is configured with the same port defaults as shown in Port 1.

Port 3 - (debug port) 9,600 baud, 8 bits per character, 1 stop bit, no parity, non-transparent mode that accepts all X_ON and X_OFF protocol characters, 8 48-byte buffers with 0 STOP buffers and 1 GO buffer, and debug break enabled.

Port 4 - 300 baud, 8 bits per character, 1 stop bit, no parity, non-transparent mode that accepts all X_ON and X_OFF protocol characters, 8 48-byte buffers with 0 STOP buffers and 1 GO buffer, and debug break disabled.

For the PS 320, Port 4 is configured with the same port defaults as shown in Port 5.

Port 5 - 19,200 baud, 8 bits per character, 1 stop bit, no parity, transparent mode that does not recognize the X_ON X_OFF protocol characters, 8 48-byte buffers, and debug break disabled.

The status of all the ports may be verified by using the SHOW INTERFACE command.

4.2.2 Changing Port Status

The following command sequence can be used to change any of the default values on Ports 1 through 5. These new values must be within the acceptable values for data characteristics as given in the previous section.

The port values are changed by entering the command:

SETUP INTERFACE <name>/<options>;

where name is the port being reconfigured, options refers to the option setting the communications interface. The command:

SHOW INTERFACE <name>;

where <name> is the port, can be used to check the values of a given port.

In using these commands, the ports names are as follows:

Port 1 is designated port10
Port 2 is designated port20
Port 3 is designated port30
Port 4 is designated port40
Port 5 is designated port50

The available options for SETUP INTERFACE are:

/SPEED=<baud rate> - input and output communications speed between 50 and 19200.

/EVEN_PARITY - establishes monitoring of parity on input and generation of parity on output, using EVEN parity.

/ODD_PARITY - establishes monitoring of parity on input and generation of parity on output, using ODD parity.

/NO_PARITY (default) - terminates the monitoring of parity on input and generation of parity on output.

/BITS_PER_CHARACTER=<number of bits per char> - sets the width of a character in bits (normally 8, even for 7-bit ASCII).

/STOP_BITS_PER_CHARACTER=<number of stop bits per char> - sets the number of stop bits for each character (normally 1).

/XON_XOFF - enables the PS 300 to use X_ON and X_OFF protocol to tell the host (or device) on this port to resume or suspend transmission. Default is to this protocol.

/NO_XON_XOFF - disables the use of XON and XOFF protocol from the PS 300 to the host (or device) on this port to resume or suspend transmission.

/HEAR_XON - enables the use of XON_XOFF protocol for the host (or device) on this port to tell the PS 300 to resume or suspend transmission. Default enables HEAR_XON.

/NO_HEAR_XON - disables the use of XON_XOFF protocol for the host (or device) on this port to tell the PS 300 to resume or suspend transmission.

/BREAK - enables the receipt of a BREAK on this port to call the ROM debugger.

/NO_BREAK - disables the receipt of a BREAK on this port to call the ROM debugger. Default is NO_BREAK.

/SPEED_EXTERNAL - sets the port speed to that of an attached modem, rather than from an internal clock. (This applies only to those ports with full modem support.)

/NO_SPEED_EXTERNAL - tells this port to use its internal clock, at the speed set by /SPEED=. Default is NO_SPEED EXTERNAL.

/BUFFERS=<number of buffers> - specifies the number of buffers in the input pool. Default is 8 buffers.

/BUFFER_SIZE=<number of bytes> - specifies the size of each buffer in the input pool. Default is 48 bytes.

NOTE

If input is received continuously, buffers will be filled until they are full. The buffer size will, in this case, specify the quantum of input being processed by subsequent functions.

If input is received at much less than the maximum baud rate, buffers will be released to waiting functions after 2 character times without receipt of a byte. In this case, the strict product of <buffer size> and <number of buffers> will not be the true amount of input buffering.

/N_STOP_BUFFERS=<number of buffers> - specifies the number of free input buffers below which the sender is told to suspend transmission. This has no effect unless the port is in /XON_XOFF mode. Default is 1 Stop Buffers. This is for host to PS 300 communication only.

/N_GO_BUFFERS=<number of buffers> - specifies the number of free input buffers above which the sender is told to resume transmission. This has no effect unless the port is in /XON_XOFF mode. Default is 2 Go Buffers.

The following four commands allow the user to specify non-standard X_ON-X_OFF characters:

/SEND_XON_CHAR=<char code> - specifies the character code as an integer (defaults to decimal 17) to be sent out from the PS 300 to tell the sender to resume transmission. This has no effect unless the port is in /XON_XOFF mode.

/SEND_XOFF_CHAR=<char code> - specifies the character code as an integer (defaults to decimal 19) to be sent out from the PS 300 to tell the sender to suspend transmission. This has no effect unless the port is in /XON_XOFF mode.

/OBEY_XON_CHAR=<char code> - specifies the character code as an integer (defaults to decimal 17) that, when received by the PS 300, allows the PS 300 to transmit.

/OBEY_XOFF_CHAR=<char code> - specifies the character code as an integer (defaults to decimal 19) that, when received by the PS 300, stops the PS 300 from transmitting.

/MASK_TO_7_BITS - specifies that incoming bytes are to have their 8th bit, normally the parity bit, stripped off.

/NO_MASK_TO_7_BITS - (default) specifies that incoming bytes are not to be masked.

/BREAK_TIME=<break time> - specifies the length of time in centiseconds that an outgoing BREAK is to be held. This defaults to 10. Maximum = 127.

/ASYNCHRONOUS - normal mode of operation.

/SYNCHRONOUS - Supported by Port 0 using the DEC/DMR-11AE Interface.

All commands are terminated with a semicolon (;) and a carriage return.

The menu available with the SHOW INTERFACE command lists only those parameters that are relevant to the interface; for example, in synchronous mode, the X_ON/X_OFF parameter would not be listed.

4.2.3 Changing PS 300/Host Interface Values Using the SITE.DAT

These port values may be changed to suit specific site requirements in two ways: the default values can be changed by using the SETUP INTERFACE commands in CI mode, or the SETUP INTERFACE commands can be entered into the SITE.DAT file. If the value needs to be changed for just one session, so that the port will go back to its default values during the next boot-up, the SETUP INTERFACE command can be entered during a PS 300 session.

Should the new port value need to be installed more permanently, with the new value booted instead of the default, the SETUP INTERFACE commands should be entered into the SITE.DAT file.

Any of the SETUP INTERFACE commands can be entered in the SITE.DAT file, using the following forms:

```
SETUP INTERFACE portn/value;
```

where n is the port name and value is the name of the feature being set. Example:

```
SETUP INTERFACE port10/XON_XOFF;
```

would enable Port 1 to use X_ON and X_OFF protocol to tell the host (or device) on this port to resume or suspend transmission.

```
SETUP INTERFACE portn/value=<p>;
```

where n is the port name, and <p> is the specified parameter.

Example:

```
SETUP INTERFACE port40/SPEED=2400/XON_XOFF;
```

would set Port 4 to a baud rate of 2400 and enable Port to use X_ON and X_OFF.

4.3 PS 300 TRANSMISSION PROTOCOL

The PS 300 Graphics System uses an X_ON-X_OFF handshaking protocol to maintain orderly data communication over a full duplex, asynchronous, serial line between itself and the host computer. The receiver of X_OFF (dec 19) is to suspend transmission as soon as possible. The receiver of X_ON (dec 17) is being prompted to resume transmission until the next reception of X_OFF. The PS 300 will suspend transmission within one character time and can accept up to one buffer full of characters after X_OFF is sent.

The following equation shows how many bytes of an empty buffer are left when an X OFF is sent. An X_OFF will be sent to the host that many bytes before input buffering is exhausted.

((Number of STOP buffers +1) * Number of bytes/buffer) - 1

4.3.1 Data Reception

The PS 300 defaults to eight 48-byte buffers available to receive data from the host computer. Transmitted characters are placed in the first free buffer starting in the first position and continue to the end of the buffer. When the buffer is full, the next available buffer is used. If all allocated buffers are full, the PS 300 will drop everything off the line until a buffer is free.

When the X_ON - X_OFF protocol is used, the PS 300 will send an X_OFF to the host (sender), when the number of free buffers is equal to the number of STOP buffers. The PS 300 will send X_ON to the host when the number of free buffers is equal to the number of GO buffers.

4.3.2 Data Transmission

X_OFF received on the host input port disables data transmission from the host to the PS 300 until the PS 300 sends X_ON. If a host transmission aborts before X_ON is transmitted, or if the host transmits X_OFF as part of the LOGOFF message, it is necessary to manually clear the X_OFF condition. X_OFF is cleared, and the port re-enabled for transmission whenever a SETUP or SHOW INTERFACE command is executed.

Rebooting the PS 300 will also clear the X_OFF condition.

Default for the PS 300 is NO_HEAR_XON_XOFF.

4.3.3 Transmission Errors

If the X_ON-X_OFF protocol is not used, and the number of available buffers is not large enough to hold the incoming data from the host (sender), data characters will be lost. These lost characters are detected and counted by the input routines. The SHOW INTERFACE command will give the current error counts for each port.

Messages characteristic of lost input characters are:

- PARSER SYNTAX ERROR due to bad syntax generated by the lost characters
- ERROR E 12 *** Message which function cannot handle

4.3.4 Data Transmission Without X ON-X OFF

Operation without support of the X_ON - X_OFF protocol is discouraged. If X_ON - X OFF protocol is not available on the host, it is up to the user to ensure that an adequate number of buffers are allocated for data reception on the PS 300.

4.4 TRANSMISSION ERROR DETECTION

The Enhanced Programmable Communications Interface (EPCI) used on PS 300 Ports 1 through 5, are able to detect three types of transmission errors. When one of these transmission errors occurs, a bit is set in the EPCI status register where it can be read by the Graphics Control Processor. The errors detected are:

- Parity errors (if parity is enabled)
- Framing errors
- Overrun errors

The SHOW INTERFACE command will display all errors detected from the last PS 300 boot.

4.4.1 Parity Errors

The parity bit follows the character bits in data transmission. If there are 7 bits/characters, and parity is enabled, the total number of bits is 8 with the parity bit being the last transmitted bit. Ignoring the start bit and stop bit(s), the letter "A" when transmitted with EVEN parity would appear as follows:

lsb						msb	
1	2	3	4	5	6	7	parity
1	0	0	0	0	0	1	0

where "lsb" is the least significant bit and "msb" is the most significant bit.

The same character transmitted with ODD parity would look like this:

lsb						msb	
1	2	3	4	5	6	7	parity
1	0	0	0	0	0	1	1

EVEN parity sets the state of the parity bit such that the number of 1s in the 8 bits is an even number.

ODD parity sets the state of the parity bit such that the number of 1s in the 8 bits is an odd number.

If parity is enabled, the EPCI determines the parity of the received character and compares this parity with the parity bit transmitted. If they do not agree, the parity error flag is set in the EPCIs status register.

From the example of the character "A", it can be seen that if the host and the PS 300 do not agree on the parity they are using, every character received or transmitted will generate a parity error.

This vertical error detection scheme can only discern an odd number of bit errors. For example, if bits 2 and 3 are erroneously changed to 1s, so that the character transmitted appears to be:

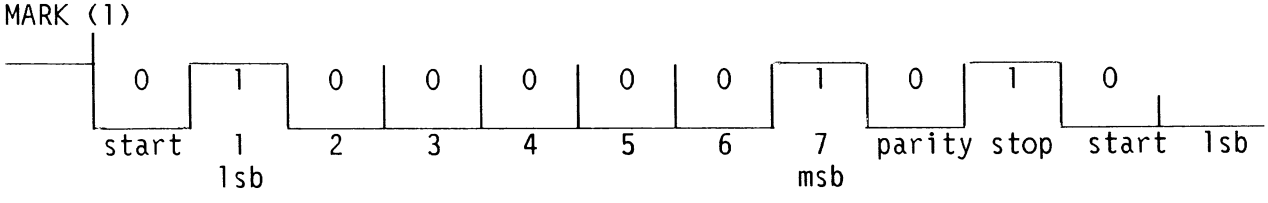
lsb						msb		
1	2	3	4	5	6	7	parity	
1	1	1	0	0	0	1	0	EVEN parity

the parity bit is correct for the character received ("G") but incorrect for the letter sent ("A").

The PS 300 supports ODD and EVEN parity, or NO parity.

4.4.2 Framing Errors

"Framing" is the process of determining which group of bits constitute a character. An error in this process is called a "framing error". Characters are framed by the start bit and the stop bit(s). Looking at the character "A" again (assume one stop bit):



The line is held in a MARK condition with current flowing when characters are not being transmitted. If for some reason the EPCI failed to detect the start bit when the signal goes to an ON, or SPACE condition, it is possible that it would assume bit 2 was the start bit, and bit 3 was the lsb, etc. At the time EPCI expected to see a stop bit, it would instead see the lsb of the next character, and a framing error would occur. When a framing error does occur, the EPCI sets the framing-error flag in the status register.

4.4.3 Overrun Errors

An overrun error occurs when the Graphics Control Processor (GCP) fails to read the characters in the holding register of the EPCI before the next character received is placed in the holding register. When this happens, the EPCI will overwrite the contents of the holding register with the next character. This overwrite causes the overrun error flag to be set in the EPCI status register.

4.5 METHODS OF COMMUNICATION OVER THE HOST LINE

Volume 1 of this document set describes the various methods of data communication that can be used over the PS 300/host line. These include standard ASCII transmission or E&S supplied host-resident software packages, the PSIOs and the GSRs. Volume 1 provides a brief description of the routing bytes that are used to channel data to the appropriate PS 300 system functions. The routing bytes and their channels are described in Chapter 5, "Local PS 300 Data Flow."

The software packages use routines on the host processor to package data prior to sending it in binary format to the PS 300. These routines use count mode (described in the next section) and the routing bytes required to channel the data to the proper PS 300 system function are embedded in the data by the routines. The routines build data 'packets' that include all the necessary information to process the data, and are in a form that is immediately acceptable by the PS 300 system function, F:CIROUTE.

In all cases, F:CIROUTE expects to receive data in a specific format called packets. These packets may be in either ASCII or binary, and for asynchronous communication, may be in either count or escape mode. When the software packages are not used for host/PS300 communication, the PS 300 system functions that interface between the system and the hardware interface, (data reception functions, such as F:DEPACKET for asynchronous, ASCII communication) are responsible for building the data packets. Over the DEC DRM-11 interface or the parallel interface, these packets are sent only in count mode. The following sections deal with the use of count and escape mode in asynchronous data transmission.

4.6 DATA COMMUNICATIONS - ESCAPE AND COUNT MODE

Data may be transported over an asynchronous line in two modes: escape mode or count mode. The mode used is dependent on application and can be selected by the user. Count mode is the faster mode, as the system function, F:DEPACKET, that converts a stream of bytes into a stream of packets does not have to check the identity of each byte.

Data is sent to the PS 300 from the host as a stream of bytes. These bytes must contain information that is intelligible to PS 300 system functions about the nature of the message and where it is to be sent internally in the PS 300. The descriptions that follow describe the data transfer modes used in host/PS 300 communication and briefly describe the system functions that accept, examine, and route data internally in the PS 300.

A system function, F:DEPACKET, accepts data input to the PS 300 from the host. F:DEPACKET converts a stream of bytes from the host into a stream of Qpacket/Qmorepacket. A Qpacket is a block of character data that can be sent from one PS 300 function to another. When data comes from the host through the F:DEPACKET function, it contains a byte for routing control. A Qmorepacket is a Qpacket that when coming from the host through F:DEPACKET, has no routing byte (i.e. a Qmorepacket has the same destination as the previous Qpacket.)

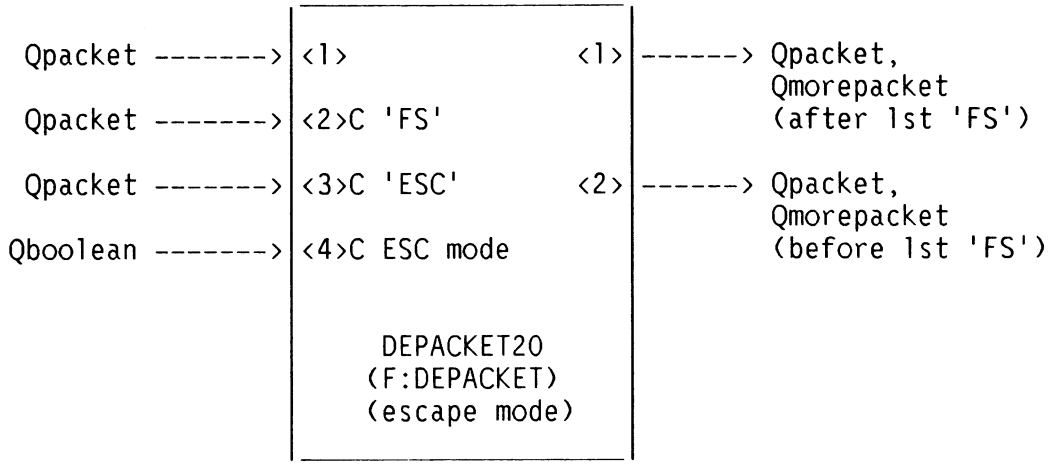
There are two instances of the F:DEPACKET function. The first, DEPACKET0, accepts all incoming bytes from the host on input <1>. It channels all incoming data through to output<2> until it sees the Start of Packet (SOP) character <ACK> (ACKNOWLEDGE - decimal character code 06 - ↑F) that signifies the start of a count mode packet.

All the data sent through to output<2> of DEPACKET0 are sent to input<1> of the second DEPACKET function, DEPACKET20, which then checks all incoming data for the SOP character <FS> (Field Separator - decimal character code 28 - ↑\) that signifies the start of an escape mode packet. It will also route all incoming bytes out output<2> until it sees the <FS> character. Output <2> of DEPACKET2 is connected to ES_TE1 (the screen).

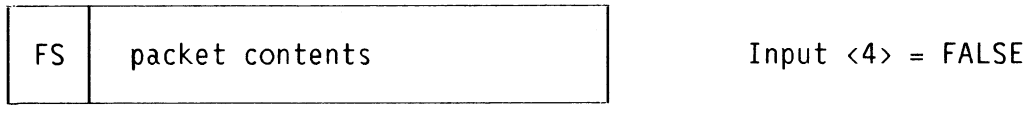
These instances of F:DEPACKET are described below. The characters that are used to signify SOP (<FS> and <ESC> characters) may be changed by the user by sending the new characters to the correct inputs of F:DEPACKET.

4.6.1 Escape Mode

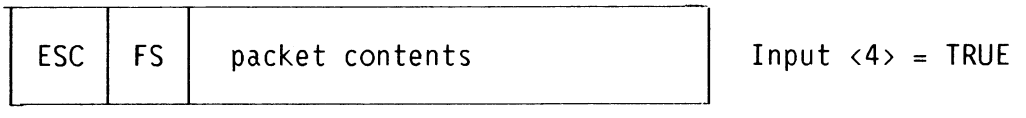
In escape mode, F:DEPACKET looks at every byte to see if it is a Start of Packet (SOP) character, which by default in escape mode is the ASCII Field Separator <FS> character, or an <ESC> character.



In escape mode, F:DEPACKET assumes that a packet is defined as either:



or



where <FS> represents the Start of Packet character that is by default the decimal character code 28 (↑\).

The definition of FS (one character) is taken from a single character Qpacket on input <2>.

In the first mode (input <4> = FALSE), any FS or ESC characters within the message packet must be escaped by prefixing them with an ESC character (i.e. the <ESC> character, decimal character code 16 (↑P)). Thus <ESC><x> becomes <x> for all values of x.

In the second mode (input <4> = TRUE), only ESC characters within the message packet must be escaped by prefixing them with an ESC character. The ESC character is defined by a single character Qpacket on Input <3>.

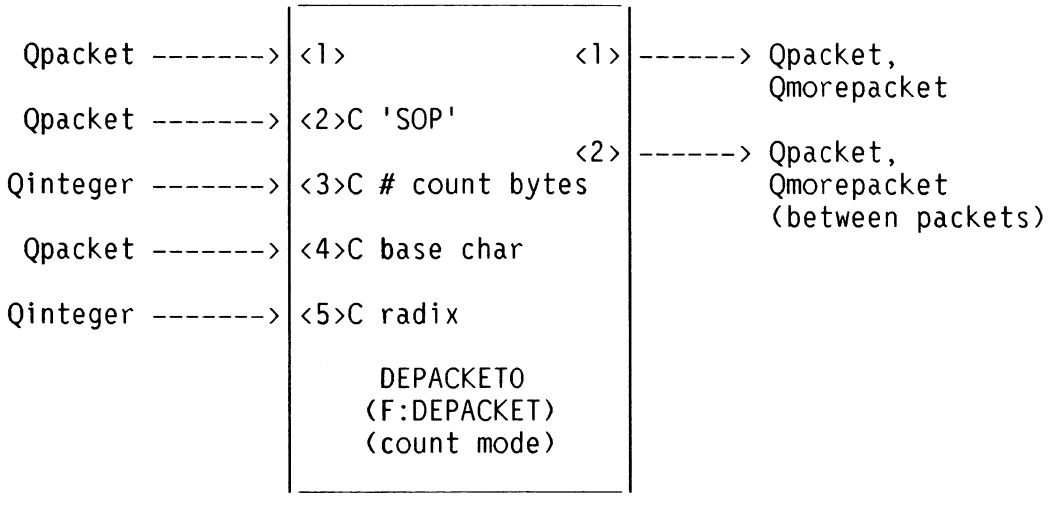
Output <1> outputs Qpacket and Qmorepackets of any messages after the first SOP control character is received. Output <2> outputs Qpackets and Qmorepackets of any messages before the first SOP control character is received. A Qpacket is output on Output <1> each time a SOP control character is received. Otherwise Qmorepackets are output.

Output <2> is normally connected the Terminal Emulator Input and Output <1> is connected to F:CIROUTE for both Count and Escape Modes.

The routing path will be used for data transfer until the multiplexing function sees another <SOP> character, and a packet with another routing byte.

4.6.2 Count Mode

In count mode, once the SOP <ACK> character is seen, F:DEPACKET merely counts the bytes until the count is reached. No attempt is made to decode any bytes until the count is reached. Because F:DEPACKET does not examine the data, it is faster than escape mode, where all bytes are checked by F:DEPACKET to see if they are <FS> or <ESC> characters. Also, count mode allows for the use of any <SOP> or <ESC> sequences as part of the data.



In count mode, F:DEPACKET assumes that a packet is defined as:

ACK	count bytes	packet contents

where <SOP> represents the Start of Packet character that is by default the the ASCII ACK character, decimal character code 06 (↑F).

The definition of SOP (one character) is taken from a single character Qpacket on input <2>.

The message count is defined by n bytes (n defined by the Qinteger on input <3>). Each count byte is offset from the base character (the base character is taken from a single character Qpacket on input <4>). After the base character is subtracted, each count byte becomes a digit of the message count whose radix is defined by the Qinteger on input <5>.

Output <1> outputs Qpackets and Qmorepackets of count mode messages. Output <2> outputs Qpackets and Qmorepackets of any messages which are not in count mode.

The <SOP> byte and the count bytes are removed from the start of the packet before the packet is sent to F:CIROUTE, that does the actual routing.

4.7 CHANGING THE <ESC>, AND/OR <SOP> SEQUENCE CHARACTERS IN THE SITE.DAT FILE

If the <ESC>, and/or <SOP> sequence characters used by E&S are incompatible with the host, or have another site-specific value, these characters can be changed by sending new values for these sequences to an instance of F:DEPACKET in the PS 300.

These new values MUST be included as PS 300 commands in the SITE.DAT file that is loaded during the system power-up. These commands should never be sent down from the host or entered in from the PS 300 keyboard during host transmission.

NOTE

If the <ESC> or <SOP> characters are changed in the SITE.DAT file, this change must be incorporated in the I/O Host-Resident Subroutines code for the PS 300, as the Subroutines use these same sequences for routing.

The PS 300 command for changing the escape mode <SOP> (default is <FS>, decimal character code 28, ASCII character '↑\') character is as follows:

```
SEND CHAR(I) to <2>DEPACKET20;
```

where I is the integer value corresponding to the new <SOP> character in escape mode.

The PS 300 command for changing the escape mode <ESC> character is as follows:

```
SEND CHAR(I) TO <3>DEPACKET20;
```

where I is the integer value corresponding to the new <ESC> sequence.

The count mode <SOP> character, (ASCII <ACK>, decimal character code 06 or ↑F), can be changed by sending the new integer value to <2>DEPACKET0:

```
SEND CHAR(I) TO <2>DEPACKET0;
```

4.8 PS 300 LOCAL DATA FLOW

The next function to accept the data is the system function F:CIROUTE. This function and an overview of local data flow in the PS 300 is discussed in the next chapter.

CONTENTS

TABLES

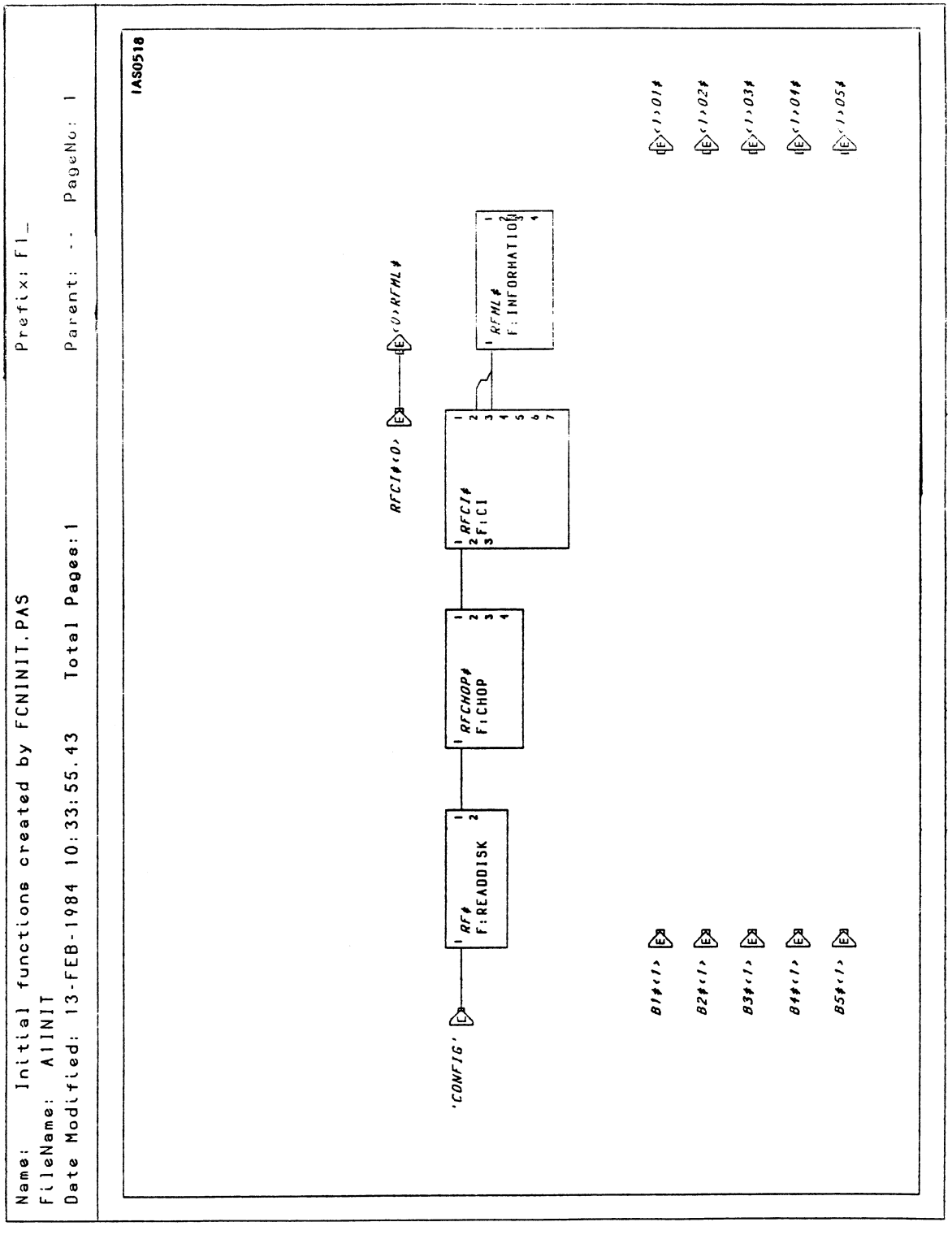
5. LOCAL DATA FLOW AND SYSTEM NETWORKING

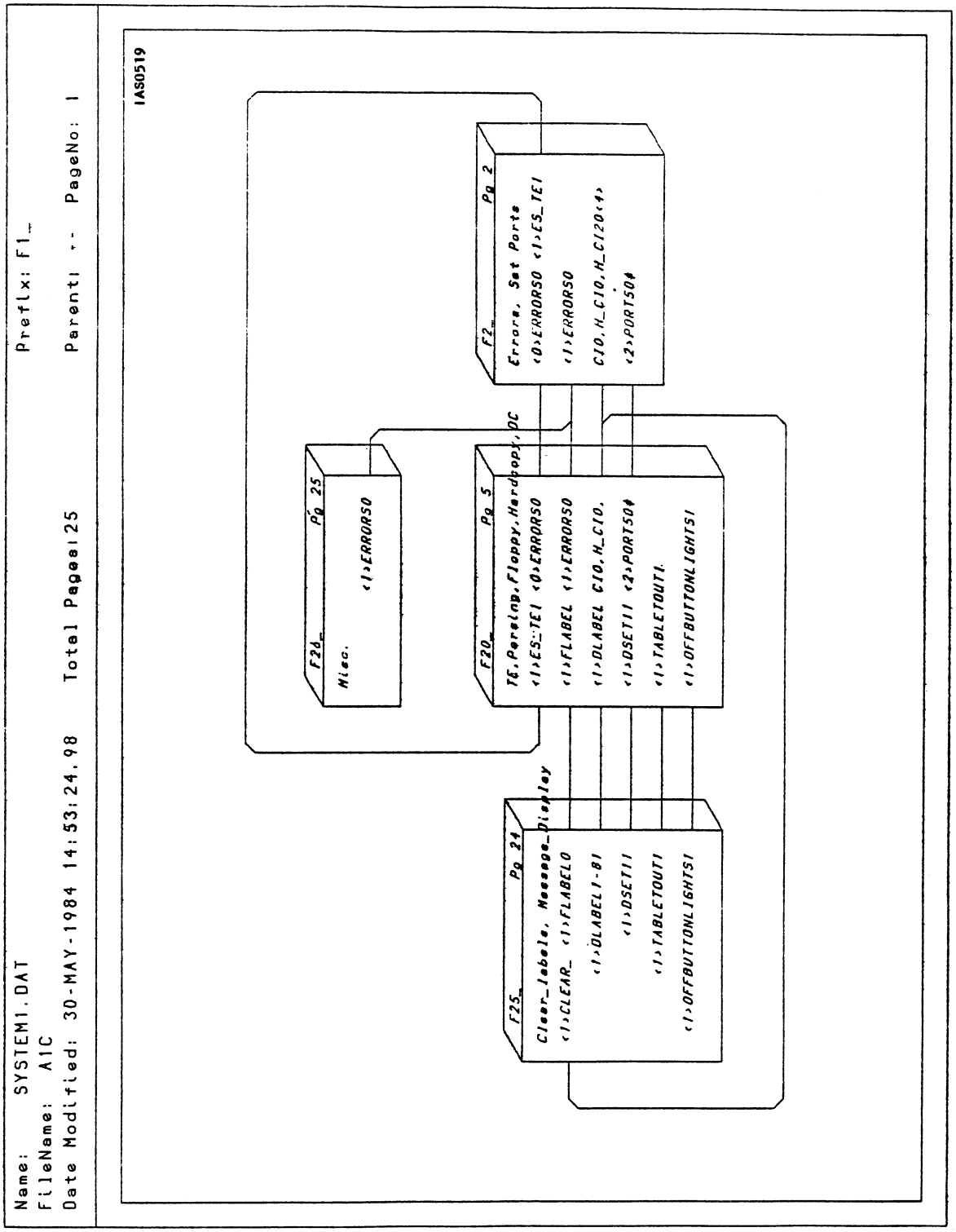
This chapter discusses internal data flow in the PS 300. Function names that appear in capital letters in this chapter are instances of system functions. The generic system function appears with the "F:" prefix. System function definitions are provided in Chapter 6 of this guide. This chapter includes block diagrams of data flow through the PS 300 system function network, a description of routing functions and routing bytes, the channels that data can be routed to, and the terminal emulator network.

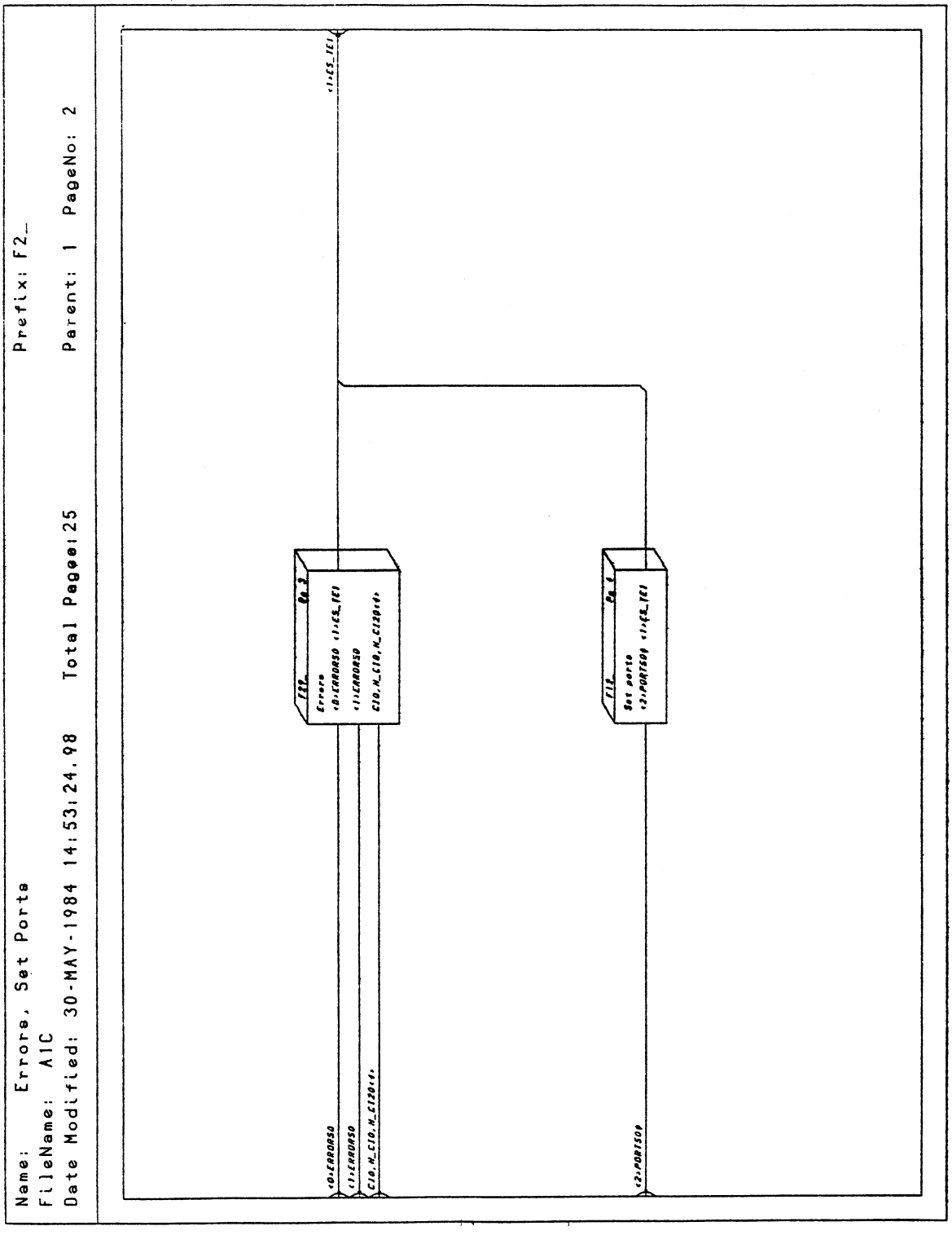
5.1 DATA FLOW BLOCK DIAGRAM

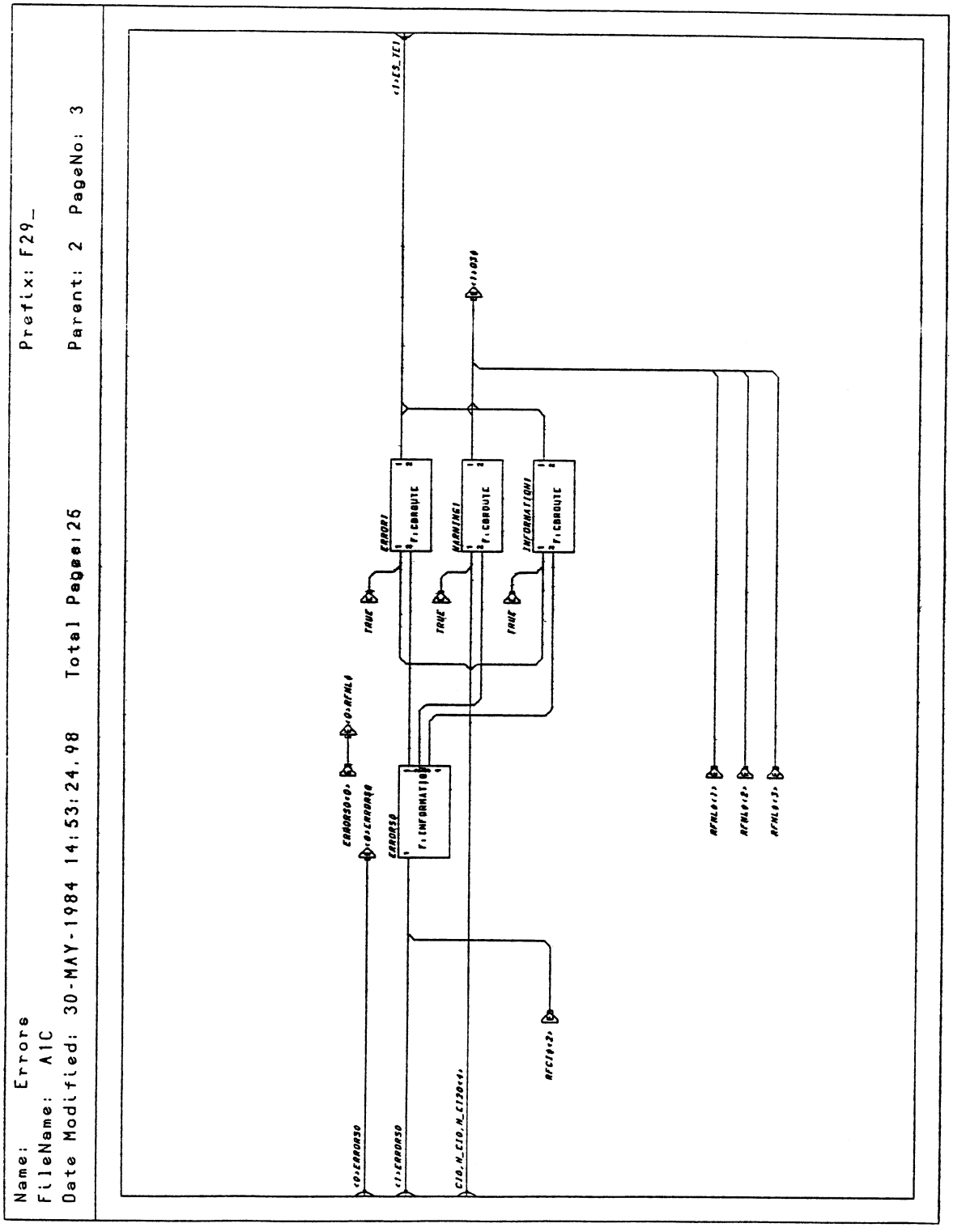
The following block diagrams show the flow of internal data through the PS 300 system function network in the PS 300 from the point of entry, B1$ for Port 1 or BIN56K1$ for Port 0 (any of the data block in functions), to HOST_MESSAGEB. Networks that support the PS 340 system, Parallel Interface, and DMR11-AE Interface are shown following the standard system network diagrams. The sections following the diagram discuss specific instances of functions that direct data flow in the PS 300.

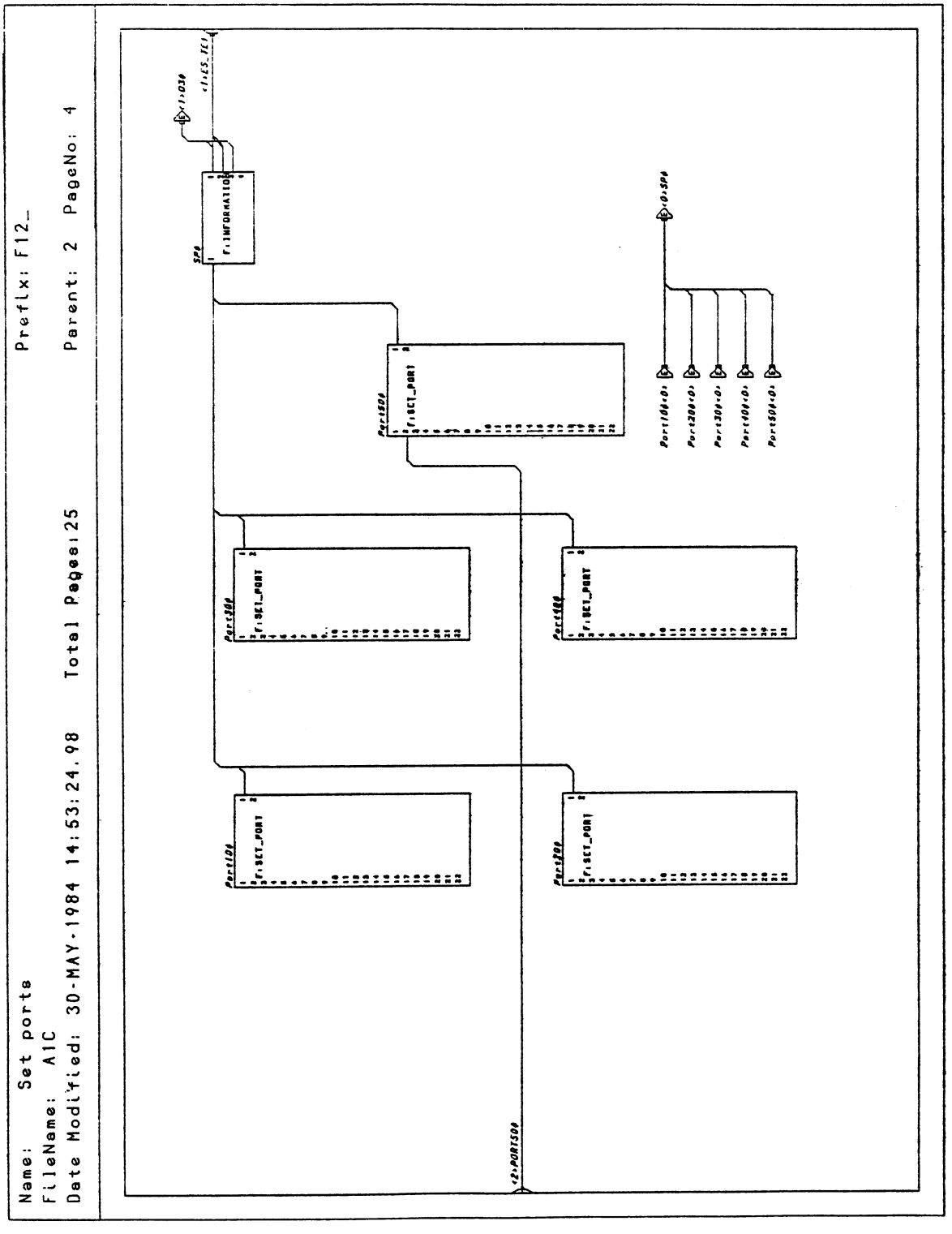
Prior to receiving data that is described as a packet (containing <SOP> characters and routing bytes) the PS 300 sends all data to ES_TE, an instance of F:VT10, that is part of the terminal emulator network. This is done to insure that any error messages sent down from the host are displayed on the PS 300 screen as text. After receiving the first SOP character, all incoming packets are then routed according to their multiplexing byte.

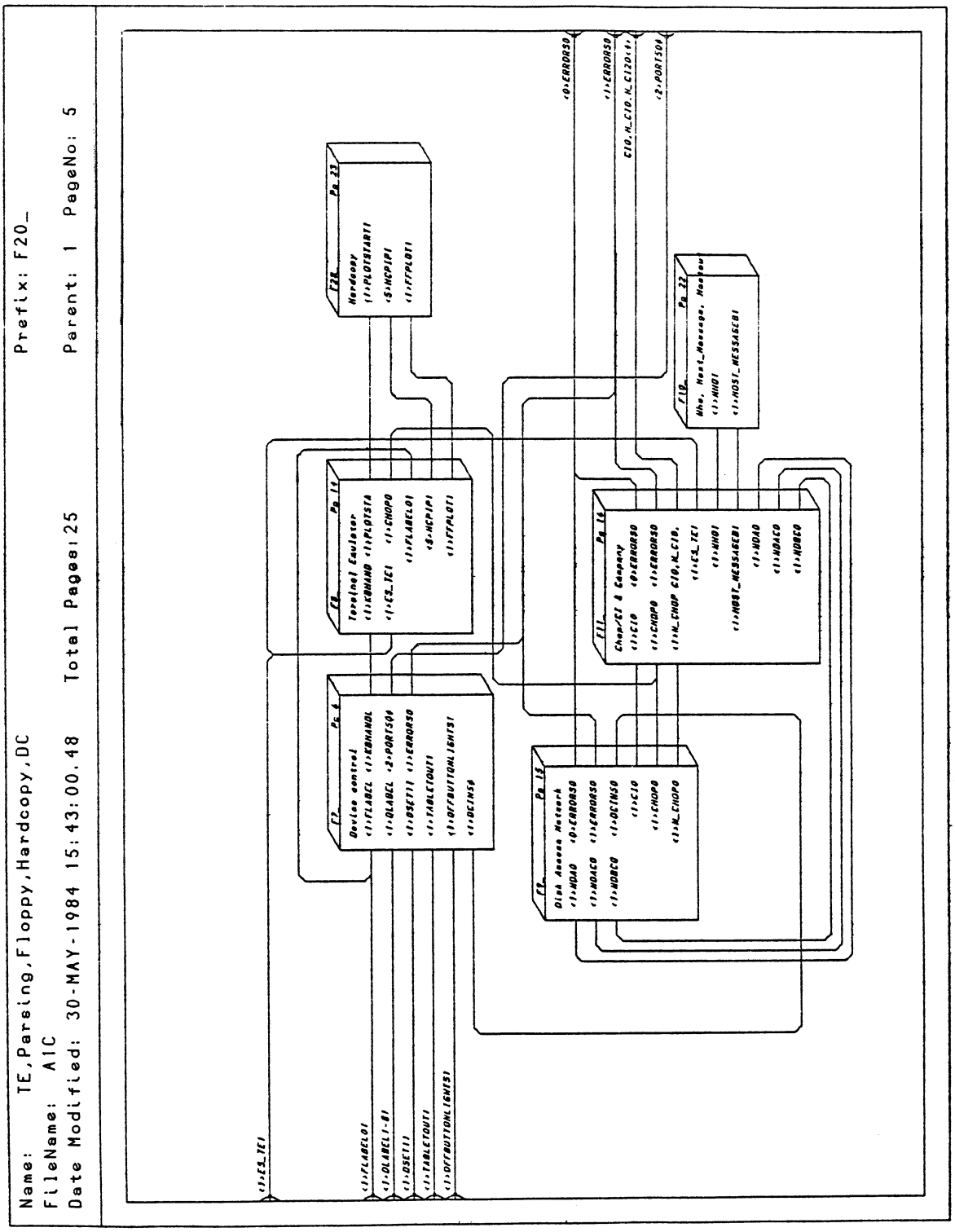

Name: Initial functions created by FCNINIT.PAS
FileName: AllNIT
Date Modified: 13-FEB-1984 10:33:55.43
Prefix: Fl_
Parent: -- PageNo: 1
Total Pages: 1
IAS0518
'CONFIG'
RF#
F:READDISK
RFCHOP#
F:CHOP
RFCI#
F:CI
RFCI#<0>
<0>:RFHL#
RFHL#
F:INFORMATION
B1#<1>
B2#<1>
B3#<1>
B4#<1>
B5#<1>
<1>:01#
<1>:02#
<1>:03#
<1>:04#
<1>:05#

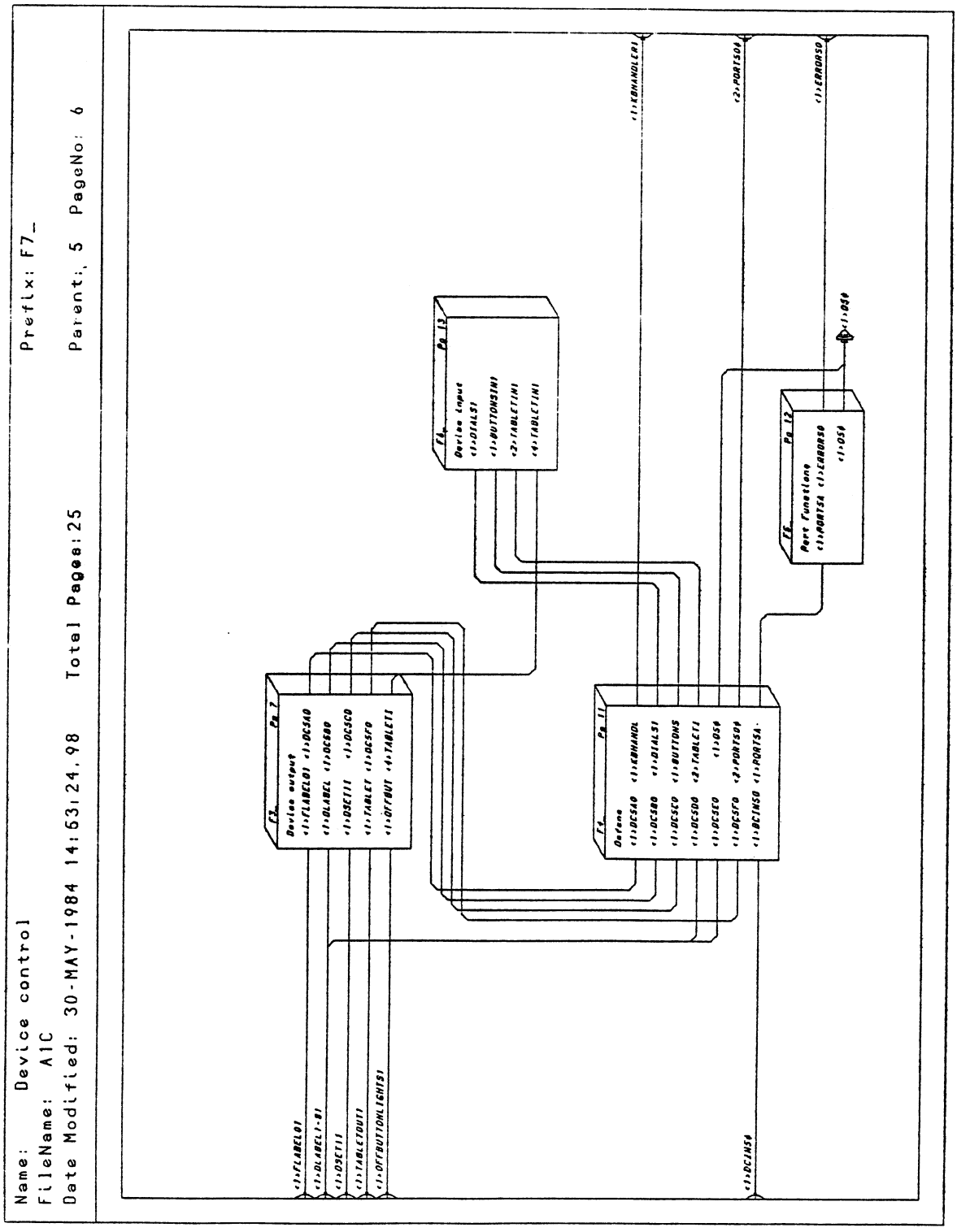

Name: SYSTEM1.DAT
FileName: A1C
Date Modified: 30-MAY-1984 14:53:24.98
Total Pages: 25
Prefix: F1_
Parent: -- PageNo: 1
IAS0519
F2_ Pg 2
Errors, Set Ports
<0>ERRORS0 <1>ES_TE1
<1>ERRORS0
CIO.H_CIO.H_CI20<4>
<2>PORTS0#
F20_ Pg 5
TE.Parallel.Floppy.Harddisk.DC
<1>ES-TE1 <0>ERRORS0
<1>FLABEL <1>ERRORS0
<1>DLABEL CIO.H_CIO.
<1>DSET11 <2>PORTS0#
<1>TABLETOUT1.
<1>OFFBUTTONLIGHTS1
F24_ Pg 25
Misc.
<1>ERRORS0
F25_ Pg 24
Clear_labels, Message_Display
<1>CLEAR_ <1>FLABEL0
<1>DLABEL1-81
<1>DSET11
<1>TABLETOUT1
<1>OFFBUTTONLIGHTS1

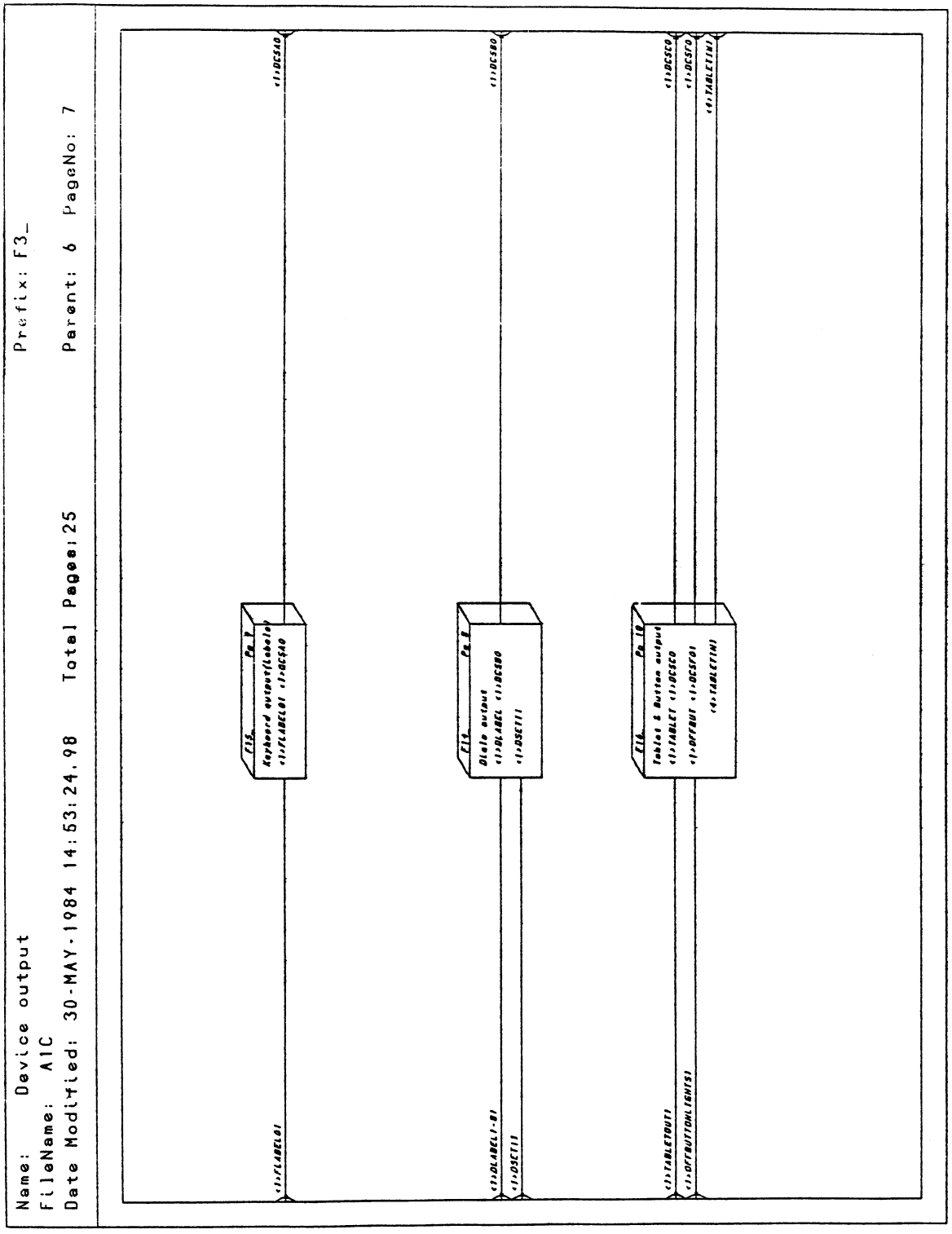

Name: Errore, Set Porte
FileName: A1C
Date Modified: 30-MAY-1984 14:53:24.98
Total Pages: 25
Prefix: F2_
Parent: 1 PageNo: 2

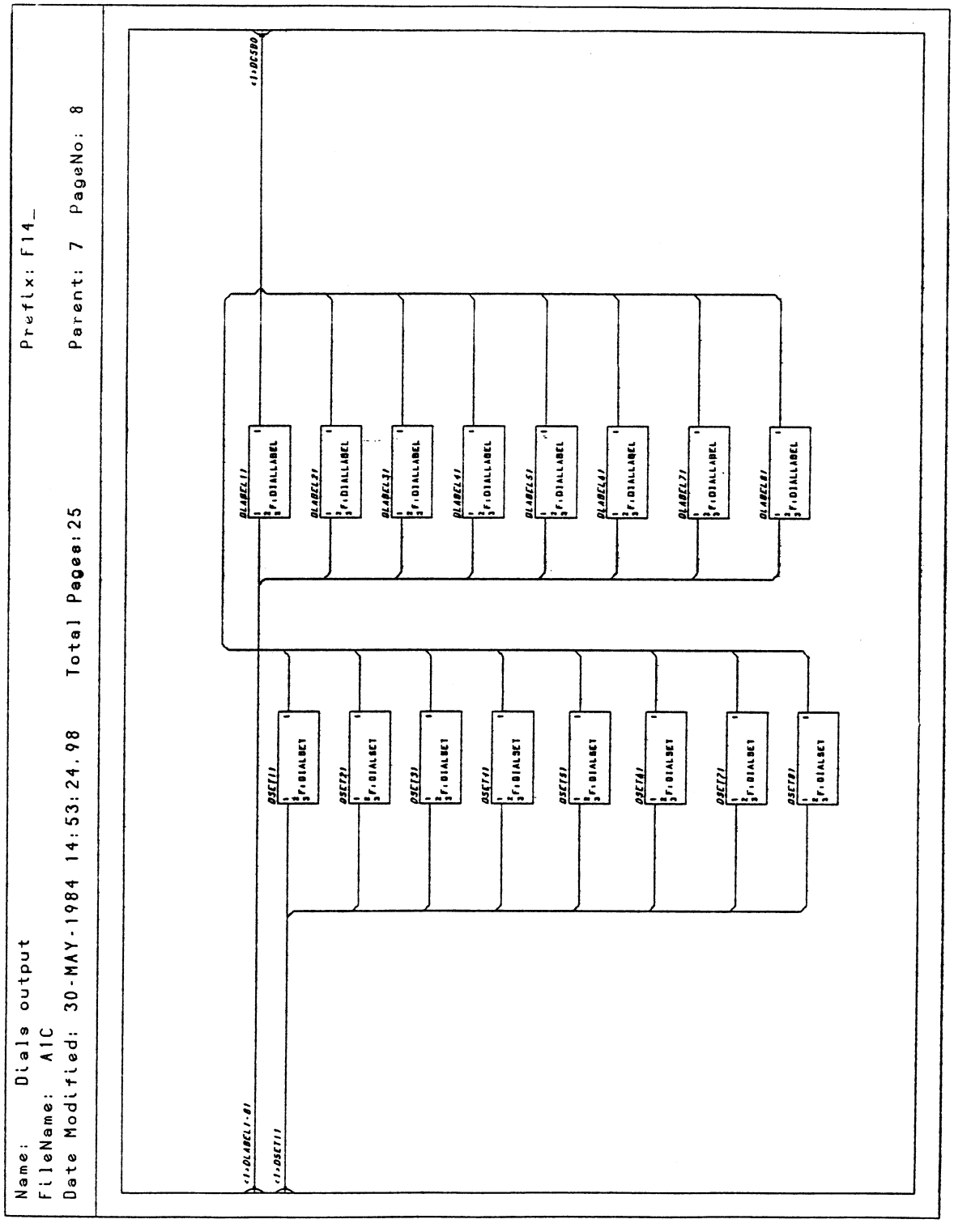

Name: Errors
FileName: A1C
Date Modified: 30-MAY-1984 14:53:24.98
Total Pages:26
Prefix: F29_
Parent: 2
PageNo: 3
ERROR1
ERROR2
ERROR3
F1:CBROUTE
F1:INFORMATION
TRUE
TRUE
TRUE
<0:ERRORS0>
<1:ERRORS0>
C10_M_C10_M_C120<1>
RFC1<2>
RFHL<1>
RFHL<2>
RFHL<3>


Name: Set ports
FileName: A1C
Date Modified: 30-MAY-1984 14:53:24.98 Total Pages: 25 Parent: 2 PageNo: 4
Prefix: F12_
SPA
F:INFORMATION
F:SET_PORT
<2>PORTSOA
<0>SPA

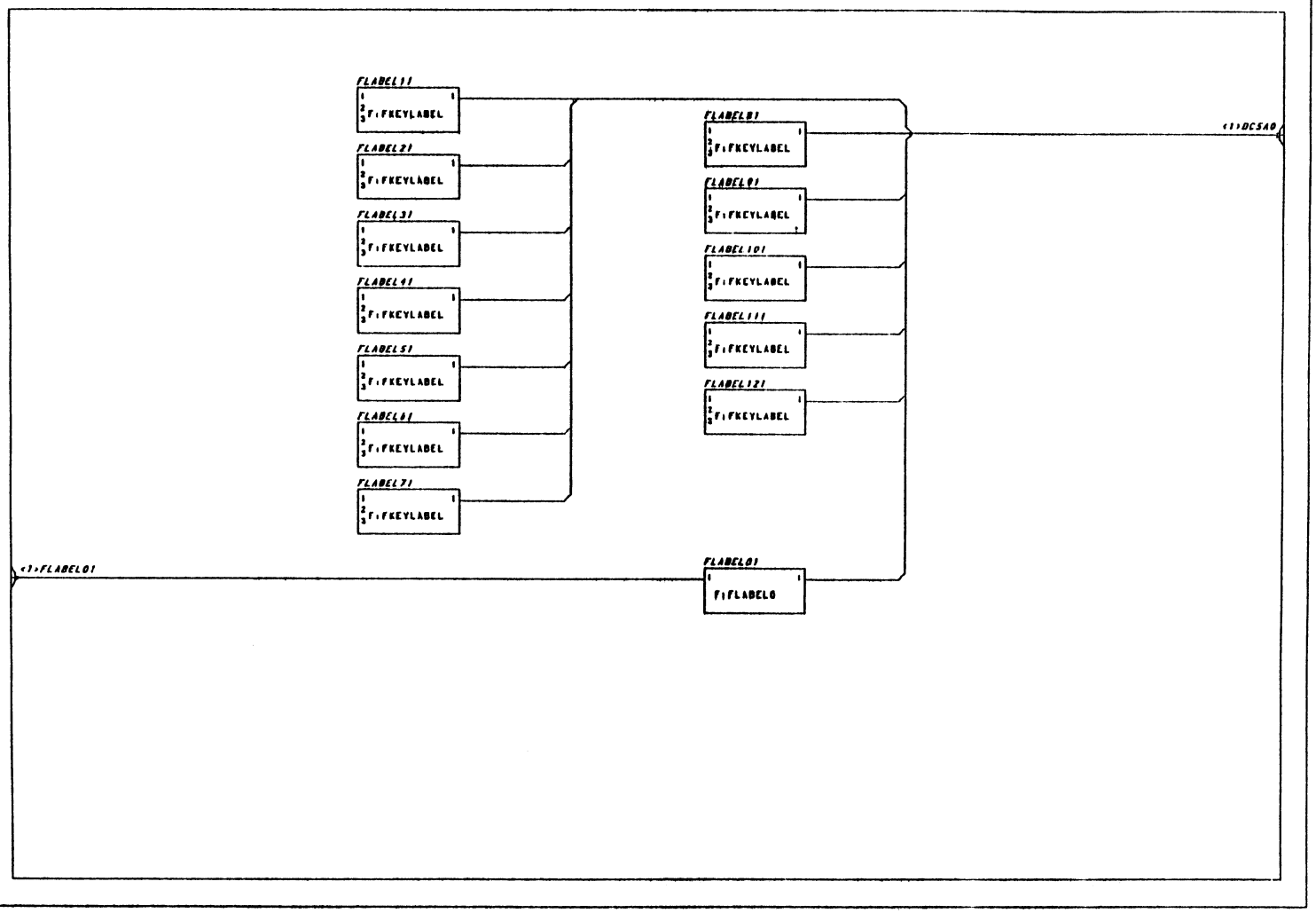

Name: TE,Parsing,Floppy,Hardcopy,DC
FileName: AIC
Date Modified: 30-MAY-1984 15:43:00.48
Total Pages:25
Prefix: F20_
Parent: 1 PageNo: 5
F20
Hardcopy
F19
Who, Host_Message, Mailbox
F18
Terminal Emulator
F17
Chap/CT & Company
F16
Disk Access Network
F15
Device control

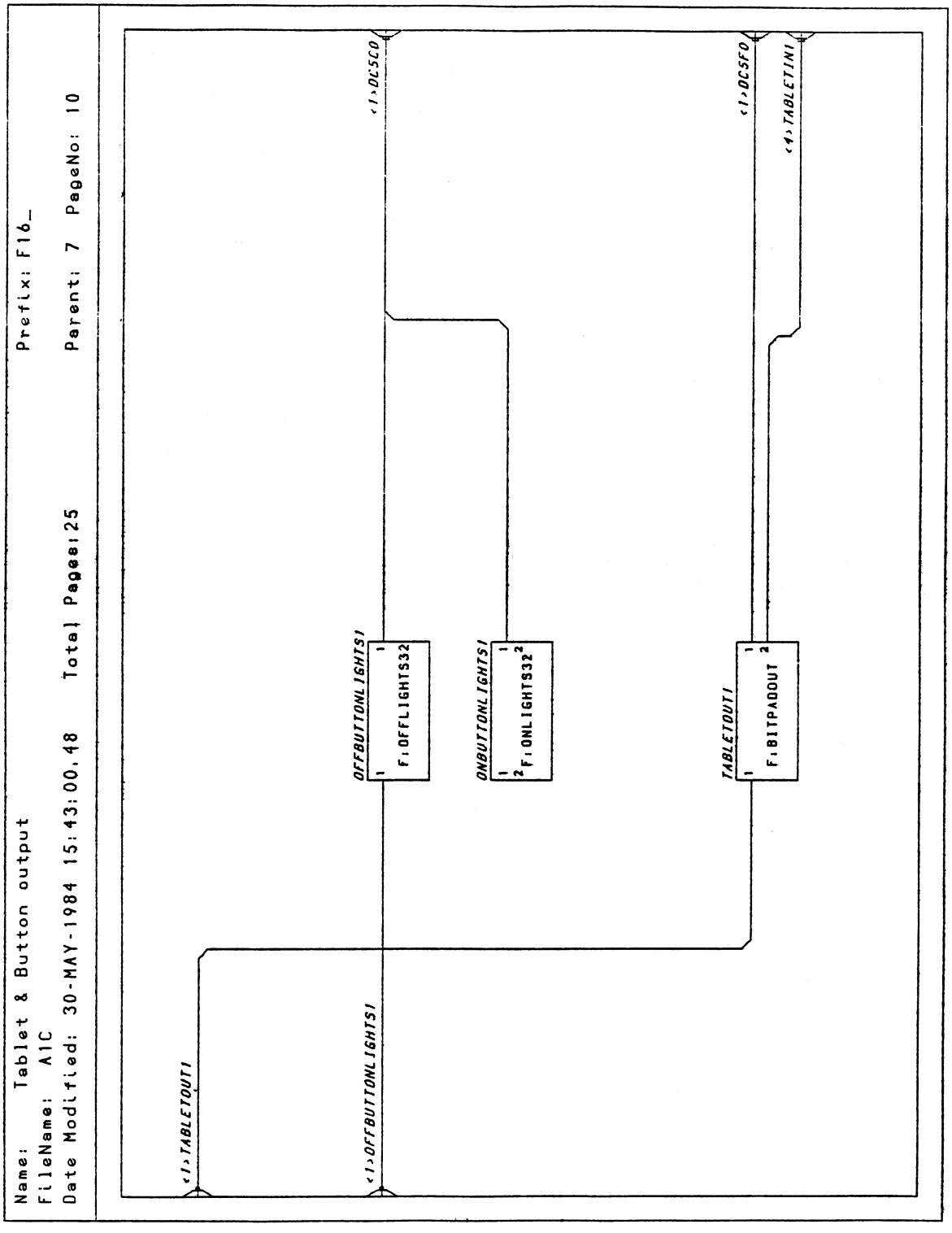

Name: Device control
FileName: A1C
Date Modified: 30-MAY-1984 14:63:24.98 Total Pages:25 Parent: 5 PageNo: 6
Prefix: F7_
Pg 7
Pg 11
Pg 12
Pg 13
<1>FLABEL01
<1>DLABEL1-01
<1>D3CT11
<1>TABLETOUT1
<1>OFFBUTTONLIGHTS1
<1>DCINS0
<1>KBHANDLER1
<2>PORTS04
<1>ERRORS0
<1>DS0

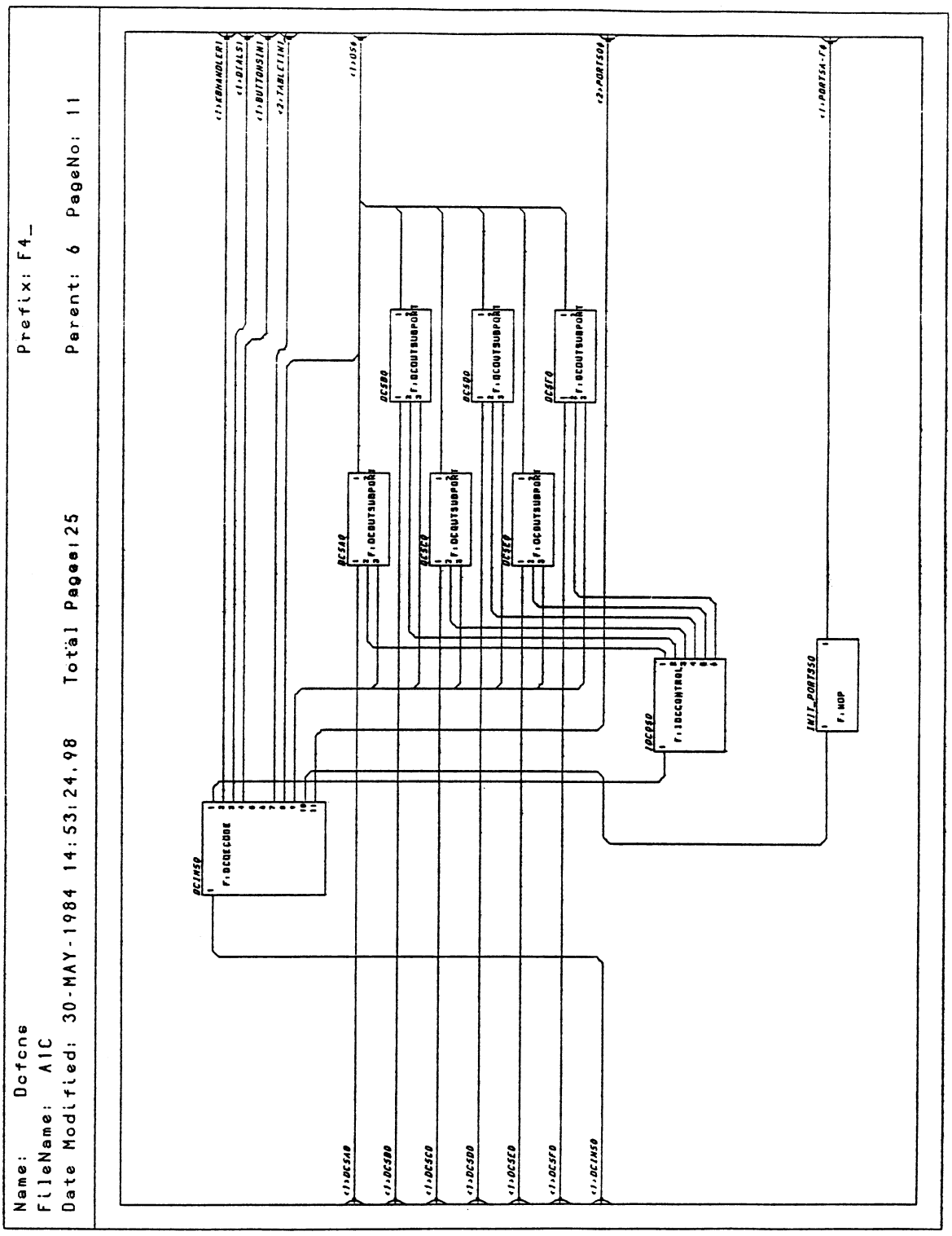

Name: Device output
FileName: A1C
Date Modified: 30-MAY-1984 14:53:24.98 Total Pages:25
Prefix: F3_
Parent: 6 PageNo: 7
F15 Pg 9
Keyboard output(labels)
F15 Pg 8
Dials output
F15 Pg 10
Tablet & Button output

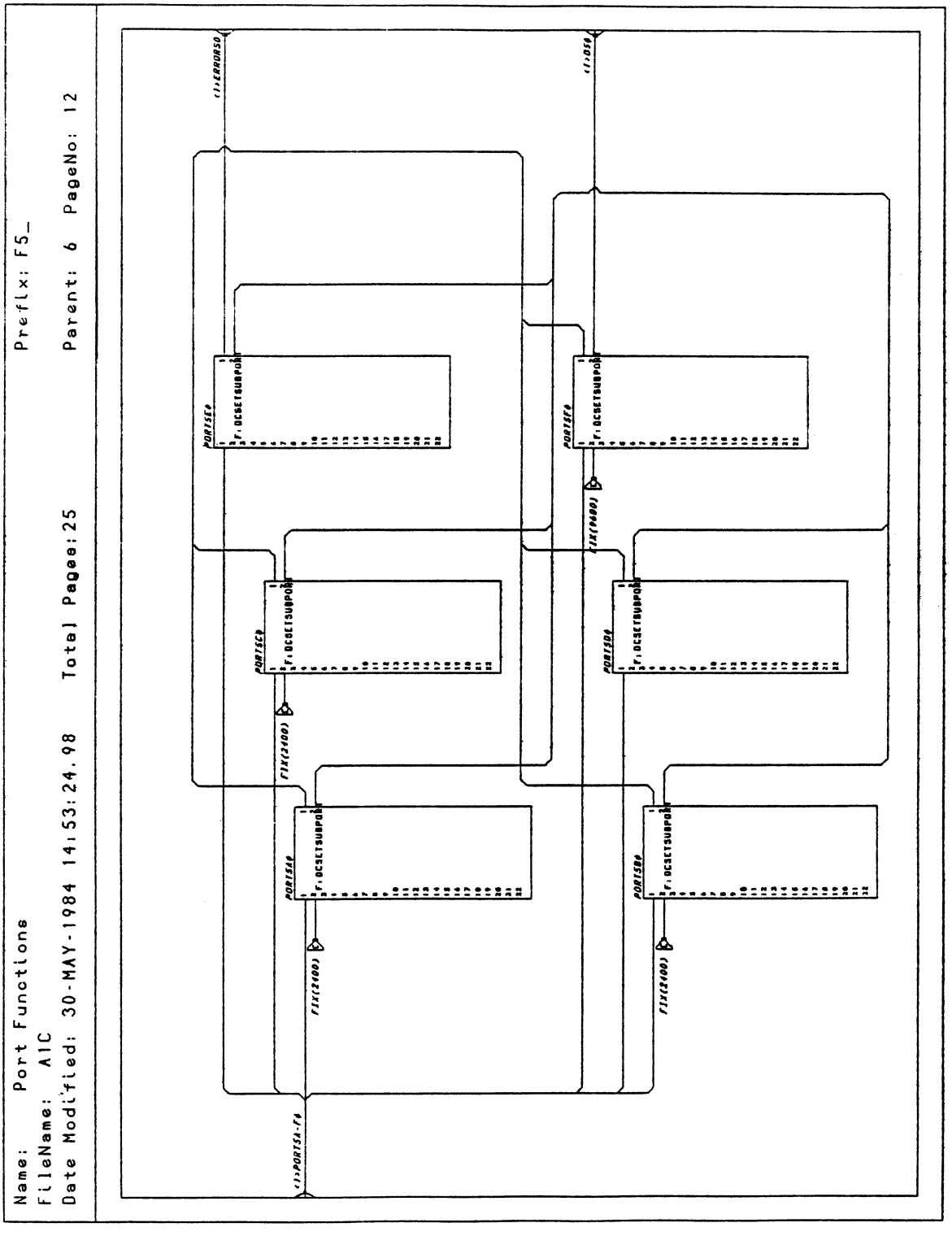

Name: Dials output
FileName: A1C
Date Modified: 30-MAY-1984 14:53:24.98 Total Pages:25 Parent: 7 PageNo: 8
Prefix: F14_
DIALLABEL
DIALSET

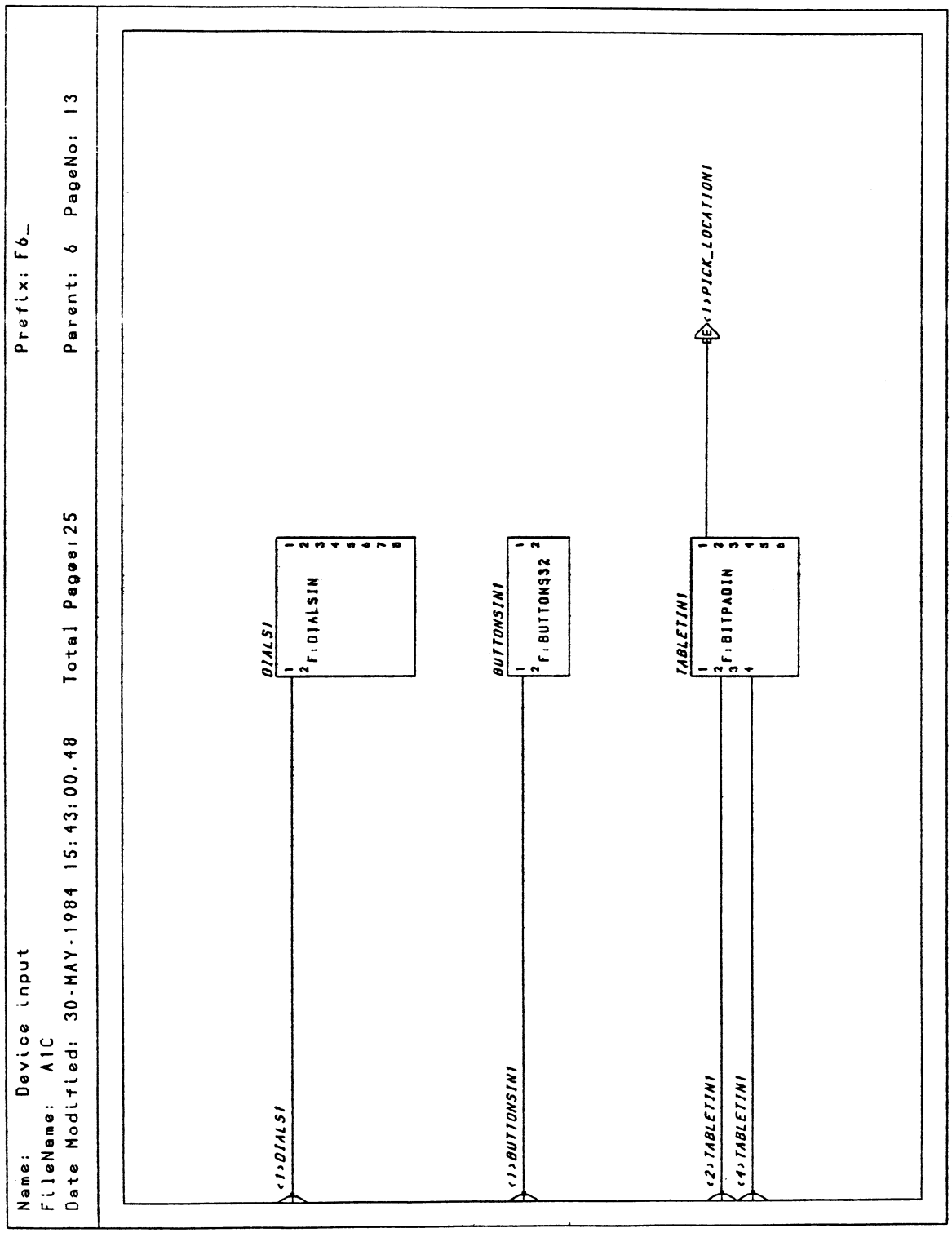
Name: Keyboard output(Labels) Prefix: F15_
FileName: A1C
Date Modified: 30-MAY-1984 14:53:24.98 Total Pages:25 Parent: 7 PageNo: 9

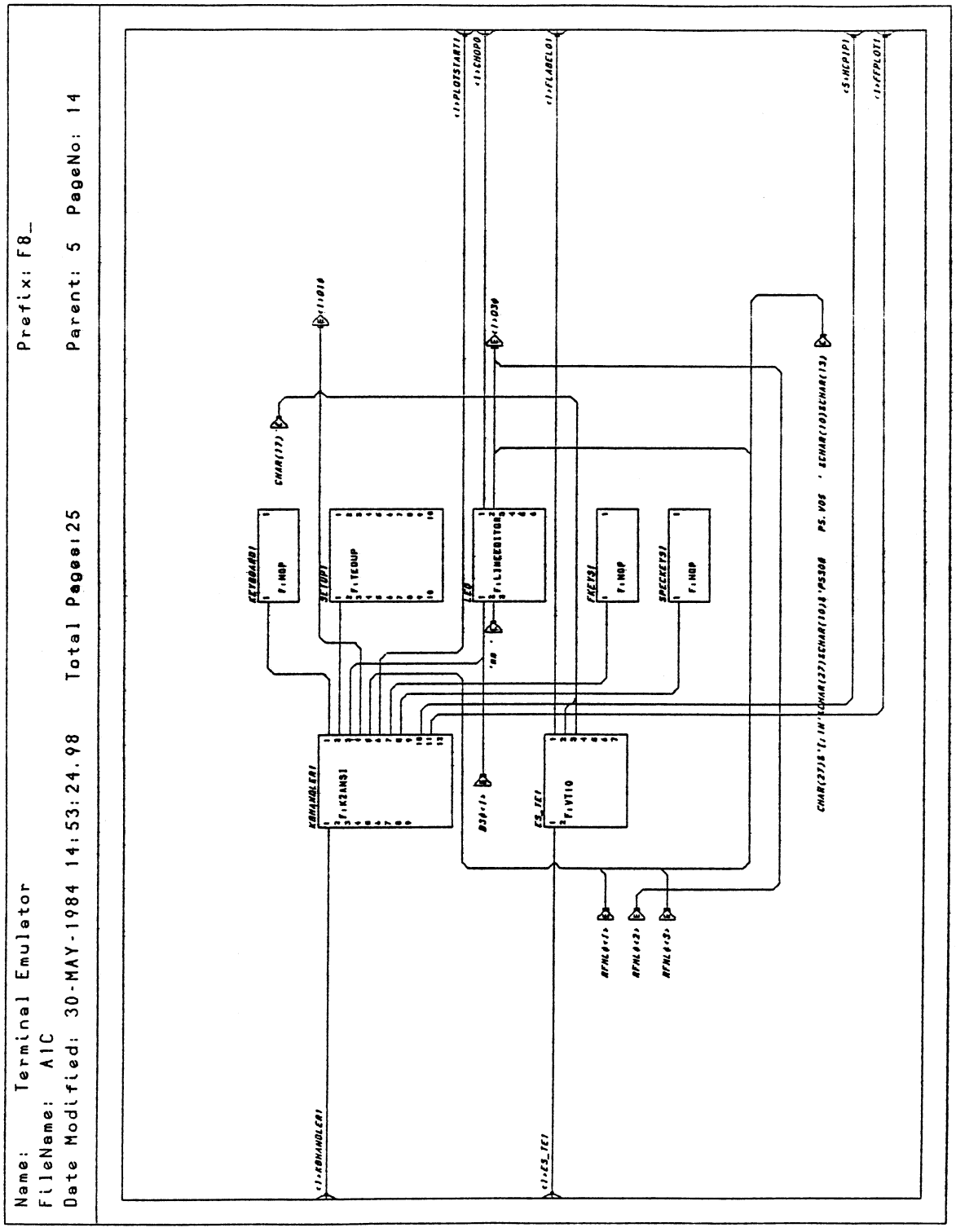


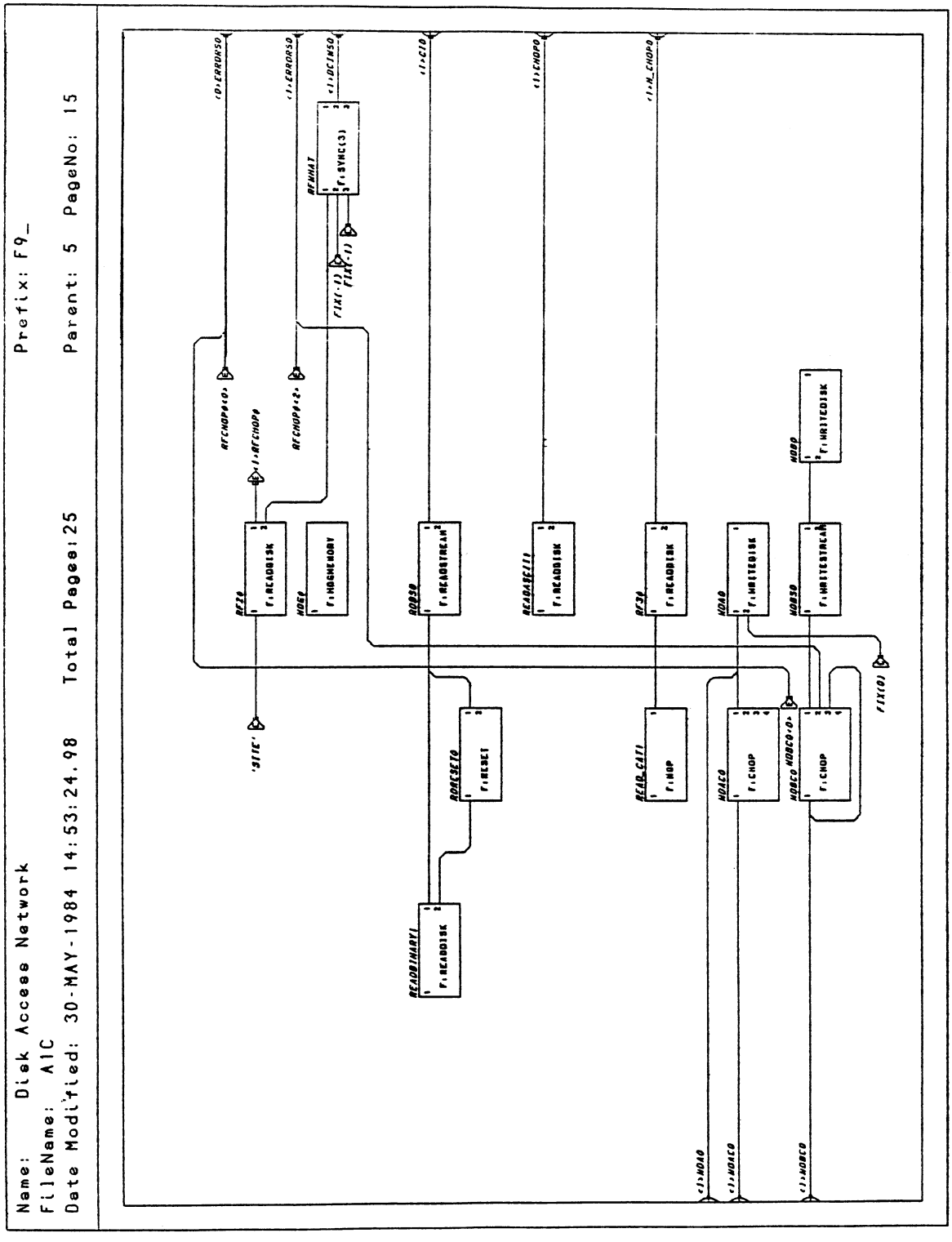

Name: Tablet & Button output
FileName: A1C
Date Modified: 30-MAY-1984 15:43:00.48 Total Pages:25
Prefix: F16_
Parent: 7 PageNo: 10
<1>TABLETOUT1
<1>OFFBUTTONLIGHTS1
<1>DC5CO
<1>DC5FO
<4>TABLETINT
OFFBUTTONLIGHTS1
F:OFFLIGHTS32
ONBUTTONLIGHTS1
F:ONLIGHTS32
TABLETOUT1
F:BITPADOUT

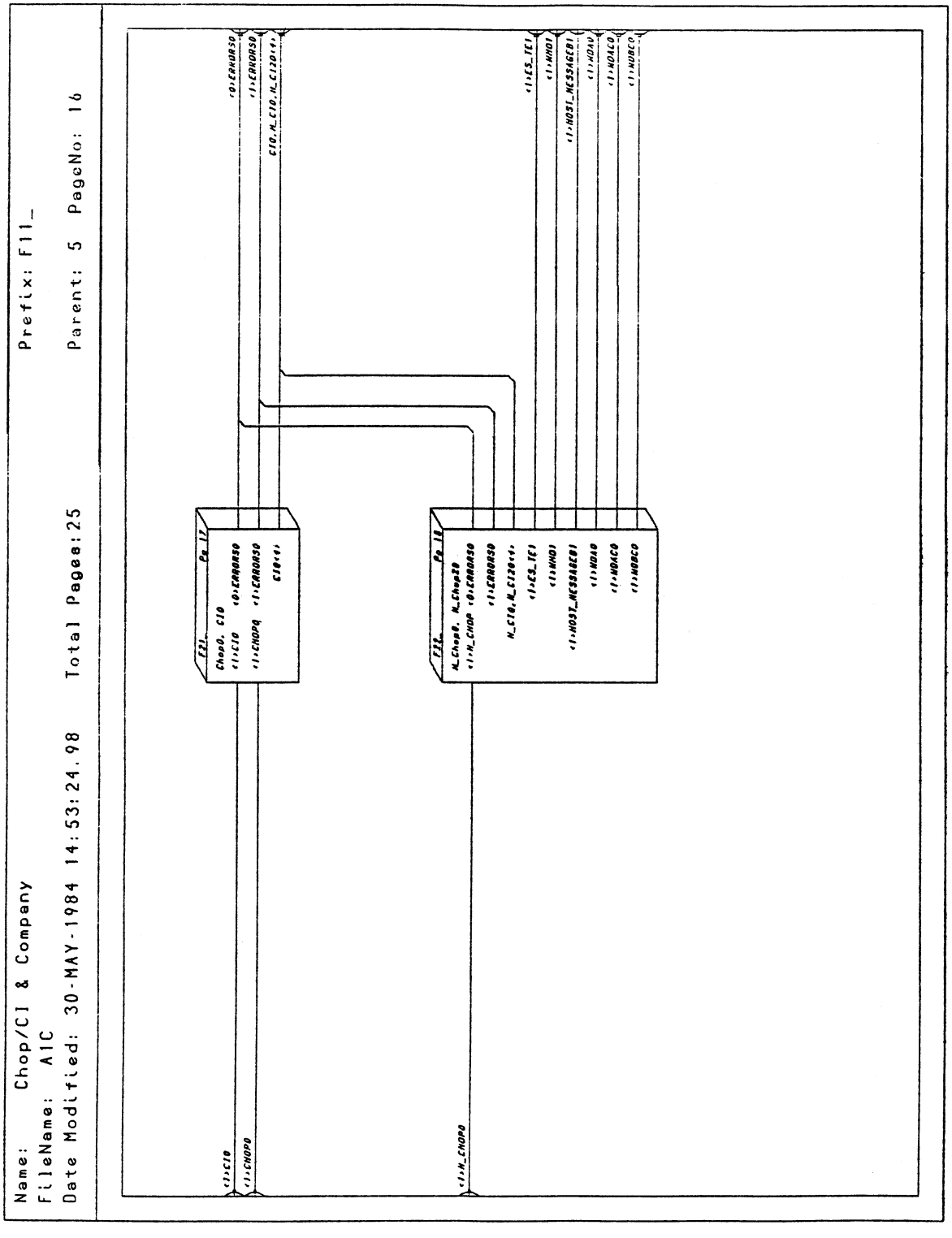

Name: Dcfcns
FileName: A1C
Date Modified: 30-MAY-1984 14:53:24.98 Total Pages: 25
Prefix: F4_
Parent: 6 PageNo: 11
DCINSO
F.DCDECODE
DCSAO
F.DCOUTSUBPORT
DCSBO
F.DCOUTSUBPORT
DCSCO
F.DCOUTSUBPORT
DCSDO
F.DCOUTSUBPORT
DCSEO
F.DCOUTSUBPORT
DCSFO
F.DCOUTSUBPORT
DCCSO
F.IOCCONTROL
INIT_PORTSO
F.NOP

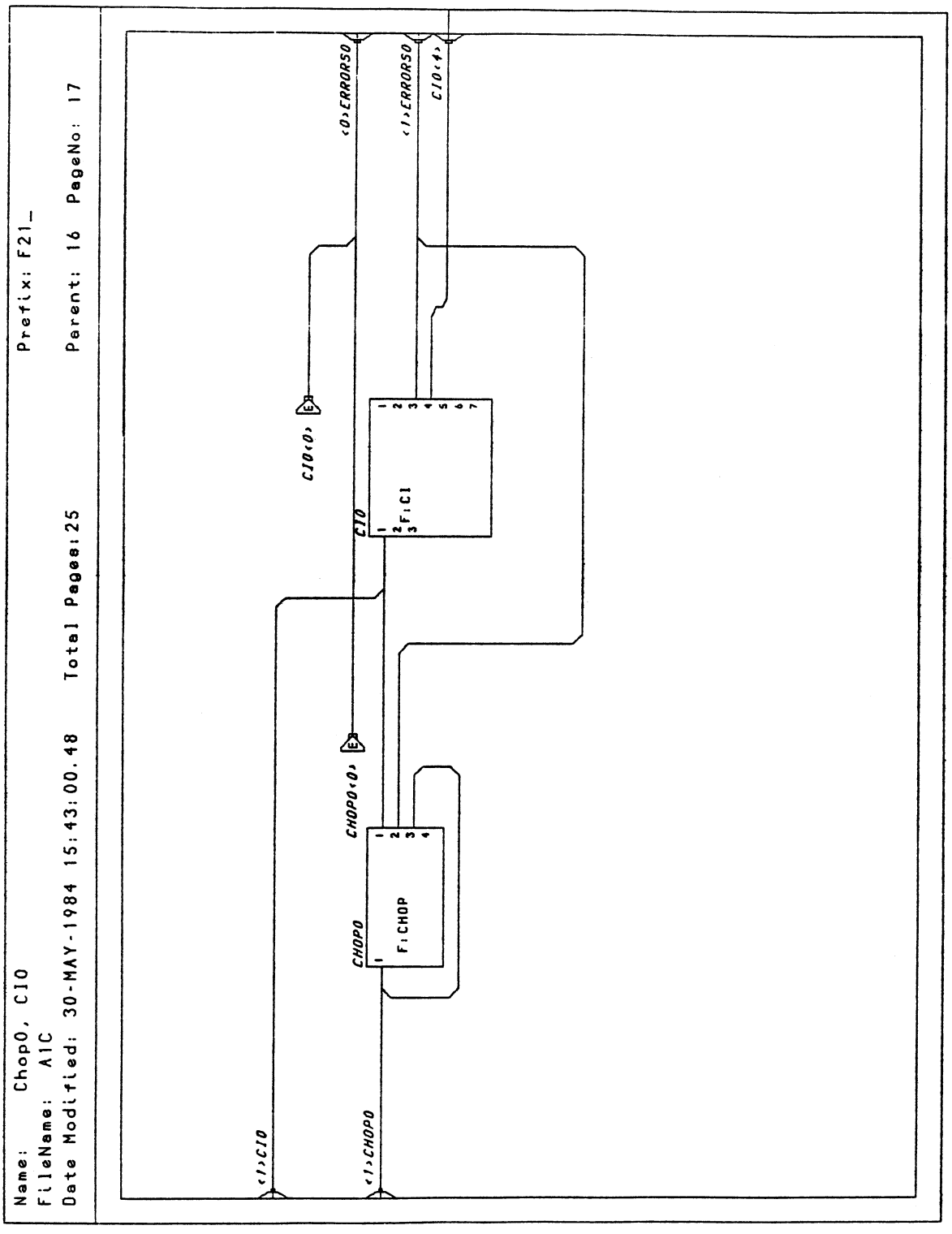

Name: Port Functions
FileName: A1C
Date Modified: 30-MAY-1984 14:53:24.98
Total Pages:25
Prefix: F5_
Parent: 6
PageNo: 12

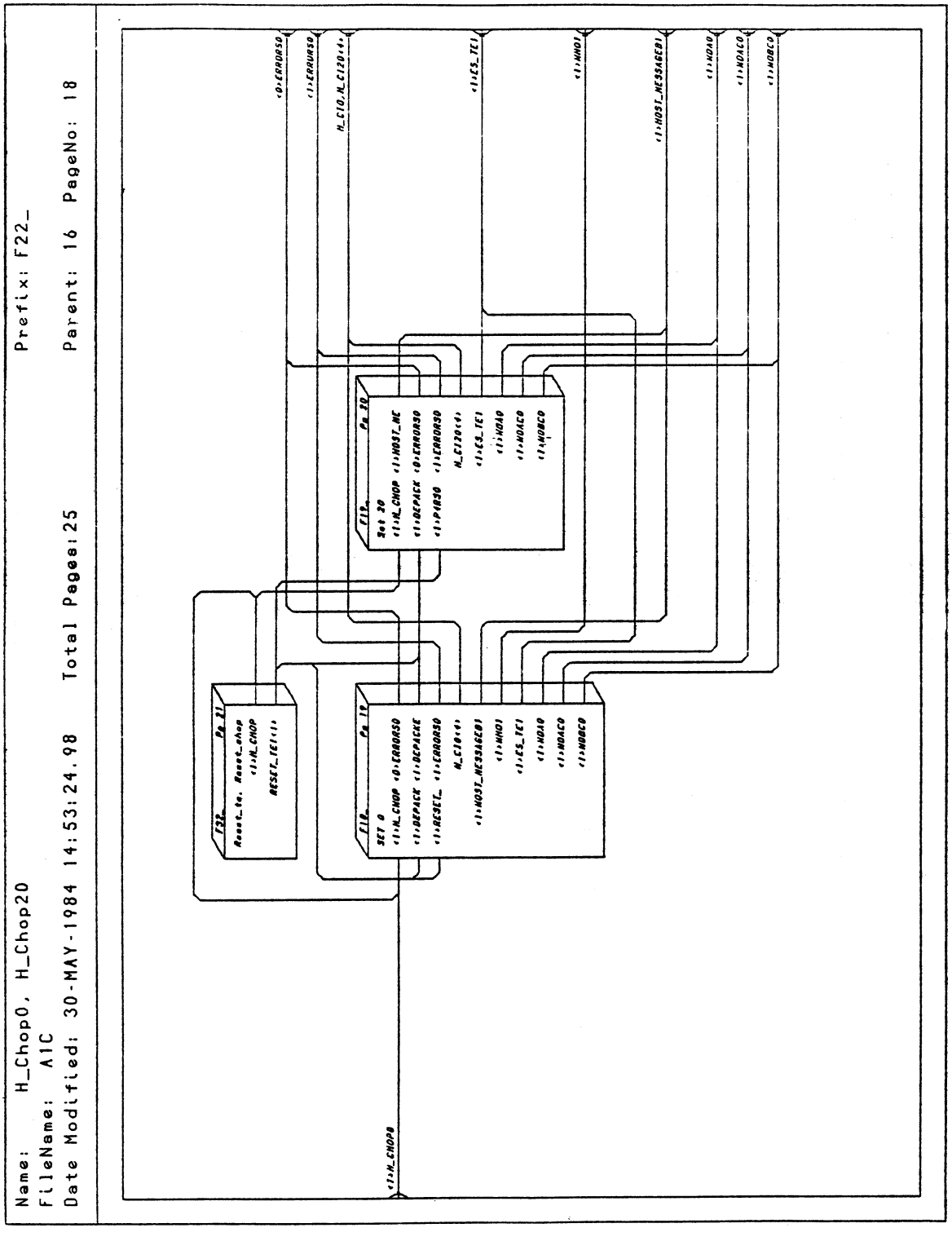

Name: Device input
FileName: A1C
Date Modified: 30-MAY-1984 15:43:00.48
Total Pages: 25
Prefix: F6_
Parent: 6 PageNo: 13
DIALS1
F:DIALSIN
BUTTONSIN1
F:BUTTONS32
TABLETIN1
F:BITPADIN
<1>PICK_LOCATION1
<1>DIALS1
<1>BUTTONSIN1
<2>TABLETIN1
<4>TABLETIN1

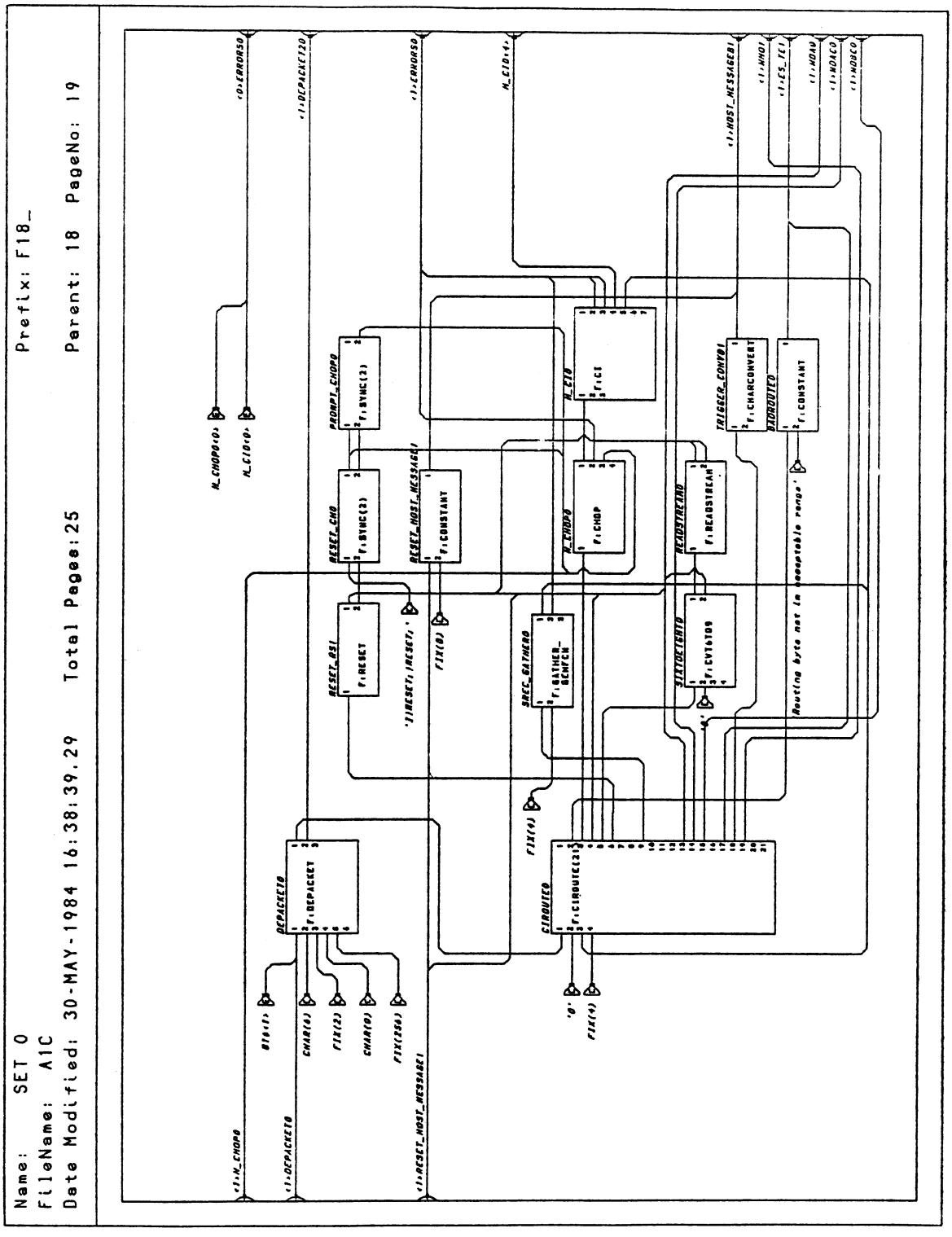

Name: Terminal Emulator
FileName: AIC
Date Modified: 30-MAY-1984 14:53:24.98
Total Pages: 25
Parent: 5
PageNo: 14
Prefix: F8_
KEYBOARD1
F.NOP
SETUP1
F.TEQUP
LED
F.LINEEDITOR
KEYS1
F.NOP
SPECKEYS1
F.NOP
KBHANDLER1
F.K2ANS1
ES_TE1
F.VT10
CHAR(17)
<1>:PLOTSTART1
<1>:ENDPO
<1>:FLABEL01
<5>:HCP1P1
<1>:FFPLOT1
<1>:KBHANDLER1
<1>:ES_TE1
CHAR(27)'[;1H'
CHAR(27)'[H'
CHAR(10)'PS300
PS.VOS
'&CHAR(10)&CHAR(13)

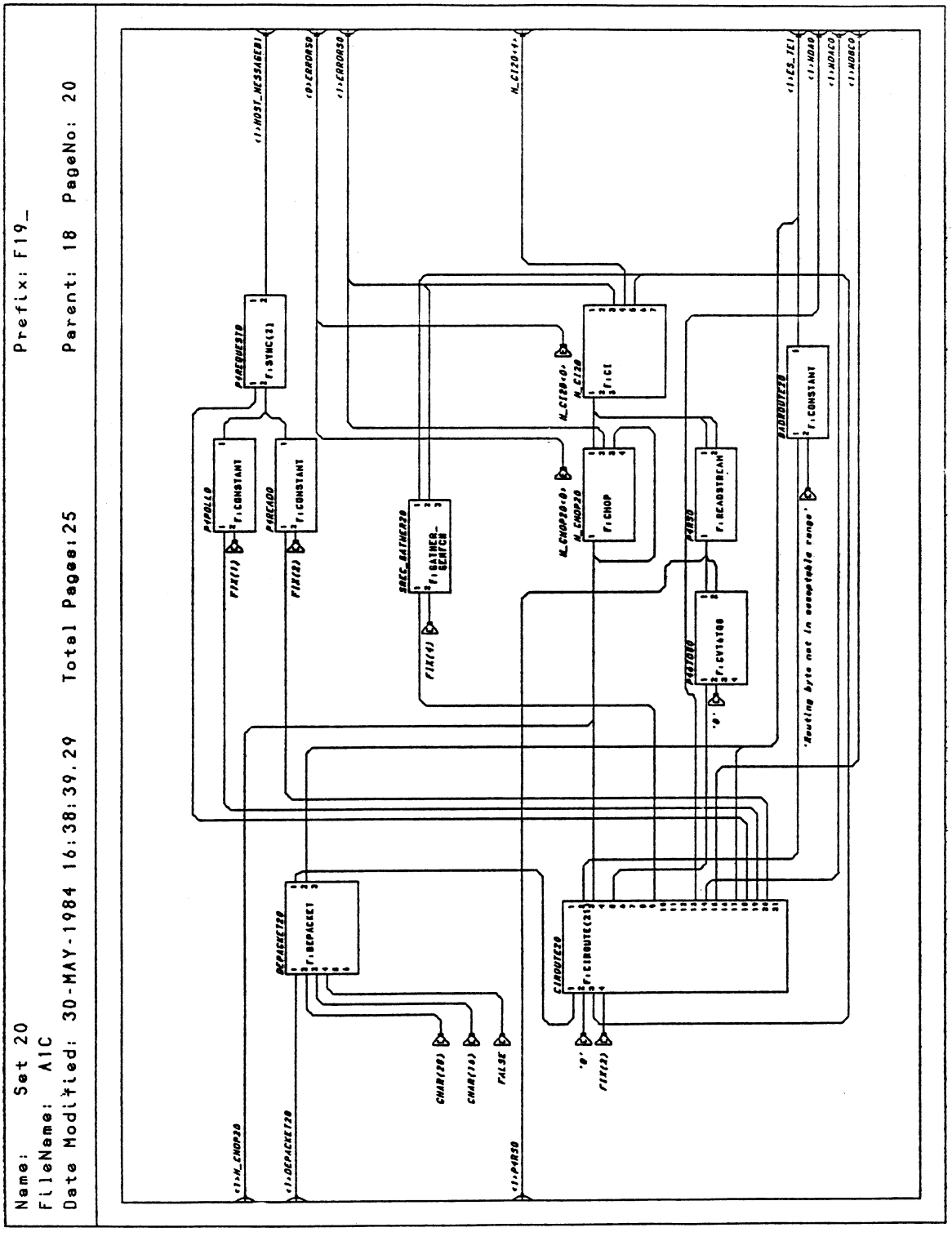

Name: Disk Access Network
FileName: A1C
Date Modified: 30-MAY-1984 14:53:24.98 Total Pages:25 Parent: 5 PageNo: 15
Prefix: F9_
F.READDISK
F.HOGMEMORY
F.READSTREAM
F.READDISK
F.READDISK
F.HRITEDISK
F.HRITESTREAM
F.READDISK
F.RESET
F.HOP
F.CHOP
F.CHOP
F.SYNC(3)
F.HRITEDISK
'SITE'
FIX(0)

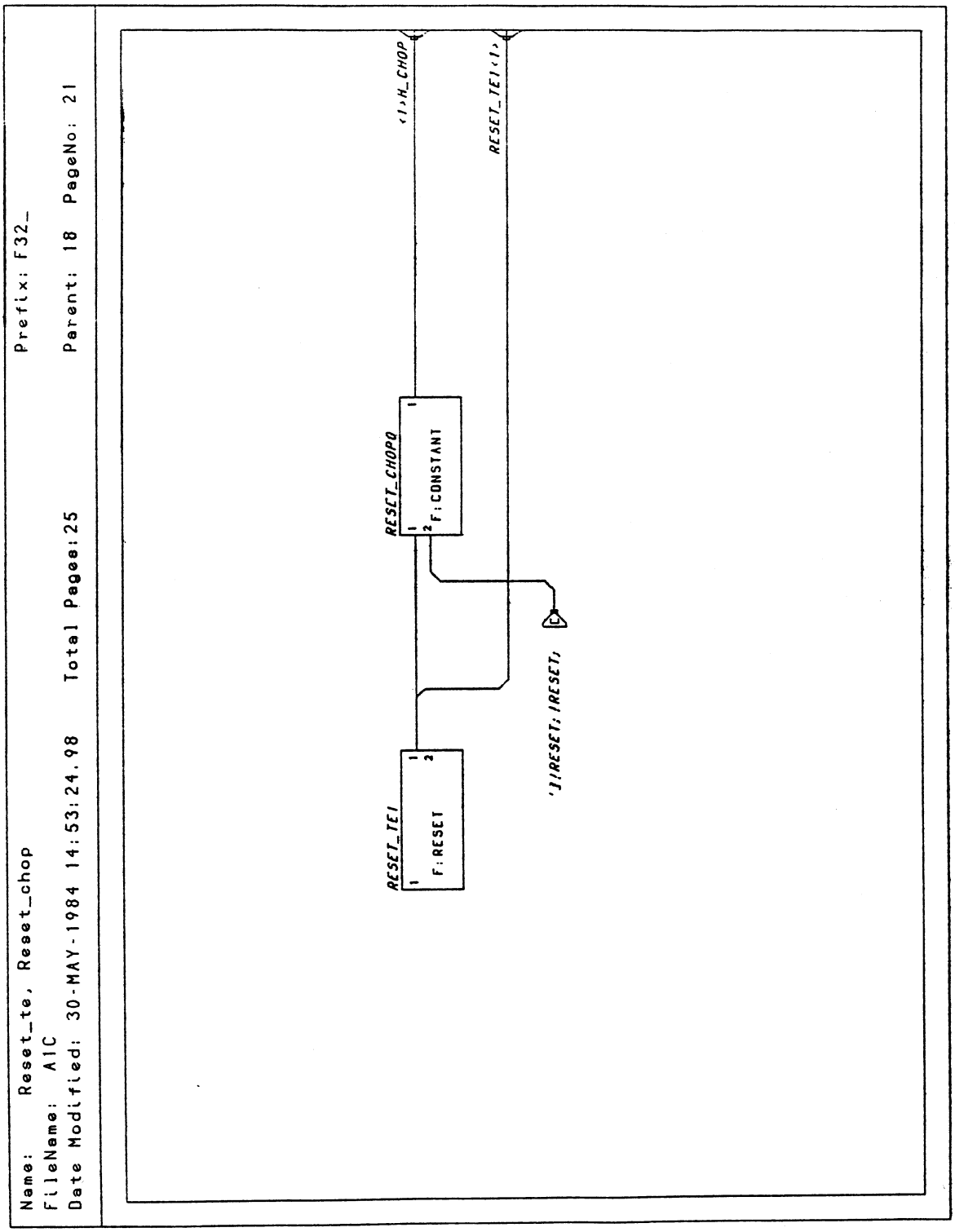

Name: Chop/CI & Company
FileName: A1C
Date Modified: 30-MAY-1984 14:53:24.98
Total Pages:25
Prefix: F11_
Parent: 5 PageNo: 16
F11_ Pg 12
Chop0, C10
F11_ Pg 18
N_Chop0, N_Chop20
(1:C10
(1:CHOP0
(1:N_CHOP0

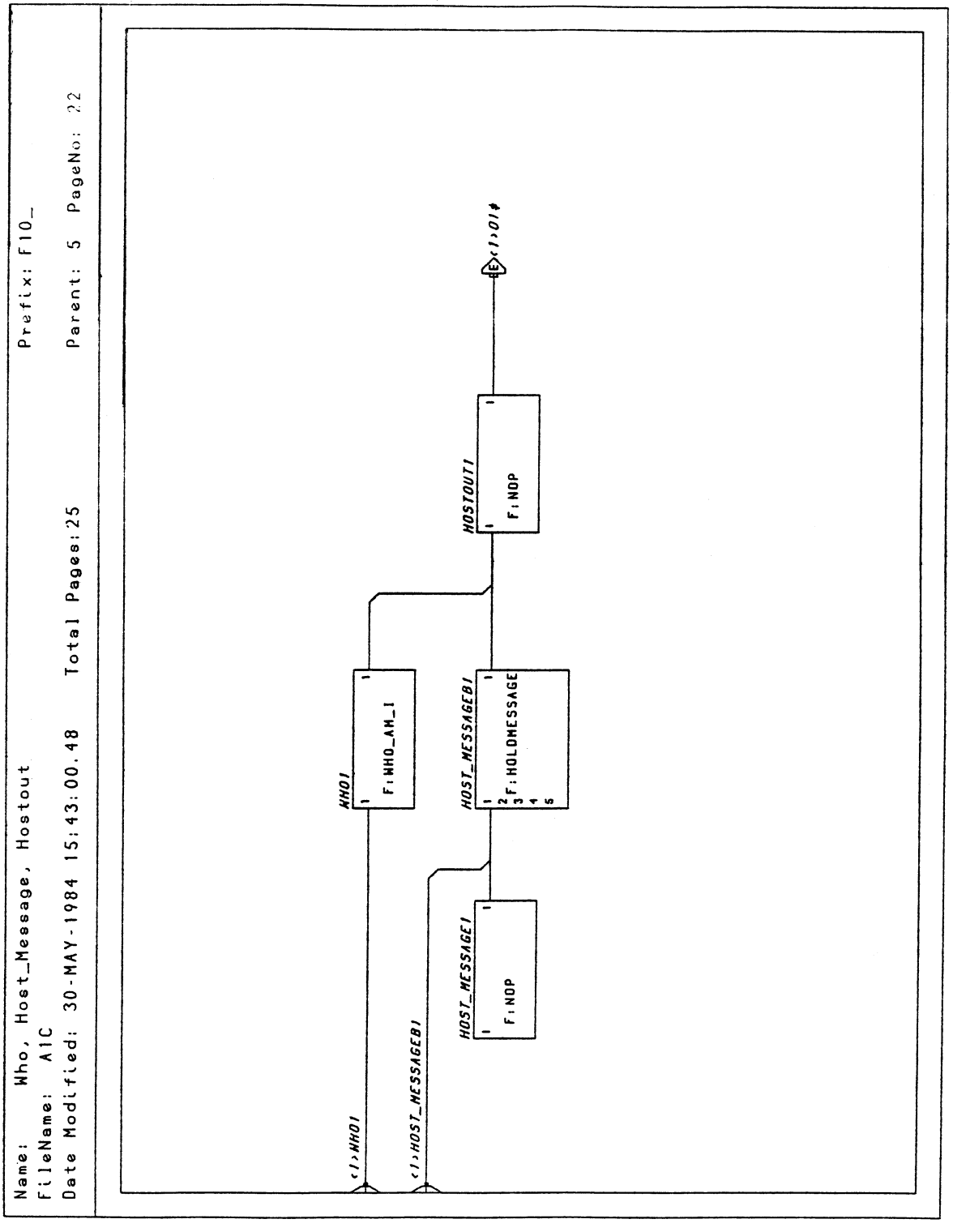

Name: ChopO, C10
FileName: A1C
Date Modified: 30-MAY-1984 15:43:00.48 Total Pages: 25 Parent: 16 PageNo: 17
Prefix: F21_
<0>ERRORSO
<1>ERRORSO
C10<4>
C10<0>
C10
F:C1
CHOPO<0>
CHOPO
F:CHOP
<1>C10
<1>CHOPO

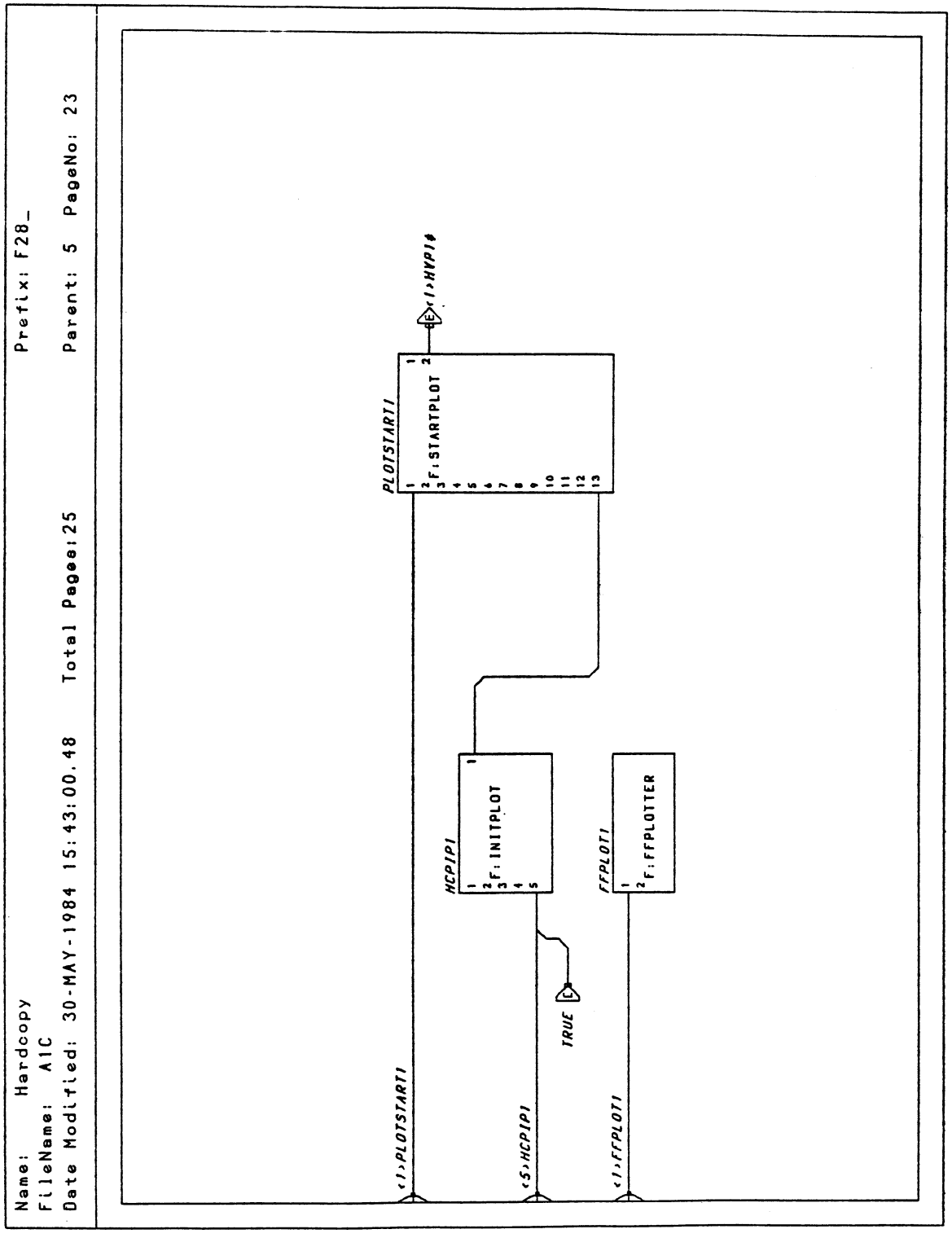

Name: H_Chop0, H_Chop20
FileName: A1C
Date Modified: 30-MAY-1984 14:53:24.98
Total Pages:25
Prefix: F22_
Parent: 16 PageNo: 18

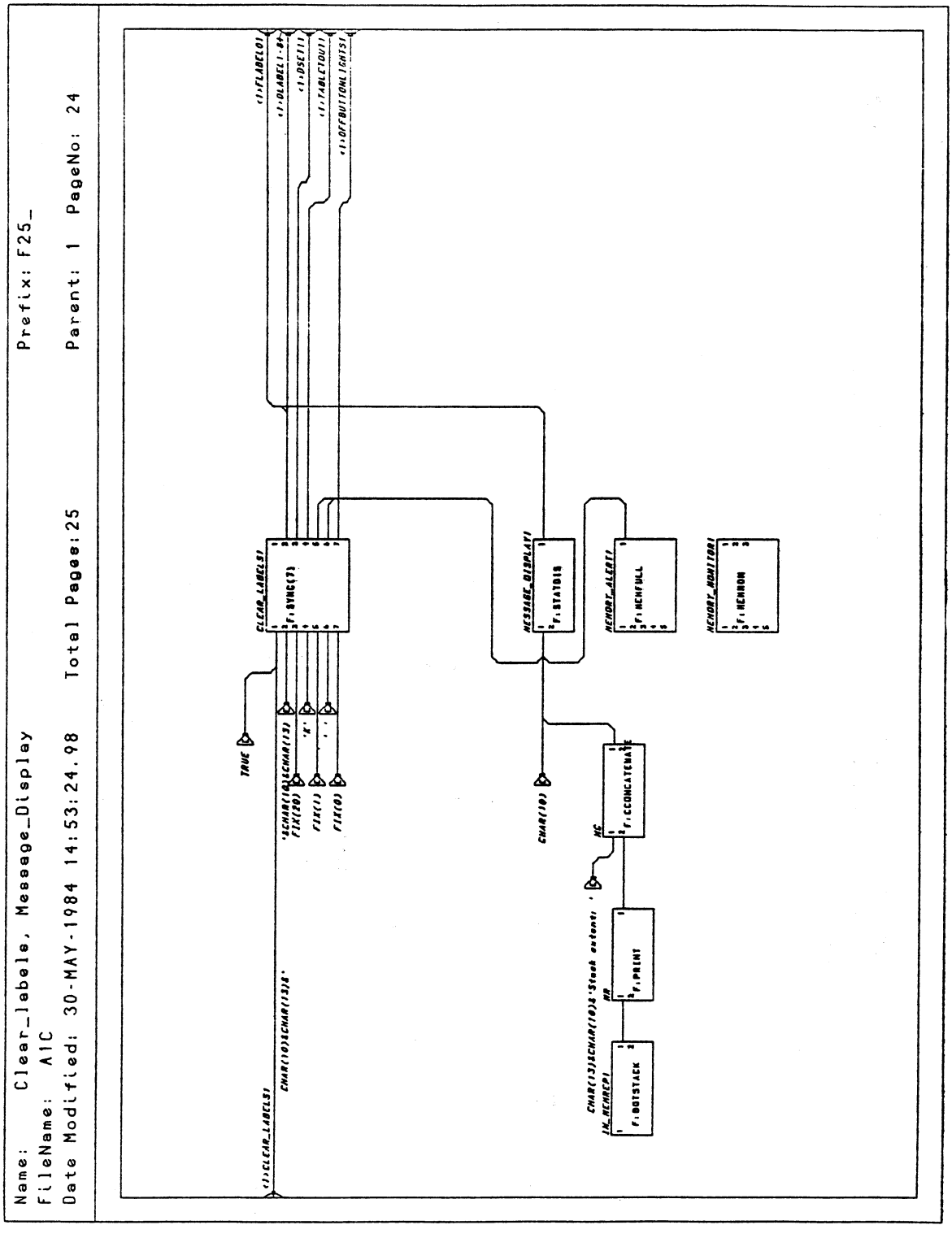

Name: SET 0
FileName: A1C
Date Modified: 30-MAY-1984 16:38:39.29 Total Pages:25 Parent: 18 PageNo: 19
Prefix: F18_
N_CHOP
DEPACKET0
RESET_ASI
RESET_CH0
RESET_HOST_MESSAGE0
PROMPT_CHOP0
N_C10
N_CHOP0
SREC_GATHER0
CIROUTE0
READSTREAM0
EVENTCIENT0
TRIGGER_CONV0
BIGROUTE0
Routing byte not in acceptable range

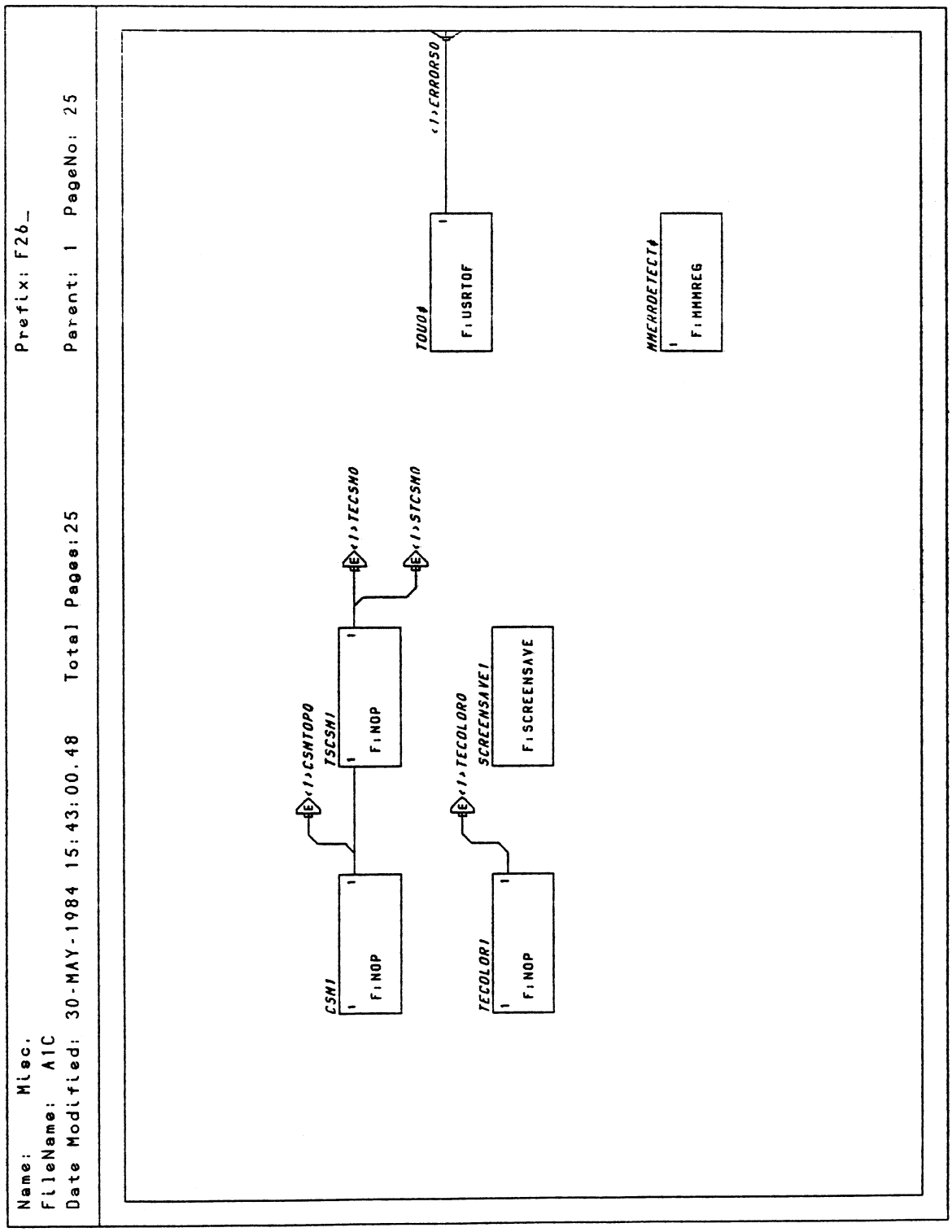

Name: Set 20
FileName: A1C
Date Modified: 30-MAY-1984 16:38:39.29 Total Pages:25 Parent: 18 PageNo: 20
Prefix: F19_
"Routing byte not in acceptable range"

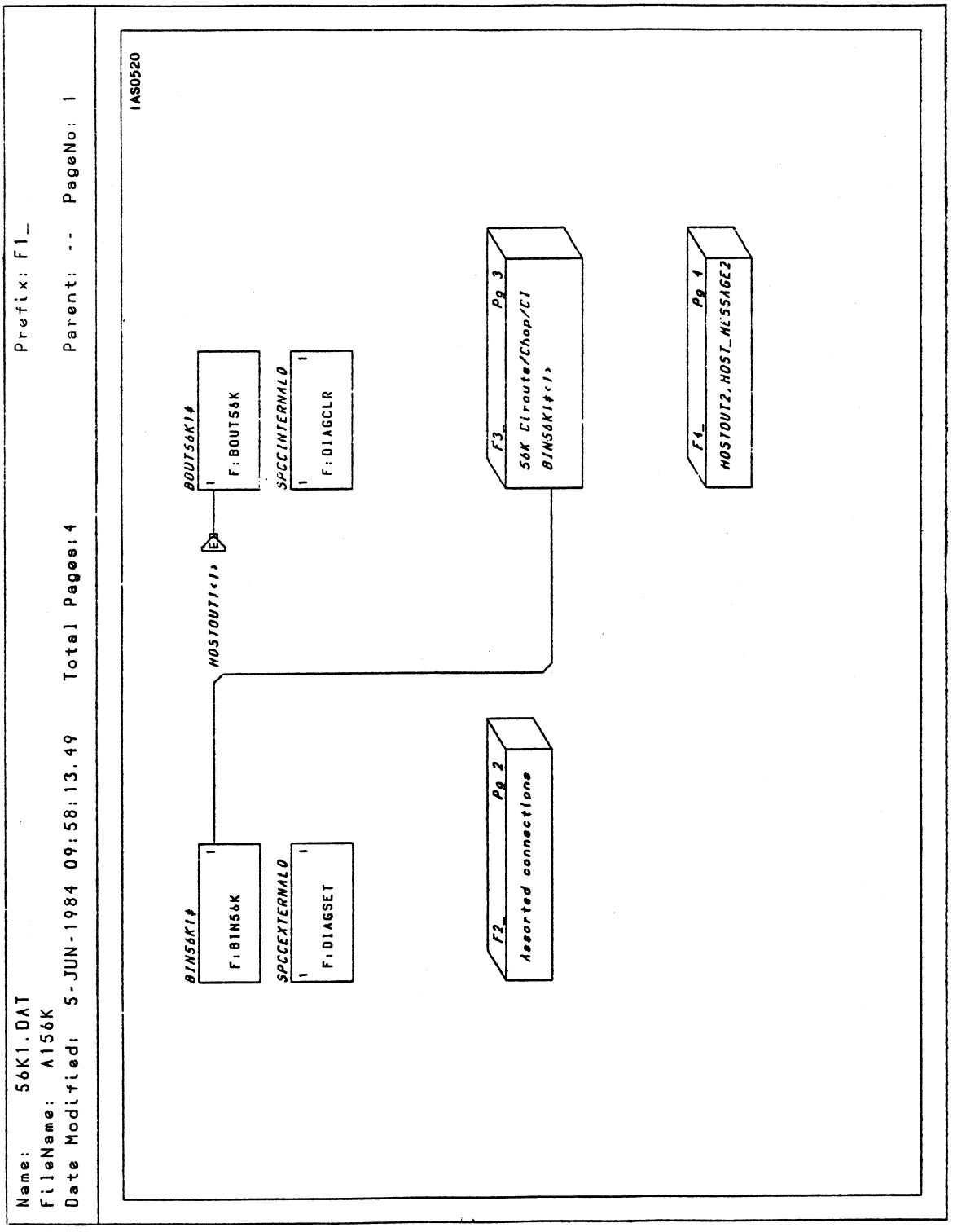

Name: Reset_te, Reset_chop
Prefix: F32_
FileName: A1C
Date Modified: 30-MAY-1984 14:53:24.98
Total Pages:25
Parent: 18
PageNo: 21
RESET_CHOPQ
F:CONSTANT
RESET_TEI
F:RESET
'J!RESET;!RESET;


Name: Who, Host_Message, Hostout
FileName: A1C
Date Modified: 30-MAY-1984 15:43:00.48
Total Pages:25
Prefix: F10_
Parent: 5 PageNo: 22
<1>WHO1
<1>HOST_MESSAGEB1
WHO1
F:WHO_AM_I
HOST_MESSAGEB1
F:HOLDMESSAGE
HOST_MESSAGE1
F:NOP
HOSTOUT1
F:NOP
<1>O1#

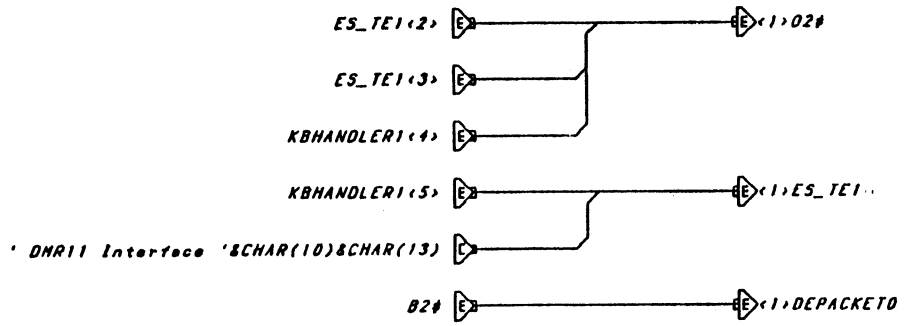

Name: Hardcopy
FileName: A1C
Date Modified: 30-MAY-1984 15:43:00.48
Total Pages:25
Prefix: F28_
Parent: 5 PageNo: 23
<1>HVPI#
PLOTSTART1
F:STARTPLOT
HCPIP1
F:INITPLOT
TRUE
FFPLOT1
F:FFPLOTTER
<1>PLOTSTART1
<5>HCPIP1
<1>FFPLOT1

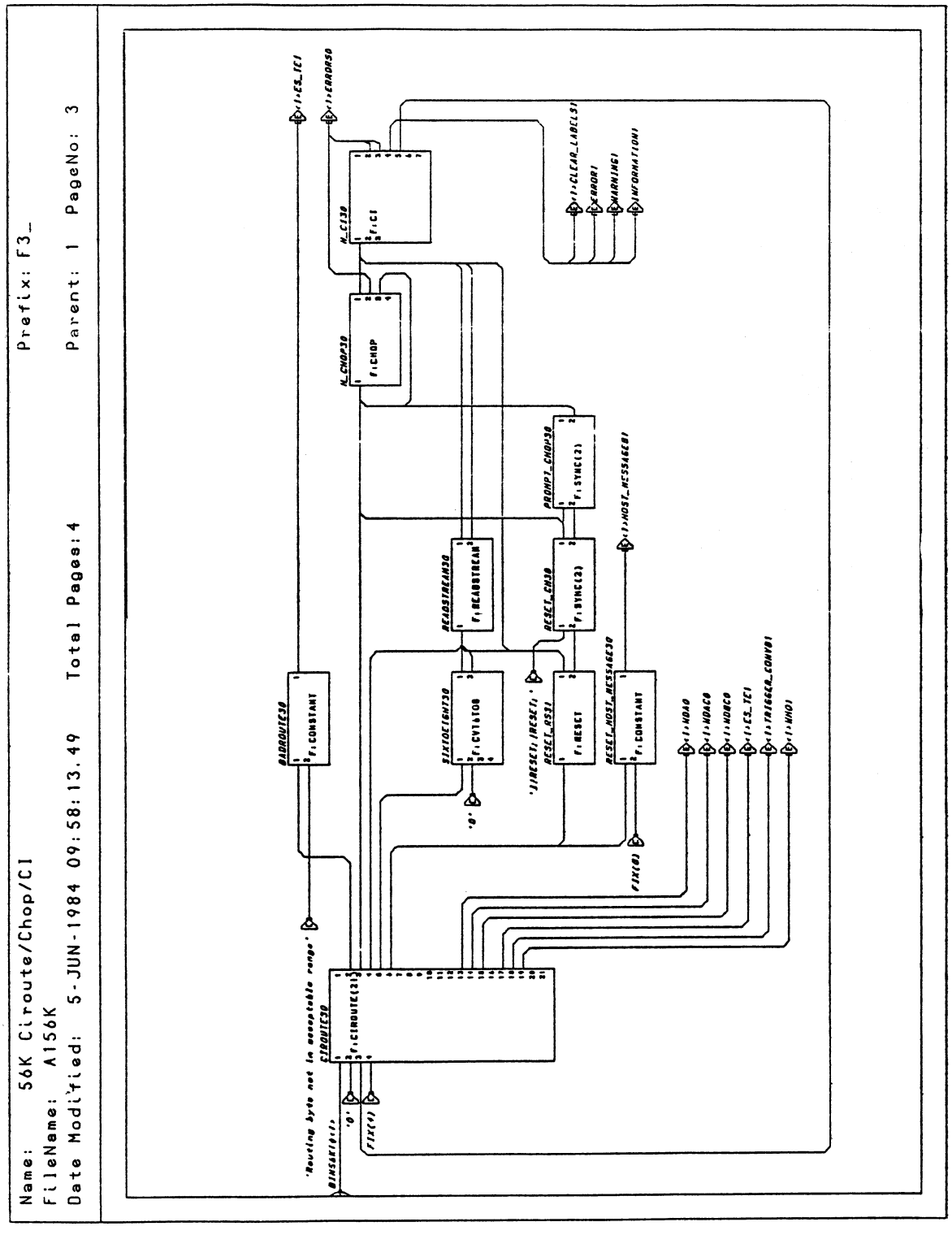

Name: Clear_labels, Message_Display
FileName: AIC
Date Modified: 30-MAY-1984 14:53:24.98
Total Pages:25
Prefix: F25_
Parent: 1 PageNo: 24
CLEAR_LABELS
F.SYNC(7)
MESSAGE_DISPLAY
F.STATUS
KCHORT_ALERT
F.KEYFULL
KCHORT_MONITOR
F.KEYROW
F.CONCATENATE
F.PRINT
F.BITSTACK
TRUE
CHAR(10)
CHAR(1)

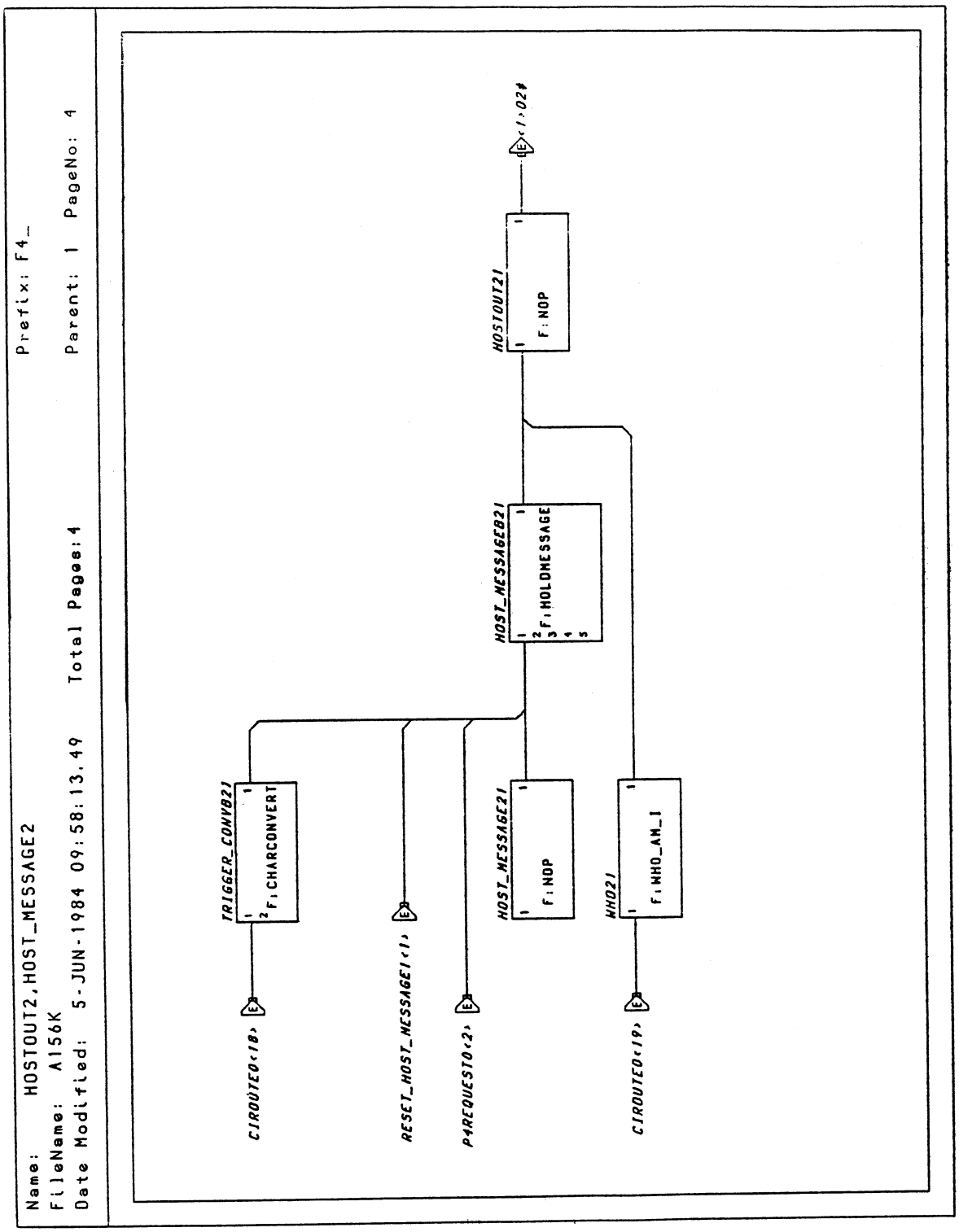

Name: Misc.
FileName: A1C
Date Modified: 30-MAY-1984 15:43:00.48 Total Pages:25
Prefix: F26_
Parent: 1 PageNo: 25
<1>ERRORSO
TOUO#
F:USRTOF
MMERRDETECT#
F:MMMREG
<E><1>TECSNO
<E><1>STCSNO
<E><1>CSNTOPO
TSCSN1
F:NOP
CSN1
F:NOP
<E><1>TECOLORO
SCREENSAVE1
F:SCREENSAVE
TECOLOR1
F:NOP

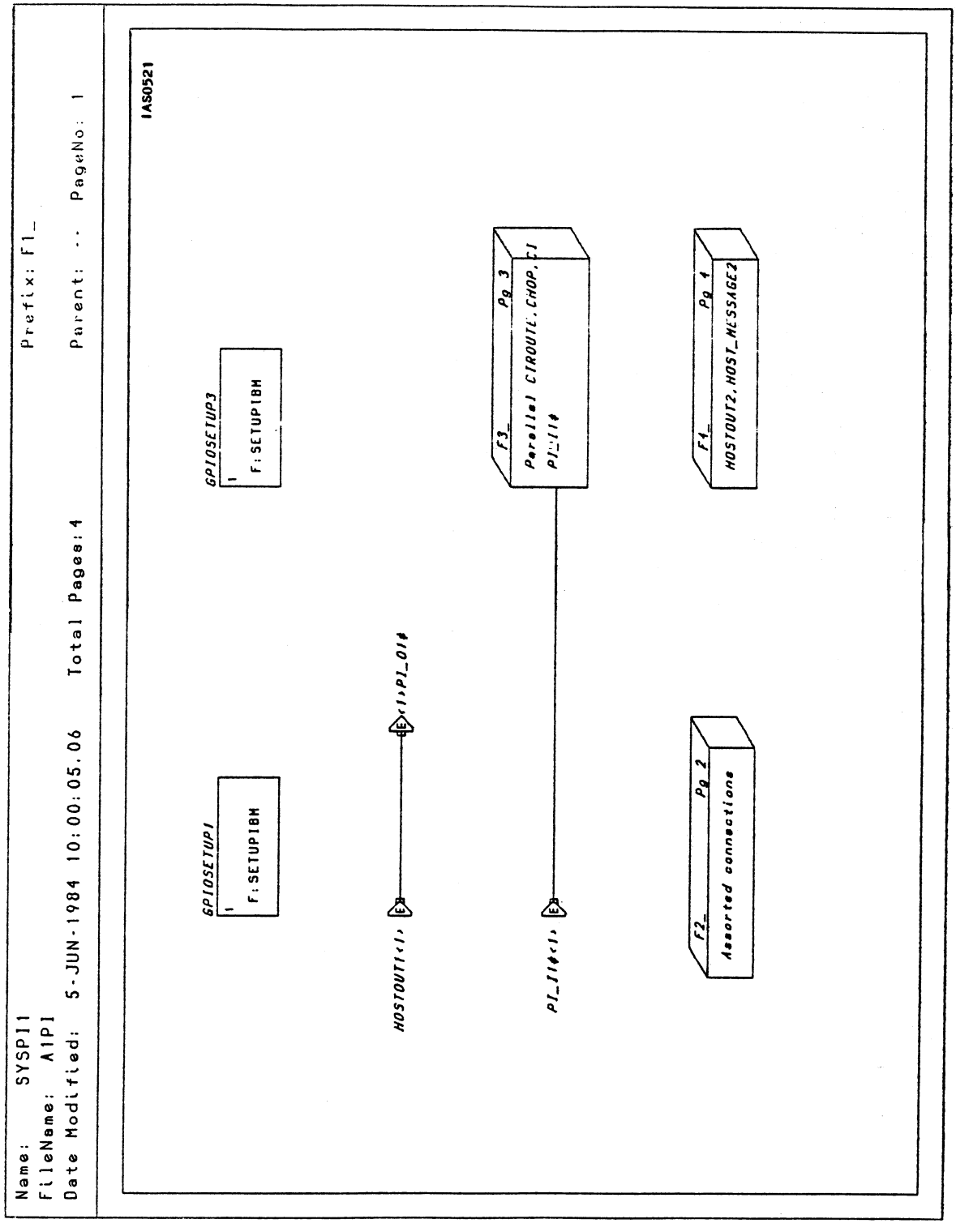

Name: 56K1.DAT
FileName: A156K
Date Modified: 5-JUN-1984 09:58:13.49
Total Pages:4
Prefix: F1_
Parent: -- PageNo: 1
IAS0520
BINS6K1#
F:BINS6K
SPCCEXTERNAL0
F:DIAGSET
HOSTOUT1<1>
BOUTS6K1#
F:BOUTS6K
SPCCINTERNAL0
F:DIAGCLR
F2_
Pg 2
Assorted connections
F3_
Pg 3
56K Circuit/Chap/CI
BINS6K1#<1>
F4_
Pg 4
HOSTOUT2, HOST_MESSAGE2

Name: Assorted connections Prefix: F2_
FileName: A156K
Date Modified: 5-JUN-1984 09:58:13.49 Total Pages:4 Parent: 1 PageNo: 2

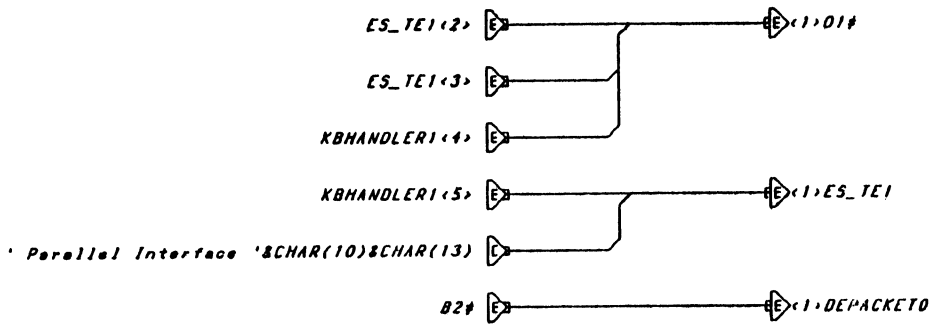


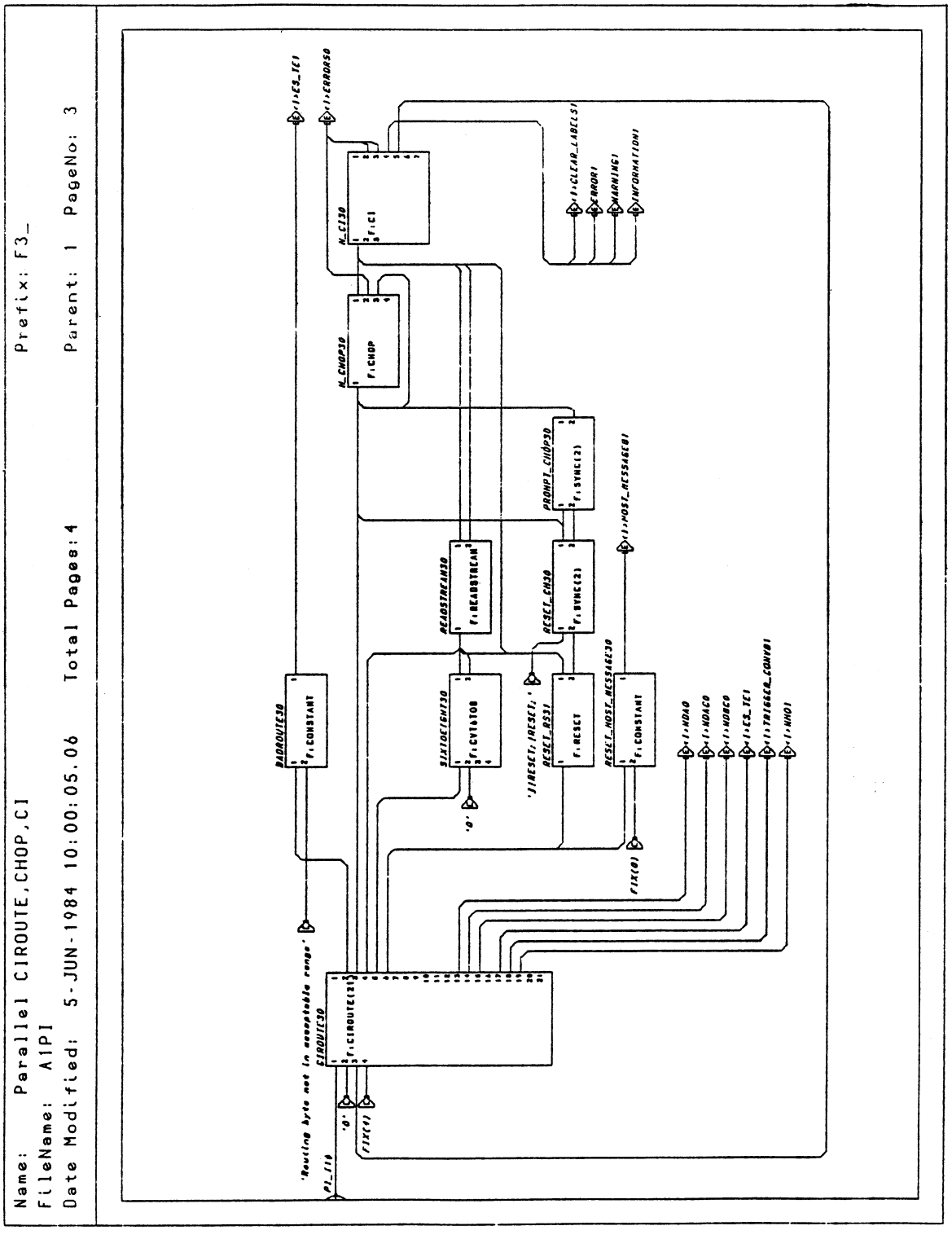

Name: 56K Circuite/Chop/CI
FileName: A156K
Date Modified: 5-JUN-1984 09:58:13.49
Total Pages:4
Prefix: F3_
Parent: 1 PageNo: 3
F1:CIROUTE(3)
F1:CHOP
F1:CONSTANT
F1:CVTATOR
F1:RESET
F1:READSTREAM
F1:SYNC(2)
F1:SYNC(3)
F1:CONSTANT
F1:C1
'Routing byte not in acceptable range'
'0'
FIX(4)
FIX(0)
I:ES_TE1
I:ERRORS0
I:CLEAR_LABELS1
I:ERROR1
I:WARNING1
I:INFORMATION1
I:HOST_MESSAGE01
I:NDA0
I:NDAC0
I:NDBC0
I:ES_TE1
I:FA166GA_CONV01
I:NHO1

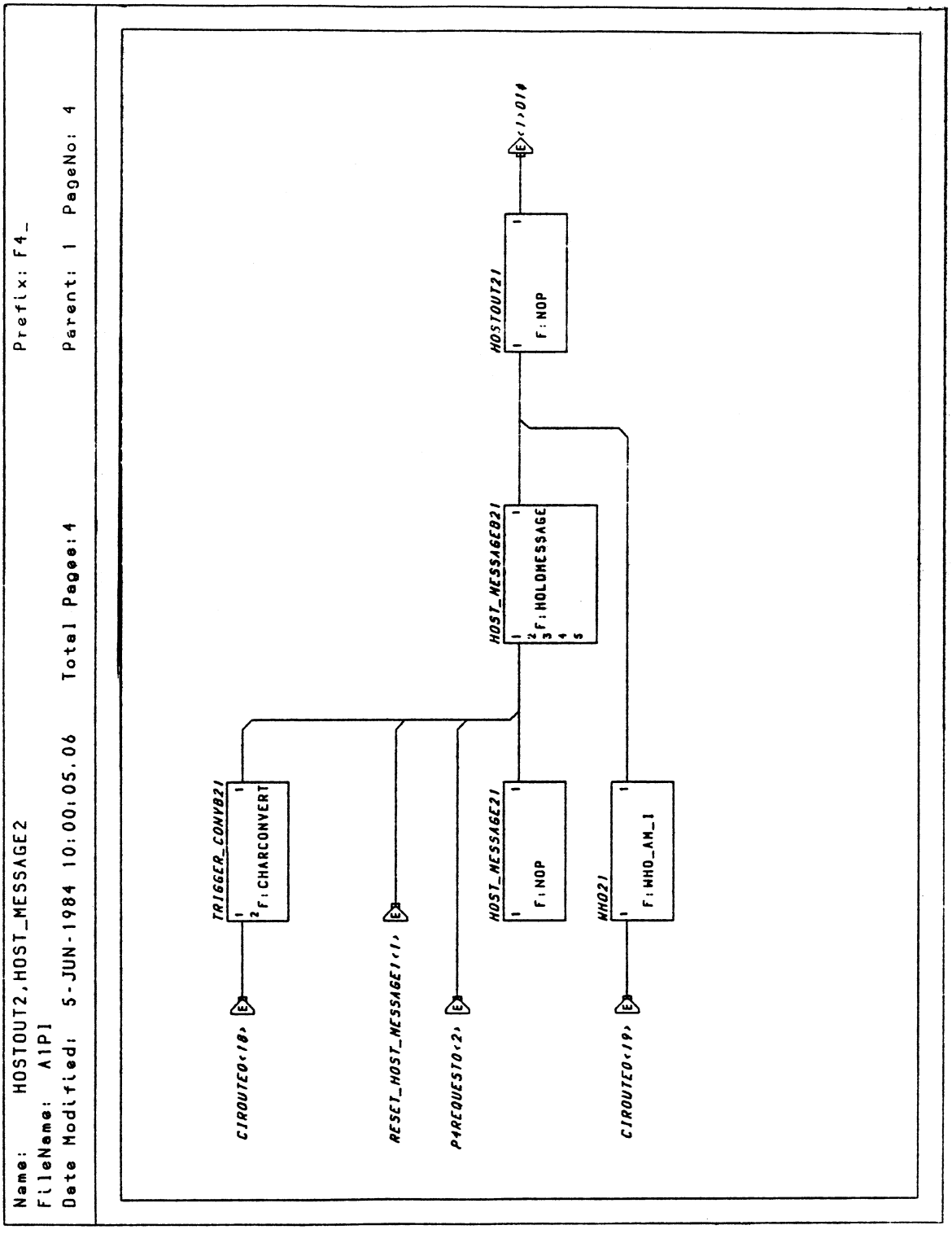

Name: HOSTOUT2,HOST_MESSAGE2
FileName: A156K
Date Modified: 5-JUN-1984 09:58:13.49
Total Pages:4
Prefix: F4_
Parent: 1 PageNo: 4
HOSTOUT21
F:NOP
HOST_MESSAGE021
F:HOLDMESSAGE
HOST_MESSAGE21
F:NOP
HHO21
F:HHO_AM_I
TRIGGER_CONV021
F:CHARCONVERT
CIROUTED<18>
RESET_HOST_MESSAGE1<1>
PXREQUEST0<2>
CIROUTED<19>

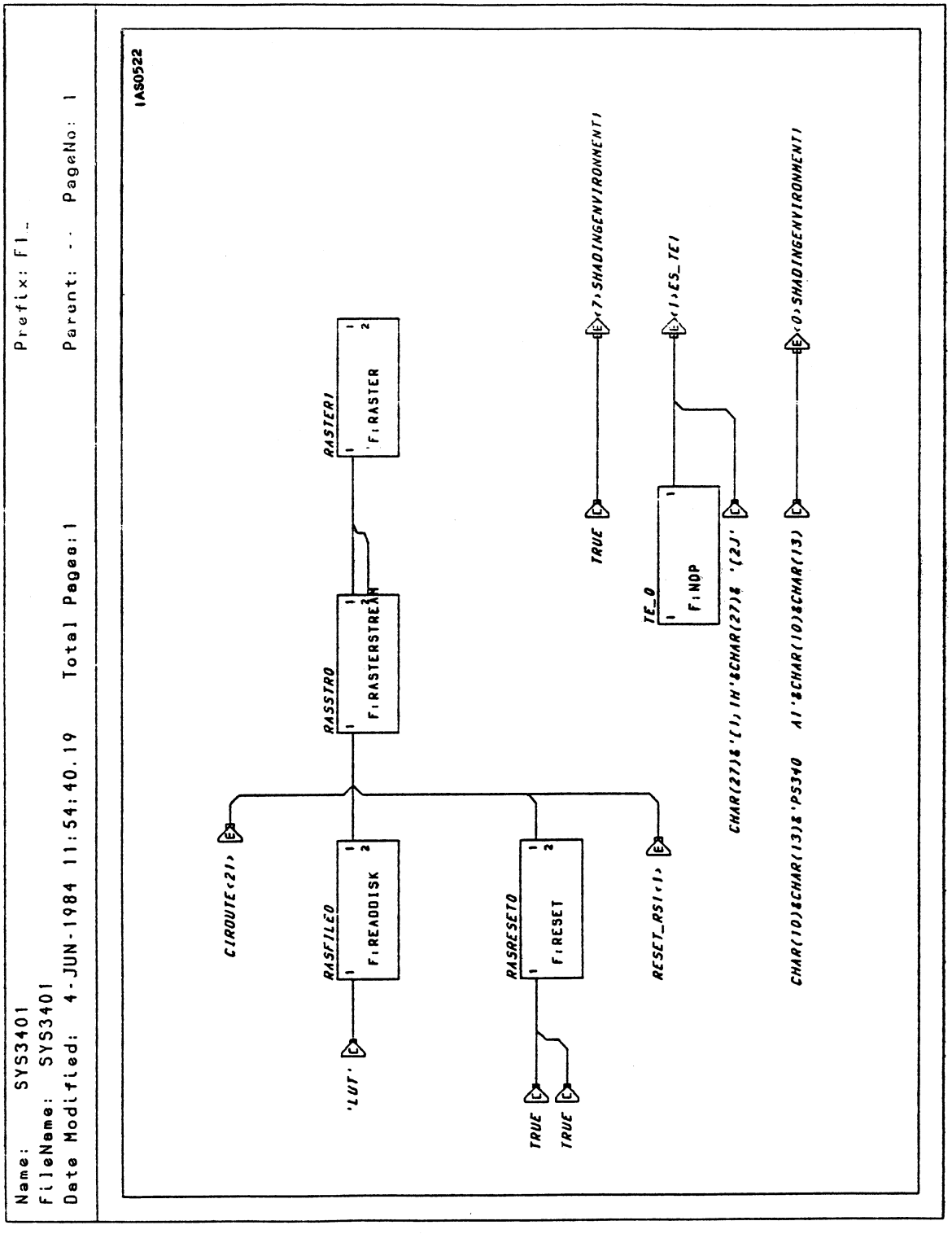

Name: SYSPI1
FileName: A1PI
Date Modified: 5-JUN-1984 10:00:05.06
Total Pages:4
Prefix: F1_
Parent: -- PageNo: 1
IAS0521
GPIOSETUP1
1
F:SETUP1BM
GPIOSETUP3
1
F:SETUP1BM
HOSTOUT1<1>
E
E<1>PI_01#
PI_11#<1>
E
F3_
Pg 3
Parallel CIRCUIT. CHOP, F1
PI_11#
F1_
Pg 1
HOSTOUT2, HOST_MESSAGE2
F2_
Pg 2
Assorted connections

Name: Assorted connections
FileName: A1P1
Date Modified: 5-JUN-1984 10:00:05.06
Total Pages: 4
Prefix: F2_
Parent: 1 PageNo: 2




Name: Parallel CIROUTE,CHOP,CI
FileName: AIPI
Date Modified: 5-JUN-1984 10:00:05.06
Total Pages:4
Prefix: F3_
Parent: 1 PageNo: 3
F.CONSTANT
F.CHOP
F.CI
F.CIROUTE(3)
F.EVTOTOB
F.READSTREAM
F.SYNC(2)
F.RESET
F.CONSTANT
'Routing byte not in acceptable range'
'0'
FIX(0)
FIX(1)
'1'RESET;'0'RESET;'
RESET_RST1
CIROUTE30
BADROUTE30
SIXTOEIGHT30
READSTREAM30
RESET_CH30
PROMPT_CHOP30
RESET_HOST_MESSAGE30
N_CHOP30
N_CI30
HOST_MESSAGE01
CLEAR_LABELS1
ERROR1
WARNING1
INFORMATION1
CES_IC1
CERROR30
HDAC0
HDAC1
HDAC2
HDAC3
CES_IC1
TRIGGER_CONV01
WHO1

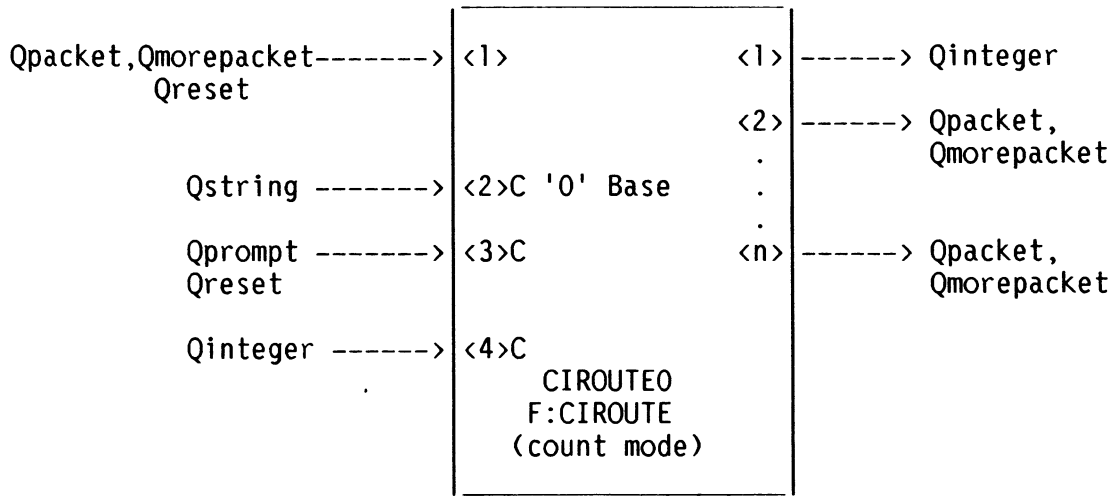

Name: HOSTOUT2,HOST_MESSAGE2
FileName: A1P1
Date Modified: 5-JUN-1984 10:00:05.06
Total Pages:4
Prefix: F4_
Parent: 1 PageNo: 4
HOSTOUT21
F:NOP
HOST_MESSAGE821
F:HOLDMESSAGE
TRIGGER_CONV821
F:CHARCONVERT
HOST_MESSAGE21
F:NOP
WHO21
F:WHO_AM_I
CIROUTEO<18>
RESET_HOST_MESSAGE1<1>
P4REQUEST0<2>
CIROUTEO<19>


Name: SYS3401
FileName: SYS3401
Date Modified: 4-JUN-1984 11:54:40.19
Total Pages: 1
Prefix: F1_
Parent: --
PageNo: 1
IAS0522
RASTER1
F:RASTER
RASSTRO
F:RASTERSTREAM
RASFILE0
F:READDISK
RASRESET0
F:RESET
TE_0
F:NOP
CIROUTE<21>
'LUT'
TRUE
TRUE
RESET_RS1<1>
TRUE
<7>SHADINGENVIRONMENT1
<1>ES_TE1
CHAR(27)&'(1,1N'&CHAR(27)&'(2J'
<0>SHADINGENVIRONMENT1
CHAR(10)&CHAR(13)&'PS340 A1'&CHAR(10)&CHAR(13)

5.2 DATA RECEPTION AND ROUTING NETWORK

F:CIROUTE

Once data have passed through either instance of F:DEPACKET (described in the previous chapter), the next function to receive it is F:CIROUTE. F:CIROUTE has two instances, one for count mode and one for escape mode. They are functionally very similar, and only the count mode instance, CIROUTE0 will be described. CIROUTE0 examines the first character it receives (the character following the count bytes in count mode and the character following the <FS> character in escape mode) to determine where the packet message is to be sent. These characters are "routing" bytes, and are used to select the appropriate channel for data in the PS 300. Data channels include lines to the terminal emulator, the PS 300 command interpreter, the Disk writing function, the Raster function (for PS 340 systems), and other system functions. A base character (defined on Input <2> of CIROUTE0) is subtracted from this routing character before it is used to select the output channel. The base character defaults to the character zero ("0").



The definitions for the inputs and outputs for F:CIROUTE are described in Chapter 6 of this guide.

5.2.1 Routing Byte Definitions

The value of the routing bytes are given in the following table.

Table 5-1. Routing Byte Definitions

	ROUTING BYTE	DESTINATION
3	'0'	Command Interpreter (CI)
4	'0' + 1 ('1')	8-bit binary data
5	'0' + 2 ('2')	6-bit binary data
6	'0' + 3 ('3')	reset network for GSRs
7	'0' + 4 ('4')	reserved for future use
8	'0' + 5 ('5')	reserved for future use
9	'0' + 6 ('6')	download user-written functions
	.	
	.	
	.	
	'0' + 10 (':')	write to disk ASCII data
	'0' + 11 (';')	close disk
	'0' + 12 ('<')	write to disk GSR binary data
	'0' + 13 ('=')	reserved for future use
	'0' + 14 ('>')	Terminal Emulator (TE) input
	'0' + 15 ('?')	HOST_MESSAGEB1 (used by PSPOLL and PSREAD)
	'0' + 19 ('@')	WHO (used by PSETUP)

NOTE

('?') is the HOST_MESSAGE request channel. <SOP>? followed by ASCII (1 or 2) requests a single message or multiple messages from HOST MESSAGEB.

('@') any message sent on this route triggers the WHO function. (Refer to the *PS 300 Host-Resident I/O Subroutine Manual for information on the WHO function.)*

5.2.2 Using the Routing Bytes for Local Data Flow

Routing can be done in a number of different ways, but every data transfer must be preceded by the <ACK> character (count mode) or an <FS> character (escape mode), and a routing byte that gives the destination of the data.

If data are to be sent from the host to the Command Interpreter (CI), the file containing the CI routing bytes would precede the data, and would contain the following characters:

↑\0

To route the line from the PS 300 Command Interpreter back to the Terminal Emulator, a file should contain this sequence:

↑\>

A simple way to include the required characters for escape mode is to build a file on the host with the routing byte to the CI, ↑\0 (<FS> character for escape mode, plus the toggle to the CI) at the beginning of the file, and the routing byte to the TE, ↑\> (<FS> character for escape mode, plus the toggle to the TE) at the end of the file.

Or, two files can be built on the host, one containing the routing byte to the CI, and one containing the routing byte to the TE. These files would act as headers and trailers for the file containing the data on the host that is to be sent to the PS 300.

The file containing the routing byte to the TE should be sent immediately after sending the file routed to the Command Interpreter so that any messages from the host are sent to the TE, rather than the CI.

Routing back to the TE is essential if the TE is being used to download the file. To get the host prompt back after downloading the file, the line must be routed back to the TE mode (↑\>). If the routing byte was not sent, the following command can be entered from the keyboard in CI mode to route back to the TE:

SEND TRUE TO <1>RESET_TE;

If the escape mode <FS> characters appear in the PS 300 command file, they must be prefixed by the escape sequence DLE (↑P). The ↑P (octal 20, decimal 16) immediately preceding the FS characters will identify the characters as being non-muxing data to be passed along.

The ↑\ <FS> character, the ↑F <ACK> character, and the escape sequence (↑P) can be changed by the user in the SITE.DAT file. This should be done when the sequences used by E&S are incompatible with the host or have another site-specific value.

Refer to Chapter 2 for information on creating and downloading the SITE.DAT file.

5.2.3 Output Port Definitions of CIROUTE0 in Count Mode

Output<1> sends out invalid routing bytes.

Output<2> sends any message that does not have a valid routing character. The message is sent to BADROUTE0, and the message 'Routing byte not in acceptable range' is output as an error message to ES_TE1 for screen display.

Output<3> sends messages that were preceded by the routing character "\0" to the H CHOP0 (PS 300 F:CHOP function) DEPACKET and CIROUTE strip the ↑F and count. The function chops and parses the input command language generating proper messages for the Command Interpreter (CI). Once chopped and parsed, the message is sent on output<1> of CHOP to the Command Interpreter (CI). F:CHOP is also responsible for generating syntax error messages.

Output<4> sends messages to Readstream0 (F:READSTREAM). (GSR or binary PSIO)

Output<5> sends messages to the SIXTOEIGHT0 function to convert six-bit to eight-bit binary, then on to the F:READSTREAM mentioned above.

Output<6> sends messages to RESET_RS1 (F:RESET)and RESET_HOST_MESSAGE1 which causes the functions accepting GSR data to be reset to initial state.

Output<9> is used for the user-written function facility.

Output<13> sends messages to WDA0 (F:WRITEDISK), which writes ASCII commands to DISK.

Output<14> sends messages to WDAC0, that is another instance of F:CHOP and is used to interpret the command to close the file sent via outputs 13 and 15 to disk.

Output<15> sends messages to WDBC0, that is used to chop binary data that will be written to the diskette.

Output<17> sends messages prefixed by '↑Fcntcnt> to ES_TE1, the terminal emulator function.

Output<18> sends messages to TRIGGER_CONVB1 (F:CHARCONVERT), that then sends messages on to input <1> of HOST_MESSAGEB.

Output<19> sends messages to the WHO1 function, that sends a package with the system information back to the host.

From the description of where packets, or messages, are sent to from F:CIROUTE, it is apparent that CIROUTE is the main function responsible for data flow from the host to the PS 300. The queues in CIROUTE are triggered by the routing bytes. For data to be directed down the proper channel, all messages that will pass through F:DEPACKET and F:CIROUTE must be prefixed with the appropriate count or escape mode Start of Packet character, and the routing byte that signifies the packet's destination.

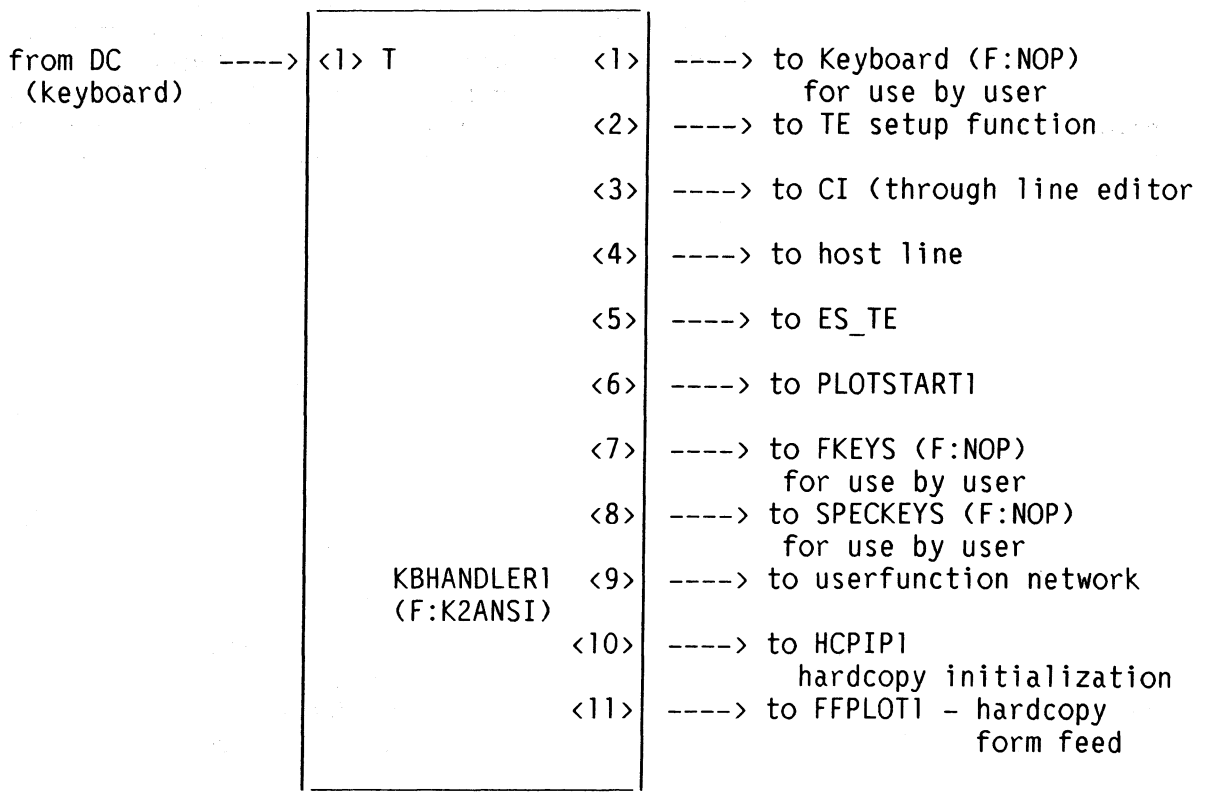
5.2.4 The Terminal Emulator Network

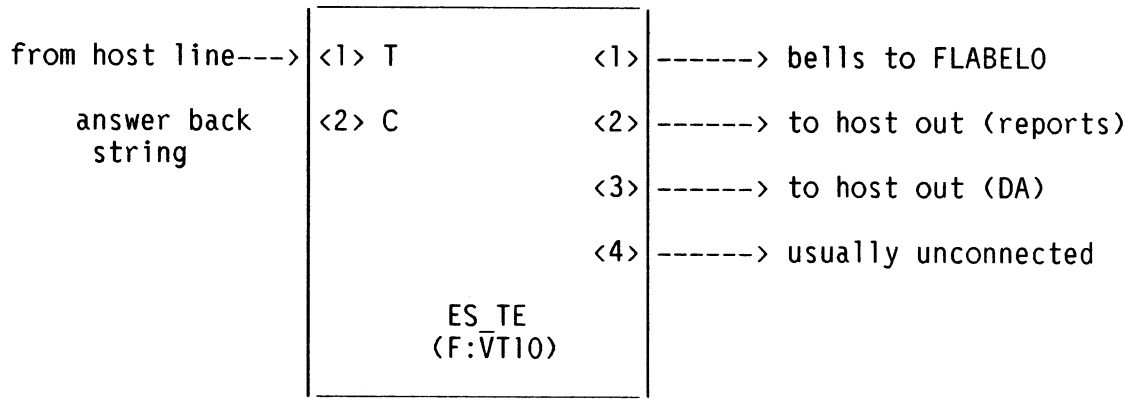
The terminal emulator network is responsible for directing the flow of data that will appear on the screen as text or graphics display, for determining certain attributes of the terminal emulator feature, for routing input from the PS 300 keyboard to the host or to local PS 300 system functions, and receiving data from the host line.

The Terminal Emulator facility is discussed in detail in Chapter 8 of this guide. This section will provide an overview of the terminal emulator data routing responsibilities.

Data enters the terminal emulator network through either KBHANDLER1, an instance of F:K2ANS1, or ES_TE, an instance of F:VT10. KBHANDLER1 is referred to as the key manager facility of the terminal emulator. It takes the stream of raw bytes from the keyboard and distributes them to output queues, translating to ANSI control sequences if necessary. It also toggles the graphics and terminal emulator displays

ES_TE, the terminal emulator display handler, receives input from a number of sources: from the host, from an error formatting function, and from the line editor that receives input from the keyboard in the CI (Command) mode. The primary task of the display handler is to route this input to the PS 300 screen.





5.2.5 Summary of the System Network

The block diagrams provided in the front of this section are the best guide to the rest of the PS 300 system network. The diagrams give both the generic function name and the instance name. Most of the generic functions shown on the block diagrams are defined in the next section, System Functions.

The Terminal Emulator is discussed in detail in Chapter 8, Terminal Emulator Modes.

CONTENTS

TABLES

6. SYSTEM FUNCTIONS

This chapter gives a brief overview of PS 300 system functions, definitions of "constant" and "trigger" queues, and a list of data types for inter-function communication. The bulk of the chapter is a System Function Summary. Each system function is shown with a box diagram, acceptable data types for input and output ports, and a description of each function.

6.1 DEFINITION OF A FUNCTION

A function is a Pascal procedure that performs one or more operations by accepting input, processing input, and sending output.

Functions must be "instanced" before they can be incorporated into a function network. Instancing is the process of creating a unique case of the function. The unique case is identified by the user-given name, the input and output connections of the function, and a user-identification number (for dual-user systems).

6.2 CONSTANT QUEUES AND ACTIVE QUEUES

Function input queues are of two different types:

1. Constant queues, where the queue retains a message until another message is received on that input, and any previous message on that queue is removed. Constant queues cannot be emptied unless the data on the queue is of an inappropriate type. The message is not "used up" by the function.

2. Trigger queues, where all data are collected on the queue, and then input to the function on a "first in first out" basis as soon as the function is activated. Messages on active queues are "used up" by the function.

A function always requires something to be on every input queue before it can execute.

The programmer has the ability to change the type of input queues on a function. The following PS 300 command:

SETUP CNESS TRUE <n>Name; {makes it a constant queue}

SETUP CNESS FALSE <n>Name; {makes it an active queue}

will set the nth queue of the function Name to be a Constant queue if TRUE is entered, and to be a Trigger queue if FALSE is entered. Unless specified otherwise, input queues are by default Trigger queues.

This feature should be used only when a function is first instanced. Input queues should not be changed between trigger and constant at any time after the function has started processing data.

This feature is accessible for most user instanceable functions. Functions which specify their queue characteristics by their name (i.e. f:Addc) will continue to be instanced with the same defaults as before. There are a few functions which, because of their nature, are not allowed to change queue characteristics. These functions are:

System functions
All I/O functions

CI
FCNTAPUP
WHO_AM_I
GATHER_GENFCN
VT10
K2ANSI
TEDUP

User functions

LINEEDITOR
CLCSECONDS
CLFRAMES
CLTICKS
BOOLEAN_CHOOSE
MATRIX
CMATRIX
INTEGER_CHOOSE
INPUTS_CHOOSE

When this command is applied to one of these functions, the following error message is given:

E 102 *** Cannot affect Cness for its generic function: (Name)

When this command is applied to a name that is not a function instance the following error message is given:

E 95 *** Name must be a function instance

6.3 QUEUE DATA TYPES

Blocks of data passed between functions are referred to as Qdata message blocks. The names and definitions of the general data types acceptable by function input queues are given in the following table:

Table 6-1. Qdata Type Definitions

Qdata Type	DEFINITION
Qreset	Dataless: reset a function instance
Qprompt	Dataless: flush the CI pipeline
Qboolean	Normal carrier of boolean values
Qinteger	Normal carrier of integer values
Qreal	Normal carrier of floating point values
Qpacket	Normal carrier of byte strings
Qmorepacket	Alternate to Qpacket as carrier of byte string on input to PS300 (only occurs as output from F:DEPACKET, F:CIROUTE)
Qmove2	2D vector including P bit

Table 6-1. Qdata Type Definitions (continued)

Qdata Type	DEFINITION
Qdraw2	2D vector including the L bit
Qvec2	2D vector with no P/L bit (normal vector)
Qmove3	3D vector including P bit
Qdraw3	3D vector including the L bit
Qvec3	3D vector with no P/L bit (normal vector)
Qmove4	4D vector including P bit
Qdraw4	4D vector including the L bit
Qvec4	4D vector with no P/L bit (normal vector)
Qmat2	2x2 matrix (all matrices use 4x4 indexing)
Qmat3	3x3 matrix
Qmat4	4x4 matrix

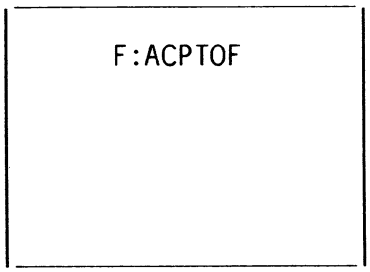
6.4 SYSTEM FUNCTION SUMMARY

The following sections provide a "black-box" diagram showing available inputs and outputs and acceptable Qdata types for generic system functions. A brief description of each function is given along with any notes or references that may be helpful.

The functions are presented in alphabetical order. Instance names of functions that are used in the system function network are shown where applicable in parentheses at the the bottom of the function diagram. The suffix is shown for a single-user system.

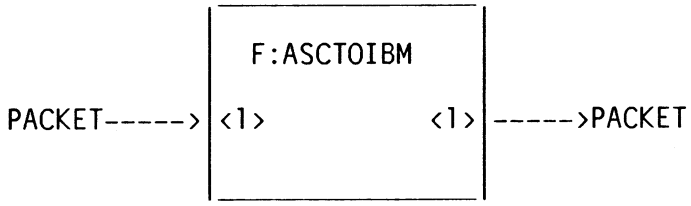
For more information on the handling and scheduling of functions, refer to Chapter 7, "Initial Data Structures."

F:ACPTOF



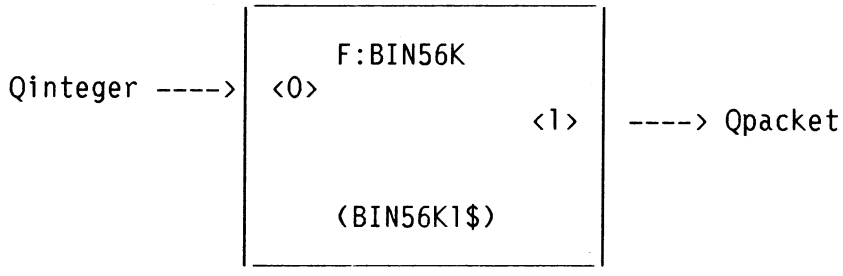
This function is executed when the ACP has been processing a frame for longer than a specified time limit. It executes after the user's display structure has been removed (see F:USRTOF) and restarts the ACP.

F:ASCTOIBM



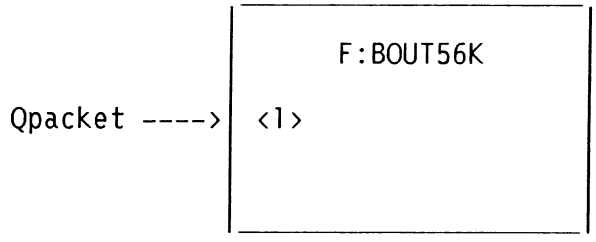
F:ASCTOIBM accepts packets of ASCII characters on input <1> and outputs packets of characters in IBM screen format on output <1>.

F:BIN56K



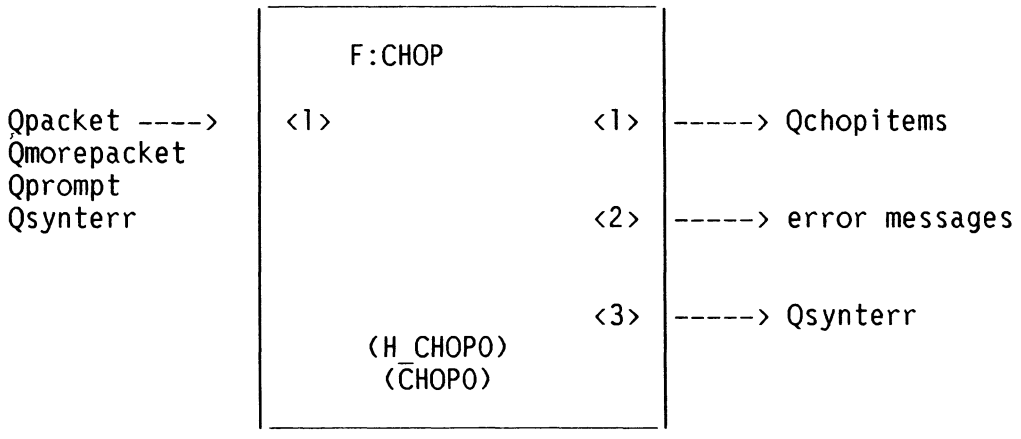
This function is the input function for the DMR11 host line. It sends the bytes it receives out output <1> as Qpacket.

F:BOUT56K



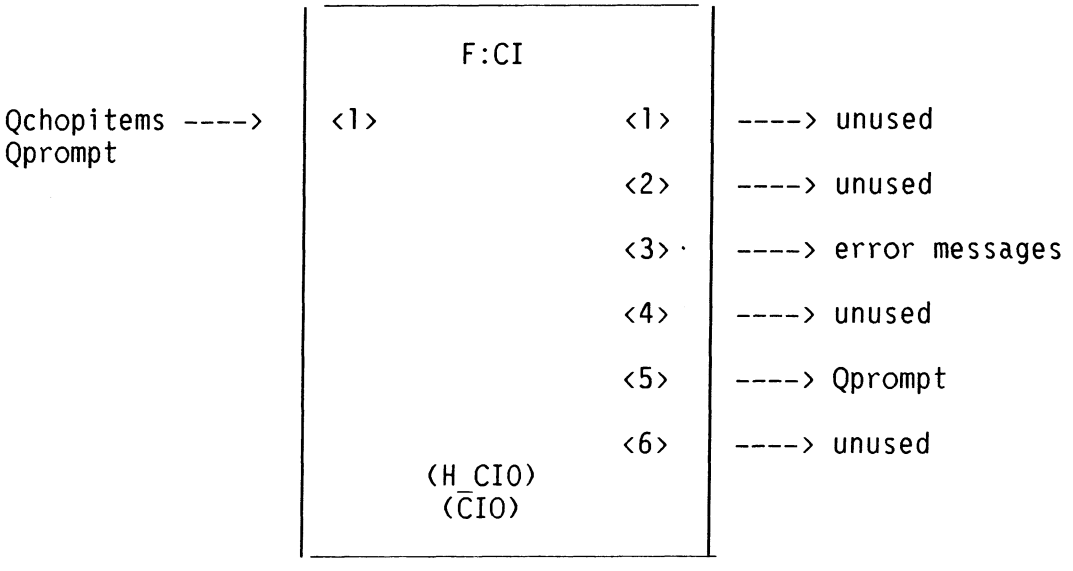
This is the output function to the DMR11 line. It sends the bytes it receives on input <1> to the host on the DMR11 line.

F:CHOP



This function chops and parses the input command language generating proper messages (Qchopitems) for the CI. Message of type Qsynterr directs printing of syntax error messages.

F:CI



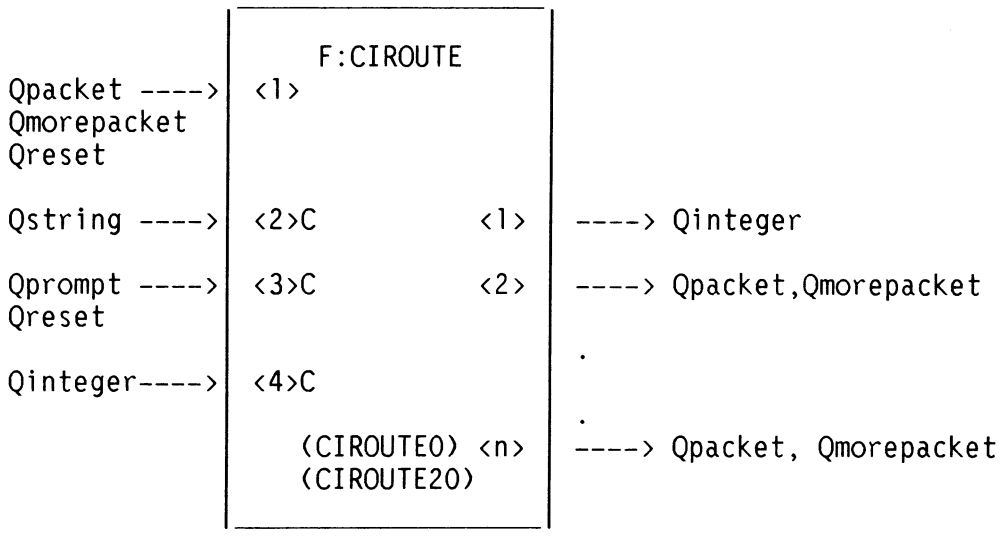
This function interprets commands, creating Display structures and function networks. It receives input either from a chop/parse function or a Readstream function (if using the GSRs).

A single parameter is given when this function is instanced (for example H CIO:=F:CI(4);). This parameter is the "CINUM" and is used to identify all names and connections this CI makes. When the CI receives an INIT command, it destroys only those connections it has made and only those structures associated with the names which have its CINUM.

Note: A name is created when that name is referenced for the first time, even if it has no associated structure. The CI that created the name is the "owner" of that name, even if the entity it refers to is created by another CI.

Note: Each function has an output <0> that is used to send error messages (such as illegal input error messages). The connection from this output is made automatically by the CI that creates the function. The CI finds the appropriate error function to connect output <0> to by looking on its own output <3>.

F:CIROUTE



DEMUX demultiplexes a stream of Qpackets/Qmorepackets from input <1> to one of the n output channels. The first byte of an incoming Qpacket is assumed to be the multiplexing byte -- equal to the base character (from input <2>) + K, where K is the channel number. If K > (n-3) or K < 0, there is no channel for this output and a pair of messages are sent on outputs <1> and <2>. These can be used to allow for later remultiplexing or further demultiplexing. An integer giving the indicated output port is sent on output <1> and the message for which there was no defined output is sent on output <2>. Whether or not K is within the limits implied by the number of outputs of DEMUX, the multiplexing byte is removed from the start of the packet.

Depacket passes incoming Qmorepackets out the current channel (as defined by the last Qpacket). Initially, after a Qreset is received, the current channel is -1.

When instancing this function, a parameter is required to specify the number of outputs.

F:CIROUTE is a special version of F:DEMUX. It assumes that it is driving parallel, asynchronous paths to a common destination (the command interpreter). It synchronizes those paths by sending out a Qprompt on a channel at the end of using that channel and waiting for it to come back around before switching to the next channel. This assumes that the common destination can strip Qprompts and send them back (which the CI does). Input <4> gives the maximum channel number, m, for which path flushing is desired. CIROUTE flushes channels 0<=K <= m with Qprompts.

F:CIROUTE (continued)

Note: because of the synchronizing nature of CIROUTE, it should be hooked to its complete destination network before its input <1> is connected.

Inputs

<1> Qpacket --- switch multiplexing channel & send
Qmorepacket --- send on current channel
Qreset --- re-init & purge queue 3.

<2>C Qstring --- base character of multiplexing byte

<3>C Qprompt (back from the CI)
Qreset --- acts like Qprompt

<4>C Qinteger --- max channel # to get prompts (default 0)
must be an individual output (not <1>,<2>)

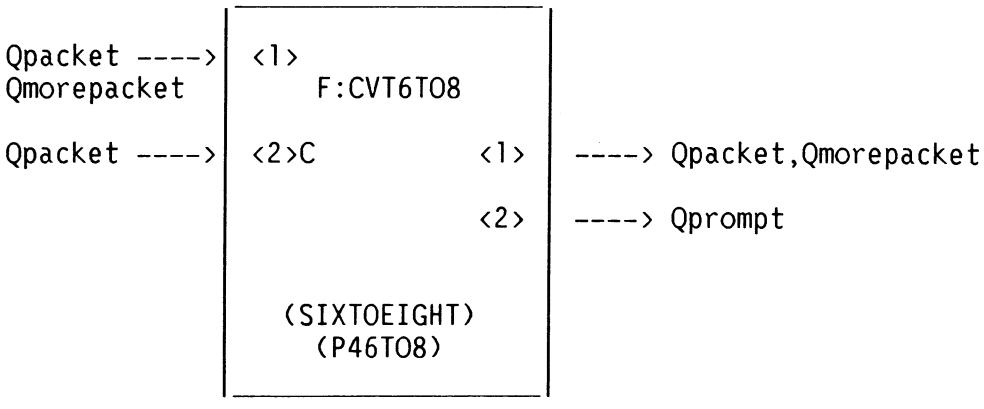
Outputs

<1> Qinteger --- (i) when output port <3+i> doesn't exist

<2> Qpacket, Qmorepacket --- stream which didn't have any valid destination

<3+i> Qpacket, Qmorepacket --- output stream (i) where destination was given as [base-char + (i)] (possibly) Qprompt (if i <= [input <4>])

F:CVT6TO8

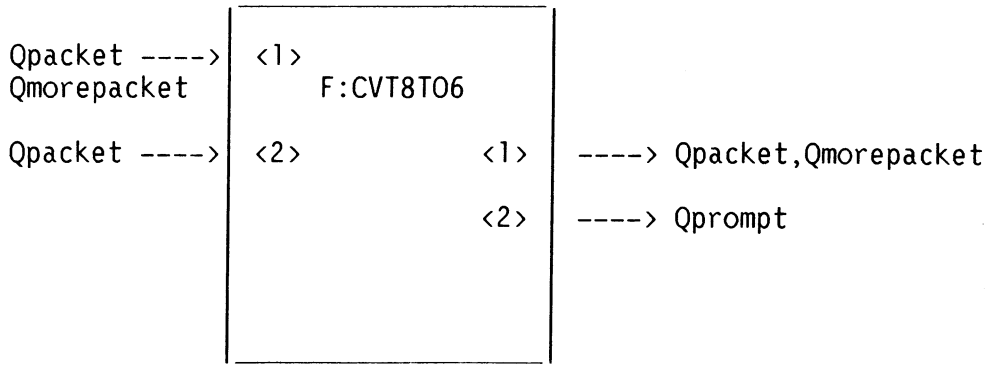


This function converts a 6-bit stream to an 8-bit stream.

Conversion is the inverse of that described in F:CVT8TO6 with the base character coming on input <2>.

Qprompts are passed out output <2>.

F:CVT8TO6

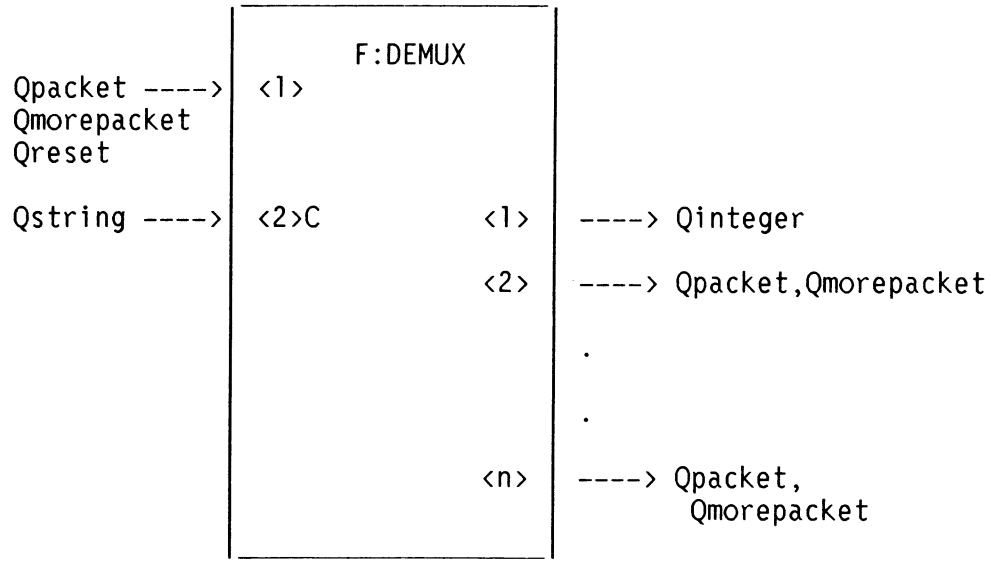


This function converts an 8-bit byte stream to a 6-bit byte stream.

The conversion yields a stream with characters from base-char (from input <2>) through base-char + 63 standing for 6-bit values in groups of 6. In addition, the special characters take care of streams that do not have a byte count = 0 mod 4. Prefixing the last group of 6 output bytes (encoded), the char: base - i: means the last i 8-bit bytes are not real.

Any Qprompts coming in on <1> are passed out output <2>.

F:DEMUX



F:DEMUX demultiplexes a stream of Qpackets/Qmorepackets from input <1> to one of the n output channels. The first byte of an incoming Qpacket is assumed to be the multiplexing byte -- equal to the base character (from input <2>) + K, where K is the channel number. If K > (n-3) or K < 0, there is no channel for this output and a pair of messages are sent on outputs <1> and <2>. These can be used to allow for later remultiplexing or further demultiplexing. An integer giving the indicated output port is sent on output <1> and the message for which there was no defined output is sent on output <2>. Whether or not K is within the limits implied by the number of outputs of DEMUX, the multiplexing byte is removed from the start of the packet.

Depacket passes incoming Qmorepackets out the current channel (as defined by the last Qpacket). Initially, after a Qreset is received, the current channel is -1.

When instancing this function, a parameter is required to specify the number of outputs.

F:DEMUX (continued)

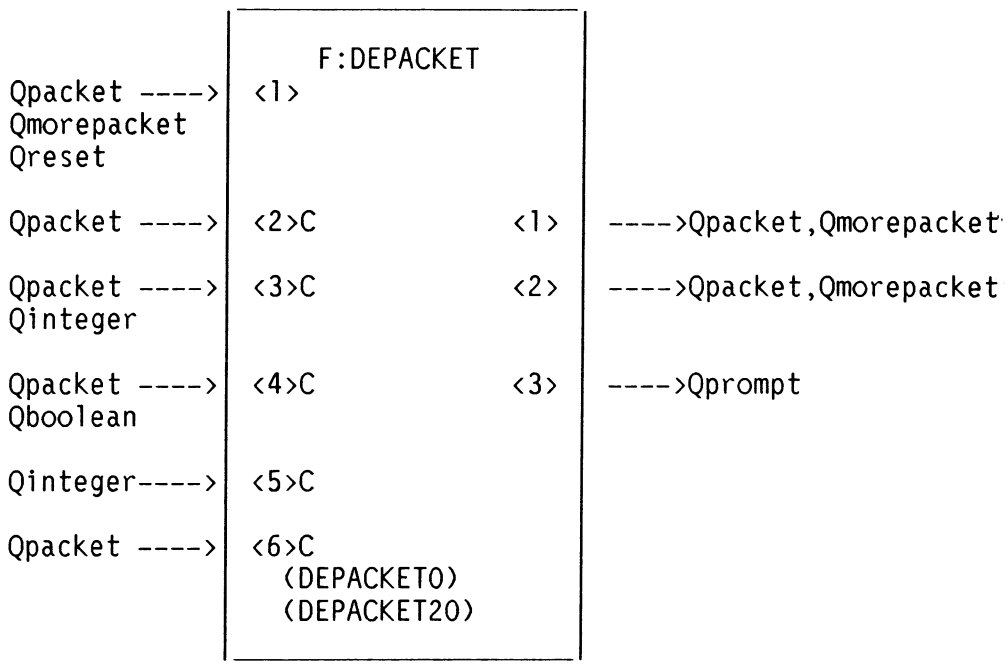
Inputs:

<1>	Qpacket --- switch multiplexing channel & send Qmorepacket --- send on current channel Qreset --- current channel becomes -1 (invalid)
<2>C	Qstring --- base character of multiplexing byte

Outputs:

<1>	Qinteger --- (i) when output port <3+i> doesn't exist
<2>	Qpacket, Qmorepacket --- stream which didn't have any valid destination
<3+i>	Qpacket, Qmorepacket --- output stream (i) where destination was given as [base-char + (i)]

F:DEPACKET



F:DEPACKET converts streams of incoming bytes to Qpacket/Qmorepacket packages

Inputs

<1> Qpacket --- source stream
Qmorepacket --- treated as if it were Qpacket
Qreset -- Return to initial state (in-between packets); send Qreset out <1>

<2>C Qpacket --- FS: start packet character

<3>C Qpacket --- ESC: escape character, if ESC mode
Qinteger -- # count bytes, if count mode (the type of message on <3> controls ESC versus Count mode)

<4>C Qpacket --- base character, if count mode
Qboolean (default: FALSE) --- is FS escaped?, if escape mode
FALSE: <FS> starts packet
TRUE: <ESC> <FS> starts packet

F:DEPACKET (continued)

<5>C Qinteger --- radix (default 10) if count mode

<6>C Qpacket --- Auto-mux prefix for between-packet streams

Outputs

<1> Qpacket, Qmorepacket --- packet stream

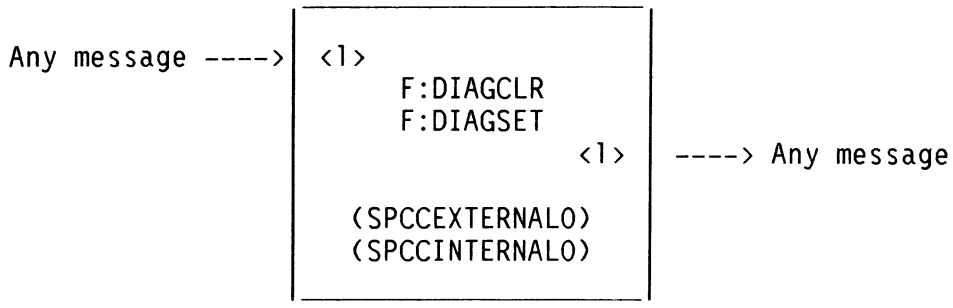
<2> Qpacket, Qmorepacket --- between-packet stream

Note: in ESC mode, once a packet is detected, nothing will ever be between packets again.

<3> Qprompt --- pass through from <1> if any show up. SR_depacket un-escapes the contents of a packet, converting <ESC> <x> to <x>

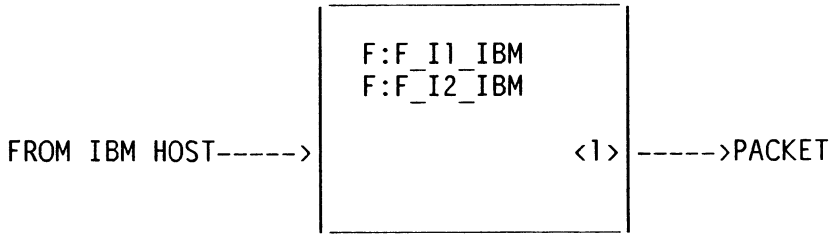
Refer to Chapter 4 for a description of the inputs and outputs of the DEPACKET function.

F:DIAGCLR
F:DIAGSET



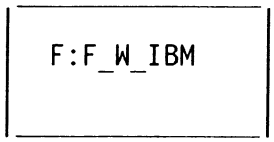
These procedures set diagsync bit in local memory maintenance register when any message is received on input <1>. Input message is sent to output <1>. The SPCCLK bit in the local memory maintenance register is also set/cleared so as to allow user control of the source of the external clock for the SPCC chip (DMR11 system).

F:F_I1_IBM, F:F_I2_IBM



F:F_I1_IBM, F:F_I2_IBM output packets of characters received from an IBM host on output <1>.

F:F_W_IBM



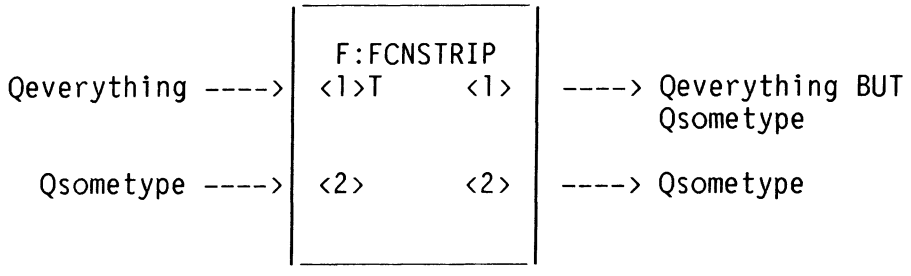
F:F_W_IBM wakes up every frame and disposes of packets which were passed to F:F_O1 IBM, F:F_O2_IBM, F:F_K1_IBM, AND F:F_K2_IBM, as well as waking up F:F_I1_IBM AND F:F_I2_IBM when data have been received from an IBM host.

It also checks to see if the GPIO board has timed out and displays the indicator character

'G'

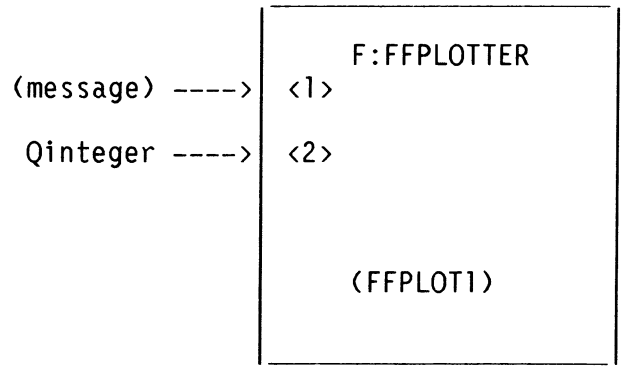
on the TE screen if a timeout has occurred.

F:FCNSTRIP



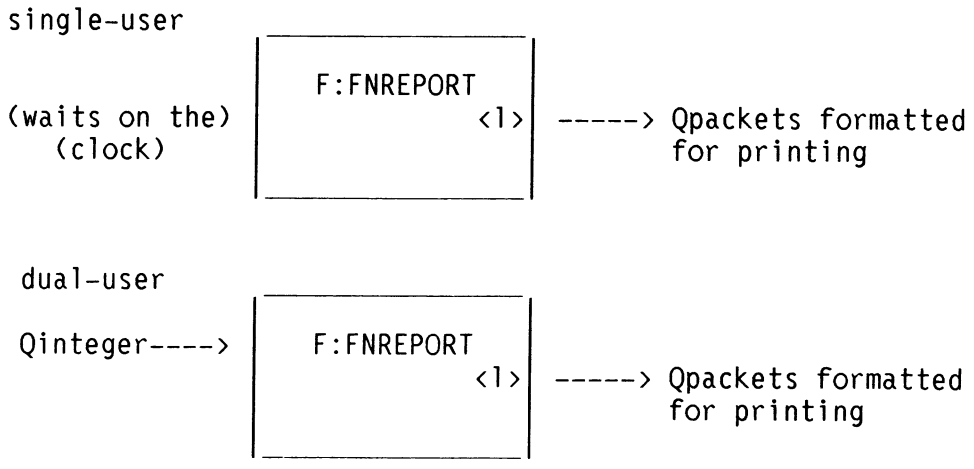
F:FCNSTRIP is used to either FILTER out some Qdata type or, alternately, to SELECT a given Qdata type. The type of incoming messages on <1> is compared to the type of the message on the constant queue <2>. If the types are different, then the incoming message is sent out output <1>, thus filtering out the type on input <2>. If they are of the same type, then output is to queue <2>, effectively selecting that type.

F:FFPLOTTER



When this function receives any message on input <1>, it causes a form feed to be sent to the indicated plotter (indicated on input <2>), if the plotter is not allocated to another user, and there is not a plot in progress.

F:FNREPORT:



F:FNREPORT is a means of gathering running statistics by generic function type. A report is given each wake up that has the following information:

1. Generic function code as an integer.
2. Number of function activations since last report.
3. Average time in microseconds for function execution.
4. Maximum time in milliseconds.
5. The tenths of percent of CPU charged to this generic function.

Note: The integer representation of each function is given in the final section of this chapter.

This function is included in all versions of the PS 330 system, and Async and 3278 versions of the PS 320 system.

F:FNREPORT (continued)

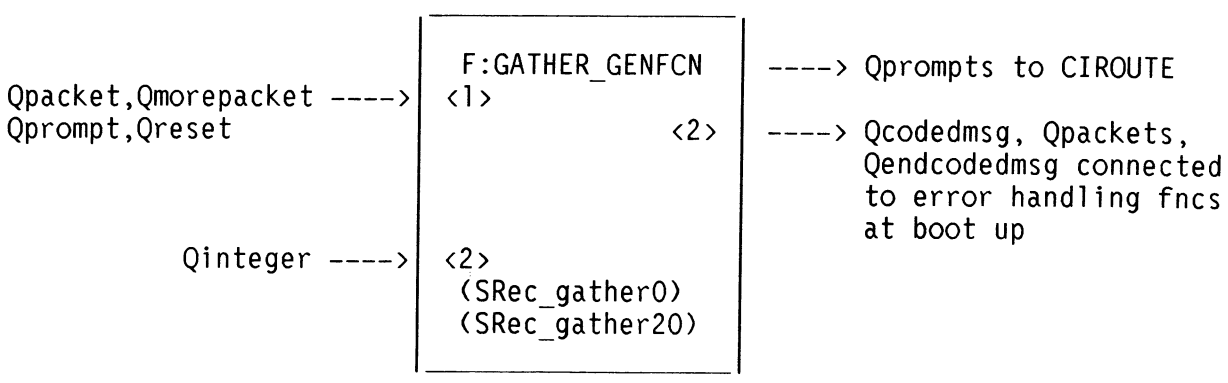
In the single-user system, the function automatically wakes up every minute. To use this function in a single-user system, the following commands should be used:

```
F:= F:FNREPORT
Configure a;        {add suffixes between this command and Finish Configuration
                     command}

Conn F1<1>:<1>03$;  {connect to wherever the report is wanted. Connecting to
                     03$ is convenient if an ASCII terminal with XON-XOFF or a
                     hardcopy terminal can be attached to Port 3}

Finish configuration;
```

In a dual-user system, statistics are kept according to user category and wake-up time is controlled by the user. Input <1> is an integer that triggers the output and also indicates the user-category. In a dual-user system, the following network can be used:

```
FN:=F:FNREPORT;
Conn FN<1>:<1>ES_TE1; {or wherever report is wanted}
{use a clock network similar to what follows for each category}
DRM:=F:clcsec;
Send false to <6>DRM; {hold up for now}
Send fix (6000) to <1>DRM {6000 csecs = 1 min}
Send fix (6000) to <2>DRM;
Send true to <3>DRM;
Send fix (0) to <4>DRM; {for user-category 0}
Send fix (0) to <5>DRM; {so always emits number on input <4>}
Conn DRM<1>:<1>FN; {for triggering report}
SEnd true to <6>DRM; {allow to run}
```

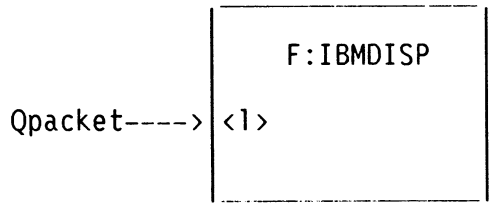
F:GATHER_GENFCN



F:GATHER_GENFCN is used to download the code for user-written functions. The first messages contain information about the user-written function, and the remainder contain Motorola S-records. It gathers the data specifying a user-written function on input <1> and creates a new function.

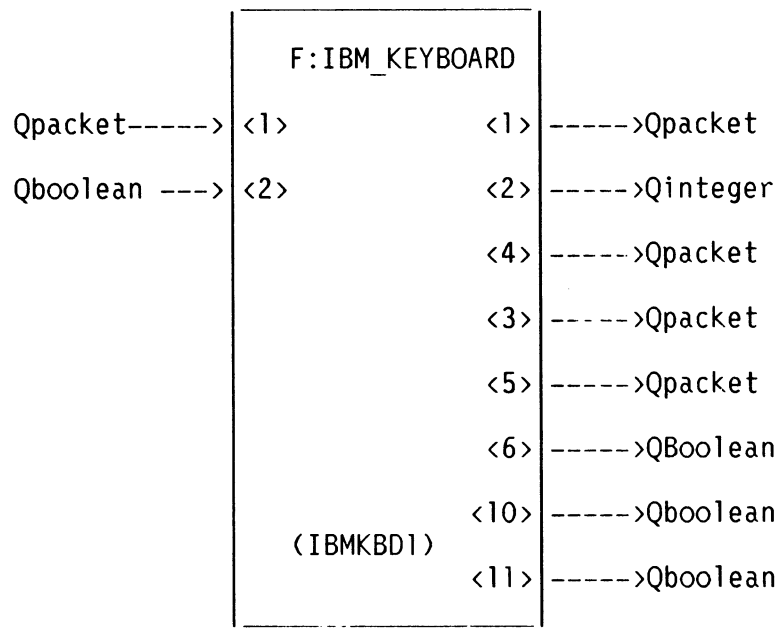
Input <2> receives a CI number to associate the function with an instance of the CI. When the CI receives an INIT command, it will remove the functions created by its associated gather_genfunction. This number corresponds to the parameter given in instancing the F:CI function. Numbers less than 10 are reserved for system use. The number 4 is the default.

F:IBMDISP



F:IBMDISP accepts packets of ASCII characters on input ‹1› and either inserts their equivalent IBM screen code into the local screen buffer used by the Command mode of terminal emulator or causes the cursor position to be adjusted in the case of a carriage return, a line feed, or a back space.

F:IBM_KEYBOARD

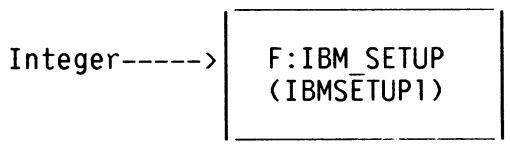


F:IBM_KEYBOARD accepts character packets from the keyboard on input <1> and based on the mode selected by the mode keys (either the LINE/LOCAL key or the HOST, LOCAL and COMMAND keys, depending on the type of keyboard used), outputs packets for use by the function network, the line editor, or an IBM host. Packets of characters for the function KEYBOARD are output on output <1>. Qintegers to be sent to the function FKEYS are output on output <2>. Qpackets of characters to be sent to the function SPECKEYS are output on output <3>. Qpackets of characters for the line editor are output on output <4>. Qpackets of IBM scan codes for an IBM host are output on output <5>. A QBOOLEAN TRUE used to trigger the hardcopy functions is output on either output <6>, output <10>, or output <11>, based on the mode of the keyboard.

Input <2> accepts a Boolean that indicates which type of keyboard is being used.

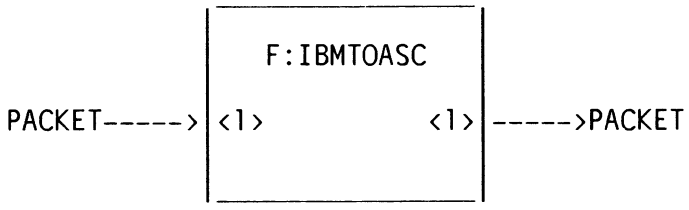
True = IBM style keyboard
False = VT100 style keyboard

F:IBM_SETUP



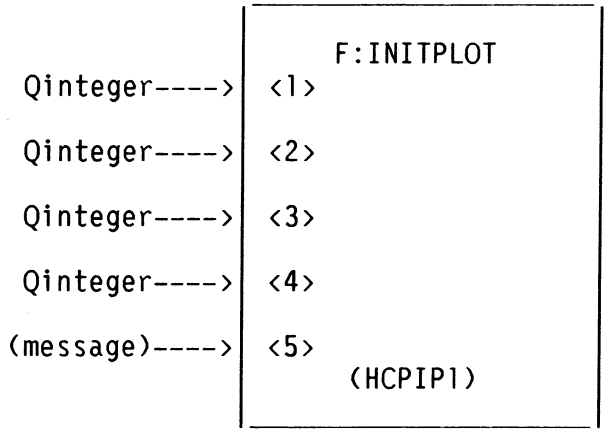
F:IBM_SETUP is used to change the parameters used by the IBM communications. Input <1> accepts an integer that specifies the maximum number of packets that can be in the pool of empty input packets.

F:IBMTOASC



F:IBMTOASC accepts packets of ECBDIC characters on input <1> and outputs packets of ASCII characters on output <1>.

F:INITPLOT



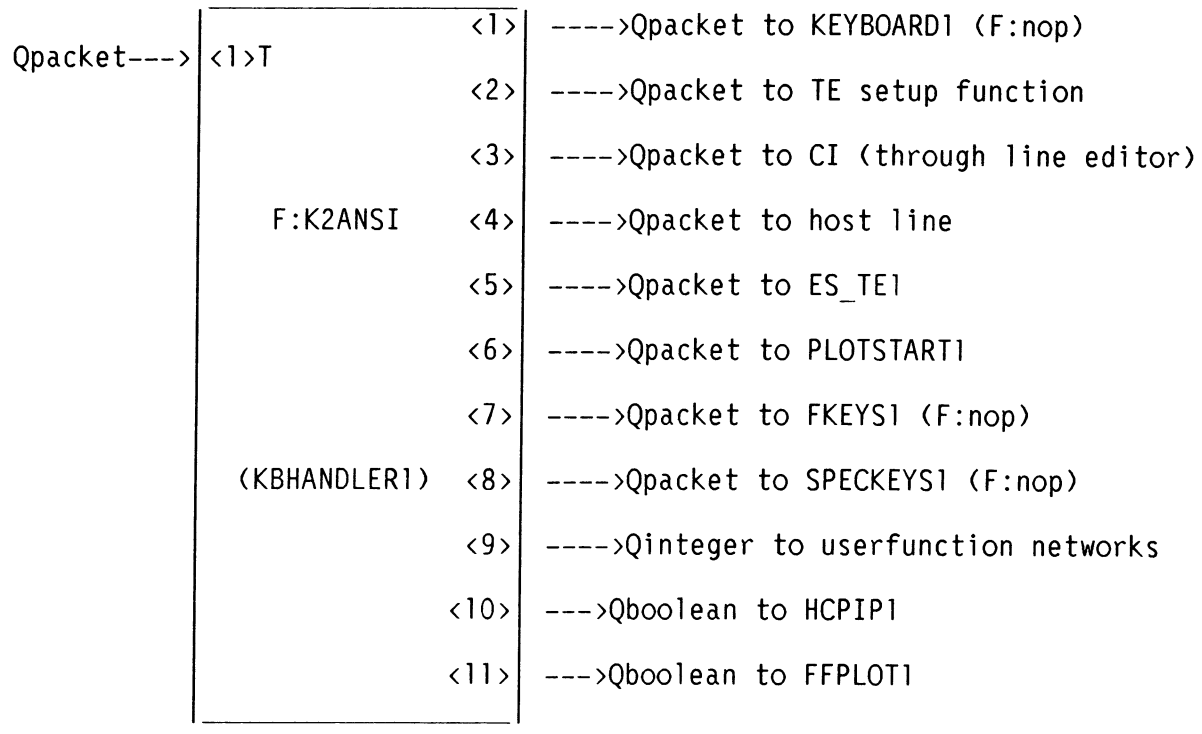
F:INITPLOT initializes the hardcopy card and associated plotters; checks that the hardcopy card and desired plotters are present and passes a subset of resident confidence tests. The function sends error indicators if an error is located.

Inputs:

<1> : Qinteger (Plotter #0 type) Default = Versatec V80
<2> : Qinteger (Plotter #1 type) "
<3> : Qinteger (Plotter #2 type) "
<4> : Qinteger (Plotter #3 type) "
<5> : (message) (Initialization trigger)

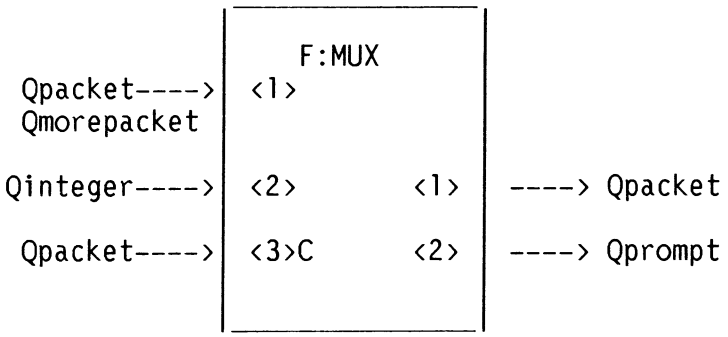
Plotter type: 1 = Versatec V80
2 = Hewlett Packard 2871G

F:K2ANSI



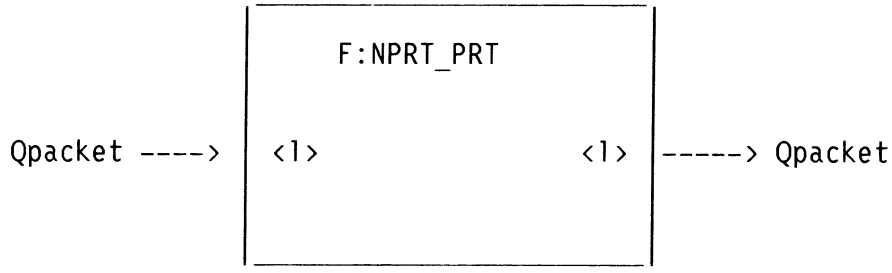
This function takes the stream of raw bytes from the keyboard and distributes them to output queues, translating to ANSI control sequences if necessary; toggles graphics and terminal emulator displays.

F:MUX



F:MUX is the inverse of F:DEMUX. It accepts Qpacket/Qmorepackets and prefixes each incoming bundle of types with the multiplexing character, the base character (from input <3>) plus the channel number from input <2>. <2> Qprompt --- any Qprompts which showed up on input <1>

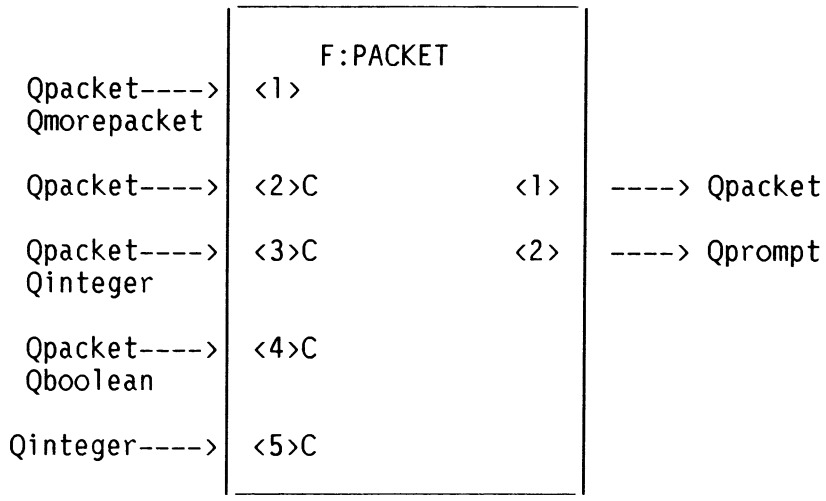
F:NPRT_PRT



This function converts strings containing non-printable characters to strings of printable characters. Example: ↑L to <FF>.

This function is a helpful debugging aid, as it allows non-printable characters to be printed. For example, this function's input could be connected to the function receiving input from the host, and its output connected to the terminal emulator. Then all characters that enter the PS 300 from the host could be seen.

F:PACKET



The F:PACKET function takes incoming Qpacket messages from input <1> and prefixes each one with the proper packet header before sending them on. (It is the inverse of the F:DEPACKET function.) It routes any Qprompts arriving at input <1> off to output <2>. Like F:DEPACKET, this function can operate in either count mode or escape mode. In count mode, F:PACKET makes a packet of the following form:

| FS | count bytes | message body from input <1> |

The definition of the FS character is taken from the single-character string on input <2>. The count is defined to have n bytes (n taken from input <3>). Each count type is offset from the base character (on input <4>). The radix of the count is on input <5>.

In escape mode, the packet is defined as:

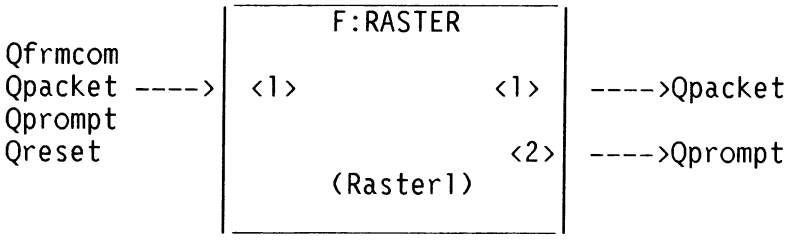
| FS | message body |

if input <4> is false. If input <4> is true, it is defined as:

| ESC | FS | message body |

The definition of FS is taken from the single-character string on input <2>. If input <4> is false, any FS character within the packet is prefixed with ESC. In either mode, any ESC byte within the packet is also prefixed wih ESC.

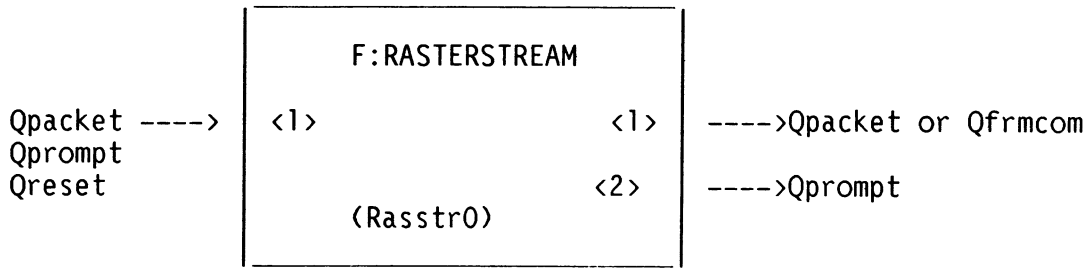
F:RASTER



The function passes the data from F:RASTERSTREAM to the ACP and disposes the information after the ACP has completed its task. When information is read back from the raster, it prefixes each set of data with a 2 byte (16 bit) byte count and then sends two bytes of 0 after the completion of all information requests. Qfrmcom either sets the mode for interpretation of subsequent Qpackets received, or requests the return of raster or color table data. Qprompt is passed to output 2, Qreset is ignored.

This function is included only in the PS 340 System. Refer to the **PS 340 Raster Programmer Guide** for information on the Raster modes.

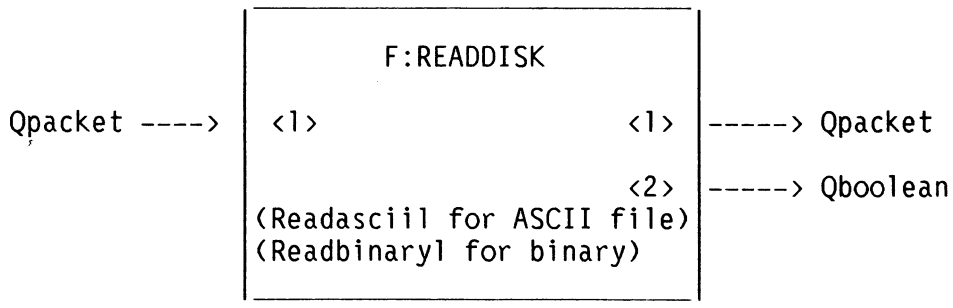
F:RASTERSTREAM



The function takes streams of bytes (usually from the host) and either makes a Qpacket or Qfrmcom (special data type used by F:RASTER) from them. It works in a count mode where it takes 2 bytes (16 bits) to specify the count. If the count = 0 then it means that a frame command is to be generated and the next two bytes are the value. If the count > 0, then it means to generate a Qpacket with that many bytes. A Qprompt is transmitted through to output <2>. A Qreset causes any data that may be stored to be disposed and to expect a new count to be received next.

This function is included only in the PS 340 system.

F:READDISK

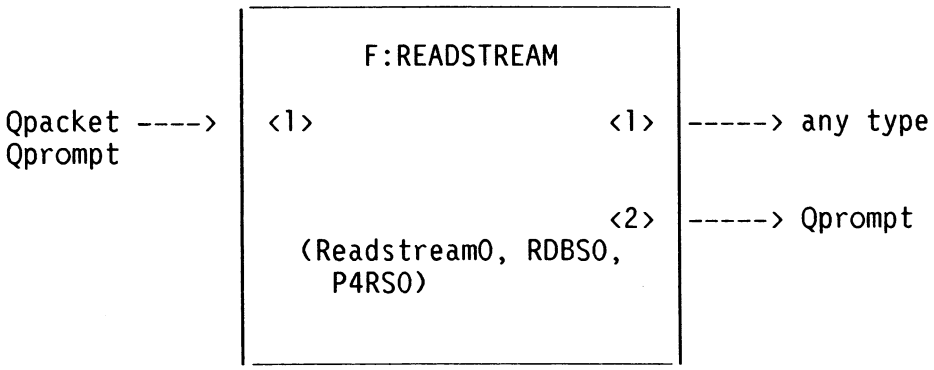


This function reads a file from the floppy disk and sends the data out output <1> in Qpackets. Input <1> accepts a Qpacket of 1 to 8 characters specifying the name of the file to be read. All disk drives are searched for the file until found; if the file is not found, an error message is produced.

A True is output from <2> when the file is found and read successfully. A False is output when the file is not found.

Note: The file name sent on input <1> should not include the file extension. The file on the disk must have the extension ".DAT".

F:READSTREAM

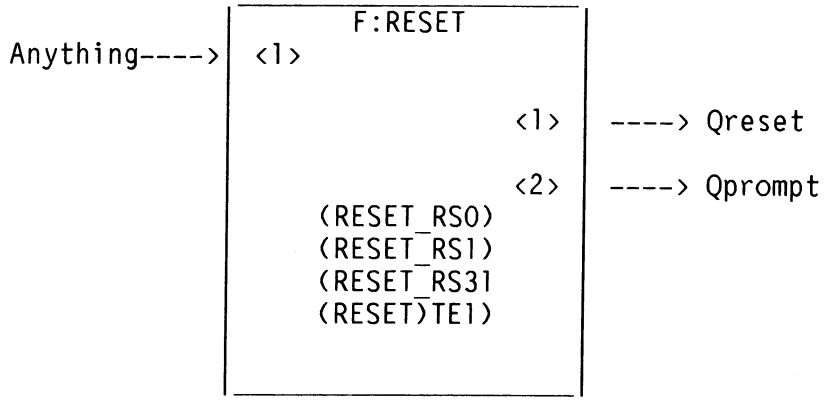


This function converts an 8-bit stream into arbitrary messages. It takes two bytes as the count of information (including message type) and creates a message of that size with the bytes of information that follow it.

The message format on input is:

2 bytes	2 bytes	
length	message type	rest of message body

F:RESET

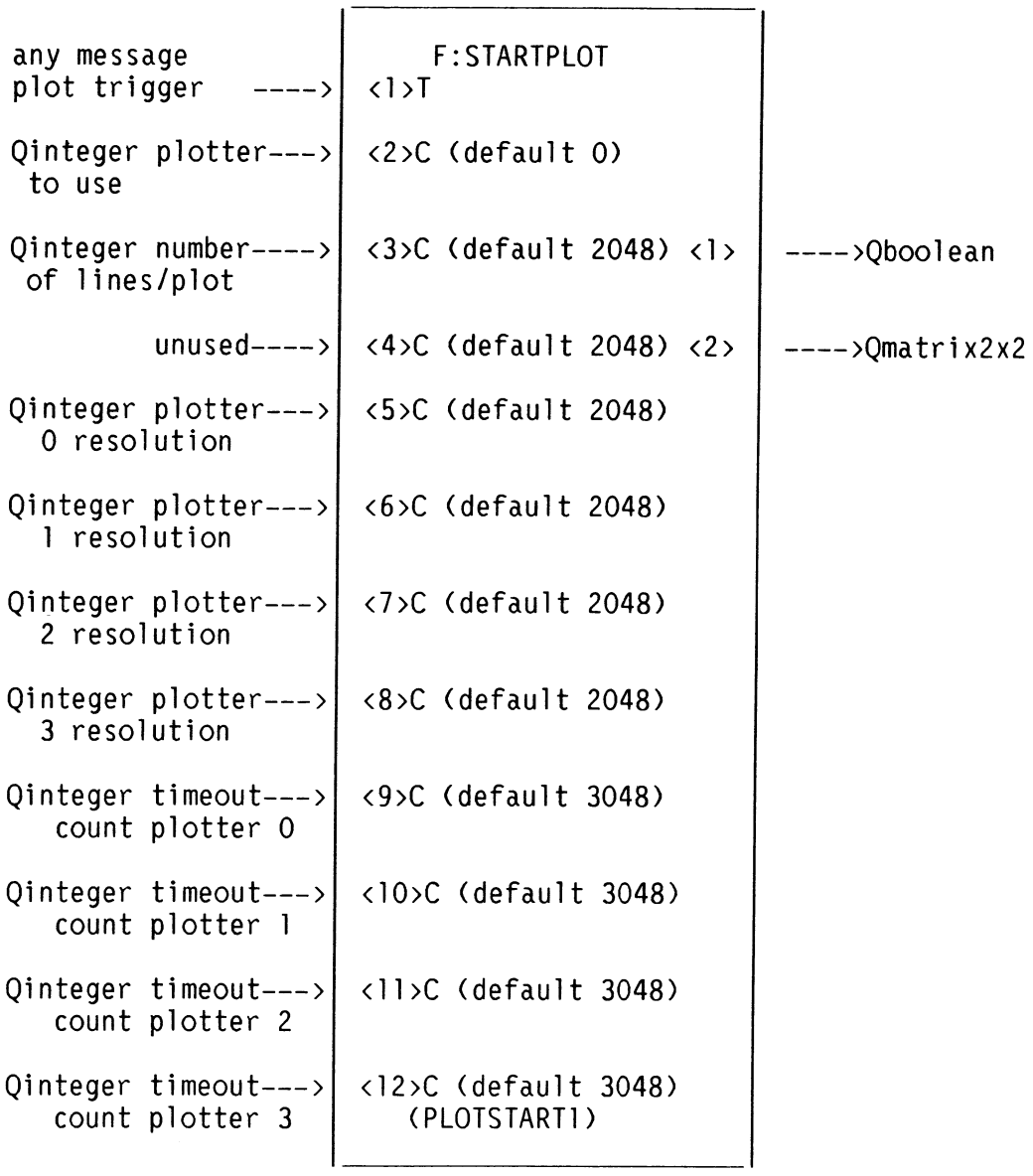


This function sends a Qreset out output <1> whenever it receives input other than a Qprompt or on input <1>. Qprompts are passed on through output <2>.

A Qreset purges a function and returns it to an initial state. Functions that respond to RESET:

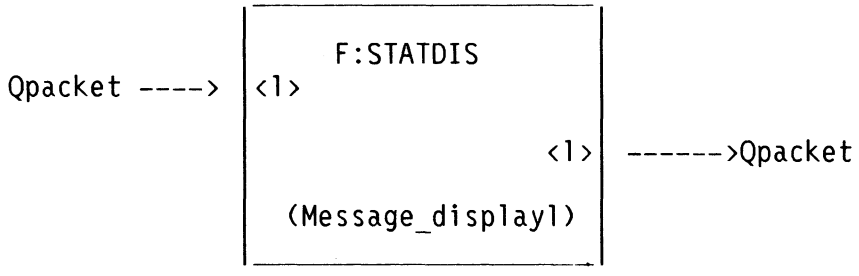
CIROUTE	PACKET
WRITESTREAM	DEPACKET
RASTER	DEMUX
RASTERSTREAM	WHO_AM_I
CI	CVT6TO8
PARSER	
gather_genfcn	

F:STARTPLOT



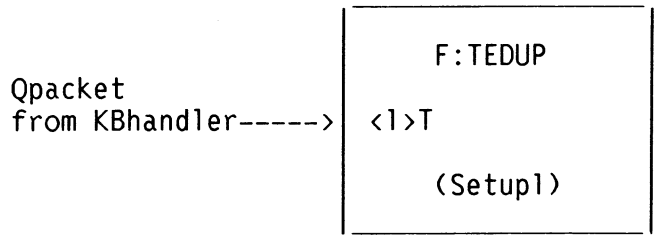
This function initiates the plotting process. It accepts information from the user concerning which plotter to use, and when to start the plot. Inputs specify which plotter ports are valid, the viewport size for each of the plotters, and the default plotter to use. Outputs are a Plot_Done signal which indicates successful completion of the plot and viewport matrix when plotter viewport must be changed (as with the Hewlett Packard plotter). This output is connected to the initial display named HVP1$ at boot time.

F:STATDIS



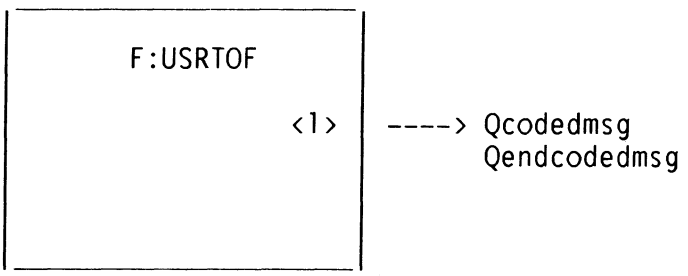
This functions causes messages to be displayed on the memory_display line of the PS 300 display. Output <1> is connected at system initialization for the function labels. Whenever this function receives a bell character, CHAR(7), it sends the character out output <1>.

F:TEDUP



The function allows users to change the VT100 terminal emulator configurable characteristics at runtime by depressing the SETUP key on the PS 300 keyboard.

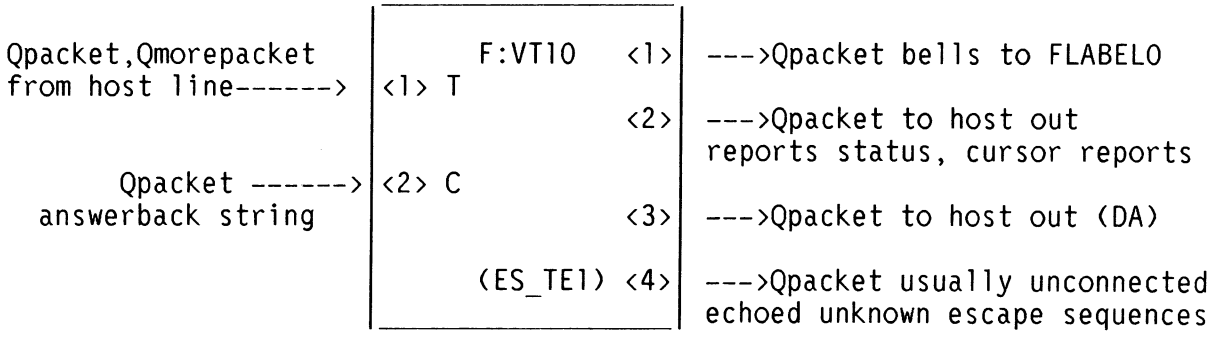
F:USRTOF



This is one of the two functions that handle ACP timeouts. When the ACP has been processing a frame for more than the specified time limit (one second, except when the PS 340 is performing a viewing operation) this function is executed. It removes all user objects being displayed and sends out a message indicating the number of timeouts since boot time.

Timeouts usually occur because a recursive structure has been displayed.

F:VT10

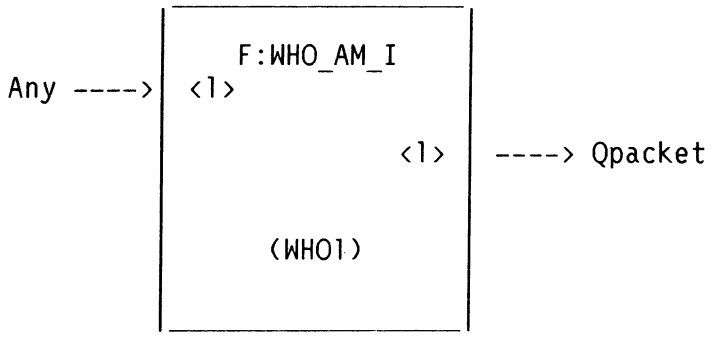


This is the VT100 terminal emulator function. It receives input from the host, the line editor, and other sources on input <1>. The primary task of this function is to route the input to the PS 300 display in an appropriate manner.

Input <2> defines the answerback string that is sent to the host when this function receives an ENQ.

Output <1> is used to make the expected "beep" on receipt of a ↑G (the beeper is in the keyboard). Output <2> sends data back to the host when the function receives command sequences, such as cursor position and terminal ID (I'm a VT-100). Output <3> is used to send the correct control sequence back to the host that identifies the terminal. Output <4> is an aid for debugging and development. It sends out all command sequences that are received, but unknown. Normally output <4> is not hooked up. It can be used to discover what kind of sequences a host program might be sending (that the terminal emulator cannot interpret) by hooking the output to a function like message_display.

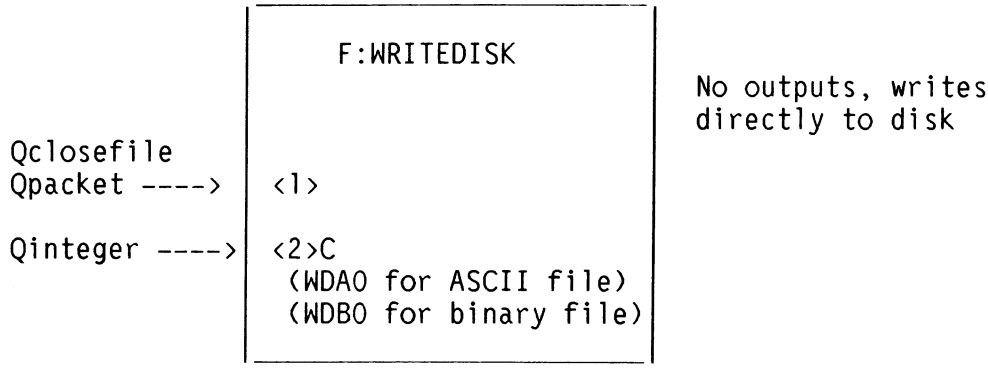
F:WHO_AM_I



F_WHO_AM_I, on any trigger, sends a string with system ID and other messages.

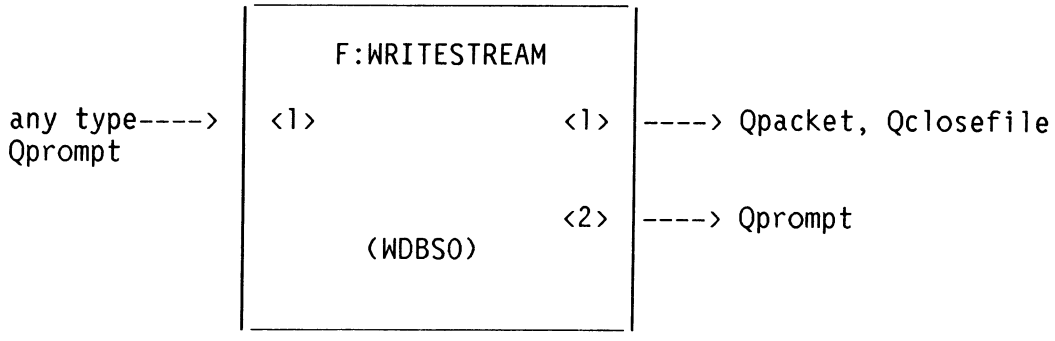
The information output by Who_Am_I includes the firmware version number, the definition of Qbindata, the definition of Qreset, and input buffer size. This information is used by the GSRs. Input buffer size is used in DMR-11 communications and is triggered by the PSIO routine, PSETUP.

F:WRITEDISK



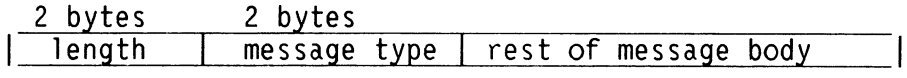
This function writes its input messages (Qpackets) to a file on the mini-floppy diskette. The message contents are collected in buffers which are stored on its private queue. When a message of type Qclosefile is received, the data in the buffers is written to disk on the drive specified on input <2> and with the file name (with ".DAT" file extension) given in the message Qclosefile. A Qclosefile message can only be obtained by sending a CLOSE "filename" command to a F:CHOP function whose output is connected to the F:WRITEDISK function.

F:WRITESTREAM



This function takes any message and turns it into a stream of bytes in a Qpacket (except for Qprompts and Qclosefiles). Qprompts are passed on through to output <2> and Qclosefiles are passed onto output <1>. This is used to create binary data files on a floppy diskette. Normally, this function receives input from a chop/parse function and sends output to a writedisk function.

The resulting Qpackets contain:



6.5 INTEGER REPRESENTATION OF FUNCTIONS FOR F:FNREPORT

The integer representation of functions reported by F:FNREPORT is shown in the following list. This integer (also referred to as the ORD) is used to identify functions and is the first item reported by F:FNREPORT. The first 28 functions are not included in the description of system functions in this chapter, but they will be reported by F:FNREPORT.

Integer code	Function
0	DISPBLK
1	UPDKILL
2	UPDFMT
3	FCNDISP
4	TICFCN
5	CSCFCN
6	FRMFCN
7	BYTEIN1
8	BYTEIN2
9	BYTEIN3
10	BYTEIN4
11	BYTEIN5
12	BYTEIN6
13	BYTEOUT1
14	BYTEOUT2
15	BYTEOUT3
16	BYTEOUT4
17	BYTEOUT5
18	BYTEOUT6
19	PICK
20	F_I1_IBM { IBM in }
21	F_I2_IBM { IBM in }
22	F_O1_IBM { IBM out }
23	F_O2_IBM { IBM out }
24	F_K1_IBM { IBM kbd }
25	F_K2_IBM { IBM kbd }
26	F_W_IBM { IBM wake }
27	RUN_GENFCN { run a user generic function }
28	FTZERO { corresponds to word # 0 in fcn name dict }
29	ADD
30	ADDC
31	SUB

Integer code	Function
32	CSUB
33	SUBC
34	MUL
35	CMUL
36	MULC
37	DIV
38	CDIV
39	DIVC
40	MOD
41	MODC
42	NOT
43	AND
44	ANDC
45	OR
46	ORC
47	XOR
48	XORC
49	LT
50	CLT
51	LTC
52	LE
53	CLE
54	LEC
55	GT
56	CGT
57	GTC
58	GE
59	CGE
60	GEC
61	EQ
62	EQC
63	NE
64	NEC
65	ROUND
66	FIX
67	FLOAT
68	SINCOS
69	SQROOT
70	LIMIT
71	DELTA
72	CEILING
73	MINMAX
74	AVERAGE
75	RANGE_SELECT
76	VEC
77	CVEC

Integer code	Function
78	VECC
79	POSITION_LINE
80	COLOR
81	CROTATE
82	XROTATE
83	YROTATE
84	ZROTATE
85	CSCALE
86	SCALE
87	XVECTOR
88	YVECTOR
89	ZVECTOR
90	DXROTATE
91	DYROTATE
92	DZROTATE
93	DSCALE
94	ATSCALE
95	ACCUMULATE
96	LOOKAT
97	LOOKFROM
98	WINDOW
99	FOV
100	MATRIX2
101	MATRIX3
102	MATRIX4
103	PARTS
104	PRINT
105	NPRT PRT
106	LINEEDITOR
107	CHARMASK
108	CHARCONVERT
109	CONCATENATE
110	CCONCATENATE
111	CONCATENATEC
112	SPLIT
113	PUT_STRING
114	PUTC_STRING
115	TAKE_STRING
116	TAKEC_STRING
117	LENGTH_STRING
118	FIND_STRING
119	CFIND_STRING
120	FINDC_STRING
121	COMP_STRING
122	COMPC_STRING
123	TRANS_STRING

Integer code	Function
124	STRING_TO_NUM
125	CLCSECONDS
126	CLFRAMES
127	CLTICKS
128	NOP
129	SYNC
130	BOOLEAN_CHOOSE
131	Inputs_choose
132	ROUTE
133	CROUTE
134	ROUTEC
135	BROUTE
136	CBROUTE
137	BROUTEC
138	CONSTANT
139	FETCH
140	EDGE_DETECT
141	PICKINFO
142	TIMEOUT
143	SCREENSAVE
144	SURFACE_OPERATION
145	XFORMDATA
146	LIST
147	USERUPDATES
148	INTEGER_CHOOSE
149	SWITCH
150	CSWITCH
151	SWITCHC
152	SYNC2
153	SYNC3
154	SYNC4
155	SYNC5
156	SELECT
157	Cselect
158	TEK4014
159	TKB4014
160	GATHER GNEFCN
161	STRIPPROMPT
162	WRITEDISK
163	READDISK
164	HOLDMESSAGE
165	LIGHTPEN
166	TF01
167	TF02
168	TF03
169	TF04

Integer code	Function
170	TF05
171	TF06
172	TF07
173	TF08
174	TF09
175	CHOP
176	CI
177	SEND
178	PCREPT
179	INFORMATION
180	DIAGCLR
181	DIAGSET
182	FKEYS
183	FLABEl0
184	KEYBOARD
185	DIALSIN
186	BITPADIN
187	BITPADOUT
188	DIALSET
189	DIALLABEL
190	FKEYLABEL
191	FNREPORT
192	DCDECODE
193	SET_PORT
194	IDCCONTROL
195	DCCONTROL
196	DCSETSUBPORT
197	DCOUTSUBPORT
198	STATDIS
199	MEMMON
200	MEMFULL
201	VT10
202	TEDUP
203	K2ANSI
204	ACPTOF
205	USRTOF
206	BIN56K
207	BOUT56K
208	TIME56K
209	BUTTONS32
210	ONLIGHTS32
211	OFFLIGHTS32
212	IBMTE
213	HOGMEMORY

Integer code	Function
214	INITPLOT
215	STARTPLOT
216	FFPLOTTER
217	MAKESTRING
218	MAKEPACKET
219	DEPACKET
220	PACKET
221	DEMUX
222	MUX
223	CMUX
224	CVT8TO6
225	CVT6TO8
226	CVTASCTOIBM
227	CVTIBMTOASC
228	WHO_AM_I
229	QUICK
230	CIROUTE
231	READSTREAM
232	WRITESTREAM
233	IBM_KEYBOARD
234	INITIBM
235	SETUPIBM
236	FCNTAPUP
237	RESET
238	FCNSTRIP
239	FCNTWDEE
240	FCNTWDUM
241	RASTER
242	RASTERSTREAM
243	WRITEBACK
244	PASSTHRU
245	MMMREG
246	MMWATCH
247	BOTSTACK
248	GATHER_STRING
249	MCAT_STRING
250	VEC_EXTRACT
251	LBL_EXTRACT
252	LABEL

CONTENTS

FIGURES

7. INITIAL DATA STRUCTURES, NAME SUFFIXING, AND SYSTEM COMMANDS

The chapter covers several loosely linked topics: initial data structures, name suffixing, and system commands. Within these main topics are other important subtopics including the system Configure mode and its uses. The first section of the chapter describes a "runtime" system and the initial data structures that are built by the PS 300 firmware. Following sections discuss the Configure mode, name suffixing procedures, and system commands.

7.1 RUNTIME SYSTEM

The PS 300 is a runtime system; there is no file system, no editor, no compiler, no symbolic debugger. In short, the PS 300 is not a personal computer and does not provide a "normal" programming environment. The runtime system is basically composed of PS 300 functions linked together to form the system function network. A PS 300 function might be viewed as self-contained program. With minor exceptions, it has no access to disk files, and deals with the world via messages and queues, one transaction at a time.

7.1.1 The Graphics Control Program

The Graphics Control Program is the name given to the collection of software that executes whenever the PS 300 is being used in the normal mode of operation as an interactive computer graphics terminal. The Control Program is loaded by the 68000 startup code, and mainly resides in local GCP memory.

The Control Program is made up of:

1. Data structuring definitions
2. The scheduler
3. The functions

7.1.2 Data Structure Definitions

The data structures are set up by Pascal procedures that define and make use of the following:

- named entities; data structure that can be named and referenced
- alpha block; data structure that contains the location of a named entity
- ACP state; contains the parameters the define the context of the Display Processor at any given time

7.1.3 The Scheduler

The PS 300 runtime system contains a scheduler that is activated once the initialization code on the firmware has loaded. The scheduler loops to schedule and execute functions. When a function is "instanced", it is assigned a default priority for execution. This priority number is used by the scheduler to determine which active function will be scheduled next to be executed.

7.1.4 The Functions

The PS 300 system functions are described in Chapter 6 of this guide. PS 300 standard functions are described in the **Function Summary** in Volume 3A of this documentation set.

7.1.5 Initial Data Structures

The initial data structures are built by the CONFIG.DAT file. These structures set up the framework that allows for displayable data structures to be built by the user. They form the top nodes of a hierarchical tree that the GCP and ACP traverse during each cycle.

The following diagram illustrates the initial structures that set up the framework allowing for displayable data structures to be built by the user.

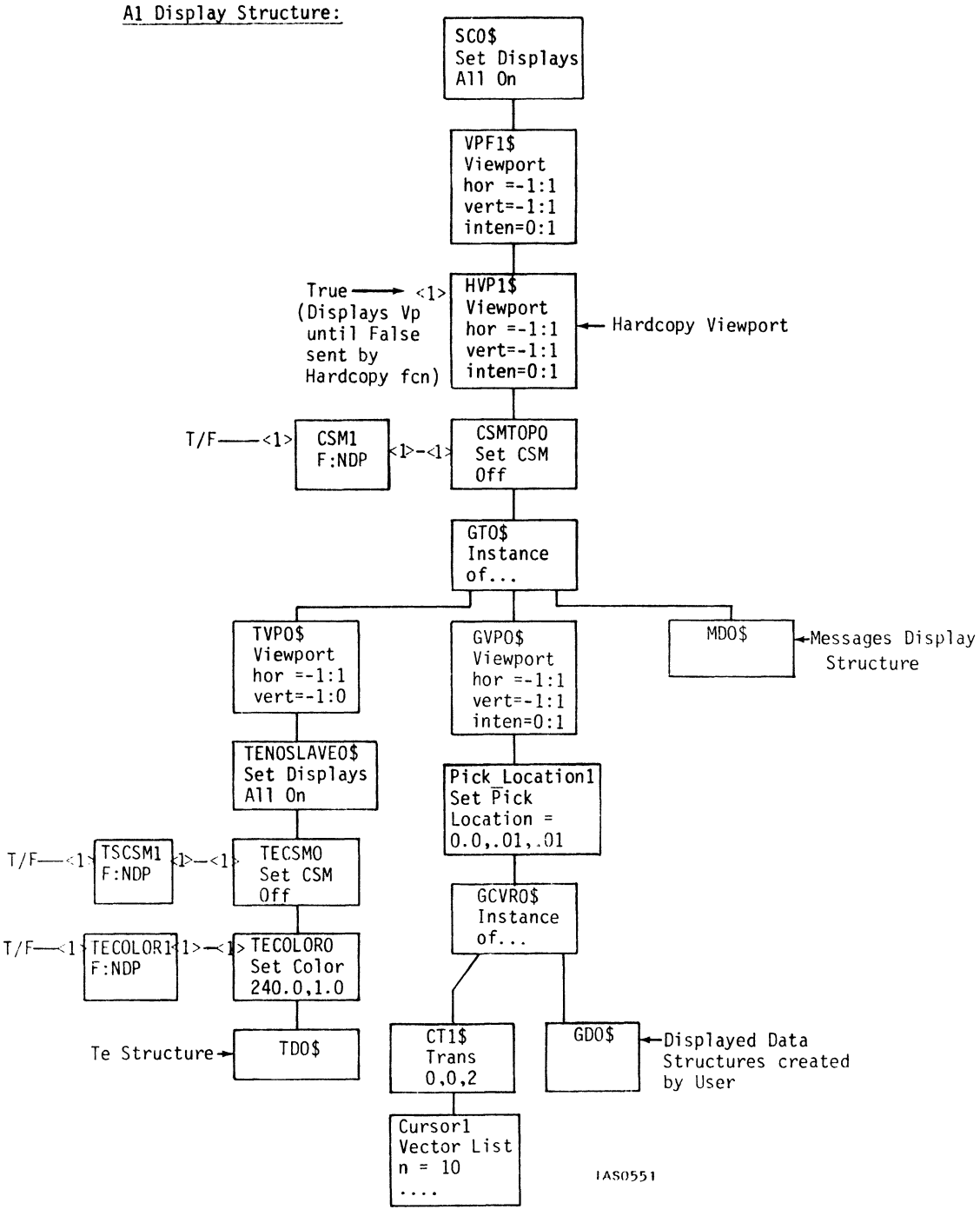


Figure 7-1. A1 Display Structure

7.1.6 Code for Initial Data Structures

The code that supports the initial data structure follows.

```
{ Initial Display Data Structure }
SC0$ := SET DISPLAYS ALL ON
   THEN VPF1$
VPF1$ := VIEW HORIZONTAL = -1:1 VERTICAL = -1:1 INTENSITY = 0:1
   THEN HVP1$;
HPF1$ := VIEW HORIZONTAL = -1:1 VERTICAL = -1:1 INTENSITY = 0:1
   THEN CSMTOP0
CSMTOP0 := SET CSM OFF
   THEN GT0$
 SEND TRUE TO <-1>HVP1$
GT0$ := INSTANCE OF GVP0$, TVP0$, MD0$

{ Graphics Display Structure }
GVP0$ := VIEW HORIZONTAL = -1.1 VERTICAL = -1:1 INTENSITY = 0:1
   THEN PICK_LOCATION1;
PICK_LOCATION1: := SET PICK LOCATION = 0,0 .01, .01
   THEN GCUR0$;
GCUR0$ := INSTANCE CT1$, GD0$; { All Display Commands append to GD0$ }
CT1$ := TRANSLATE BY 0,0,2
   THEN CURSOR1;
CURSOR1 := VECTOR_LIST ITEMIZED N=10
   p .035,.035 l -.035,-.035 p -.035,.035 l .035,-.035
   p .035,.035 l -.035,-.035 p -.035,.035 l .035,-.035;

{ Terminal Emulator Display Structure }
TVP0$ := VIEW HORIZONTAL = -1:1 VERTICAL = -1:0
   THEN TENOSLAVE0$;
TENOSLAVE0$ := SET DISP ALL ON
   THEN TECSM0;
TECSM0 := SET CSM OFF
   THEN TECOLOR0;
TECOLOR0 := SET COLOR 240.0, 1.0
   THEN TD0$; { The Terminal Screen is appended to TD0$ }
```

```
{ Crash Message Display Structure }
CRASH_MSGS$:=BEGIN_STRUCTURE
    IF LEVEL =  17 THEN C17$;
    IF LEVEL =  15 THEN C16$;
    IF LEVEL =  16 THEN C16$;
    IF LEVEL >  17 THEN C16$;
    IF LEVEL <  0 THEN C16$;
    IF LEVEL =  0 THEN C0$;
    IF LEVEL =  1 THEN C1$;
    IF LEVEL =  2 THEN C2$;
    IF LEVEL =  3 THEN C3$;
    IF LEVEL =  4 THEN C4$;
    IF LEVEL =  5 THEN C5$;
    IF LEVEL =  6 THEN C6$;
    IF LEVEL =  8 THEN C8$;
    IF LEVEL =  9 THEN C9$;
    IF LEVEL =  10 THEN CA$;
    IF LEVEL =  11 THEN CB$;
    IF LEVEL = 12 THEN CC$;
    IF LEVEL = 13 THEN CD$;
    IF LEVEL = 14 THEN CE$;
END_STRUCTURE;
C16$:=CHAR 'Unknown crash';
C0$:=CHAR 'Mass memory Exhausted';
C1$:=CHAR 'OKINT/NOINT imbalance';
C2$:=CHAR 'Free block size invalid';
C3$:=CHAR 'Attempt to activate non-functin or nil';
C4$:=CHAR 'NEW call in Nomemsched failed to find memory';
C5$:=CHAR 'Attempt to queue where fcn already waiting';
C6$:=CHAR 'Systemerror';
C7$:=CHAR 'TRAP7';
C8$:=CHAR 'Mass Memory Error';
C9$:=CHAR 'TRAP9';
CA$:=CHAR 'Multiple DISPOSE of same block';
CB$:=CHAR 'Block exponent not big enough';
CC$:=CHAR 'TRAP C';
CD$:=CHAR 'PASCAL Error';
CE$:=CHAR 'PASCAL Error';
C17$ := CHAR 'Unexpected exception';
```

```
{ Setup Mode Display Structure }
SVP0$ := VIEW HORIZONTAL=-1:1 VERTICAL = -1:1
    THEN SZ0$;
STCSM0 := SET CSM OFF
    THEN SS0$;
SS0$ := CHAR SCALE 0.03
    THEN SS0$;
SS0$ := INSTANCE OF
    S10$, S20$, S30$, S40$, S50$, S60$, S70$, S80$, S90$;
S10$ := CHAR -1,.9 'SETUP';
S20$ := CHAR -1,.8 ' ';
S30$ := CHAR -1,.7 'F2-SRM :T F3=Awrp:F F4=ANSI:T F5=VT52:F ';
S40$ := CHAR -1,.6 'F6=KPM :F F7=CKM :F F8=Cnum:T F9=Knum:T ';
S50$ := CHAR -1,.5 ' ';
S60$ := CHAR -1,.4 'F10= Define breakkey :^V      ';
S70$ := CHAR -1,.3 'F11= Move TE viewport, lower left corner ';
S80$ := CHAR -1,.2 'F12= Move TE viewport, upper right corner ';
S90$ := CHAR -1,.1 'Mode: TE  Term: On  Graf: On    ';
SA0$ := CHAR -1,.0 'Press special key to be breakkey, F1 to exit. ';
SB0$ := CHAR -1,.0 'Move corner with cursor keys, F1 to exit. ':
```

7.2 CONFIGURE MODE

Before any of the data structures can be built by the CONFIG.DAT file, the file must first be read by the system. One of the first things done by the firmware is the creation of a simple function network that consists primarily of an instance of the readdisk function and the command interpreter.

This network reads the CONFIG.DAT file from the diskette. This file contains the initial instance of system level commands and functions. While reading this file, the command interpreter is in a "privileged" mode of operation called the CONFIGURE mode. The command interpreter must be in the CONFIGURE mode to create or modify any system functions.

The command interpreter that processes the CONFIG.DAT file is separate and distinct from the command interpreter that handles user commands These "user" command interpreters are initially in a non-privileged mode called the Command' mode.

7.3 NAME SUFFIXING

Whenever a user names anything or instances a function, the command interpreter (CI) assigns a specific suffix to that name. The suffix is determined by the suffix that has been assigned to that instance of the command interpreter. In a dual user system, this allows both users to share memory while remaining independent of each other. Name suffixing is also used to separate system level names and instances from user-originated names and instances.

Whenever working in the CONFIGURE mode, with system-level or user-level names, the programmer is responsible for correctly suffixing any function that is instanced. This includes appending the proper suffix to the command interpreter to assure that other functions created by this CI will have the appropriate suffix. This is particularly important in a dual-user system. When a programmer creates and names an instance of a command interpreter that will be used by "user1", it must be given a instance name with the correct suffix. Then, when that command interpreter is executed, it will append the correct suffix to the names it sees.

The suffix assignments for the PS 300 are as follows (the suffixing is tailored to allow for a dual-user system):

0 - suffix for system related functions associated with user 1. Names with this suffix are not directly accessible to user 1 and user 2.

1 - suffix for user-defined and accessible names associated with user 1. All names with this suffix are accessible to user 1.

(If the CI used is suffixed with a 0 or a 1, it will suffix names that it creates with 1.)

2 - suffix for system related functions associated with user 2. Names with this suffix are not directly accessible to user 2 or user 1.

3 - suffix for user-defined and accessible names associated with user 2. All names with this suffix are accessible to user 2.

(If the CI used is suffixed with a 2 or 3, the names it creates will be suffixed with a 3.)

System names are usually identified and distinguished from all other by having the suffix "$".

Users are never required to suffix any entity they name or specify the suffix when accessing a name, except in CONFIGURE mode. In Command mode, this is all done by the command interpreter.

Note: When the F:CI(n) function is instanced, the function creates PICK[suffix]. Therefore, the CI should be created before downloading the remaining program (or given a suffix that will create the PICK[suffix] that is being used in the program). Otherwise, all connections from PICK[suffix] that were made before instancing will be lost.

7.4 USING THE CONFIGURE MODE

To access system-level functions, the system manager must be able to access any name, regardless of the suffix. This is done by entering the CONFIGURE mode, the privilege mode of operation. In this privileged mode, the system manager has the capability of reconfiguring system functions.

To enter the CONFIGURE mode, the following command is entered while in the normal mode of operation (command):

```
CONFIGURE password;
```

where password is a string that has been defined by using the setup password command (explained in the System Commands section). If no password has been defined (the default case), any string can be entered.

Since no name suffixing occurs while the command interpreter is in the Configure mode, the system manager must explicitly include the suffix on any names to affect a specific user. For example, if the SITE.DAT file (which is read from the diskette when the CI is in Configure Mode) contains commands to send a site message to each user's FLABEL0, the appropriate suffix must be included at the end of the name. Using the name suffixing convention, the commands in the SITE.DAT file would appear as follows.

```
SEND 'E&S System 11, Site Manager - Scot Jones' to <1> FLABEL01;
SEND 'E&S System 11, Site Manager - Scot Jones' to <1> FLABEL03;
```

7.5 SYSTEM COMMANDS

The following commands are available to the system manager.

7.5.1 Configure Mode Commands

The following commands allow the system manager to enter and exit the Configure mode.

1. SETUP PASSWORD password;

where "password" is the established string.

This command allows the system manager to establish and modify the password required to enter the CONFIGURE mode. This command can be included in the SITE.DAT file, or may be set up by the system manager at any time. This command can only be entered while in Configure mode.

2. CONFIGURE password;

where "password" is the established string.

This command allows the system manager to enter the CONFIGURE mode (privileged mode). The password can be defined in the SITE.DAT file using the SETUP PASSWORD command. If no password has been set up using the SETUP PASSWORD, any string may be used as a password.

3. FINISH CONFIGURATION;

The FINISH CONFIGURATION command takes the PS 300 out of CONFIGURE mode and must be used at the end of any session that has modified any part of the CONFIG.DAT file or accessed any system level functions.

7.5.2 System Configuration Commands

System configuration commands are used to show or change the default values on Ports 1 through 5 on the PS 300 Control Unit.

The command:

SHOW INTERFACE <name>;

where <name> is the port, can be used to check the values of a given port.

The menu available with the SHOW INTERFACE command (when no parameter is given) lists only those parameters that are relevant to the interface; for example, in synchronous mode, the X_ON/X_OFF parameter would not be listed.

NOTE

If using the DMR-11AE interface, the SHOW and SETUP commands cannot be used for port 1.

The following command sequence can be used to change any of the default values. (These new values must be within the acceptable values for data characteristics. The default values and the acceptable range of values are described in Chapter 4 of this manual, Host/PS 300 Communications.)

SETUP INTERFACE portn/option=<n>;

where portn is the port being reconfigured, option is the name of the value that will be changed, and <n> is a legal parameter that is specific to the option.

For example, the command:

SETUP INTERFACE port10/SPEED=300;

where port10 is the port being reconfigured, SPEED refers to the baud rate, and 300 is a legal speed for the communications interface, changes the effective baud rate of port10 to 300.

In using these commands, the ports names are as follows:

Port 1 is designated port10
Port 2 is designated port20
Port 3 is designated port30
Port 4 is designated port40
Port 5 is designated port50

The available options for SETUP INTERFACE are given in Chapter 4 of this guide, Host/PS 300 Communications.

The PS 300 does not have to be operating in CONFIGURE mode for either of the commands described above.

7.5.3 Command Status Command

The command:

COMMAND STATUS;

directs the command interpreter to print the status of the command stream. The message output lists the number of open BEGIN...END and BEGIN_STRUCTURE...END_STRUCTURE commands, and indicates if the privileged state is operative. The message also indicates if the optimize structure model is in effect.

7.5.4 Reboot Command

The command

REBOOT password;

reboots the PS 300 as if from power-up. If no password has been setup, then any character string will do. Otherwise entering an incorrect password will give an error message. The REBOOT command can appear anywhere; it can occur within BEGIN...END and BEGIN_STRUCTURE...END_STRUCTURE as well as without. It may be named or not. However, it cannot be within a quote or comment.

The command causes the PS 300 to reboot just as if it had been powered up (starts the confidence tests at "A", etc.).

7.5.5 Notes On Using the CONFIGURE Mode

E&S reserves the right to change the content of the CONFIG.DAT file and the implementation of the CONFIG.DAT file without prior notice. Use of any named entities or networks instanced in CONFIGURE mode that have names identical to any names found in the CONFIG.DAT file will result in unpredictable system behavior. E&S will not use any names that are preceded with the three characters CM_.

CONTENTS

TABLES

8. TERMINAL EMULATOR MODES AND FUNCTIONS

This chapter discusses the PS 300 terminal emulator from several perspectives. The terminal emulator will first be discussed in terms of the ANSI modes and control sequences that are used to implement the VT-100 terminal emulations capabilities of the PS 300. Many of Digital Equipment Corporation private sequences and modes for the VT-100 are referred to in the discussion. More information on these sequences and modes is found in DEC's VT-100 User Guide (EK-VT100-UG-002).

The second discussion will cover the system functions that form the terminal emulator network and how data is received and passed between them.

The third discussion will go over the three communication modes of operation of the keyboard and how certain keys are translated within these modes. Operator information for the three communication modes used by the PS 300 keyboard is covered in Volume 1, *User Operation and Communication.*

The terminal emulator facility has characteristics and features that can be changed fairly easily by system programmers. Information for changing and adapting these features will be covered throughout the chapter.

8.1 ANSI MODES OF OPERATION

The PS 300 operates under ANSI (and certain VT52) modes wherein it recognizes and responds to certain coded sequences whose syntax and semantics are in accordance with ANSI specifications. These modes determine how other coded sequences are to be interpreted and how the terminal will respond in certain situations.

Escape sequences are interpreted as control functions that set the mode of operation, (i.e. sending a particular escape sequence from the host to the terminal will determine whether the right-hand keypad on the PS 300 keyboard generates the numeric value of the keycaps or the escape sequences that are used for EDT editing commands). The interpretation of the escape sequence is dependent on the mode the terminal is operating in. The modes can be set or reset by sending escape sequences from the host to the terminal.

It is difficult to categorize the modes in a straightforward manner because some of them are dependent on the settings of other modes: if the ANSI (VT-100) mode is set to FALSE (or OFF), then logically, the terminal will not be able to respond to any other ANSI control sequences. Some of these modes are standard to the DEC VT-100, and some are peculiar to the PS 300. The modes and the escape sequences that can be used to set them will all be discussed in later sections. The list below gives some idea of the modes and what they do.

- Send-Receive Mode (SRM (Local echo/Nolocal echo)) - determines whether keyboard input will be echoed to the display.
- ANSI Mode - (DECANM) determines whether the PS 300 will generate and respond to standard ANSI (VT-100) escape sequences.
- VT52 Mode - allows the PS 300 to recognize VT52 coded sequences.
- Keypad numeric mode (DECKPNM) - causes the numeric keycap values to be sent from the right-hand keypad to the host.
- Keypad application mode (DECKPAM) - causes the keys on the right-hand keypad to transmit an escape sequence which begins with <ESC>O to the host.
- Cursor key Mode (DECCKM) - causes the cursor keys to transmit the ANSI control sequences that cause the cursor movements indicated on the cursor keycaps.

The modes listed above, as well as other modes that are specific to the PS 300, can be changed using the terminal emulator SETUP facility. A definition of these modes, their defaults, and how to change them is discussed in Volume 1, ***User Operation and Communication.***

8.1.1 Definition of Escape Sequences

An escape sequence is a sequence of characters that is used for control purposes to perform a control function and whose first character is the escape <ESC> (the ASCII Hex "1B") control character. They are used to set and reset modes, as well as tell the terminal how to respond to coded sequences. These characters are not displayed as text on the screen, but instead cause the terminal to perform some action or change some internal parameter of operation. Control sequences are also used to change or define characteristics of the terminal. A control sequence is an escape sequence that provides supplementary controls and begins with the control sequence introducer (CSI). In the VT-100 emulation, the CSI is <ESC>[.

The sequences that the terminal emulator deals with take two general forms: those that may have parameters, and those that do not. Those not having parameters take the form <ESC>c, where c is a single character. Those that may have parameters take the form:

<ESC>[P1;P2;...Pnc

where:

<ESC>[is the control sequence introducer

P1....Pn are the parameters (none need be present)

';' is used to separate parameters

'c' is the final character that determines which control sequence is being defined.

The parameters are numbers expressed in their ASCII form. In sequences that use private or non-standard parameters, the first character of the parameter string is "?" for DEC private sequence and ">" for E&S private sequences.

8.1.2 SET and RESET - SM, RM

The SET and RESET control sequences are used to set and reset certain modes of the terminal. These control sequences for setting or resetting these modes are sent from the host. The modes that can be set or reset are listed below, along with the set and reset escape sequences.

- Send-Receive Mode (SRM)
- ANSI-VT52 (DECANM)
- Cursor Key Mode (DECCKM)
- E&S private sequences

The SET and RESET control sequences are:

SM: <ESC>[Pnh

RM: <ESC>[Pn*l**

where 'n' is the parameter that determines which mode is to be set, i.e.

<ESC>[?1h

would set the cursor key mode (DECCKM).

8.1.3 Send-Receive Mode (SRM) - Local Echo/ Nolocal Echo

The SRM mode can be set or reset from the host by sending the proper control sequence, by using the SETUP facility of the terminal emulator package, or by including the appropriate ASCII characters in the SITE.DAT file. (Refer to Volume 1, ***User Operation and Communication*** for SETUP, or to Chapter 2 of this guide for the SITE.DAT file.) This mode determines whether the screen receives the input from the keyboard on the host line, or from a PS 300 system function. If the host line is half duplex, the host does not echo the keys as they are sent from the Terminal Emulator to the host. This mode must be reset so that the characters that are received by the TE from the keyboard will be displayed on the screen.

If the line to the host is full-duplex, the host retransmits the keys it receives from the keyboard back to the terminal, and they are then displayed on the screen. In this case, SRM should be set so that the characters will not appear on the screen twice: once as they are keyed in, and once as they are received back from the host.

8.1.4 Send-Receive Mode (SRM) Escape Sequences

"12" is the parameter that designates SRM.

<ESC>[12h	SET SRM. Do not send keyboard input to the display.
<ESC>[12*l**	RESET SRM. Send keyboard input to the display, with [CR] (Carriage Return) displaying as [CRLF] (Carriage Return-Line Feed).

* The last character of this escape sequence is a lowercase L.

8.1.5 ANSI - VT52 Mode Escape Sequences

The ANSI - VT52 modes can be set or reset with the (SM/RSM) control sequences. The VT52 set state causes VT52 compatible escape sequences to be interpreted and executed. The ANSI set state causes only ANSI (VT-100) compatible escape sequences to be interpreted and executed. The ANSI-VT52 modes are private, using a private string parameter. The first character in the string must be "?" and "2" designates ANSI-VT52 mode. The recognition of VT52 sequences may may also be turned off by the <ESC>< sequence when in the VT52 mode.

<ESC>[?2h or — SET ANSI mode. Escape sequences will be interpreted as ANSI; keys will be translated accordingly.

<ESC>[?2l* — SET VT52 mode. Escape sequences will be interpreted as VT52; keys will be translated accordingly.

8.1.6 Left-Hand Keypad Cursor Keys - (DECCKM)

The four cursor keys on the left-hand keypad of the keyboard have a single mode that may be set or reset using the SM/RM control sequences. The Cursor Key mode is similar to DEC's DECCKM. When this mode is reset (the default at power-up), the cursor keys transmit the ANSI control sequences that cause cursor movement as indicated by the arrows on the key caps. When the cursor key mode is set, the keys are in an application mode, and like the right-hand keypad, transmit escape sequences.

When the VT52 mode is in effect, the sequences have no intermediate characters, and are the same regardless of the setting of the cursor key mode. The following table shows what is transmitted in the Reset and Set modes.

Table 8-1. CURSOR KEY Transmission

CURSOR KEY	VT52 (SET - MODE (RESET)	ANSI MODE RESET MODE	ANSI MODE SET MODE
Up	<ESC>A	<ESC>[A	<ESC>OA
Down	<ESC>B	<ESC>[B	<ESC>OB
Right	<ESC>C	<ESC>[C	<ESC>OC
Left	<ESC>D	<ESC>[D	<ESC>OD

* The last character of this escape sequence is a lowercase L.

8.1.7 Cursor Key Mode Escape Sequences

The Cursor Key mode is also a private mode and uses the private parameter string. The first character in the string must be "?". "1" is the parameter that designates Cursor Key mode.

<ESC>[?1h	SET Cursor Key Mode. Cursor keys will now cause <ESC>Oc sequences to be sent.
<ESC>[?1l*	RESET Cursor Key Mode. Cursor keys will now cause the <ESC>[c sequences to be sent.

* The last character of this escape sequence is a lowercase L.

8.1.8 E&S Private ANSI Commands for Function Keys, Right-Hand Keypad and Cursor Keys

The Function keys, the right-hand keypad, and the cursor keys can be placed under the control of the user-application program. The modes to do so may be set or reset using E&S private ANSI commands. When "FKEYs always" is set, the output of the Function Buttons is always sent to FKEYS<1>. When the mode controlling the routing of the output of the right-hand keypad or the cursor keys is set, the numeric value (as input to function networks) of these keys are available to PS 300 function networks in any of the three communication modes (Terminal Emulator, Command, or Interactive). When any of these modes are set, these keys (right-hand keypad or cursor) cause integers to appear at output<9> of KBHANDLER. When reset, the key values will be sent through KBhandler to the host, the command interpreter, SPECKEYS, etc., depending on the communication mode of the keyboard.

Multiple Pn' are allowed per command, i.e., <ESC>[>10;11;12h would cause all Function keys, cursor keys, and keypad keys to go to the user application.

ANSI SEQUENCES	DESCRIPTION
<ESC>[>2h	Set no/local echo. The TE will not locally echo keys. When reset, the TE locally echoes keys.
<ESC>[>3h on <ESC>[>1h	Set auto-wrap. The TE adds <crlf> if it receives more than 80 characters without getting <crlf>. When reset, the TE puts additional characters in column 80, overwriting the last one.

ANSI SEQUENCES	DESCRIPTION
<ESC>[>4h	Set ANSI. The TE recognizes ANSI control sequences. When reset, the TE responds like a teletype terminal. When the reset sequence is sent from the host to the PS 300, all further ANSI commands are ignored (including <ESC>[>4l).
<ESC>[>5h	Set VT52. The TE will recognize VT52 control sequences. When reset, the TE will not recognize VT52 control sequences.
<ESC>[>6h	Set KPM. The right-hand keypad sends control sequences. When reset, the right-hand keypad sends numbers.
<ESC>[>7h	Set CKM. The cursor keys send control sequences. When reset, the cursor keys send cursor control sequences.
<ESC>[>8h	Set Cnum. The right-hand keypad sends numbers in CI mode. When reset, the right-hand keypad sends ↑Vc in CI mode.
<ESC>[>9h	Set Knum. The right-hand keypad sends numbers in KB mode. When reset, the right-hand keypad sends ↑Vc in KB mode.
<ESC>[>10h	Set FKeys always. Except in TE SETUP, the numeric value of the Function Keys will always appear at FKEYS<1>, regardless of the PS 300 communication mode. When reset, the Fkeys become VT100 keypad keys.
<ESC>[>11h	Set cursor-keys always. Except in TE SETUP, the numeric value of the cursor keys will always appear at KBhandler<9> regardless of the PS 300 communication mode.
<ESC>[>12h	Set keypad-keys always. Except in TE SETUP, the numeric value of the right-hand keypad keys will always appear at KBhandler<9> regardless of the PS 300 communication mode.

There are four other E&S private set and reset escape sequences that can be used to set display features of the PS 300. These escape sequences change the status of the displays affected by the TERM and GRAPH keys on the left-hand keypad.

ANSI SEQUENCES	DESCRIPTION
<ESC>[>13h	Turns the TE display ON.
<ESC>[>131*	Turns the TE display OFF.
<ESC>[>14h	Turns the GRAPH display ON.
<ESC>[>141*	Turns the GRAPH display ON.
<ESC>[>101*, etc.	Reset the various modes. When reset, the keys function under the modes in effect. (For example, if "Fkeys always" is reset, and DECKPAM is set, the last four Function Keys will generate control sequences used by DEC's EDT and KED editing programs.

Any of the above modes may be set or reset by entering the appropriate characters in the SITE.DAT file, or by sending the appropriate sequence to <1>ES_TE1.

8.1.9 Values Appearing at KBhandler<9>:

When "keypad keys always" is set, the right-hand keypad numeric keys pass their own value (except 0); i.e. pressing the 5 key in "keypad keys always" mode causes an integer 5 to come out of KBhandler<9>. The remaining keys spiral out from the "9" key:

'-' is 10
',' is 11
ENTER is 12
'.' is 13
'0' is 14

Cursor keys:

Up cursor is 15
Down cursor is 16
Left cursor is 17
Right cursor is 18

These modes may be set or reset by entering the appropriate ASCII characters in the SITE.DAT file. Example:

```
SEND CHAR(27) & '[>10h' to <1>ES_TE1;
```

would set the "Fkeys always" mode.

* The last character of this escape sequence is a lowercase L.

8.1.10 Right-Hand Keypad - (DECKPNM and DECKPAM)

The characters or sequences transmitted by the right-hand keypad are dependent on a number of modes and configurations that can be set by the programmer. Normally, the right-hand keypad transmits the codes shown on the key caps; the 10 digits and some characters useful when keying in numerical data. However, in some host applications (DEC's editor utilities EDT and KED), these keys need to be interpreted as program function keys that cause some action to take place.

To differentiate these keys from the number and character keys on the main keyboard, the right-hand keypad has two modes; a keypad numeric mode, and a keypad application mode with a known sequence that the keys transmit in the application mode. (Refer to Tables 8-2 and 8-3.)

8.1.11 Right-Hand Keypad Escape Sequences

The keypad modes are set up by sending two different escape sequences from the host and eventually to the terminal emulator network (KBHANDLER).

<ESC>>	causes the keycap values (numeric and other) to be sent to the host when the keys are pressed. This is the keypad numeric mode that corresponds to DEC's DECKPNM.
<ESC>=	puts the keypad in the keypad application mode (DEC's DECKPAM). In this mode, pressing the keys causes them to transmit an escape sequence that begins with <ESC>O.

Setting the DEC VT52 mode (as opposed to the ANSI mode) will also affect the translation of these keys. Table 8-2 shows what is transmitted in the two modes when ANSI is set. Table 8-3 shows what is transmitted in the two modes when VT52 is set.

Table 8-2. Keypad Transmissions in ANSI Mode

	KEY CAP	NUMERIC MODE (DECKPNM)	APPLICATION MODE (DECKPAM)
	0	0	<ESC>Op
	1	1	<ESC>Oq
	2	2	<ESC>Or
	3	3	<ESC>Os
	4	4	<ESC>Ot
	5	5	<ESC>Ou
	6	6	<ESC>Ov
	7	7	<ESC>Ow
	8	9	<ESC>Oy
	-	-	<ESC>Om
	,	,	<ESC>Ol*
	.	.	<ESC>On
	ENTER	[CR]	<ESC>OM
**	P1(F9)	<ESC>OP	<ESC>OP
**	P2(F10)	<ESC>OQ	<ESC>OQ
**	P3(F11)	<ESC>OR	<ESC>OR
**	P4(F11)	<ESC>OS	<ESC>OS

* The last character of this escape sequence is a lowercase L.

** The P1 - P4 keys are the last four PS 300 Function Keys in the Terminal Emulator mode.

Table 8-3. Keypad Transmissions in VT52 Mode

	KEY CAP	NUMERIC MODE (DECKPNM)	APPLICATION MODE (DECKPAM)
	0	0	<ESC>?p
	1	1	<ESC>?q
	2	2	<ESC>?r
	3	3	<ESC>?s
	4	4	<ESC>?t
	5	5	<ESC>?u
	6	6	<ESC>?v
	7	7	<ESC>?w
	8	9	<ESC>?y
	-	-	<ESC>?m
	,	,	<ESC>?l*
	.	.	<ESC>?n
	ENTER	[CR]	<ESC>?M
**	P1(F9)	<ESC>P	<ESC>P
**	P2(F10)	<ESC>Q	<ESC>Q
**	P3(F11)	<ESC>R	<ESC>R
**	P4(F12)	<ESC>S	<ESC>S

* The last character of this escape sequence is a lowercase L.

** The P1 - P4 keys are the last four PS 300 Function Keys in the Terminal Emulator mode.

8.1.12 Escape Sequences that Affect Screen Display

There are a number of escape sequences that can be sent from the host causing some action to take place in the terminal that affect the screen display. These include cursor position, scrolling, deletion of text, scrolling regions, and selective graphic rendition.

The following sections describe the escape sequence commands that implement these actions.

8.1.13 Cursor Movement Command Escape Sequences

The cursor movement commands UP, DOWN, FORWARD and BACK are identical in form except for the final character. They take the form

<ESC>[Pc

where P is the number of positions to move, and c is A for UP, B for DOWN, C for FORWARD, and D for BACK. If P is 0, 1, or absent, it is interpreted to be 1.

These sequences, with P absent, are generated by the cursor keys when the cursor key mode is reset.

If a given cursor command causes the cursor to move out of the display area, the cursor is set at the edge of the display area in the direction of the move. Scrolling does not take place. If the cursor were on the bottom line, and the Terminal Emulator received the <ESC>[26B, nothing would happen. The cursor would remain on the bottom line, and no scrolling would take place.

<ESC>[PA CUU - Move the cursor P lines upward.

<ESC>[PB CUD - Move the cursor P lines down.

<ESC>[PC CUF - Move the cursor P columns forward.

<ESC>[PD CUB - Move the cursor P columns back (left).

The cursor position and horizontal vertical position (CUP, HVP) commands take the same form except for the final character. They take the form

<ESC>[Pl;PcC

where Pl is the line number to move to, Pc is the column to move to, and C is "H" for CUP and "f" for HVP (VT100 editor function). If one of these commands would cause the cursor to move out of the Display, it is set at the edge of the display area in the direction of the move. Scrolling does not take place. With no parameters present, it is equivalent to a cursor to home action.

<ESC>[Pl;PcH CUP - Move cursor to line Pl, column Pc.

<ESC>[Pl;Pcf HVP - Move cursor to line Pl, column Pc.

8.1.14 Index, Next Line, Reverse Index Command Escape Sequences (IND, NEL, RI)

These commands move the cursor, but may also cause scrolling to occur. All of them take the form <ESC>c.

<ESC>D IND – Move the cursor down one line, maintaining column positioning. If the TE is at the bottom line of the scrolling window when IND is received, a scroll-up is performed.

<ESC>E NEL – Move the cursor down one line and to column 1. If the TE is at the bottom line of the scrolling window when NEL is received, a scroll-up is performed.

<ESC>M RI – Move the cursor up one line, maintaining column position. If the TE is at the top line of the scrolling window when RI is received, a scroll-down is performed.

8.1.15 Erase Commands Escape Sequences (ED, EL)

The Erase in Display (ED) command takes the form

<ESC>[PJ

where P selects a specific erasing action. If P is absent, it is interpreted to be 0.

<ESC>[J or
<ESC>[0J Erase the display from the cursor to the end of the screen.

<ESC>[1J Erase the display from the beginning of the screen to the cursor.

<ESC>[2J Erase the entire screen.

The Erase in Line (EL) command takes the form:

<ESC>[PK

where P selects a specific erasing action. If P is absent, it is interpreted to be 0.

<ESC>[K or
<ESC>[0K Erase from the cursor to the end of the line.

<ESC>[1K Erase from the beginning of the line to the cursor.

<ESC>[2K Erase the entire line.

8.1.16 Set Top and Bottom Margins Command Escape Sequence (DECSTBM)

This command allows a scrolling window to be defined. Inside the given scrolling window, the lines scroll as they normally would for the entire screen. Outside of the window, lines do not scroll. This command also causes the cursor to be positioned in the upper-left corner of the scrolling region as defined.

The form of this command is:

‹ESC›[Pt;Pbr

where Pt is the top line of the scrolling window, Pb is the last line of the scrolling window, and r designates this command.

This command also requires that Pt < Pb since the scrolling window must be logical and contain a minimum of two lines. Should an illegal set of parameters be defined, the current setting of the window remains unchanged.

‹ESC›[Pt;PbrMake the scrolling window Pt to Pb, inclusive.

8.1.17 Set Graphic Rendition Command Escape Sequences (SGR)

The intent of this command is to make some part of the text displayed on the screen stand out, in contrast to the rest of the screen. The form of the command is

‹ESC›[Pm

where P selects some form of graphic rendition. If P is absent or 0, then the command means to turn off all forms of graphic rendition. As the most common methods used to make the contrast are difficult or expensive to implement on the PS 300, we interpret the command by underscoring the selected text in the display.

‹ESC›[Pm where P <> 0. Begin underscoring the text in the display.

‹ESC›[0m

or

‹ESC›[m Stop underscoring the text in the display

8.1.18 Report to the Host Command Escape Sequences (CPR, DSR)

These commands involve a query command from the host, and a response by the terminal. The query command takes the form:

<ESC>[Pn

where P selects the type of report requested. Two values of P are recognized: 5, which is a device status report, and 6, which requests a cursor position report.

The response takes the forms:

<ESC>[0n

that means "Ready, no malfunctions detected" and

<ESC>[Pl;PcR

where Pl is a two-digit ASCII number giving the current line (line 1 is at the top) and Pc is a two-digit ASCII number giving the current column.

Host:	<ESC>[5n	Please report status.
TE:	<ESC>[0n	Ready, no malfunctions detected.
Host:	<ESC>[6n	Please report active (cursor) position.
TE:	<ESC>[Pl;PcR	Cursor is at line Pl, column Pc.

8.1.19 VT52 Command Escape Sequences

In the PS 300 Terminal Emulator, all of the VT52 commands, except one, take the form:

<ESC>c

The exception, Direct Cursor Addressing, is discussed in the last paragraph of this section.

<ESC>A Move cursor up one position. Do not scroll.

<ESC>B Move the cursor down one position. Do not scroll.

<ESC>C Move cursor right one position.

<ESC>D Move the cursor left one position.

<ESC>H Move cursor to line 1, column 1.

<ESC>I Reverse line feed; reverse scroll if at top.

<ESC>J Erase to end of screen.

<ESC>K Erase to end of line.

<ESC>= Enter alternate keypad mode.

<ESC>> Exit alternate keypad mode.

<ESC>< Enter ANSI mode.

Direct Cursor addressing requires a 4-character sequence. The first two characters are <ESC>Y. The next two characters indicate the line and the column to move to. The desired number is obtained by subtracting 31 (Hex 1F or Octal 37) from the ASCII character code of the character. The first character that gives the line will be in the range of "!" (line 1) to "8" (line 24), and the second that gives the column will be in the range of "!" to "p" (column 80). For example:

<ESC>Y/@ Move the cursor to line 15, column 32.

The keypad mode commands are always recognized apart from the VT-52 emulation.

8.2 PS 300 TERMINAL EMULATOR FUNCTION NETWORK

The actual networking of the functions that build the terminal emulator is shown in Chapter 6 of this guide, System Functions. This section will discuss the three main terminal emulator functions in more detail.

8.2.1 Keyboard Manager - (KBhandler1)

The keyboard manager takes the stream of raw bytes from the keyboard and distributes them to output queues, translating to ANSI control sequences, if necessary; toggles graphics and terminal emulator displays.

Inputs:

<1>:Strings originating at keyboard; connected to data concentrator demultiplexing function.

Outputs:

<1>: To KEYBOARD
<2>: To SetUp
<3>: To CHOP PARSE
<4>: To host
<5>: to display handler function
<6>: to PLOTSTART1 (queues a hardcopy plot on the plotter)
<7>: To FKEYS.
<8>: To SPECKEYS
<9>: To user function-networks
<10>: to HCPIP1 (initializes hardcopy)
<11>: to FFPLOT1 (causes a form feed sequence to be sent to the hardcopy plotter)

Private:

None.

The first four outputs, <1> to <4>, are keyboard routes for four different tasks that the keyboard performs. <1> ultimately goes to a user function network that has been connected to KEYBOARD. This output is used when the keyboard is in the "Interactive" (KB) communication mode (Shift/Line Local). <2> goes to the TE SETUP function; hitting the SETUP key toggles this mode on and off. <3> is the output for the "Command" (CI) communication mode (Control/Line Local). It goes through a line editor function to a chop, parse, CI line for the interpretation of PS 300 commands. <4> is the Terminal Emulator (TE) output port and output is sent to the host computer.

The other outputs are minor and special purpose to some extent. <5> goes directly to the TE display data handler function for two reasons: the first is to pass commands resulting from the CLEAR/HOME key being pressed, the second is to implement the local-echo option of the terminal emulator. When outputting to this queue, the key handler expands CR to CRLF.

Output <6> is used to trigger the hardcopy function when the HARDCOPY key is pressed.

Output <7> sends out the proper Qinteger when an Fkey is pressed and input to a user function network is desired via FKEYS. <8> was added so that the cursor keys can be used in user function networks via SPECKEYS. <9> allows the numeric value of the right-hand keypad keys and the cursor keys to be passed to user function networks. <10> initializes hardcopy, and <11> causes a form feed sequence to be sent to the hardcopy plotter.

8.2.2 Terminal Emulator Display Handler (F:VT10) - ES_TE1

This function receives input from the host, from an error formatting function, and from the line editor that receives input from the keyboard in "Command" (CI) mode. The primary task of the data display handler is to make this input show up on the PS 300 screen.

F:VT10 TE display handler - ES_TE1

Inputs:

<1>:Qpackets, Qmorepackets. Input to the TE.

<2>:Qstring. answerback string.

Outputs:

<1>:Qpackets. Bells for the keyboard.
<2>:Qpackets. Status, cursor reports (to host).
<3>:Qpackets. Terminal ID (VT52 or VT100 to host)
<4>:Qpackets. Echoed unknown escape sequences.

Private:

None.

Users may send an answerback string to input <2> of ES_TE1. When the host sends ENQ (↑E or %X5), the answerback string is sent to the host. As most of the input stream will have an effect on the screen, or show up as displayable data, the outputs are minor. <1> is used to make the expected "beep" on receipt of a ↑G (the beeper is in the keyboard). <2> sends data back to the host when the function receives command sequences, such as cursor position and terminal ID (I am a VT-100). <3> is used to send the correct control sequence back to th host that identities the terminal. <4> is an aid for debugging and development. It sends out all command sequences that are received, but unknown by the function. Normally output <4> is not hooked up. It can be used to discover what kind of sequences a host program might be sending (that the terminal emulator cannot interpret) by hooking the output to a function like message_display.

8.2.3 Terminal Emulator Setup

TE_setup changes the characteristics of the terminal emulator.

Inputs:

<1>:Messages from key_manager

Outputs:

None.

Private:

None.

The Setup function gets input from the keyboard function and uses it to change the characteristics of the terminal emulator as a whole. Like the display handler function, the setup function manipulates a display structure that appears on the PS 300 screen and changes in response to actions by the user. Setup is interactive and uses menus and the function keys.

8.3 TE INITIAL DATA STRUCTURES

The data structures used by the terminal emulator are set up by the CONFIG.DAT file and then completed by a function TE_build.

The CONFIG.DAT file contains a CSM node and a color node. The CSM node, when set on, heightens the intensity of the TE display, SETUP, message_display, and data structures displayed on the screen. Both these nodes are accessible by sending the appropriate values to CSM1 and TECOLOR1. There is also a TSCSM1 CSM node that affects only the TE and the SETUP display.

TE_build add a set node, a 4x4 matrix, a matcon2 (to scale characters), and a set node called the line set to the name TD<suffix>$ that is established by the CONFIG.DAT file. From the line set, a structure for each line is hung and a structure is hung for the cursor. The display handler function keeps various pointers into this structure and uses them to get data on the screen, perform scrolls, etc.

8.4 KEYBOARD COMMUNICATION MODES

The three modes of operation, Terminal Emulator (TE), Command (CI), and Interactive (KB) are all modes of operation that are established by pressing a key (or combination of keys) on the PS 300 keyboard. The term "mode" is slightly misleading. Mode is also used to describe the operation of the keypads, cursor keys, and other terminal emulator features. The modes referred to here are actually determined by what output port is used by the key_manager.

The command sequences that can be sent to <1>KBhandler1 to toggle these communication modes are:

```
CI  CHAR (22) & CHAR (18)
KB  CHAR (22) & 'R'
TE  CHAR (22) & 'r'
```

For example, to boot up in KB mode, the following command would be placed in SITE.DAT:

```
SEND CHAR (22) & 'R' to <1>KBhandler1;
```

The keys and the output that are generated from the keyboard_manager (KBhandler) in the three modes is discussed in the following section.

8.4.1 Keys and Outputs

SETUP is the easiest to summarize. Except for the left keypad, all special keys (and all central keys) are passed on.

In any of the modes listed below, if "keypad-keys always" or "cursor-keys always" is set, the numeric value of the keys will be sent to any user function network connected to <9>KBhandler. If "Fkeys always" is set, an integer is output from FKEYS<1>.

In Interactive mode (KB), the right-hand keypad keys will be translated into the keycap numbers if Knum is true; otherwise they will be passed out KEYBOARD as ASCII characters (or to output <9> as the numeric value of the key if keypad-keys always is set). Fkeys always go out the FKEYS queue as Qintegers and the cursor keys always go out to SPECKEYS as <char>xyzw (or to output <9> as <char> if cursor-keys always is set).

In Command mode (CI), the right-hand keypad keys will be passed as numbers if Cnum is true and as sequences, ↑V<char>. Output here is only to the CI queue. Finally, cursor keys will always go out the CI queue as ↑V<char>.

The Terminal Emulator (TE) mode is the most complicated. The treatment of the right-hand keypad depends on two modes: DECKPM and DECCKM. In DECKPM, when false, the keys are translated to their keycap values (numeric mode) or when true, the keys are translated into escape sequences. The escape sequences that are generated depend on whether VT52 is true(<ESC>?<char>) or false (<ESC>O<char>). Fkeys is identical to CI mode. Otherwise, the keys become a superset of DEC's PF keys and send out <ESC>O<char> sequences like the right-hand keypad in DECKPM mode. The cursor keys send out escape sequences (unless cursor-keys always is set). If VT52 is true, the sequences are <ESC><char>. If the TE is emulating a VT100, then the sequence depends on DECCKM. If DECCKM is true, then <ESC>O<char> is sent, so that the cursor keys look like PF keys (or the right-hand keypad in DECKPM mode); otherwise the sequence is <ESC>[<char>, which is the ANSI command sequence to make the cursor to one place in the direction of the arrow.

The left-hand keypad keys need to be covered individually, except for the GRAPH and TERM keys. GRAPH and TERM allow the user to toggle the graphics and the TE displays on and off. The viewports and the set are created in the CONFIG.DAT file.

The HARDCOPY key causes a trigger message to be sent out the hardcopy queue. The SETUP key toggles the SETUP mode. In SETUP mode, all keys are passed to the SETUP function. At the end of SETUP, caused again by hitting the SETUP key, the last use and queue are pulled. CLEAR/HOME causes an ANSI control sequence to be passed to the display handler function. The key used by itself causes two sequences to be passed: one that clears the screen and one that causes the cursor to return to the "home" position (upper left corner of the TE display).

The SHIFTED CLEAR/HOME key sequence causes the cursor to move to the home position, but the screen is not cleared. The CLEAR/HOME-CONTROL key sequence causes the screen to clear, but the cursor will maintain the same position.

The LINE/LOCAL key is used to multiplex the keyboard between the communication modes (except SETUP). When used alone LINE/LOCAL = TE mode, Shift LINE/LOCAL = KB mode, and Control LINE/LOCAL = CI mode.

Current mode information is available through SETUP, which is functional in any mode.

The following diagram is an attempt to illustrate the keys, the modes, the output of the keys in the modes, and any other combinations that are useful.

The representation in the diagram assumes that ANSI is set.

	TE MODE	CI MODE	KB MODE
FUNCTION KEYS	Final 4 keys used w/ DECKPAM (EDT)	↑V char	Qinteger to FKEYS
Fkeys always	Qinteger to FKEYS	Same as TE	Same as TE
CURSOR KEYS			
DECCKM - Set	Application functions to host	Ignored	Ignored
DECCKM - Reset (w/DECKPAM set)	Cursor control commands to host	Ignored	<char>to SPECKEYS
Cursor keys always	Qinteger to KBhandler<9>	Same as TE	Same as TE
RIGHT-HAND KEYPAD			
DECKPNM	Numeric value passed to host	Ignored	Ignored
DECKPAM	Transmits control sequences to host for EDT utility	Ignored	Ignored
Keypad keys always	Qinteger to KBhandler<9>	Same as TE	Same as TE

8.4.2 Using the SITE.DAT File to Change Features of the Terminal Emulator

The SITE.DAT can be used to set bootable values for the SETUP features of the terminal emulator. A full description of the SETUP features and the default values for the features is given in Volume I of this documentation set. The following section gives the PS 300 commands that can be used to change features or defaults of the PS 300 Terminal Emulator.

TE characteristics are changed by sending sequences to <1>KBhandler1. These sequences will have the same effect as if they had been keyed in the SETUP mode of the Keyboard and Display. (Refer to Volume I, ***User Operation and Communication*** for a description of the SETUP feature of the Terminal Emulator.)

There are four groups of commands: toggles, breakkey, mode, and displays, each of which is handled differently.

Toggles

These are TE options that have two values, true and false or on and off. In SETUP, they are changed by pressing a single Function Key that changes the present value to its opposite. To put a command in the SITE.DAT file so that the TE feature comes up in its desired value at bootup, the toggling sequence must be sandwiched between two sequences that represent the pressing of the SETUP key. The header and trailer sequence for the SETUP key is CHAR(22) & "o".

The following chart gives the SETUP name, the definition, the default value, and the PS 300 command sequence to change the default.

Setup Name	Definition	Default Setting	Sequence to toggle
SRM	Local Echo	OFF	CHAR (22) & 'b'
Awrp	Automatic line wrap	OFF	CHAR (22) & 'c'
ANSI	ANSI sequences obeyed	ON	CHAR (22) & 'd'
VT52	VT52 mode	OFF	CHAR (22) & 'e'
KPM	Keypad Application Mode	OFF	CHAR (22) & 'f'
CKM	Cursor Key Mode	OFF	CHAR (22) & 'g'
Cnum	Keypad Numeric CI Mode	ON	CHAR (22) & 'h'
Knum	Keypad Numeric KB Mode	ON	CHAR (22) & 'i'

Example: To setup the TE for local echo (host is noecho) and for automatic line-wrap, the following command would be placed in the SITE.DAT file:

```
SEND CHAR(22) & 'o' & CHAR(22) & 'b' & CHAR (22) & 'c'
   & CHAR(22) & 'o' to <1>KBhandler1;
```

It is recommended, when possible, that the E&S private escape sequences used to set/reset the various modes of the terminal emulator be used in the SITE.DAT file. These commands are more compact, and take up less space on the diskette. Example: To setup the TE for local echo (host is noecho) and for automatic line-wrap, the following commands can be sent to ES_TE1:

```
Send CHAR(27) & '[>1;2h' to <1>ES_TE1;
```

The BREAK Key

The BREAK key, like the toggles, must be sandwiched between sequences representing the SETUP key. It also has an inner sandwich, telling SETUP that it is the BREAK key and the end of the definition. The important sequence in these two outer wrappings represents the special key designated by the user to be the BREAK key. For example, to set the HARDCOPY key as the BREAK key, the following command would be placed in the SITE.DAT file:

```
SEND CHAR(22) & 'o' & CHAR(22) & 'j' & CHAR (22) & 'n'
   & CHAR (22) & 'a' & CHAR(22) & 'o' to <1>KBhandler1;
```

where:

CHAR(22) & 'o' is the header/trailer sequence for the SETUP key (to enter SETUP)

CHAR(22) & 'j' is the sequence for Function Key #10 (to enter the set/BREAK key mode)

CHAR (22) & 'n' is the sequence designating the HARDCOPY key as BREAK key

CHAR (22) & 'a' is the sequence for Function Key #1 (exiting out of set/BREAK key)

CHAR(22) & 'o' is the header/trailer sequence to exit SETUP.

Mode

To put the keyboard into interactive mode on bootup, the following should be put in the user's SITE.DAT file:

```
Send CHAR(22) & 'R' to <1>KBhandler1;
```

The PS 300 normally comes up in Terminal Emulator Mode (TE) mode; that is, the keyboard outputs to the initial instance of ES_TE. To change to the other two modes (either CI or KB), the following sequences may be inserted in the SITE.DAT file. Note that these do not have to be sandwiched between SETUP key sequences.

MODE	SEQUENCE
CI	CHAR (22) & CHAR (18)
KB	CHAR (22) & 'R'
TE	CHAR (22) & 'r'

Displays

The two displays are the TE display and the Graphics display. They are toggled by the TERM and GRAPHICS keys and normally are on. To turn them off at boot time, special sequences may be sent.

DISPLAY	SEQUENCE
TE	CHAR(22) & 's'
Graphic	CHAR(22) & 'p'

For example, to turn the TE display off at boot time, the following command would be placed in SITE.DAT:

```
SEND CHAR(22) & 's' to <1>KBhandler1;
```

The only TE characteristic that cannot be conveniently set by a SITE.DAT file is the size and placement of the TE display.

8.4.3 Using the SITE.DAT To Send Control Sequences to the Terminal

Control sequences that affect the screen display (as well as any other escape sequences) can be placed in the SITE.DAT file as ASCII sequences. The terminal emulator function ES_TE1 can accept and translate these sequences. The escape sequence in the SITE.DAT should take the following form:

SEND <char> n &'[P1;P2;...Pnc' to ES_TE1;

where "[" is the control sequence introducer and P1 through Pn are the parameters that may or maynot be present.

8.4.4 Dual User SITE.DAT File Information

If the system supports two graphics stations (PS 320), then sequences that control modes or display characteristics must be suffixed by the appropriate user identification number. Those sequences that are to affect usernumber1 station should be sent to KBhandler1 or ES_TE1. Those sequences that are to affect usernumber2 station should be sent to KBhandler3 or ES_TE3 in the SITE.DAT file.

CONTENTS

9. SYSTEM ERROR MESSAGES

This chapter provides a description of the system error messages that a programmer might encounter during standard operation of the PS 300 Graphics System. There are three lists provided. The first list is the error number and brief description of the "traps" or software induced exceptions that might cause the system to fail. Following the list of traps is a list of the error numbers (with error definitions) of system errors that might be caused by a programmer using the User-Written Function (UWF) facility provided with the A1 release of the Graphics Firmware. The final list is a comprehensive list of the system error numbers. Most system errors are generated only during the development process of the Graphics Firmware and are rarely, if ever, seen during normal system operation. If any of the error numbers are reported, other than those that might be caused by user-written functions when using the UWF feature (shown in Table 9-2), E&S Customer Engineering Software Support should be notified.

TABLE 9-1 PS 300 TRAPS and Their Meanings

NUMBER	DEFINITION
0	Not enough available memory to come up or handle request.
1	E&S Firmware Error.
2	Memory corrupted or over-written (could be caused by UWF).
3	Memory corrupted or over-written (could be caused by UWF).
5	Attempt to wait on queue when function is waiting on another device (CLOCK, I/O) (could be caused by UWF).
6	System Errors (see next list).
7	Double-bit mass memory error if address on LEDs is between 200 and 300; unexpected interrupt on a vector with no routine if address is between 300 and 400.
10	Memory corrupted or over-written (could be caused by UWF).
11	E&S Firmware Error.
12	Pascal in-line runtime error: usually caused by Case statement in Pascal with no Otherwise clause (could be caused by UWF).

TABLE 9-2 User-Written Function Error Descriptions

ERROR NUMBER	DEFINITION
SYSTEMERROR #7F	If exit function before re_queuing function (not following template).
SYSTEMERROR #80	Bad parameter passed to text utility routine: Text_text, B1 < 0.
SYSTEMERROR #81	Bad parameter passes to text utility routine: Char_text, b < 0.
SYSTEMERROR #8E	Bad parameters passed to Updates utilities: AnnounceUpdate List tail = nil; head <> nil.
SYSTEMERROR #B9	Nil or invalid parameter passed to Illegal_Input handling routines.
SYSTEMERROR #C9	User written function stack overflow.
SYSTEMERROR #CB	Improper redefinition of user written function name.
SYSTEMERROR #D9	Call to Ckinputs has Nmin < 0.
SYSTEMERROR #DA	Call to Ckinputs has Nmin > Nmax.
SYSTEMERROR #DB	Call to Ckinputs has Nmax > total number of inputs for function.
SYSTEMERROR #DE	Multiple call to Qsendcopymsg on the same input.
SYSTEMERROR #E0	Function was not in state Running when Ckinputs was called; Cleaninputs returned a FALSE and still called Ckinputs; Cleaninputs was not called before calling Ckinputs the second time.
SYSTEMERROR #E1	Function was not in state Mid_running when Cleaninputs was called.

TABLE 9-2 User-Written Function Error Descriptions (continued)

ERROR NUMBER	DEFINITION
SYSTEMERROR #E9	Qillmessage, or Qillvalue was called for input which does not exist.
SYSTEMERROR #EA	Qillmessage, or Qillvalue was called for input which was already dealt with; previous call to Qillmessage, Qillvalue, or Qsendcopymsg.
SYSTEMERROR #110	Tolerance on FCnearzero is too small.
SYSTEMERROR #111	Set node has no dummy control block.
SYSTEMERROR #68	Possible overwrite of block boundary: Sending to an unrecognized Namedentity.
SYSTEMERROR #A1	Possible overwrite of block boundary: AppendVector, Invalid Acpdata type.
SYSTEMERROR #92	Possible overwrite of block boundary: ByteIndex Invalid Acpdata type.
SYSTEMERROR #9C	Possible overwrite of block boundary: Unrecognized type of Namedentity.
SYSTEMERROR #9D	Possible overwrite of block boundary: Hasstructure.
SYSTEMERROR #A3	Possible overwrite of block boundary: Nomemsched, Bad.Status for a fcn.
SYSTEMERROR #103	Possible overwrite of block boundary: Curfcn was not active at entry.
SYSTEMERROR #109	Possible overwrite of block boundary: ContBlock, nil block.
SYSTEMERROR #10D	Possible overwrite of block boundary: GetVector, Not an Acpdata block.
SYSTEMERROR #10E	Possible overwrite of block boundary: GetVector, Not a vector Acpdata block.

TABLE 9-2 User-Written Function Error Descriptions (continued)

ERROR NUMBER	DEFINITION
Motorola Exceptions	
These exceptions could be due to E&S software exceptions or to UWF memory overwrites:	
Exception 2	Bus Error.
Exception 3	Address Error.
Exception 4	Illegal Structure.

TABLE 9-3 List of System Error Messages

The following list is the comprehensive list of system error messages. The messages are shown in two categories: software exceptions possibly caused by use of the UWF facility (Possible UWF error) and software exceptions that indicate that Customer Engineering Software Support should be notified (E&S Firmware Error).

Error Number	Definition
#64	E&S Firmware Error
#65	E&S Firmware Error
#66	E&S Firmware Error
#67	E&S Firmware Error
#68	Possible UWF Error
#69	E&S Firmware Error
#6A	E&S Firmware Error
#6B	E&S Firmware Error
#6C	E&S Firmware Error
#6	E&S Firmware Error
#6E	E&S Firmware Error
#6F	E&S Firmware Error
#70	E&S Firmware Error
#71	E&S Firmware Error
#72	E&S Firmware Error
#73	E&S Firmware Error
#74	E&S Firmware Error
#75	E&S Firmware Error
#76	E&S Firmware Error
#77	E&S Firmware Error
#78	E&S Firmware Error
#79	E&S Firmware Error
#7A	E&S Firmware Error
#7B	E&S Firmware Error
#7C	E&S Firmware Error
#7D	E&S Firmware Error
#7E	E&S Firmware Error
#7F	Possible UWF Error
#80	Possible UWF Error
#81	Possible UWF Error
#85	E&S Firmware Error
#86	E&S Firmware Error
#87	E&S Firmware Error
#88	E&S Firmware Error
#8A	E&S Firmware Error

TABLE 9-3 List of System Error Messages (continued)

Error Number	Definition
#8D	E&S Firmware Error
#8E	Possible UWF Error
#8F	E&S Firmware Error
#90	E&S Firmware Error
#91	E&S Firmware Error
#92	Possible UWF Error
#93	E&S Firmware Error
#94	E&S Firmware Error
#95	E&S Firmware Error
#96	E&S Firmware Error
#97	E&S Firmware Error
#98	E&S Firmware Error
#99	E&S Firmware Error
#9C	Possible UWF Error
#9D	Possible UWF Error
#9E	E&S Firmware Error
#A1	Possible UWF Error
#A3	Possible UWF Error
#A9	E&S Firmware Error
#AA	E&S Firmware Error
#AB	E&S Firmware Error
#AC	E&S Firmware Error
#AD	E&S Firmware Error
#AE	E&S Firmware Error
#AF	E&S Firmware Error
#B0	E&S Firmware Error
#B3	E&S Firmware Error
#B4	E&S Firmware Error
#B8	E&S Firmware Error
#B9	Possible UWF Error
#BA	E&S Firmware Error
#BD	E&S Firmware Error
#BF	E&S Firmware Error
#C0	E&S Firmware Error
#C1	E&S Firmware Error
#C2	E&S Firmware Error
#C3	E&S Firmware Error
#C9	Possible UWF Error
#CA	E&S Firmware Error
#CB	Possible UWF Error
#CC	E&S Firmware Error

TABLE 9-3 List of System Error Messages (continued)

Error Number	Definition
#CD	E&S Firmware Error
#CF	E&S Firmware Error
#D0	E&S Firmware Error
#D1	E&S Firmware Error
#D2	E&S Firmware Error
#D3	E&S Firmware Error
#D4	E&S Firmware Error
#D5	E&S Firmware Error
#D6	E&S Firmware Error
#D7	E&S Firmware Error
#D8	E&S Firmware Error
#D9	Possible UWF Error
#DA	Possible UWF Error
#DB	Possible UWF Error
#DC	E&S Firmware Error
#DE	Possible UWF Error
#DF	E&S Firmware Error
#E0	Possible UWF Error
#E1	Possible UWF Error
#E2	E&S Firmware Error
#E5	E&S Firmware Error
#E6	E&S Firmware Error
#E7	E&S Firmware Error
#E8	E&S Firmware Error
#E9	Possible UWF Error
#EA	Possible UWF Error
#EB	E&S Firmware Error
#ED	E&S Firmware Error
#EE	E&S Firmware Error
#EF	E&S Firmware Error
#F0	E&S Firmware Error
#F1	E&S Firmware Error
#F3	E&S Firmware Error
#F6	E&S Firmware Error
#F7	E&S Firmware Error
#F8	E&S Firmware Error
#F9	E&S Firmware Error
#FC	E&S Firmware Error
#FD	E&S Firmware Error
#FE	E&S Firmware Error
#FF	E&S Firmware Error
#100	E&S Firmware Error

TABLE 9-3 List of System Error Messages (continued)

Error Number	Definition
#101	E&S Firmware Error
#102	E&S Firmware Error
#103	Possible UWF Error
#104	E&S Firmware Error
#105	E&S Firmware Error
#106	E&S Firmware Error
#107	E&S Firmware Error
#108	E&S Firmware Error
#109	Possible UWF Error
#10A	E&S Firmware Error
#10B	E&S Firmware Error
#10C	E&S Firmware Error
#10D	Possible UWF Error
#10E	Possible UWF Error
#10F	E&S Firmware Error
#110	Possible UWF Error
#111	Possible UWF Error
#112	E&S Firmware Error
#113	E&S Firmware Error

CONTENTS

10. UTILITY COMMANDS

This chapter provides a reference for the Utilities commands that are on the PS 300 Diagnostic diskettes. The utilities are used to back-up diskettes and for diskette file management. The first section of the chapter tells how to access the Utility program that contains the commands and gives a list and a short description of the commands. The second part of the chapter provides the steps that are taken when backing up the Graphics Firmware, or any other system diskette, and describes the downloading procedure.

10.1 DIAGNOSTIC UTILITY COMMANDS

This section provides a listing of the diagnostic utility commands available on the Diagnostic diskettes. This section can be used as a reference section for a listing of the commands and how to access them.

Any PS 300 Diagnostic Diskette can be loaded to access the Utility commands. Load the diskette using the following steps:

1. The system should be powered down, and the activity light off.

2. Dismount any diskette in the PS 300 disk drive.

3. Insert any PS 300 Diagnostic diskette into the drive. The E&S label on the diskette should face up and the covered write-protect slot should face to the left.

4. Power up the PS 300 Control Unit and the Display. An auxiliary terminal or the PS 300 Keyboard may be used to enter in the diagnostic commands. If an auxiliary terminal is used, it can be connected to Port 1-5 of the Control Unit Connector Panel. Usually, the debug port (Port 3) is used.

5. Wait until the PS 300 Keyboard (or auxiliary terminal) displays "O" and beeps before continuing.

6. Repeatedly type in:

   <CTRL> P

   (strike P while depressing the CTRL key) on the terminal or the PS 300 Keyboard until the system responds. The system will respond with:

   ```
   PS 300 Diagnostic operating system Ax.Vxx
   Disk name = PS 300 Diagnostic Disk X Ax.Vxx
   Type "HELP" for help.
   =
   ```

   The = prompt indicates the diagnostics have been successfully loaded.

The Utility program in the Diagnostic operating system contains commands used to format diskettes, to check, copy, delete, modify, download, and send back files. The Utility program also has Terminal Emulator capabilities. These commands and their descriptions are listed following the loading procedures. The Utilities program is selected by typing in UTILITY, followed by RETURN. The > is the prompt for the Utilities program. The program indicates it has been loaded as follows:

```
=Utility
UTILITY; 1 loaded

PS 300 file and download Utility Px.Vxx
Type HELP for additional help.

Utility>
```

Utility then waits for a command to be entered and executes that command.

A command is entered by typing in an alphabetic string that is long enough to identify the command. For example, typing in CH is enough to identify the command CHECK. Typing in CHECK would also work. Some commands containing parameters require more than one line when they are being entered. The system prompts for any required parameters that are not entered by the operator. If the first character in the entered command is not alphabetic, or if the first word in the entered command is incorrect, the system responds with:

```
Invalid command.
```

The following file utility commands are available:

CHECK	Reads the entire diskette to check for diskette errors and to determine if the file structure is valid.
COMPARE	Compares two diskettes or two files to determine if they are the same.
COMPRESS	Compresses a diskette by copying each file over any empty space on the diskette until all empty space resides in one contiguous block at the end of the diskette.
COPY	Copies a file from one diskette to another diskette or copy a file from one place on a diskette to another place on the same diskette.
COPYDISK	Copies the contents of an entire diskette onto another diskette.
CREATE	Creates a file from data in memory.
CREATEBOOT	Creates a boot file from an existing file.
DATE	Displays and/or changes the date.
DELETE	Deletes a file.
DIRECTORY	Displays the diskette directory.
DRIVE	Selects a diskette drive.
DUMP	Dumps a file from the diskette into memory.
EXIT	Returns to the Diagnostic Operating System monitor.
FORMAT	Formats and initializes a diskette.
FREE	Indicates the number of free blocks on a diskette.
HELP	Displays a list of available commands and information about each command.
INITIALIZE	Initializes a diskette without formatting it.
MEMORY	Displays memory size and allows use of either local or mass memory.
MODIFY	Modifies the host communication parameter values for baud rate, parity, port number, etc.

PURGE	Deletes all but the latest version of each file or all but the latest version of one specific file from the diskette.
REMOVE	This is a query "delete". It will selectively delete any files on the disk as it prompts the user through the file names.
RENAME	Renames a file.
RENAMEDISK	Changes the diskette title.
RESTORE	Restores one of the saved (see SAVE command) files containing the host communication value parameters.
SAVE	Saves host communication parameter values modified using the MODIFY command.
SENDBACK	Transfers a file from the PS 300 to the host.
TERMINAL	Software resident on the diagnostic diskettes lets the user access a line to the host system. The communication parameters must be correct for this command to work.
TRANSFER	Transfers a file from the host to the PS 300.
TYPE	Types the contents of a file to the terminal.

Use the HELP command for a brief description of the function and syntax of each Utility command. Once the name of the command to be used is known and entered after the Utility> prompt, the Utility program steps through a series of prompts that will complete the command.

10.2 BACKING UP FIRMWARE AND DIAGNOSTIC DISKETTES

This section describes how to make a copy of the Graphics Firmware or any of the PS 300 diskettes. The diskettes should be copied as they are received.

Steps to be taken in backing up a diskette are:

1. Load PS 300 Diagnostic diskette.

2. Access the Utility program on the Diagnostic diskette.

3. Format a blank diskette to be used as the copy diskette.

4. Using either mass memory or local memory, store data off the original diskette so that it can be transferred to the copy diskette.

A blank diskette(s) must be available to use as the copy disk(s).

10.2.1 Formatting the Destination Diskette

The Utility program in the diagnostic operating system is used to format the blank diskette and copy the PS 300 Graphics Firmware. The procedural steps are as follows:

1. Load a Diagnostic diskette. Access the Utility Program by typing in Utility.

1. Dismount the Diagnostic Software Diskette.

2. Mount the blank diskette (destination diskette) to be formatted in the diskette drive. As the Graphics Firmware (or Diagnostic software) is to be copied onto this diskette, make sure the "write-protect tape" has been removed.

3. When the destination diskette is mounted, type in FORMAT on the terminal.

4. The system responds with:

```
Utility> Format
ENTER DISK NAME
```

5. Although the system asks for a disk name, a response is not necessary. The Firmware or Diagnostic diskette is copied onto the destination diskette, name included. Enter a RETURN to continue.

6. The system should now format the diskette. If the diskette is found to be write-protected, the system returns with this message:

```
*****ERROR: Disk write protected.
```

   Should this message appear, make sure the write-protect tape has been removed. If the tape has been removed, and the system still responds with this message, a new diskette should be used.

7. After the destination diskette has been successfully formatted, the Utility program prompt "Utility>" is again displayed.

Difficulties experienced in getting a disk formatted are usually the result of a bad disk. Other problems may include faulty mounting of the diskette in the drive.

The CHECK command can be used to determine if a diskette has been properly formatted.

10.2.2 Copying the PS 300 Diskettes

After the Utility> prompt sign appears, use the following procedures to copy the Graphics Firmware or Diagnostic diskettes.

If mass memory is to be used in copying the disk, it must be initialized at this time. It is faster to copy diskettes using mass memory, rather than local memory. To initialize the mass memory, type "Mem" for Memory.

The system will respond with:

Memory is currently set to use xxxK of local memory.

Do you want to change using mass memory?

With a negative response (N or NO), or a RETURN, the system uses local memory.

With an affirmative answer (Y or YES), the system displays the current amount of mass memory available:

xxxK of memory is available in mass memory.

and initializes mass memory to be used in the COPYDISK command.

1. Type in COPYDISK at the keyboard.
2. The system responds with:

   Copy using 1 or 2 drives? (See Section 10.2.3 for 1 disk drive.)

3. The number 1 or 2 are the only valid responses. If only one drive is available, enter 1.
4. The system then asks:

   Which drive? (Press RETURN for standard.)

   The diskette drives are number 0 and 1. In single-drive systems, the response would be a 0 followed by a RETURN, or just RETURN.

If a "2" is entered in response to Step 2, the following prompts will appear. (If a "1" is entered, follow the steps listed in Section 10.2.3.)

1. The system will then prompt:

   Enter source drive number.

   The drive number of the drive that contains the source diskette should be entered. The system will then prompt:

   Enter destination drive number:

   Enter the number of the drive containing the newly formatted diskette.

2. The system then prompts:

   Please insert source and destination disks, then press RETURN.

3. The system will now load the data from the source file into memory and copy it onto the destination diskette. The next system prompt that appears is:

   The disk has been copied.
   Do you want another copy of the same disk?

   (Second prompt will only appear if mass memory is used.)

10.2.3 COPYDISK Using Mass Memory With One Drive

If a YES or Y is entered after the Memory request, the system uses mass memory to store the data from the PS 300 diskette and then initializes mass memory to be used in the COPYDISK command. With a response of "1" to #3 of COPYDISK, the next system message is:

Please insert the source disk, then press RETURN.

1. Remove the newly formatted blank destination diskette from the drive and mount the PS 300 diskette to be copied. Do not take off the write-protect tape.

2. Press RETURN to continue. With enough available mass memory, the system copies all the data from the source disk into mass memory.

3. When copying is complete, the system responds with the message:

   Please insert the destination diskette, then press RETURN.

4. Remove the source diskette and insert the destination diskette.

5. Enter RETURN. The system now copies all files that were written into mass memory from the source diskette onto the destination diskette.

6. When all data from the source diskette do not fit into mass memory, more than one pass is necessary to accomplish the copying. If this is the case, the system again prompts:

    Please insert the source disk and press return to go on.

If this is the case, repeat steps (1) through (7) until the system responds with:

    The disk has been copied.
    Do you want another copy of the same disk?
(Second prompt will only appear if mass memory is used.)

A negative response or a RETURN will bring up the UTIL prompt. If another copy of the disk is desired, enter an affirmative answer, and the system will prompt for the insertion of a new destination diskette.

10.2.4 COPYDISK Using GCP Local Memory

If mass memory is not available for temporary use with the COPYDISK utility, the system uses the GCP local memory. After a NO is entered in response to the system query:

    Do you want to change using mass memory?

the system uses the available local memory for temporary storage.

The system uses the same system messages and prompts for the copy procedure using mass memory.

10.2.5 Error Messages During COPYDISK

There are several error messages that may appear during Copydisk. If the system displays the following:

    *****ERROR: Record not found during write.

reformat the diskette and try COPYDISK again.

Refer to the list of Utility commands at the front of this chapter for more commands that may be helpful in backing up the PS 300 diskettes.

10.2.6 The CHECK Command

The CHECK command can be used to determine if a diskette has been properly formatted. The CHECK command will respond with a detailed report of the number of blocks in the header, footer, and body of each file. The message will appear as:

```
Header              0
Directory           1
File Name.Ex_;26    2-43
* Empty *           44-719
```

If the system comes back with the Utility› prompt after entering the CHECK command, and no error messages appear, the diskette has been examined by the Diagnostics Operating System and found to be properly formatted.

10.2.7 Checking the Files

When the file copying is complete, the Directory utility DIR can be used to see if all files were copied from the source disk. The CHECK utility can be used to read the diskette and display the name and number of sectors of each file, or the COMPARE utility can be used to compare the newly copied disk with the source disk.

10.2.8 The DELETE Command

The DELETE command is used to delete a file from the diskette.

If the PS 300 Diagnostic Software has not been loaded prior to requesting the DELETE command, refer to the front of this chapter for information on loading the Diagnostic Software.

This Utility command should be used to delete the original SITE.DAT file from the copy of the Graphics Firmware before downloading the new version.

If the extension is not specified, the first file found on the diskette that has a matching file name and version number is deleted. The version number must always be specified.

For example, to delete the file SITE.DAT;4 from the diskette the following would be entered:

```
Utility>DELETE
Enter name of file to be deleted: SITE.DAT;4
File deleted successfully.
Utility>
```

The DELETE command may also be entered on one line:

```
Utility>DELETE SITE.DAT;4
```

The file name must be valid, and must include a version number. If the version number is not specified, the system advises the operator of the error with the following message:

```
Error, version number must be specified.
```

Upon completion of the above example, the file SITE.DAT;4 would no longer exist on the diskette; the space formerly occupied by the file would be available for other files.

10.3 DOWNLOADING USING THE DIAGNOSTIC UTILITIES

The following steps should be used when downloading from the host onto a PS 300 System diskette using the commands available on the Diagnostics diskettes.

1. Load a PS 300 Diagnostics diskette.

2. Access the Utility program.

3. At the prompt, type in:

   MODIFY

   followed by a carriage return.

   The modify command allows parameters that affect system operation to be modified. When the Modify command mode is entered, the following is displayed:

   ```
   Utility> MODIFY
   Type 0 to display the menu.
   ```

If 0 is entered, the following menu is displayed on the screen if a terminal is used:

```
The following may be modified:
 0) Re-display the menu.
 1) Display all current modes.
 2) Set host port.
 3) Set host port through data concentrator.
 4) Set host port baud rate.
 5) Set XON character.
 6) Set XOFF character.
 7) Set sendback string.
 8) Set sendback terminator string.
 9) Set transfer string.
10) Set transfer terminator string.
11) Set S-record offset value.
12) Set terminal exit character.
13) Set file name specifier character.
14) Set brief mode on.
15) Set brief mode off.
16) Set end of line.
17) Set parity and character length.
18) Set terminal page size and screen width.
All others: exit.

Enter a number:
```

The Modify Option 1) displays the current value of all modes that can be changed by the Modify command.

The two options to be used, with the possible exception of options 2) – 4) that modify the host line, are Option 9), which allows the user to specify the string to be used to cause the host to send a file to the PS 300 (the appropriate terminal interface copy command for the system), and Option 10), which allows the user to set the appropriate transfer string terminator character.

4. Enter the number "9" and a carriage return. This option defaults to TYPE <fn><CR>, where <fn> is a placeholder. This placeholder will later be filled in with the name of the file that has been created on the host computer. The TYPE command is the copy command under VMS. For RSX-11M, "PIP TI:= <fn><CR>" should be entered as the appropriate copy command.

   Option 9) will begin by displaying the current transfer string and prompting for the new string to be entered.

5. Select a character not in the transfer string. Enter that character, followed by the appropriate transfer string, followed by <CTRL F>, and ending with the beginning character. For example, if the character selected is "@", and the appropriate transfer string is "TY", this sequence would be entered:

   @TY <CTRL F><CR> @

   The <CTRL F> is entered by striking the F key while depressing the CTRL key. The Modify sub-command will echo the <CTRL F> back out to the terminal as the four characters "<fn>". Similarly, the carriage return will be echoed back as "<CR>".

   If a mistake is made while entering the transfer string, simply repeat its redefinition by re-entering the number "9" and a carriage return in response to the prompt "Enter a number:".

6. You will see the prompt:

   Enter a number:

   Enter the number "10" and a carriage return. The system will display the current terminator string and prompt for the new character. Enter three characters,"/<terminator>/, where <terminator> is the character that was used to end the file of PS 300 commands created on the host. If the character / was selected as the file terminator, this sequence would be entered:

   @/@

7. You will see the prompt:

   Enter a number:

   Type a carriage return to exit the Modify option of DOWNLOAD. The Utility> prompt will appear on the screen.

   Should a superfluous carriage return be entered during Steps 8 - 10, the system will return to the Utility> prompt. Should this occur, typing in Modify, followed by a carriage return will bring up the Modify menu and the "Enter a number" prompt.

8. Type "Terminal". This attaches the terminal to the host computer. Sign on to the host in an appropriate account that will allow the copy utility, set as the transfer string in Step 9, to be performed. Do NOT perform the transfer yet.

If possible, the host should be set not to echo back the characters sent from the PS 300. This is done by entering "SET TERM/NOECHO" under VMS, and "SET/NOECHO=TTnn:" under RSX-11M. If this is not done, the "transfer string" will be the first data included on the SITE.DAT file. If the host cannot be set to noecho, the "transfer string" in the SITE.DAT will only produce a Parser Syntax Error, when the first line in the SITE.DAT is a semicolon.

9. Type <CTRL A>. The Utility> prompt will reappear.

10. Replace the Diagnostic diskette in the floppy diskette drive with the destination diskette.

11. Type

    TRANSFER

    followed by a carriage return. The system will prompt for the name of the file on the host. Enter that name (the name of the file created in Step 2), followed by a carriage return. This is the name that will be substituted for the <fn> of the "transfer string".

12. The system will now prompt for the name of the file as it should appear on the Graphics Firmware work diskette. Type:

    filename

    followed by a carriage return.

13. The next prompt will be for the type of transfer. Type:

    ASCII

    followed by a carriage return.

14. The system may now prompt for the date. Enter the current date. If successive downloads have been performed, the system will remember the date, and no prompt will appear.

15. Periods will be displayed on the terminal or the LEDs of the PS 300 Keyboard (whichever is being used). Wait until the message:

    File successfully transferred.

    appears on the terminal or the LEDs. The file from the host now exists as the highest version of filename on the diskette. This is the version the system will use when the modified diskette is loaded.

16. Type:

    Terminal

    followed by a carriage return. This again attaches the terminal to the host computer. If the terminal has been set to noecho, reset to echo. Sign off the host computer.

17. The downloading procedure is now complete.

CONTENTS

11. INSTALLING HOST-RESIDENT SOFTWARE

This chapter contains the installation instructions for the PS 300 Host-Resident I/O Subroutines (PSIO) and Graphics Support Routines (GSRs). Requirements for both packages are given with the package description in *User Operation and Communication*, in Volume 1 of this documentation set.

11.1 PSIO INSTALLATION UNDER DEC RSX-11M

This section describes the installation of the PS 300 Host-Resident I/O Subroutines under the DEC RSX-11M operating system.

The installation instructions are for systems using either standard asynchronous transmission, or synchronous 56KB transmission with the DEC/DRM-11AE interface.

Under RSX-11M, the DMR11 Drive must be built before the Host-Resident Subroutines can be run on the host. The steps for installing the Driver are found in a later section of this chapter.

The firmware and the installation guide necessary to support the PS 300-DEC/DMR11-AE Interface are available through Evans & Sutherland.

11.1.1 Preparation

The magtape enclosed with this release of the PS 300 Graphics Firmware Version contains the RSX-11 FORTRAN and MACRO source files necessary to Task Build with the user application programs.

The file names and contents are listed below.

- PSIO.FTN - contains the subroutines PSSEND, PSREAD, PSPOLL, PSVECS, PSEXIT, PSETUP, PSCHAR, and PSFIXI.
- RSXPSIO.FTN - contains the second level subroutines that are machine dependent.
- RSXPSSER.MAC - contains the routines for communication to the serial line.
- RSXDMR.MAC - contains the routines for communication to the DMR11-AE.

The process of compiling, task building, and running the FORTRAN and ASSEMBLER routines for running with the user programs is outlined below.

11.1.2 Compiling the FORTRAN Programs

The FORTRAN files should be compiled in the following manner:

```
>F4P PSIO=PSIO
>F4P RSXPSIO=RSXPSIO
```

or

```
>FOR PSIO=PSIO
>FOR RSXPSIO=RSXPSIO
```

using the format

```
user_program-object=user_program_source
```

11.1.3 Assembling the MACRO Code

The MACRO files should be assembled in the following manner:

```
>MAC RSXPSSER=RSXPSSER
>MAC RSXPSDMR=RSXPSDMR
```

using the format

```
user_program-object=user_program_source
```

11.1.4 Task Building (Linking) the Programs

The programs should be task built with the user application program in the following manner:

```
TKB>user_program_task_image=user_program_object,PSIO,
   RSXPSIO,RSXPSSER,RSXPSDMR
TKB>LB:[1,1]F4POTS/LB or LB:[1,1]FOROTS/LB
TKB>/
ENTER OPTIONS:
TKB>UNITS=10
TKB>ASG=serialline_id:m
TKB>ASG=XM:n
TKB>//
```

where m is the logical unit number passed as the first argument in PSETUP and n is the logical unit number passed as the second argument in PSETUP. Refer to the description of the subroutine PSETUP in Chapter 1 for a description of the arguments.

NOTES

1. The explicit statement of a FORTRAN library is not required if the library is part of the system library SYSLIB.

2. F4P under RSX-11M has a facility for virtual array declaration. Do not declare the vector passed to PSVECS as a VIRTUAL array.

3. If the user is running RSX-11M on a system with floating point processor, the /FP switch must be included when task building.

   TKB> TEST/FP = TEST, ...

Example:

```
TKB>CUBES=CUBES,PSIO,RSXPSIO,RSXPSSER,RSXPSDMR, LB:[1,1]F4POTS/LB
TKB>/
ENTER OPTIONS:
TKB>UNITS=10
TKB>ASG=TT2:7 (serialline_id)
TKB>ASG=XM:8 (DMR_id)
TKB>//
```

In this example terminal port TT2: is assigned to logical unit number 7 and the XM device (serial line) is assigned the logical unit number 8. It is assumed that the PS 300 is connected to TT2:.

11.1.5 Execution

To execute a user program task image that has been created by the task builder, submit a RUN command to the RSX-11M operating system as follows:

Before running the program, make sure the "XM" driver is loaded. Run the user application program:

```
RUN user_program_task_image
```

Example: RUN CUBES

11.2 INSTALLING THE DEC/DMR11-AE DRIVER

The DMR11 Driver must be installed before the Host-Resident Subroutines can be run on the host. The steps for installing the driver are as follows.

1. When installing the DMR11 Driver, the user must be logged in under a privilege UIC ([1,54], for example). The RSX Executive must support loadable drivers. If it does not, the RSX Executive must be rebuilt to include this support.

   The following tasks must be installed when the driver is being built: MAC, PIP, SLP, TKB.

   The following RSX-11M files are required to generate the driver:

RSXMC.MAC	Normally located on the System Disk at UIC [1,1]
EXEMC.MLB	Normally located on the System Disk at UIC [1,1]
EXELIB.OLB	Normally located on the System Disk at UIC [1,1]
XMDRV.MAC	Normally located on the RSX-11M Source Disk
RSX11M.STB	Must be located on the System Disk at UIC [1,54]

2. To start the installation, copy the following files from the E&S distribution media to UIC [300,300] on a work disk:

   XMDRVBLD.CMD
   GENXMT.CMD
   EDRSXMC.SLP

3. Now, the command file should be started by typing

   @XXn:[300,300]XMDRVBLD

   where XXn is the physical name of the work disk.

4. Next, type in the physical name of the System Disk and the work disk when prompted by the system. The system will ask for the location of the files RSXMC.MAC, EXEMC.MLB, and EXELIB.OLB. If the file is on the System Disk at UIC [1,1], just type a RETURN to the question. If it is not located at UIC [1,1], type in the physical disk name and UIC where the file is located.

5. The system will now prompt for the location of the file XMDRV.MAC. Respond with the physical disk name and the UIC where the file is located. The files XMDRV.MAC and RSXMC.MAC will now be transferred to the work area.

6. The system will ask if the following equates are present in the file RSXMC.MAC:

   LD$XM=0 (Defining that XMDRV is loadable.)
   X$$M11=1 (Defining that XMDRV controls 1 DMR11)

   If the answer is no, the above equates will be edited into RSXMC.MAC on the work disk.

7. The Driver Symbol Table will now be generated. The system will ask for the Interrupt Level, Interrupt Vector and Device address for the DMR11. The Driver and Symbol Table will then be assembled. After the assembly is complete, the system will ask if there were any errors. An affirmative response will exit the command file. The source of the error will need to be located, and the command file run again.

   A negative response will generate the Task Build file for the driver.

8. The system will prompt for the name of the partition where the driver will be loaded. Normally, this will be DRVPAR. The driver will now be task built, and stored on the System Disk at UIC [1,54].

9. The driver will be loaded, completing the installation, and the command file will exit.

Special Notes

The following notes apply to the use of the PS 300 I/O Subroutines under the DEC/DMR11-AE Interface.

1. If a vector list is to be block normalized, only one call to PSVECS is allowed in defining the list. This restriction is not the case for vector normalized vector lists. It is not imposed on the later case as the normalization occurs on a 'per vector' basis, rather than on a set of vectors. Once the block normalized vectors have been sent to the PS 300, no mechanism is currently available to renormalize them, should subsequent calls to PSVECS require it.

2. Passing a vector count in a call to PSVECS that is larger than the actual vector array declaration will produce a FORTRAN access violation error.

3. The vector list command sent (via PSSEND) to the PS 300, prior to the initial PSVECS call, must agree in type to the vector type parameter passed to PSVECS.

4. The configuration of the DMR Port is setup through Port10. The DMR port uses the clock of Port10, which prevents Port10 from being used when the 56KB Interface is used.

11.3 INSTALLATION UNDER DEC VAX/VMS

This section describes the installation of the PSIOs under VMS with the user programs that call the subroutines.

The installation instructions are for system using either standard asynchronous transmission, or synchronous 56KB transmission with the DEC/DMR-11AE Interface.

The firmware and the installation guide necessary to support the PS 300-DEC/DRM-11AE Interface are available through Evans & Sutherland.

11.3.1 Preparation

The magtape enclosed with this release of the PS 300 Graphics Firmware Version P5.V01 contains the VMS FORTRAN and MACRO source files necessary to LINK with the user application programs.

The file names and contents are listed below.

- PSIO.FOR – contains the subroutines PSSEND, PSREAD, PSPOLL, PSVECS, PSEXIT, PSETUP, PSCHAR, and PSFIXI.
- VAXPSIO.FOR – contains the second level subroutines that are machine dependent.
- VAXPSSER.MAR – contains the routines for communication to the serial line.
- VAXPSDMR.MAR – contains the routines for communication to the DMR11-AE.

The process of compiling, linking, and executing the FORTRAN and ASSEMBLER routines for running with user programs is outlined below.

11.3.2 Compiling the FORTRAN Code

The FORTRAN files should be compiled in the following manner:

```
$FOR PSIO
$FOR VAXPSIO
```

11.3.3 Assembling the MACRO Codes

The MACRO code should be assembled in the following manner:

```
$MACRO VAXPSSER
$MACRO VAXPSDMR
```

11.3.4 Linking the Routines

The routines should be linked with the user program in the following manner:

```
$LINK user_program_object,PSIO,VAXPSIO,VAXPSSER,VAXPSDMR
```

Example:

```
$LINK CUBES,PSIO,VAXPSIO,VAXPSSER,VAXPSDMR
```

11.3.5 Allocating and Assigning the Serial Line or DMR11

The serial line has the logical name PS, and the DMR11 line has the logical name XM. To run the program, the physical name must be assigned to the logical name.

Example:

```
$ALL TTA0:
$ASSIGN TTA0: PS
```

or

```
$ALL XMD0:
ASSIGN XMD0: XM
```

Where TTA0: is the RS-232 serial line connected to Port 1 of the PS 300, and XMD0 is the DMR11 synchronous communication line connected to Port 0 of the PS 300.

Refer to the description of the PSETUP routine in Chapter 1 on selecting the proper communication line at the time of program execution.

11.3.6 Program Execution

To execute a user program that has been created by the linking, submit the following command to the VMS operation system:

```
$RUN user_program_executable_image
```

Example:

```
$RUN CUBES
```

11.4 INSTALLATION OF THE PS 300 I/O SUBROUTINES PACKAGE UNDER THE DEC/DMR11-AE INTERFACE

E&S distributes a special Graphics Firmware diskette to support the DEC/DMR11-AE interface with the PS 300. This diskette must be booted when using the I/O subroutines. To use the 9600 Asynchronous Host Line, a standard Graphics Firmware floppy must be used.

After the hardware installation of the DEC/DMR11-AE interface is complete, DMR11 Driver is automatically installed by autoconfiguration when the VAX/VMX is powered up. Before running the program, allocate and assign the logical name "XM" for the DMR11. For example, if the DMR11 is physical device "XMB0":

```
$ALLOCATE XMB0:
$ASSIGN XMB0: XM
```

Run the application program.

Special Notes

The following notes apply to the use of the PSIOs under the DEC/DMR11-AE Interface.

1. If a vector list is to be block normalized, only one call to PSVECS is allowed in defining the list. This restriction is not the case for vector-normalized vector lists. It is not imposed on the later case as the normalization occurs on a 'per vector' basis, rather than on a set of vectors. Once the block-normalized vectors have been sent to the PS 300, no mechanism is currently available to renormalize them, should subsequent calls to PSVECS require it.

2. Passing a vector count in a call to PSVECS that is larger than the actual vector count will produce a FORTRAN access violation error.

3. The vector list command sent (via PSSEND) to the PS 300, prior to the initial PSVECS call, must agree in type to the vector type parameter passed to PSVECS.

4. The configuration of the DMR port is setup through Port10. The DMR port uses the clock of Port10, which prevents Port10 from being used when the 56KB Interface is used.

11.5 INSTALLATION OF THE FORTRAN GSRs

This section contains brief installation instructions for the DEC/VAX FORTRAN-77 version of the PS 300 Graphics Support Routines (GSRs). The GSRs will compile only under a FORTRAN-77 compiler and are supported under VMS Version 3.2 and higher. PS 300 Graphics Firmware Version P5.V03 or higher is required to run the Graphics Support Routines.

The GSRs are distributed on 1600-bpi magtape. The source files for the DEC VAX/VMS FORTRAN-77 version of the Graphics Support Routines are listed below along with a description of each file.

File Name	Description
GSRF.FOR	Source file for the GSRs.
PROFORLIB.FOR	Source file for the intermediate code between GSRF.FOR AND PROLIB.MAR.
PROIOLIB.MAR	Macro Source file for the low-level I/O subroutines used by the GSRs to communicate with the PS 300.
PROCOMF.FOR	Contains the global definitions of the FORTRAN-77 VAX GSR's and is INCLUDED by GSRF.FOR.
PROFORCOM.FOR	Contains the global definitions for the PROFORLIB.FOR.
PROCONST.FOR	Contains file that may be INCLUDed by the user in an application program. Contains the constant definitions.

The object module library containing ALL of the DEC VAX/VMS FORTRAN-77 GSR subroutines is contained in the file:

GSRF.OLB

To link your program with the DEC VAX/VMS FORTRAN-77 GSRs, enter the following command:

$ LINK <pgm>,<...any additional user object modules...>,GSRF/LIB

The files:

CIRCLEF.FOR

BLKLEVF.FOR

respectively contain the source code of two VAX FORTRAN-77 GSR programs demonstrating some of the subroutine calls. These will need to be compiled and linked with the GSR library.

To recreate the DEC VAX/VMS FORTRAN-77 GSRs from the original source files, the following DCL command sequence should be used:

```
$ FORTRAN GSRF
$ FORTRAN PROFORLIB
$ MACRO  PROIOLIB
$ LIBRARY/CREATE GSRF GSRF,PROFORLIB,PROIOLIB
$
```

General Notes

To read the GSR files off of the accompanying tape, the following DCL command sequence should be used:

```
$ ALLOCATE MTnn:          (MTnn: refers to the physical device name of
$ MOUNT  MTnn: PSIO        the appropriate tape drive unit)
$ COPY Mtnn:*.*;* *
$ DISMOUNT MTnn:
$ DEALLOCATE MTnn:
$
```

The magtape contains many other files that are not related to the Graphics Support Routines. Refer to the file README.TAP for a description of the contents of these other files. READFOR.GSR contains a short description of each of the FORTRAN GSR files.

11.6 INSTALLATION OF THE PASCAL GSRs

This section contains brief installation instructions for the DEC VAX PASCAL V2 version of the Graphics Support Routines (GSRs). The GSRs will compile only under a VAX PASCAL V2 compiler and are supported under VMS Version 3.2 and higher. PS 300 Graphics Firmware Version P5.V03 or higher is required to run the Graphics Support Routines.

The GSRs are distributed on 1600-bpi magtape (E&S #904015-004). The source files for the DEC VAX/VMS PASCAL V2 version of the Graphics Support Routines are listed below along with a description of each file.

File Name	Description
GSRP.PAS	Source file for the GSRs
PROPASLIB.PAS	Source file for the intermediate I/O procedures conceptually residing between GSRP.PAS AND PROIOLIB.MAR
PROIOLIB.MAR	Macro Source file for the low-level I/O procedures used by the GSRs to communicate with the PS 300
PROCONST.PAS	File that should be INCLUDEd by the user in an application program. Contains CONSTant definitions.
PROTYPES.PAS	File that should be INCLUDEd by the user in an application program. Contains TYPE definitions.
PROEXTRN.PAS	File that should be INCLUDEd by the user in an application program. Contains EXTERNal definitions.

The object module library containing ALL of the DEC VAX/VMS PASCAL V2 GSR procedures is contained in the file:

GSRP.OLB

To link your program with the DEC VAX/VMS PASCAL V2 GSRs, enter the following command:

$ LINK <pgm>,<...any additional user object modules...>,GSRP/LIB

The files:

CIRCLEP.PAS

BLKLEVP.PAS

contain the source code of two VAX PASCAL V2 GSR programs demonstrating some of the procedures. These will need to be compiled and linked with the GSR library.

To recreate the DEC VAX/VMS PASCAL V2 GSRs from the original source files, the following DCL command sequence should be used:

```
$ PAS GSRP
$ PAS PROPASLIB
$ MACRO  PROIOLIB
$ LIBRARY/CREATE GSRP GSRP,PROPASLIB,PROIOLIB
$
```

General Notes

To read the GSR files off of the accompanying tape, the following DCL command sequence should be used:

```
$ ALLOCATE MTnn:          (MTnn: refers to the physical device
$ MOUNT/ MTnn: PSIO       name of the appropriate tape drive unit)
$ COPY Mtnn:*.*;* *
$ DISMOUNT MTnn:
$ DEALLOCATE MTnn:
$
```

The magtape contains many other files that are not related to the Graphics Support Routines. Refer to the file README.TAP for a description of the contents of these other files. The file READPAS.GSR contains a short description of the Pascal GSR files.

CONTENTS

TABLES

FIGURES

12. INTERACTIVE DEVICES

This chapter describes how the PS 300 Interactive Devices hardware works. Each section contains the hardware functional description for a specific device, and information related to maintaining these devices.

Included are descriptions of the standard and IBM versions of the PS 300 Keyboards, the Data Tablet hardware, operating modes, and power requirements, the Data Tablet/PS 300 Interface, cables and connectors, the PS 300 Function Buttons power source, RS-449 serial communications interface, hardware, light driver interface, switch driver interface, and communications protocol, and the PS 300 Control Dials hardware, power, and operating modes.

12.1 PS 300 KEYBOARD

The PS 300 Keyboard has two main functions:

1. The generation and transmission of ASCII displayable characters, ASCII control characters, and PS 300 system sequences. These data are transmitted serially to the Graphics Control Processor (GCP), the controlling system processor that is located in the PS 300 Control Unit; the transmitted data may ultimately specify displayed characters, commands, menu/table selections, etc.

2. The display of full-line or segmented alphanumeric messages on an optional 1 X 96-character LED array. These displayed characters most often function as labels for the keyboard's 12 user-programmable function keys. The LED characters may also be used "in tandem" to present a single message up to 96 characters long; this mode is typically used by a test technician for diagnostic purposes.

12.1.1 Physical Configuration

The PS 300 Keyboard (see Figure 12-1) is a modular unit that connects to the system with a single interface cable. The main processor in the keyboard controls LED displays and I/O data transmissions. The keys are mounted on a PC card that contains the microprocessor that decodes the key locations and produces an ASCII code that corresponds to the pressed key or keys. Like the other interactive devices, the keyboard is microprocessor-controlled to provide limited local processing capabilities.

The keyboard's enclosure is made of LEXAN* resin and structural foam. It is a simple, two-piece unit that accommodates a 95-key Micro Switch keyboard, three 32-character LED display cards, and a Keyboard Interface card. A 2.5-inch speaker mounts on the bottom half of the keyboard enclosure assembly. Figure 12-1 shows the layout of the keyboard.

CAUTION

Care should be taken when cleaning the PS 300 Keyboards and Control Dials. These devices have Lexan covers that accumulate static electricity when buffed. To safely clean these devices, use a damp cloth. DO NOT VIGOROUSLY BUFF THE CONTROL DIALS OR KEYBOARDS.

*A registered trademark of the General Electric Company

Figure 12-1. PS 300 Keyboard Layout

The assembled keyboard measures 21.1 inches (53.6 cm) long by 8.25 inches (20.9 cm) deep. The keyboard stands 3.5 inches (8.9 cm) high on four rubber feet.

The only operator controls located on the keyboard are the 95 keys used for data input.

LED Indicators

The LEDs are configured in a single row above the twelve keyboard function keys. They are arranged in twelve 8-character groups that are numbered from left to right, 0 through 11. Each LED group may serve as a label for its associated function key, or all LED characters may be used together to display a single message. A space of one character separates each 8-character LED group from the next.

Each 4-character alphanumeric LED display is a separate 12-pin, clear plastic, encapsulated package. Each character in the package has a separate immersion lens that provides high brightness and low off-angle distortion (to $\pm$ 50 degrees).

The system's audible alarm sounds through the 2.5-inch speaker that mounts on the bottom half of the enclosure.

Interface Cable

An 8-conductor, flexible cable with locking modular plugs connects the PS 300 Keyboard to the Data Concentrator at the rear of the Display. The cable is similar in function and appearance to a standard telephone "flex" cord. The cable may be stretched to permit many different work station arrangements. The modular plugs are identical, allowing the cable to be connected in either direction.

12.1.2 KEYBOARD OPERATION

The PS 300 Keyboard is a dual-purpose interactive device that:

- Allows the operator to input ASCII characters and other sequences to the Graphics Control Processor by means of a typewriter-like keyboard.
- Allows the system to communicate alphanumeric messages to the operator by means of an optional multi-character LED display.

Both aspects of keyboard operation are discussed in detail in the following paragraphs.

Data Entry

The PS 300 Keyboard features a TTY/typewriter-compatible layout that makes data entry fast and easy. All keys except REPEAT and CAPS LOCK are momentary-closure devices. REPEAT and CAPS LOCK toggle on (key down) and off (key up) like a standard typewriter's LOCK key.

The 95 keys fall into eight general categories.

1. Keyboard Function Control
2. Alphabetic
3. Standard Numeric
4. Special Character
5. Terminal Function
6. PS 300 Function
7. Numeric/Application Mode
8. PS 300 Device Control

The Keyboard Function Control keys are used to modify the codes produced by other keys. In this way, characters are defined as uppercase, lowercase, control, etc. No codes are transmitted when these keys are depressed individually or in combination with each other.

The Alphabetic, Standard Numeric, Special Character, and Terminal Function keys all generate standard ASCII characters. Depressing any of these keys alone or in combination with SHIFT and/or CTRL causes 7-bit character codes or control codes to be transmitted from the keyboard to the Graphics Control Processor.

The PS 300 Function Control, Numeric/Application Mode, and Device Control keys are system-oriented. Depressing any of these keys alone or in combination with SHIFT and/or CTRL causes special two-byte PS 300 system sequences to be generated and transmitted to the GCP. This process is described later.

Keyboard Function Control Keys

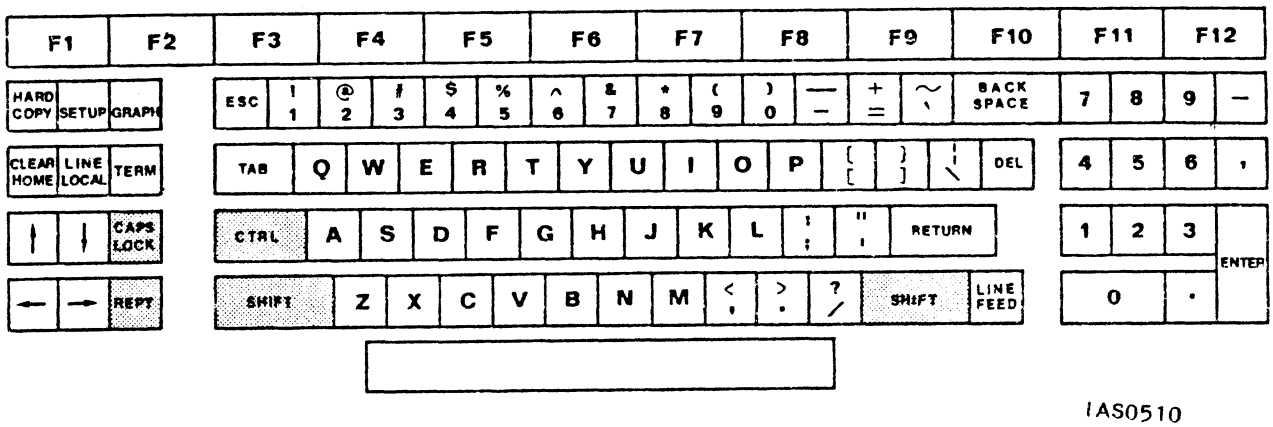


Figure 12-2. Keyboard Function Control Keys

The Keyboard Function Control keys, shown in gray in Figure 12-2, are unencoded, local controls. No codes are transmitted when these keys are struck individually or in combination with each other. The Keyboard Function Control keys are used to modify the codes transmitted by other keys.

When either SHIFT key is depressed simultaneously with a displayable character key, the uppercase code for that key is generated. If the key does not have an uppercase function, the SHIFT key is ignored. For example, striking the A key causes the code B'01100001' for the character a to be transmitted; the sequence SHIFT A causes the code B'01000001 for the character A to be transmitted. Bit 6 is forced low to define an uppercase character.

When CTRL is depressed simultaneously with one of keys A-Z (uppercase only), the space bar, or the Special Character keys \, [,], \, or ?, an ASCII control code is generated. For example, the CTRL Z keyboard sequence causes the code B'00011010' to be generated. The only difference between this code and that for Z (B'01011010') is that bit 7 is forced low to define the control code.

When the SHIFT and CTRL keys are depressed simultaneously, the shift function is selected in most cases. The only exceptions occur with the \ and ? keys. The SHIFT CTRL \ sequence causes the control character RS (B'00011110') to be transmitted. The SHIFT CTRL ? sequence causes the control character US (B'00011111') to be transmitted.

When the REPEAT key is locked down, the auto-repeat feature is enabled on all keys except: F1 - F12, HARD COPY,SETUP, GRAPH, CLEAR/HOME, LINE/LOCAL, TERM, CAPS LOCK, CTRL, SHIFT (both keys), RETURN, and all numeric pad keys. When any other key is held down with the keyboard in auto repeat mode, repeated character transmission occurs. The initial rate is less than 2 Hz, but this increases to about 11 Hz in a little less than two seconds. Striking the REPEAT key a second time causes it to release upwards, cancelling the auto repeat feature.

Depressing the CAPS LOCK key causes it to assume a locked-down position, asserting the "caps lock" function. This is actually a limited shift operation that applies to the alphabetic (A-Z) keys only. Alphabetic keys struck while the keyboard is in "caps lock" mode generate uppercase characters. Depressing the CAPS LOCK a second time causes it to release upward, cancelling the "caps lock" mode.

Alphabetic Keys

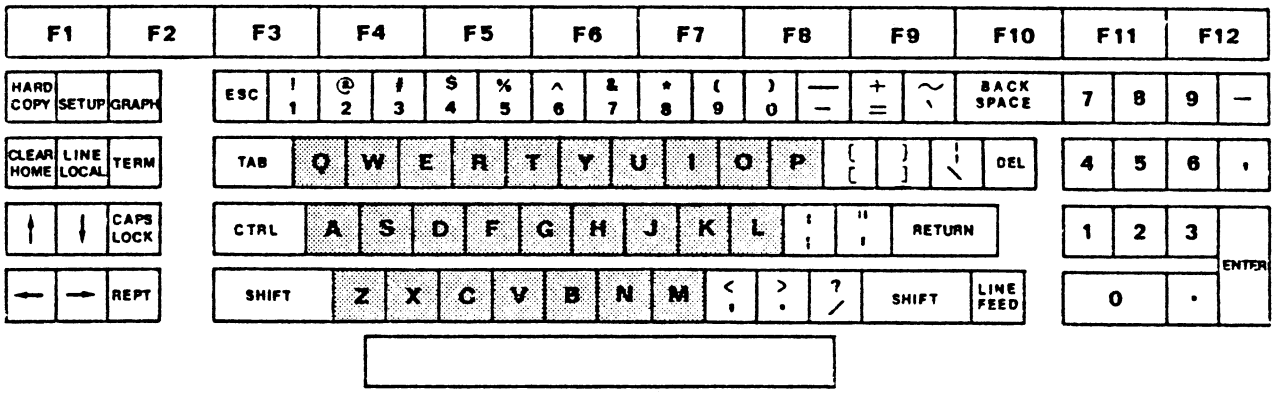


Figure 12-3. Keyboard Alphabetic Keys

The Alphabetic Keys, shown in gray in Figure 12-3, are used to produce uppercase and lowercase ASCII displayable character codes, and ASCII control codes. Table 12-1 shows the code and character produced when each key is struck alone, with the SHIFT key, or with the CTRL key.

Table 12-1. Alphabetic Key Codes

KEY LABEL	KEY CODE	ALONE CHARACTER	KEY + SHIFT CODE	SHIFT CHARACTER	KEY + CTRL CODE	CTRL CHARACTER
A	X'61' O'141'	a	X'41' O'101'	A	X'01' O'001'	SOH
B	X'62' O'142'	b	X'42' O'102'	B	X'02' O'002'	STX
C	X'63' O'143'	c	X'43' O'103'	C	X'03' O'003'	ETX
D	X'64' O'144'	d	X'44' O'104'	D	X'04' O'004'	EOT
E	X'65' O'145'	e	X'45' O'105'	E	X'45' O'005'	ENQ
F	X'66' O'146'	f	X'46' O'106'	F	X'06' O'006'	ACK
G	X'67' O'147'	g	X'47' O'107'	G	X'07' O'007'	BEL
H	X'68' O'150'	h	X'48' O'110'	H	X'08' O'010'	BS
I	X'69' O'151'	i	X'49' O'111'	I	X'09' O'011'	HT
J	X'6A' O'152'	j	X'4A' O'112'	J	X'0A' O'012'	LF
K	X'6B' O'153'	k	X'4B' O'113'	K	X'0B' O'013'	VT
L	X'6C' O'154'	l	X'4C' O'114'	L	X'0C' O'014'	FF
M	X'6D' O'155'	m	X'4D' O'115'	M	X'0D' O'015'	CR
N	X'6E' O'156'	n	X'4E' O'116'	N	X'0E' O'016'	SO

Table 12-1. Alphabetic Key Codes - Continued

KEY LABEL	KEY CODE	ALONE CHARACTER	KEY + SHIFT CODE	SHIFT CHARACTER	KEY + CTRL CODE	CTRL CHARACTER
O	X'6F' O'157'	o	X'4F' O'117'	O	X'0F' O'017'	SI
P	X'70' O'160'	p	X'50' O'120'	P	X'10' O'020'	DLE
Q	X'71' O'161'	q	X'51' O'121'	Q	X'11' O'021'	DC1
R	X'72' O'162'	r	X'52' O'122'	R	X'12' O'022'	DC2
S	X'73' O'163'	s	X'53' O'123'	S	X'13' O'023'	DC3
T	X'74' O'164'	t	X'54' O'124'	T	X'14' O'024'	DC4
U	X'75' O'165'	u	X'55' O'125'	U	X'15' O'025'	NAK
V	X'76' O'166'	v	X'56' O'126'	V	X'16' O'026'	SYN
W	X'77' O'167'	w	X'57' O'127'	W	X'17' O'027'	ETB
X	X'78' O'170'	x	X'58' O'130'	X	X'18' O'030'	CAN
Y	X'79' O'171'	y	X'59' O'131'	Y	X'19' O'031'	EM
Z	X'7A' O'172'	z	X'5A' O'132'	Z	X'1A' O'032'	SUB

Standard Numeric Keys

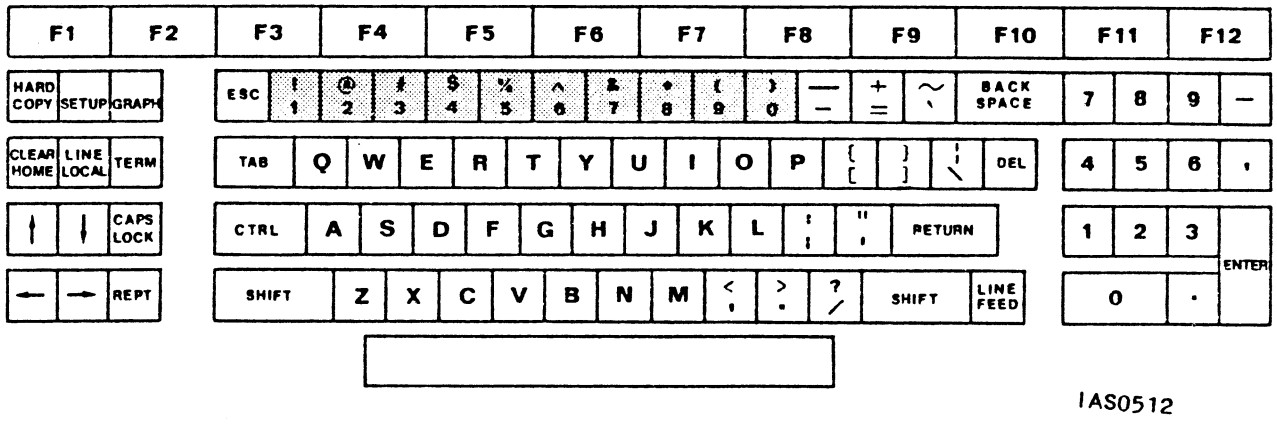


Figure 12-4. Keyboard Standard Numeric Keys

The shiftable Standard Numeric keys, shown in gray in Figure 12-4, are similar to the shiftable numeric/symbol keys that appear on a typewriter; they generate ASCII displayable numbers and symbols. The CTRL key is ignored when used with these keys. Table 12-2 shows the code and character produced when each key is struck alone, with the SHIFT key, or with the CTRL key.

Table 12-1. Alphabetic Key Codes - Continued

KEY LABEL	KEY CODE	ALONE CHARACTER	KEY + SHIFT CODE	SHIFT CHARACTER	KEY + CTRL CODE	CTRL CHARACTER
O	X'6F' O'157'	o	X'4F' O'117'	O	X'0F' O'017'	SI
P	X'70' O'160'	p	X'50' O'120'	P	X'10' O'020'	DLE
Q	X'71' O'161'	q	X'51' O'121'	Q	X'11' O'021'	DC1
R	X'72' O'162'	r	X'52' O'122'	R	X'12' O'022'	DC2
S	X'73' O'163'	s	X'53' O'123'	S	X'13' O'023'	DC3
T	X'74' O'164'	t	X'54' O'124'	T	X'14' O'024'	DC4
U	X'75' O'165'	u	X'55' O'125'	U	X'15' O'025'	NAK
V	X'76' O'166'	v	X'56' O'126'	V	X'16' O'026'	SYN
W	X'77' O'167'	w	X'57' O'127'	W	X'17' O'027'	ETB
X	X'78' O'170'	x	X'58' O'130'	X	X'18' O'030'	CAN
Y	X'79' O'171'	y	X'59' O'131'	Y	X'19' O'031'	EM
Z	X'7A' O'172'	z	X'5A' O'132'	Z	X'1A' O'032'	SUB

Standard Numeric Keys

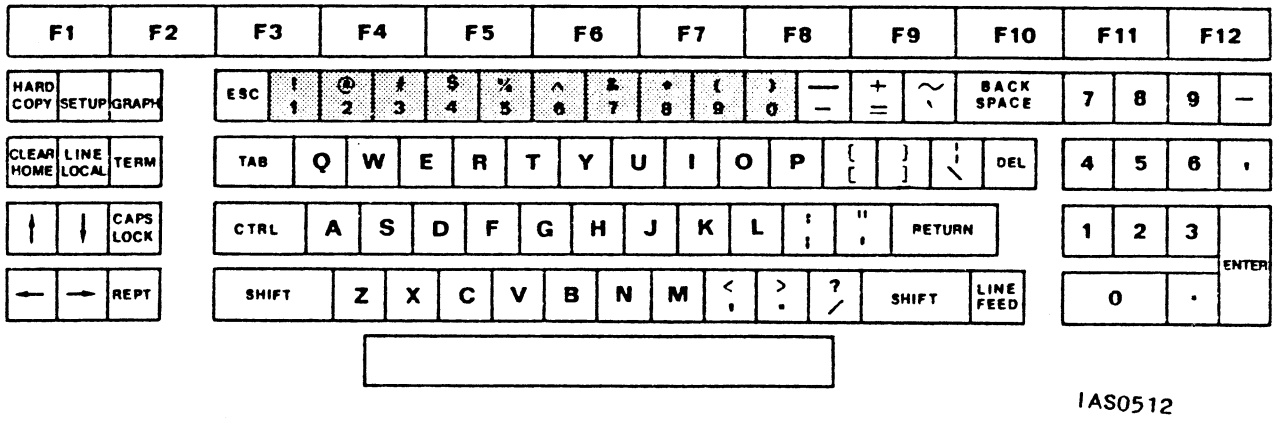


Figure 12-4. Keyboard Standard Numeric Keys

The shiftable Standard Numeric keys, shown in gray in Figure 12-4, are similar to the shiftable numeric/symbol keys that appear on a typewriter; they generate ASCII displayable numbers and symbols. The CTRL key is ignored when used with these keys. Table 12-2 shows the code and character produced when each key is struck alone, with the SHIFT key, or with the CTRL key.

Table 12-2. Standard Numeric Key Codes

KEY LABEL	KEY CODE	ALONE CHARACTER	KEY + SHIFT CODE	SHIFT CHARACTER	KEY + CTRL CODE	CTRL CHARACTER
0	X'30' O'060'	0	X'29' O'051'	)	X'30' O'060'	0
1	X'31' O'061'	1	X'21' O'041'	!	X'30' O'061'	1
2	X'32' O'062'	2	X'40' O'100'	@	X'32' O'062'	2
3	X'33' O'063'	3	X'23' O'043'	#	X'33 O'063'	3
4	X'34' O'064'	4	X'24' O'044'	$	X'34' O'064'	4
5	X'35' O'065'	5	X'25' O'045'	%	X'35' O'065'	5
6	X'36' O'066'	6	X'5E' O'136'		X'36' O'066'	6
7	X'37' O'067'	7	X'26' O'046'	&	X'37' O'067'	7
8	X'38' O'070'	8	X'2A' O'052'	*	X'38' O'070'	8
9	X'39' O'071'	9	X'28' O'050'	(	X'39' O'071'	9

Special Character Keys

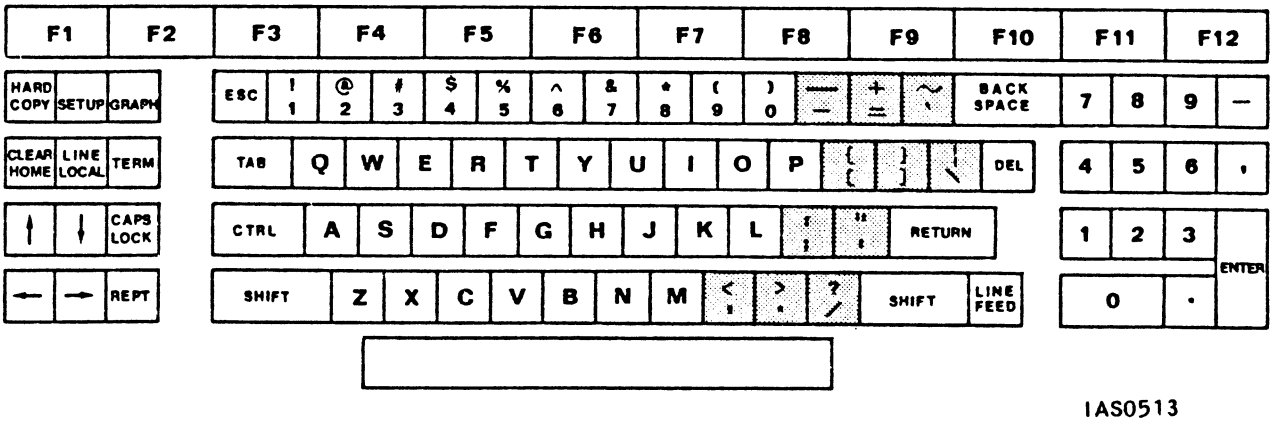


Figure 12-5. Keyboard Special Character Keys

The shiftable Special Character keys, shown in gray in Figure 12-5, are used to produce both ASCII displayable characters and ASCII control characters. Table 12-3 shows the codes and characters produced when these keys are activated alone, with the SHIFT key, and with the CTRL key. Note the varying response given to the CTRL key; in some instances, the unshifted key character is produced. In other cases, a control character is generated. In two cases, O'036' and O'037', both the SHIFT and CTRL keys must be used with the Special Character key to produce the control code shown in Table 12-3.

Table 12-3. Special Character Key Codes

KEY LABEL	KEY ALONE		KEY + SHIFT		KEY + CTRL	
	CODE	CHARACTER	CODE	CHARACTER	CODE	CHARACTER
_ -	X'2D' O'055'	- (minus)	X'5F' O'137'	(underline)	X'2D' O'055'	- (minus)
+ =	X'3D' O'075'	=	X'2B' O'053'	+	X'3D' O'075'	=
\ ~	X'60' O'140'	\	X'7E' O'176'	~	X'1E' O'036'	RS*
{ [	X'5B' O'133'	[	X'7B' O'173'	{	X'1B' O'033'	ESC
}]	X'5D' O'135'	]	X'7D' O'175'	}	X'1D' O'035'	GS
¦ \	X'5C' O'134'	\	X'7C' O'174'	¦	X'1C' O'034'	FS
: ;	X'3B' O'073'	;	X'3A' O'072'	:	X'3B' O'073'	;
" '	X'27' O'047'	'	X'22' O'042'	"	X'27' O'047'	'
< ,	X'2C' O'054'	,	X'3C' O'074'	<	X'2C' O'054'	,
> .	X'2E' O'056'	.	X'3E' O'076'	>	X'2E' O'056'	.
? /	X'2F' O'057'	/	X'3F' O'077'	?	X'1F' O'037'	US*

*These control codes may also be produced by depressing both SHIFT and CTRL in conjunction with the indicated key.

Terminal Function Keys

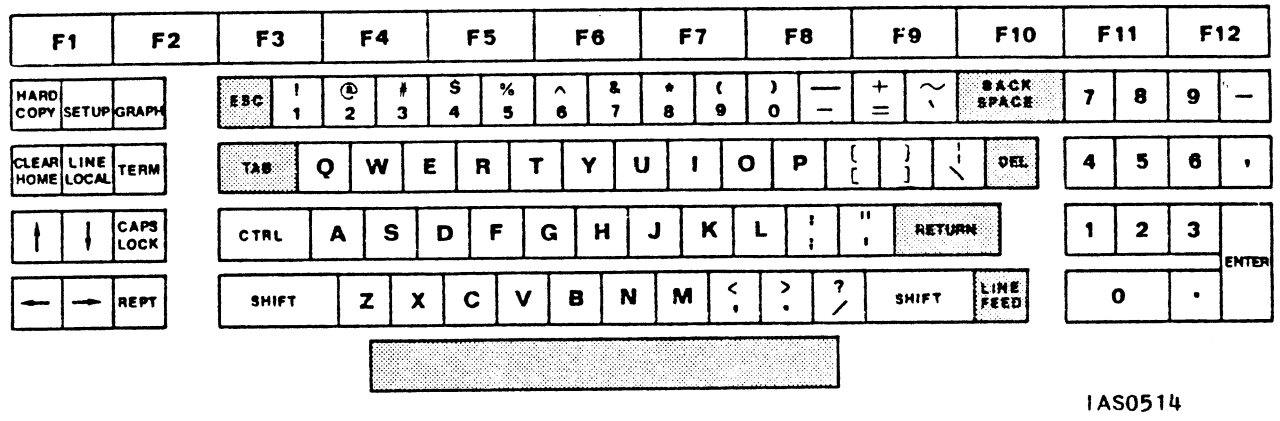


Figure 12-6. Keyboard Terminal Function Keys

The Terminal Function keys, shown in gray in Figure 12-6, produce codes used by a typical video display terminal. These keys enable an operator to generate any commonly used terminal control character with a single keystroke. The codes produced by these keys are identical to those generated by the conventional two-key control sequences described in Table 12-4.

The SHIFT and CTRL keys have no effect on the codes produced by the Terminal Function keys, except for the CTRL Space Bar sequence that generates an ASCII NUL character.

Table 12-4 lists the codes and characters generated by the Terminal Function keys.

Table 12-4. Terminal Function Key Codes

KEY LABEL	KEY CODE	ALONE CHARACTER	KEY + SHIFT CODE	SHIFT CHARACTER	KEY + CTRL CODE	CTRL CHARACTER
BACK SPACE	X'08' O'010'	BS	X'08' O'010'	BS	X'08' O'010'	BS
DEL	X'7F' O'177'	DEL	X'7F' O'177'	DEL	X'7F' O'177'	DEL
RETURN	X'0D' O'015'	CR	X'0D' O'015'	CR	X'0D' O'015'	CR
LINE FEED	X'0A' O'012'	LF	X'0A' O'012'	LF	X'0A' O'012'	LF
ESC	X'1B' O'033'	ESC	X'1B' O'033'	ESC	X'1B' O'033'	ESC
TAB	X'09' O'011'	HT	X'09' O'011'	HT	X'09' O'011'	HT
(none; space bar)	X'20' O'040'	(space)	X'20' O'040'	(space)	X'00' O'000'	NUL

PS 300 Function Keys

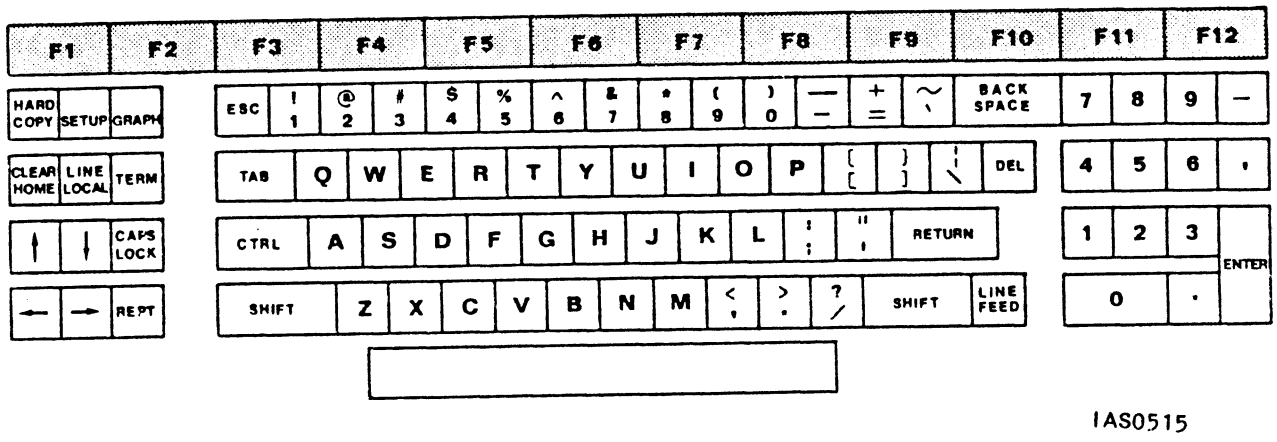


Figure 12-7. Keyboard PS 300 Function Keys

The PS 300 Function keys, shown in gray in Figure 12-7, are used to transmit special 2-byte system sequences to the Graphics Control Processor. These keys can be used individually or in combination with SHIFT and/or CTRL. When a keyboard entry is made, the following events occur:

1. An untransmitted, 8-bit code is initially produced. This code consists of a 7-bit ASCII character that is flagged with an asserted bit eight.

2. The high eighth bit informs the Keyboard Interface processor that the code is a special PS 300 system sequence that must be modified before it can be transmitted serially to the GCP. The processor accordingly converts the 8-bit code to the following 2-byte format.

| CTRL V | CODE |

These two bytes are transmitted to the GCP as the Function key is depressed. The second byte is the original code with bit eight cleared--an "official" ASCII character.

Exactly the same transmitted codes can be generated by entering CTRL SHIFT V, followed by an appropriate alphanumeric or symbol key alone or in combination with SHIFT and/or CTRL. Since most operators prefer a one-key entry, the Function keys are provided as a convenience feature.

Table 12-5 illustrates the codes produced by each Function key as it is used individually or in combination with SHIFT and/or CTRL. Each transmitted code is preceded by CTRL V (B'0010110').

Table 12-5. PS 300 Function Key Codes

Note: All codes are preceded by CTRL V (B'0010110')

KEY LABEL	KEY ALONE		KEY + SHIFT		KEY + CTRL	
	CODE	CHARACTER	CODE	CHARACTER	CODE	CHARACTER
F1	X'61 O'141'	a	X'41' O'101'	A	X'01' O'001'	SOH
F2	X'62 O'142'	b	X'42' O'102'	B	X'02' O'002'	STX
F3	X'63' O'143'	c	X'43' O'103'	C	X'03' O'003'	ETX
F4	X'64' O'144'	d	X'44' O'104'	D	X'04' O'004'	EOT
F5	X'65' O'145'	e	X'45' O'105'	E	X'05' O'005'	ENQ
F6	X'66' O'146'	f	X'46' O'106'	F	X'06' O'006'	ACK
F7	X'67' O'147'	g	X'47' O'107'	G	X'07' O'007'	BEL
F8	X'68' O'150'	h	X'48' O'110'	H	X'08' O'010'	BS
F9	X'69' O'151'	i	X'49' O'111'	I	X'09' O'011	HT
F10	X'6A' O'152'	j	X'4A' O'112'	J	X'0A' O'012'	LF
F11	X'6B' O'153'	k	X'4B' O'113'	K	X'0B' O'013'	VT
F12	X'6C' O'154'	l	X'4C' O'114'	L	X'0C' O'014'	FF

Numeric/Application Mode Keys

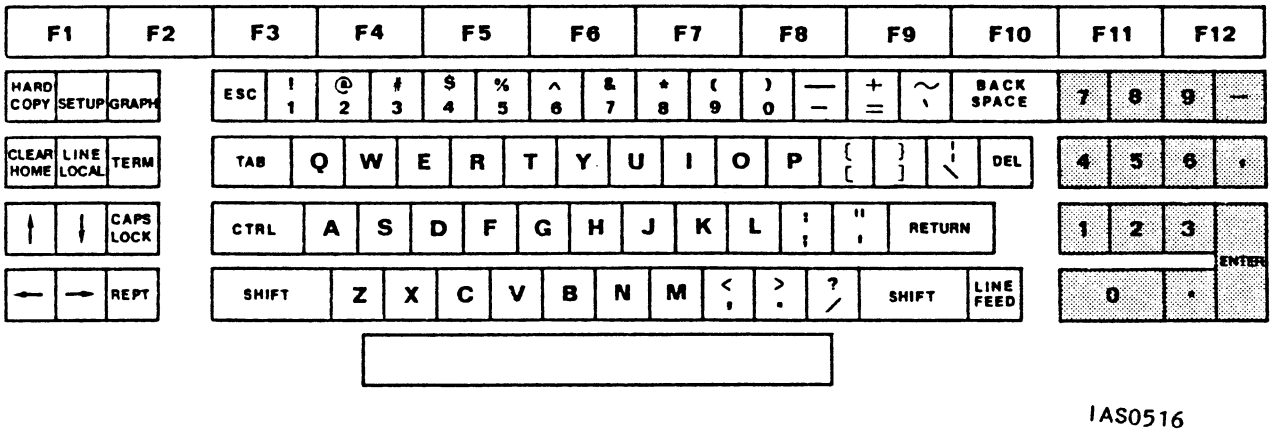


Figure 12-8. Keyboard Numeric/Application Mode Keys

The Numeric/Application Mode keys, shown in gray in Figure 12-8, generate special 2-byte PS 300 system sequences similar to those produced by the PS 300 Function keys.

Neither SHIFT nor CTRL affects the ENTER key, and no codes are modified by the CTRL key.

Any code generated by a Numeric/Application Mode key may be duplicated by entering CTRL SHIFT V, followed by the appropriate displayable character or control character.

Table 12-6 illustrates the codes and characters produced by the Numeric/Application Mode keys.

Table 12-6. Numeric/Application Mode Key Codes

KEY LABEL	KEY CODE	ALONE CHARACTER	KEY + SHIFT CODE	SHIFT CHARACTER	KEY + CTRL CODE	CTRL CHARACTER
0	X'30' O'060'	0	X'29' O'051'	)	X'30' O'060'	0
1	X'31' O'061'	1	X'21' O'041'	!	X'31' O'061'	1
2	X'32' O'062'	2	X'40' O'100'	@	X'32 O'062'	2
3	X'33' O'063'	3	X'23' O'043'	#	X'33' O'063'	3
4	X'34' O'064'	4	X'24' O'044'	$	X'34' O'064'	4
5	X'35' O'065'	5	X'25' O'045'	%	X'35' O'065'	5
6	X'36' O'066'	6	X'5E' O'136'		X'36' O'066'	6
7	X'37' O'067'	7	X'26' O'046'	&	X'37' O'067'	7
8	X'38' O'070'	8	X'2A' O'052'	*	X'38' O'070'	8
9	X'39' O'071'	9	X'28' O'050'	(	X'39' O'071'	9
.	X'2E' O'056'	.	X'3E' O'076'	>	X'2E' O'056'	.
,	X'2C' O'054'	,	X'3C' O'074'	<	X'2C' O'054'	,
-	X'2D' O'055'	(minus) -	X'5F' O'137'	(underline) _	X'2D' O'055'	-
ENTER	X'0D' O'015'	CR	X'0D' O'015'	CR	X'0D' O'015'	CR

PS 300 Device Control Keys

The Device Control keys, shown in gray in Figure 12-9, generate two-byte sequences similar to those described in the two previous sections. The codes produced by these keys are modified by SHIFT and CTRL as shown in Table 12-7.

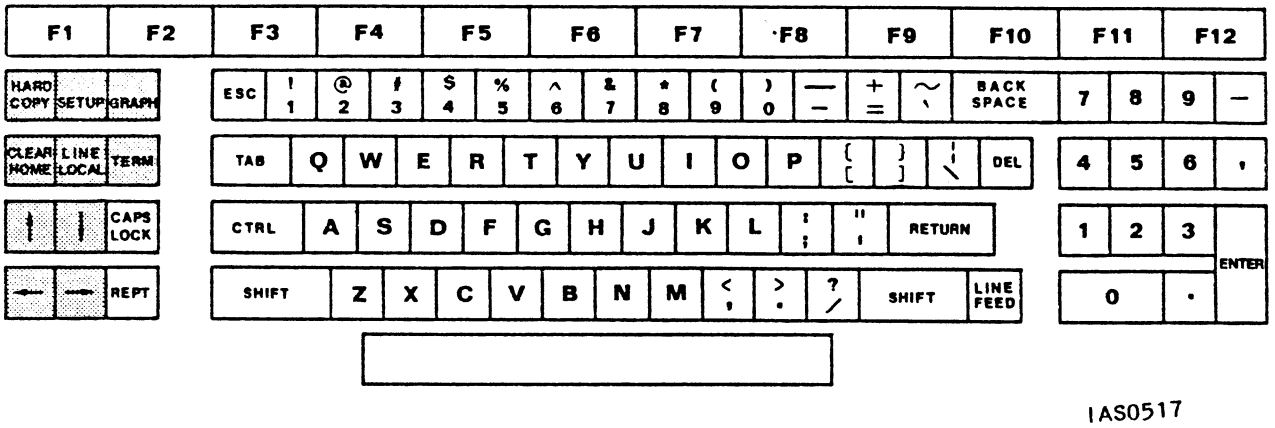


Figure 12-9. PS 300 Keyboard Device Control Keys

Any code generated by a Device Control key may also be produced by entering CTRL SHIFT V, followed by the appropriate displayable character or control character.

Table 12-7. PS 300 Device Control Key Codes

Note: All codes are preceded by CTRL V (B'0010110')

KEY LABEL	KEY CODE	ALONE CHARACTER	KEY + SHIFT CODE	SHIFT CHARACTER	KEY + CTRL CODE	CTRL CHARACTER
HARD COPY	X'6E' O'156'	n	X'4E' O'116'	N	X'0E' O'016'	SO
SET UP	X'6F' O'157'	o	X'4F' O'117'	O	X'0F' O'017'	SI
GRAPH	X'70' O'160'	p	X'50' O'120'	P	X'10' O'020'	DLE
CLEAR HOME	X'71' O'161'	q	X'51' O'121'	Q	X'11' O'021'	DC1
LINE LOCAL	X'72' O'162'	r	X'52' O'122'	R	X'12' O'022'	DC2
TERM	X'73' O'163'	s	X'53' O'123'	S	X'13' O'023'	DC3
<—	X'77' O'167'	w	X'57' O'127'	W	X'17' O'027'	ETB
—>	X'78' O'170'	x	X'58' O'130'	X	X'18' O'030'	CAN
↑	X'79' O'171'	y	X'59' O'131'	Y	X'19' O'031'	EM
↑	X'7A' O'172'	z	X'5A' O'132'	Z	X'1A' O'032'	SUB

The Cursor Up key becomes Scroll Up when shifted.
The Cursor Down key becomes Scroll Down when shifted.

12.1.3 OPTIONAL KEYBOARD LED DISPLAY

The Keyboard LED Display will recognize and display the following ASCII characters:

!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[]¢_

In addition to the above characters, Control E, Control G, Control V, backspace, delete, carriage return, space, and lowercase alphabetic characters are recognized.

Lowercase alphabetic characters are converted to uppercase and displayed. Control E causes the keyboard to send the following message to the PS 300:

KBxxxy

where xxx is the PROM Version Number in the keyboard. If the keyboard has the optional LED Displays, y is "D". If it does not, y is "N" and is sent to the PS 300.

Control G generates a bell tone. Control V, backspace, delete, and carriage return are used as described below. Any other characters are ignored.

Keyboard Display Modes

The keyboard display operates in two modes:

1. Line Mode
2. Label Mode

Optional Keyboard LED Line Mode

In Line mode, the LEDs fill from left to right as characters for display are received over the serial port. A left-justified line up to 96 characters long (including spaces) can be displayed.

The delete and backspace characters are processed only in Line mode.

The backspace character (BS) causes the entire display to logically move left one LED display position.

The delete character (DEL) causes the most recently entered character to be deleted.

All data transmitted to the LEDs for display in Line mode must be terminated with a return character (CR).

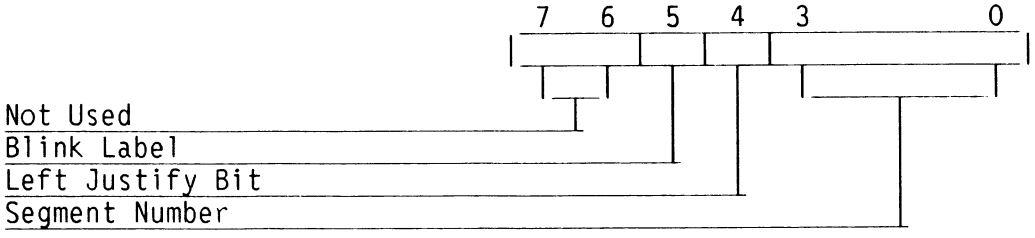
After a return character (CR) is entered, the display is cleared when the next valid character is received. The received character is output to the leftmost LED character, and the LEDs are filled left to right as before.

Keyboard LED Segment Mode

Segment mode is used to provide a descriptive label for each Function key. The serial data input to the keyboard for this purpose must conform to the following format:

| CTRL V | Label Parameter Byte | 0 to 8 Characters | CR |

The Label Parameter byte specifies blinking, left justification, and LED segment number; its format is as follows:



When the Blink Label bit is 1, the characters in the label location (0 - 11) specified in bits 0-3 blinks. When this bit is 0, the segment does not blink. To blink or unblink an existing label, it is only necessary to send CTRL V, the Label Parameter Byte, and a CR.

When the left justify bit is 1, the function key label is left-justified in the specified label location (0 -11); spaces are placed in any unused characters. When this bit is 0, the label is automatically centered in the segment location.

The segment number bits specify label location. Label 0 is the leftmost 8-character LED segment, and Label 11 is the segment on the far right of the LED array.

The "0 to 8 Characters" specified in the above label format constitute actual ASCII characters to display. The CR (O'015') is a required terminator that must appear following each LED label string.

To describe segment mode, examples of function key labels and the data required to produce and modify them are provided below.

1. To center an "unblinking" label X AXIS over key F6 the following hexadecimal string is used:

Byte	Meaning
X'16'	CTRL V
X'05'	Don't Blink; Center; use Segment 5
X'58'	X
X'20'	Space
X'41'	A
X'58'	X
X'49'	I
X'53'	S
X'0D'	CR

2. To make the existing label blink, the following hexadecimal string is used:

Byte	Meaning
X'16'	CTRL V
X'25'	Blink; Center; use Segment 5
X'0D'	CR

3. The following string is used to "unblink" the existing label:

Byte	Meaning
X'16'	CTRL V
X'05'	Don't Blink; Center; use Segment 5
X'0D'	CR

4. To code the label Y TRANS for presentation over key F12 with the label left-justified and blinking, use the following hexadecimal string:

Byte	Meaning
X'16'	CTRL V
X'3B'	Blink; Left-justify; Use Segment 11
X'59'	Y
X'20'	Space
X'54'	T
X'52'	R
X'41'	A
X'4E'	N
X'53'	S
X'0D'	CR

12.1.4 Communications Interface

The keyboard communicates with the GCP through the Data Concentrator using an on-card RS-449 differential line receiver and differential line driver. The keyboard operates at 2400 baud.

12.2 PS 300 DATA TABLET

The PS 300 Data Tablet in Figure 12-10 transforms graphic information into digital data suitable for transmission to the Graphics Control Processor (GCP). The data tablet uses a stylus or a four-button cursor to identify coordinates on the data tablet. The cursor (an X) moves around the screen in response to movement of the stylus across the surface of the data tablet.

The data tablet operates on the magnetostrictive principle. Current is pulsed along a send wire that lies perpendicular to a mesh of magnetostrictive wires laid on a substrate beneath the tablet writing surface. This current pulse changes the dimensions of the magnetostrictive material and a strain wave propagates simultaneously down all the wires of the axis in one direction. The stylus and cursor contain "receive" coils that sense the passing of the strain wave. A binary counter in the control unit times the delay required for the strain wave to reach the receive coil. An 8-bit microprocessor formats this binary count for output as X and Y coordinate data. The data tablet uses an MM5303 serial communications device to receive serial data and convert it to parallel data for the microprocessor and to receive parallel data from the microprocessor that it converts to serial data for transmission to the PS 300.

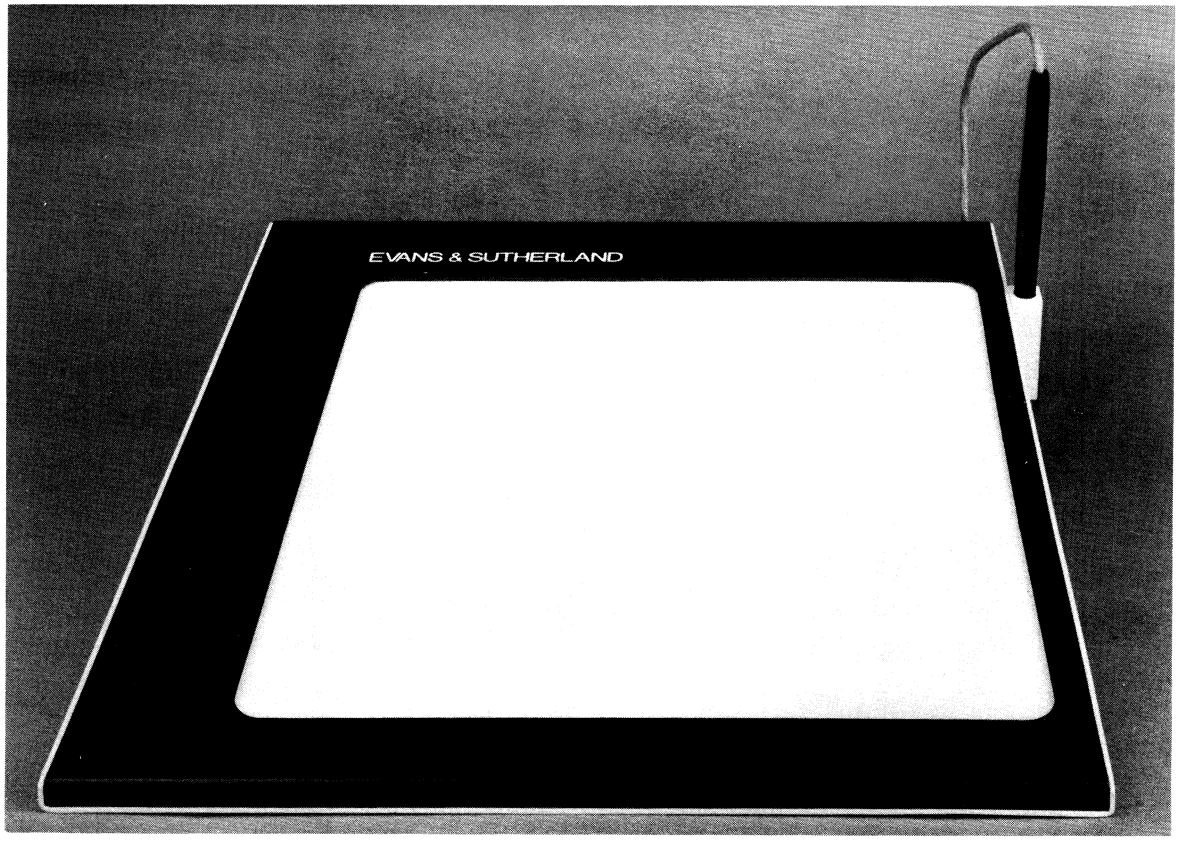


Figure 12-10. The PS 300 Data Tablet

12.2.1 Operating Modes

Data tablet modes and sampling rates can be controlled externally under program control or internally by switches on the logic card. The positions of the internal switch determine the power-up mode and sampling rate. The following operating modes are available:

- Point Mode

  Pressing the stylus on the tablet or pressing a button on the cursor outputs one X, Y coordinate pair (sample) in the appropriate format.

- Stream Mode

  X, Y coordinate pairs (samples) are generated continuously at the selected sampling rate when the stylus or cursor is near the active area of the tablet. Pressing the stylus to the tablet or depressing a button on the cursor puts the flag character (F) bit in the output string. (See TABLETIN Function in the *Command Summary*.)

- Switched Stream Mode

  Pressing the stylus or a button on the cursor continuously outputs X, Y coordinate pairs at the selected sampling rate until the stylus is lifted or the button is released.

The data tablet has a six-position switch that sets the mode of operation and the rate at which the coordinate data are output to the processor.

The Mode and Rate Controls on the data tablet are mounted on SW 2. Positions 1 and 2 are mode switches and Positions 3, 4, and 5 are rate switches. Switch 6 is not used. The system reset switch is mounted externally at the rear of the lower frame.

Both the mode and the sampling rate may be changed under program control from the PS 300 by sending the data tablet an ASCII character. Table 12-8 lists the ASCII codes.

Table 12-8. Binary Data Transmission Codes

Mode	Binary Rate	Uppercase ASCII Character
Stop	-	S
Point	-	P
Switched Stream	2	@
	4	A
	10	B
	20	C
	35	D
	70	E
Stream	141	F
	141	G
	2	H
	4	I
	10	J
	20	K
	35	L
	70	M
	141	N
	141	O

Rate is calculated as coordinate pairs per second at 19,200 baud. All other rates are dependent on baud rates.

12.2.2 Power Requirements

The data tablet is shipped with a connector that mates with the power input connector located at the rear of the data tablet. The pin assignments that apply to this connector are shown in Table 12-9.

12.2.1 Operating Modes

Data tablet modes and sampling rates can be controlled externally under program control or internally by switches on the logic card. The positions of the internal switch determine the power-up mode and sampling rate. The following operating modes are available:

- Point Mode

  Pressing the stylus on the tablet or pressing a button on the cursor outputs one X, Y coordinate pair (sample) in the appropriate format.

- Stream Mode

  X, Y coordinate pairs (samples) are generated continuously at the selected sampling rate when the stylus or cursor is near the active area of the tablet. Pressing the stylus to the tablet or depressing a button on the cursor puts the flag character (F) bit in the output string. (See TABLETIN Function in the *Command Summary*.)

- Switched Stream Mode

  Pressing the stylus or a button on the cursor continuously outputs X, Y coordinate pairs at the selected sampling rate until the stylus is lifted or the button is released.

The data tablet has a six-position switch that sets the mode of operation and the rate at which the coordinate data are output to the processor.

The Mode and Rate Controls on the data tablet are mounted on SW 2. Positions 1 and 2 are mode switches and Positions 3, 4, and 5 are rate switches. Switch 6 is not used. The system reset switch is mounted externally at the rear of the lower frame.

Both the mode and the sampling rate may be changed under program control from the PS 300 by sending the data tablet an ASCII character. Table 12-8 lists the ASCII codes.

Table 12-8. Binary Data Transmission Codes

Mode	Binary Rate	Uppercase ASCII Character
Stop	-	S
Point	-	P
Switched Stream	2	@
	4	A
	10	B
	20	C
	35	D
	70	E
Stream	141	F
	141	G
	2	H
	4	I
	10	J
	20	K
	35	L
	70	M
	141	N
	141	O

Rate is calculated as coordinate pairs per second at 19,200 baud. All other rates are dependent on baud rates.

12.2.2 Power Requirements

The data tablet is shipped with a connector that mates with the power input connector located at the rear of the data tablet. The pin assignments that apply to this connector are shown in Table 12-9.

Table 12-9. Data Tablet Pin Assignments

Power Input Cable Connector	Power Cable Connector	PC Board Pins
1 Spare	1	7
2 Spare (Key)	2 (Key)	6 (Key)
3 -12 Volts +/-5%	3	5
4 +12 Volts +/-5%	4	4
5 + 5 Volts +/-5%	5	3
6 + 5 Volts +/-5%	6	2
7 Ground	7	1

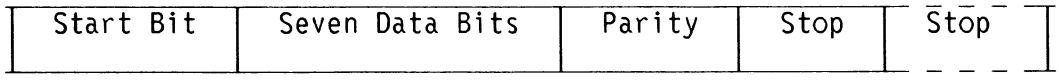
Please note pin reversal from Cable to PC Card.

12.2.3 Data Tablet/PS 300 Interface

The data tablet is a stand-alone microprocessor-driven device that can be remotely programmed. The Graphics Control Processor controls remote operation of the data tablet. The following conditions must exist for remote control of the data tablet.

- All internal mode and rate controls (SW 2) must be inactive or in the OFF condition.
- Data going to the data tablet must be at the same baud rate as the data transmitted from the data tablet.
- Data tablet command data must be input on J1 Pin 3 with a bit polarity of low level mark, high level space.
- One of the binary data transmission codes shown in Table 2-8 must be selected.

The data tablet communicates with the PS 300 via an RS-232 asynchronous cable. Each character is transmitted as a complete self-contained message consisting of an ASCII data character with even or odd parity (POE) preceded by a start bit and followed by one or two stop bits, depending on the strap option selected (HCB). The bit polarity of the transmitted data is low level mark, high level space in the following format:



Binary Data Format (Switch 1, Position 7 ON)

The binary formatted RS-232 interface is a five byte count output. Binary format is shown in Table 12-10.

Table 12-10. Binary Format

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	P	1	F3	F2	F1	F0	0	0
2	P	0	X5	X4	X3	X2	X1	X0
3	P	0	X11	X10	X9	X8	X7	X6
4	P	0	Y5	Y4	Y3	Y2	Y1	Y0
5	P	0	Y11	Y10	Y9	Y8	Y7	Y6

Cables and Connectors

The data tablet output connector uses a 25-pin female RS-232-C AMPHENOL type 206584-1 cable that mates with a DB-25P. Pin assignments are shown in Table 12-11.

Table 12-11. RS-232 Pin Assignments--Output Connector (J1)

Pin	Signal	Code
1	Ground	AA
2	Transmit Data	BA
3	Receive Data	BB
4	Request to Send	CA
5	Clear to Send	CB
7	Ground	AB
11	Transmit Data TTL	SA
20	Data Terminal Ready	CD

RS-232 Unit Switch Settings and Strap Options for 600 Series PROMs

Data tablets are shipped in a standard setting from the factory. The following sections describe strap and switch settings for non-standard strap and switch settings. Refer to the *Summagraphics Bit Pad One Users Manual*, for all switch and strap locations.

Switch 1 (Format/Calibration)

This nine-position switch controls the output data format as follows:

Position	Effect
1	Do not adjust--Factory Set
2	Do not adjust--Factory Set
3	Do not adjust--Factory Set
4	Do not adjust--Factory Set
5	Do not adjust--Factory Set
6	Not Used
7	ON--Serial Binary Output (No CRLF transmitted when in Serial Binary (Position 8) *OFF-ASCII BCD Output
8	*ON-Carriage Return Line Feed (CRLF). This adds line feed to the end of output data format. OFF-Carriage Return (CR) only.
9	*ON--English (0.005" Resolution) OFF--Metric (0.1 mm Resolution)

* Factory settings.

NOTE

Switches 1-5 are factory set calibration switches. They should not be changed unless the tablet portion of the device is changed. Refer to the Bit Pad One User's Manual.

Switch 2 (Mode/Rate)

This six-position switch controls the sampling mode (Point, Switch Stream, or Continuous Stream) and the sampling rate (X-Y coordinate pairs per second). The switch is factory set in the Continuous Stream Mode at 200 samples per second. To operate under program control, set all internal position switches to OFF.

The PS 300 Data Tablet operates at 9600 baud, sending 105 serial binary samples per second. Due to the limitations of serial baud rate transmit time, the maximum sampling rate is automatically limited to the sampling rates shown in Table 12-12.

Table 12-12. Data Tablet Sampling Rates

Baud Rate	Serial ASCII BCD (Maximum Sampling Rate)	Serial Binary
28800	85	166
19200	68	141
9600	46	105
4800	28	65
2400	16	37
1200	9	20
300	2	5

Switch 7 and Pluggable Program Strap BA (Baud Rate)

Both Switch 7 and the Pluggable Strap BA must be set to select the desired baud rate. One of the ten positions on Switch 7 must be set to ON and the blue pluggable strap must be over the center pin and the A pin (or over the center pin and the B pin). Only one position on Switch 7 may be on at a time. The baud rate is factory set with Position 2 ON on Switch 7 and pluggable strap BA over Pin B and the center pin. Table 12-13 shows the baud rates that may be selected.

Table 12-13. Baud Rate Selection

Switch 7 (Position ON)	Blue Pluggable Strap BA	
	Strap A	Strap B
1	19200	19200
2*	28880	9600*
3	14400	4800
4	7200	2400
5	3600	1200
6	1800	600
7	900	300
8	450	150
9	225	75
10	112.5	--

* Factory setting.

Parity and Stop Bits

Strapping allows the data tablet to use even or odd parity and one or two stop bits. Standard data tablets are shipped strapped for even parity, one stop bit.

12.2.4 Biasing

The active writing surface of the PS 300 Data Tablet must be magnetically biased to ensure that response to the stylus or 4-button cursor is uniform and precise. E&S will bias the data tablet at installation before it is placed into operation.

Further, the data tablet should be biased anytime it is not responding properly. If any magnetic materials are inadvertently laid on the tablet's surface, re-biasing is necessary.

The large, rubberized bar magnet shipped with the Data Tablet is used for biasing. To bias the Data Tablet follow these steps:

1. Orient the Tablet so the Tablet connectors are on the side opposite you.

2. Place the magnetic bar diagonally across the rear left-hand corner of the Tablet.

3. Grasp the bar magnet. The arrows on the magnet must point toward you.

4. Hold the magnet firmly against the tablet corner; pull the magnet slowly across the tablet to the front right-hand corner in one smooth, continuous motion. Firm contact between the tablet surface and magnet must be maintained throughout this wiping process.

5. After wiping the tablet, store the magnet away from the tablet. Care should be taken not to bias tapes, disks, or cassettes.

12.3 PS 300 FUNCTION BUTTONS

The PS 300 Function Buttons Unit, shown in Figure 12-11, gives an expanded capability for program selection, providing 32 programmable function buttons in addition to the 12 function keys on the PS 300 Keyboard. Power and communications for the PS 300 Function Buttons Unit are provided through a single modular phone cord that connects to the data concentrator.

As with the function keys on the PS 300 Keyboard, pressing a function button results in a user-specified action.

The PS 300 Function Buttons are arranged with one row of four buttons, four rows of six buttons, and a final row of four buttons. The buttons are numbered from left to right, beginning at the top row of four buttons, with the first button labeled 0. Buttons can be programmed from the PS 300 to light when activated and go out when not activated.

The PS 300 Function Buttons Unit is powered directly from the PS 300 and has a microprocessor-controlled serial communications interface used to communicate with the PS 300.

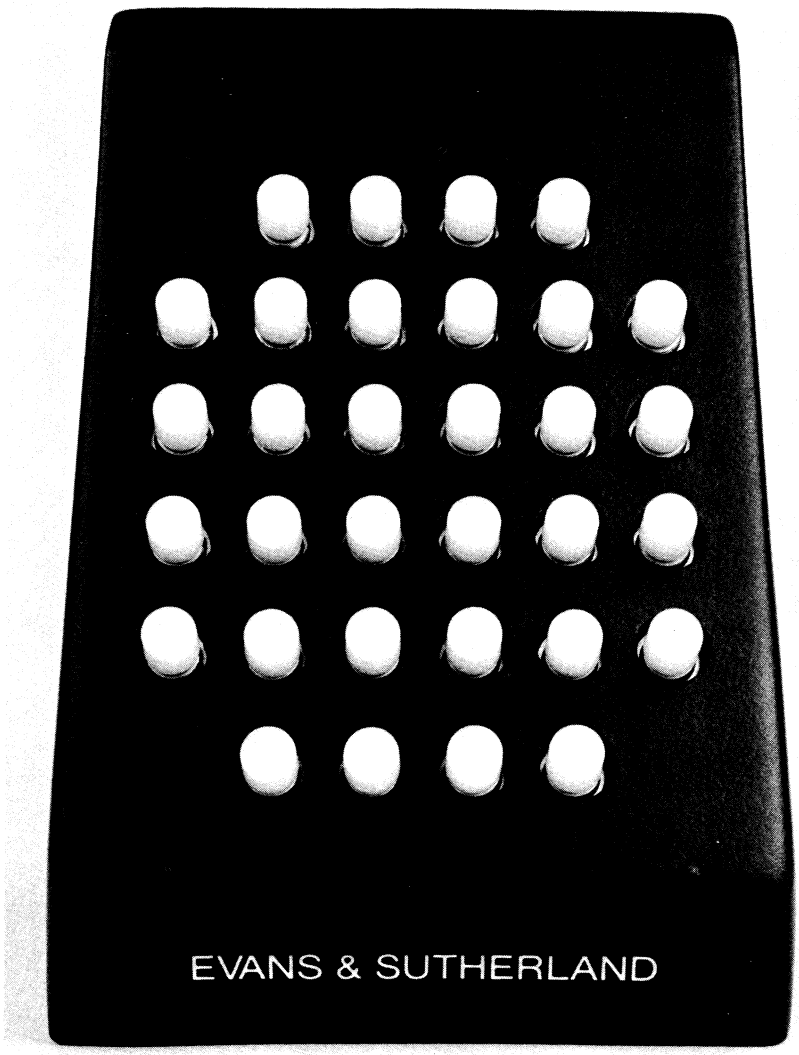


Figure 12-11. PS 300 Function Buttons Unit

The Function Buttons Unit has the following modes of operation.

- Test Mode

  On power up, or after a reset, the Function Buttons Unit comes up in Test Mode. In this mode when a button is up, the button light is off; when a button is held down, the button light is on. No characters are sent to the PS 300 while the buttons are in this mode. The Function Buttons Unit monitors the serial communication line for characters at 2400 Baud. If any valid character is received by the Function Buttons Unit, the unit exits from Test Mode, enters Run Mode, and processes the character.

- Run Mode

  In Run Mode, the function buttons respond to valid characters from the PS 300 and send a character to the PS 300 if a button is pressed. Inputs to the PS 300 from Function Buttons and sent to the appropriate function network determine all button functions. The activity of the lights backing the buttons is determined by messages sent from the PS 300 to the Function Buttons Unit.

12.3.1 PS 300 Function Buttons Power Source

Power for the PS 300 Function Buttons is supplied through a modular phone cord that connects to the Data Concentrator on the PS 300 Display Station. Two wires in the modular phone cord provide +12V @ 2A power to the Control Card and two wires act as a ground reference. Two wires are required because of the current drawn by the Function Buttons Unit. The position of the rocker switch on the back of the Function Button Unit determines which voltage is used and gives the user a dim and a bright setting for the lights.

12.3.2 RS-449 Serial Communications Interface

Communication between the Function Buttons Unit and the PS 300 Data Concentrator takes place through two pairs of differential duplex RS-449 lines that transmit and receive asynchronously through the modular phone cord that connects the two devices.

The Function Buttons Microprocessor outputs a single TTL signal that is converted on the Function Button Control Card to an RS-449 signal pair. This signal pair is then transmitted to the PS 300 Data Concentrator.

A second RS-449 signal pair is received from the PS 300 Data Concentrator. This signal pair is converted on the Function Button Control Card to a single TTL input signal to the GCP.

The serial link operates at 2400 Baud.

12.3.3 Communications Protocol of PS 300 32 Function Buttons Unit

On power up, or after a reset, the Function Button Unit comes up in a Test Mode. In this mode, when a button is up, the light in it is turned OFF; when a button is down, the light is turned ON. No characters are sent down the RS-449 line from the Function Buttons to the PS 300. If the Function Buttons Unit receives any valid character (any digit or character used in the protocols described below), it exits Test Mode, enters Run Mode and processes the character.

While in Run Mode, the PS 300 and the Function Button Unit use the communications protocol outlined below.

NOTE

The displayed messages (such as (X'05'), Ctrl E) in this section show both the ASCII equivalent (X'05') and the actual character (Ctrl E). When "KEY" is entered as the actual character, it indicates the key entered by the user.

Enquire Message (From PS 300)

This message is sent to a peripheral when the PS 300 needs to have that peripheral identified:

‹ENQ›,(X'05'),Ctrl E

Button Response Message (From Buttons)

This response to the Enquire Message tells the PS 300 first that this is the Button Unit (BTN) and also includes a three ASCII character description of the level of the firmware located in the Button Unit PROM:

```
(X'42'),B
(X'54'),T
(X'4E'),N
  x1
  x2
  x3
‹CR›,(X'0D')
‹LF›,(X'0A')
```

Request Status of Lights Message (From PS 300)

This message is sent to the Button Unit when the PS 300 wants to know the status of all the lights:

```
‹DC2›,(X'12'),Ctrl R
```

Status of Lights Response Message (From Buttons)

This response to the Request Status of Lights Message tells the PS 300 the status of all 32 lights:

```
‹STX›,(X'02'),Ctrl B
      y1
      y2
      y3
      y4
‹ETX›,(X'03'),Ctrl C
```

First, a start-of-response delimiter is sent, followed by four binary bytes of data, followed by a final delimiter. Each byte of data contains the status of eight lights with the smaller number lights in the less significant bits. The first byte of data contains the status of the first eight lights, the second byte the second set of eight, etc. A binary zero means the light is OFF and a binary one means that it is ON. The chart below shows each button, by number, arranged in 8 bit bytes.

MSB	LSB
BYTE 1	07 06 05 04 03 02 01 00
BYTE 2	15 14 13 12 11 10 09 08
BYTE 3	23 22 21 20 19 18 17 16
BYTE 4	31 30 29 28 27 26 25 24

Turn ON All Lights Message (From PS 300)

This message from the PS 300 turns ON all 32 lights in the Button Unit:

‹SI›,(X'0F'),Ctrl O

Turn OFF All Lights Message (From PS 300)

This message from the PS 300 turns OFF all 32 lights in the Button Unit:

‹SO›,(X'0E'),Ctrl N

Turn ON Light KEY Message (From PS 300)

This message from the PS 300 turns ON one of the thirty-two lights in the Button Unit (no other lights are affected):

(X'40'+ KEY)

The value chosen for KEY (which should be a hex number from [X'00] to [X'1F']) determines the specific light selected. If the designated light is already ON, this message has no affect.

Turn OFF Light KEY Message (From PS 300)

This message from the PS 300 turns OFF one of the thirty-two lights in the Button Unit (no other lights are affected):

(X'60'+ KEY)

The value chosen for KEY (which should be a hex number from [X'00] to [X'1F']) determines the specific light selected. If the designated light is already OFF, this message has no affect.

Key Down, Light ON Message (From Buttons)

This message from the Button Unit reports to the PS 300 that a KEY has been pressed down and that the status of the light in that KEY is "ON":

(X'40'+ KEY)

The value of KEY should be a number (X'00') to (X'1F') corresponding to one of the thirty-two keys in the Button Unit.

Key Down, Light OFF Message (From Buttons)

This message from the Button Unit reports to the PS 300 that a KEY has been pressed down and that the status of the light in that KEY is OFF:

(X'60'+ KEY)

The value of KEY should be a number (X'00') to (X'1F') corresponding to one of the thirty-two keys in the Button Unit.

Run-Confidence-Test Message (From PS 300)

This message from the PS 300 directs the Button Unit to run a series of confidence tests designed to test various aspects of the hardware including the Internal RAM, the ROM and the External RAM:

‹DC4›,(X'14'),Ctrl T

Confidence-Test-Response Message (From Buttons)

This response to the Run Confidence Test Message tells the PS 300 that the Confidence Tests are being run as requested.

(X'41'),A

The B, C, and D characters are returned from the Buttons Unit upon the successful completion of the associated test. If the associated test fails, the corresponding letter is not returned and the next test is executed. Upon completion of all three tests the letter "E" is sent out.

Int RAM	(X'42'),B
ROM	(X'43'),C
Ext RAM	(X'44'),D

12.4 CONTROL DIALS UNIT

The PS 300 Control Dials Unit, shown in Figure 12-12, is a modular interactive device that is microprocessor controlled. It is much like the PS 300 Keyboard in physical appearance and construction. Power, ground, and communication lines are routed through a modular phone cord from Port B of the Data Concentrator to the Control Dials Interface card. It uses a single, 8-conductor flexible interface cable with locking modular plugs. The dials are used to communicate dynamic, incrementing, and decrementing data to the PS 300.

CAUTION

Care should be taken when cleaning the PS 300 Keyboards and Control Dials. These devices have Lexan covers that accumulate static electricity when buffed. To safely clean these devices, use a damp cloth. DO NOT VIGOROUSLY BUFF THE CONTROL DIALS OR KEYBOARDS.

Each control dial is fully programmable. An optional 8-character LED label is available for each dial and is used to display static or dynamic information about the dial or dial operations.

The eight knobs on the face of the Control Dials Unit provide direct, one-to-one input to the shaft encoders. The encoders are continuous-turn digital devices that send pulses to the Control Dials Interface card each time a dial is turned.

Figure 12-12. PS 300 Control Dials Unit

12.4.1 Shaft Encoders

Two channels in quadrature (i.e., Channel A offset from Channel B by 90 degrees) are provided from each shaft encoder to give the relative position of the encoder shaft compared to a previously-read shaft position. These two channels output two pulse trains that determine the direction of shaft rotation.

Shaft encoders are a great improvement over conventional potentiometers. The main advantages offered include:

- No dead spots
- Improved resolution

  While the basic encoder specification states a resolution of 256 counts per turn, a transition occurs four times per count on either Channel A or Channel B. Thus, an effective resolution of 1024 counts per turn is obtained.

- Very little noise

12.4.2 Operating Modes

The Control Dials operate in the following modes.

- Local Loopback

  When the Control Dials Unit is powered up it enters Local Loopback mode. This is a self-test period in which the operator may verify correct operation of both the LEDs and the dials. This mode is exited when the first serial input is received from the GCP.

- Message

  The Control Dials output rotational values in message mode only when enabled to do so by a setup command from the Graphics Control Processor. Each dial is individually programmable. The message mode may be entered any time after initial power-up and is entirely under the control of the GCP.

- LED Segment Mode

  This mode allows each 8-character LED label to be individually defined.

12.4.3 Control Dials Unit Local Loopback Test

Like the keyboard, the Control Dials Unit enters a Local Loopback mode when powered up. In this mode, each dial and corresponding LED label may be separately tested for proper operation.

The dials are tested by turning any dial slowly and noting the count displayed on the corresponding LED label. Turning the dial clockwise increments the accumulated decimal count appearing on the corresponding LED label; the number will increase from 0 to 1023, reset to 0, and begin counting up to 1023 again. Rotating the dial counterclockwise causes the accumulated decimal count to decrement to 0, reset to 1023, and decrement to 0 again. Each dial may be tested in this manner.

The Control Dials Unit remains in Local Loopback mode until it receives a character from the GCP. The Unit returns to Local Loopback mode when a Control D (Hex 04) character is received from the GCP.

12.4.4 Control Dials Operation

The PS 300 Control Dials Unit outputs pulses when any of the dials are turned. From the pulses, the Control Dials Interface determines:

- Which dial is being turned.
- What direction the dial is turning.
- How far the dial is rotated. Dial position is evaluated in terms of the number of changes of delta.

After the Control Dials interface analyzes dial motion, rotational information is transmitted to the Graphics Control Processor. The following paragraphs describe data formats and codes exchanged in dial and LED display operation.

Two messages set up the operating mode for the PS 300 Control Dials. One command specifies the minimum rotation count delta required before a sample is output to the PS 300, and the other command specifies the maximum rate at which the Control Dials send a new delta update to the PS 300. The dials output relative delta values only; that is, the position of each dial is reported in terms of its last sampled location. These inputs can come from the initial function DSET1...DSET8. and must be done before any output can occur after power up.

The message to specify the rotation count delta for a particular control dial consists of the following four-byte sequence:

CTRL V	*Control byte*	*MSB*	*LSB*

The control byte specifies the dial number in the following format:

1x0xxnnn

where the n's specify the dial number between 000 and 111 (0 - 7.), and the x's may be either zero or one.

The MSB (most significant byte) and LSB (least significant byte) together specify the 16-bit delta value. This number may be any value between 1 and 65535; use of negative or zero values is not recommended.

The message that specifies the maximum update in seconds is in the following four-byte format:

CTRL V	*Control byte*	*Reserved*	*Time Count*

The control byte is in the following format:

1x1xxxxx

where all x's may be either zero or one. This means that the specified maximum update applies to all dials.

The next byte is reserved for future use.

The final byte consists of a binary number that specifies the sample time value. The following sample times are available:

Hex	Decimal	Updates/Sec.
05	5	60
0A	10	30
1E	30	10

12.4.5 Dial Setup Programming Examples

To specify a maximum update rate of 10 updates per second:

Byte	Meaning
X'16'	CTRL V
X'90'	Setup maximum update rate
X'00'	Reserved byte
X'1E'	10 updates per second (decimal 30)

To set Dial 4 for the minimum rotation count delta required before a sample is output:

Byte	Meaning
X'16'	CTRL V
X'84'	Setup Dial 4
X'00'	Delta MSB
X'06'	Delta LSB

The data format that is output from the Control Dials Unit takes the following form:

CTRL V	*Dial Number*	*Sample MSB*	*Sample LSB*

12.4.6 Control Dials LED Display Operation

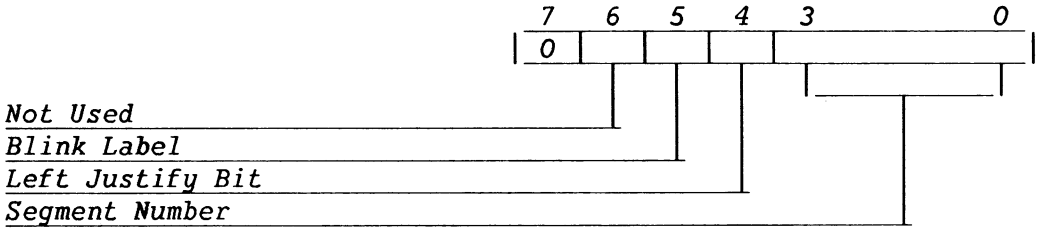
The PS 300 Control Dials Unit has eight 8-character LED displays. Each display functions as a label for a dial. The LED displays are much like those on the PS 300 Keyboard, displaying the same characters and responding to the same codes. The Control Dials LEDs are always operated in segment mode. That is, each display is separately programmed and functions independently of the other LEDs. Line mode is not used because of the wider physical separation between the 8-character groupings.

The LED label message format is as follows:

CTRL V	*Control Byte*	*0 to 8 Characters*

The CTRL V character indicates the beginning of a command string.

The control byte specifies blinking, left justification, and LED segment number.



The control byte format is as follows:

- Bit 7 in the control byte is always 0.
- Bit 6 is not used.

When the Blink Label bit (bit 5) is 1, the label blinks. When this bit is 0, the label does not blink.

When the Left Justify bit (bit 4) is 1, the label is left-justified in the specified label location. When this bit is 0, the label is automatically centered in the label location.

The Segment Number bits (bits 3–0) specify the LED label location (0 through 7).

The "0 to 8 characters" are the ASCII characters to be displayed on the selected LED segment. If there is no label message (character count = 0), then the current message in the LED segment is set up according to the values of bit 5 in the control byte (that is, the LED will blink or not blink).

DATA FORMATTING

CONTENTS

13. DATA FORMATTING

This chapter describes the data formats expected by PS 300 system functions, in particular, the command interpreter (CI). The information is made available to provide programmers whose systems are not supported by the PS 300 Graphics Support Routines with the information necessary to write their own. The chapter will follow data paths from the host to the PS 300 command interpreter, giving examples of the data formats that are expected by the functions that receive and pass data through the PS 300.

Information in this chapter relies heavily on other chapters of this Guide. Where helpful, information will be duplicated here for clarity. Otherwise, references will be given to other chapters.

The first section gives the formats for the different data types. It is important to note that data are received a byte at a time by the GCP so where there is a MSB - LSB specified the MSB must be sent first. Also given is the format for an error - reset command. This command should be sent anytime there is an error detected in the sending of other commands. This will cause the CI to reset to begin command interpretation again. This section also describes the system functions that receive, pass, and route data internally.

The second section of this chapter provides the 6-bit binary encoding method used by the PS 300. This format uses 2D or 3D vector normalized data as an example.

The final section of this chapter provides the data formats for each of the commands that can be sent to the CI. The format shows the data type followed by the data that should be sent. The data will be expressed as a number, a Boolean, an identifier or an expression. If an identifier begins with the letter Q, it is a subcommand type and the value of the subcommand to be used is shown after the equal sign (=). If the identifier is SIZE it refers to the size or length of the string or id about to be transferred. All other identifiers are user supplied variables.

13.1 DATA TYPES

This section gives the formats for different data types. Data types that can be passed internally in the PS 300 are defined below:

```
{ 0} Qreset,          { dataless: reset a function instance }
{ 1} Qprompt,         { dataless: flush the CI pipeline }
{ 2} QBoolean,        { normal carrier of Boolean values }
{ 3} Qinteger,        { normal carrier of integer values }
{ 4} Qreal,           { normal carrier of floating point values }
{ 5} Qstring,         { original carrier of byte strings, not used }
{ 6} Qpacket,         { carrier of byte strings }
{ 7} Qmorepacket,     { continuation Qpacket carrier of byte strings }
{ 8} Qmove2,          { 2D vector including P bit }
{ 9} Qdraw2,          { 2D vector including L bit }
{10} Qvec2,           { 2D vector with no P/L bit (normal vector) }
{11} Qmove3,          { 3D vector including P bit }
{12} Qdraw3,          { 3D vector including L bit }
{13} Qvec3,           { 3D vector with no P/L bit (normal vector) }
{14} Qmove4,          { 4D vector including P bit }
{15} Qdraw4,          { 4D vector including L bit }
{16} Qvec4,           { 4D vector with no P/L bit (normal vector) }
{17} Qmat2,           { 2x2 matrix }
{18} Qmat3,           { 3x3 matrix }
{19} Qmat4,           { 4x4 matrix }
{20} Qbindata         { definition of binary data (data part of vector
                        list)}
{21} Qusertype        { type that user may use to define own message }
     .
     .
     .
     { Qdtype is padded with 260 miscellaneous elements to ensure that a 16-bit
     field is allocated by the Pascal compiler rather than the 8-bit field that
     would be allocated otherwise. }
```

The Qdtype is used to specify the different types of Qdata message blocks available in the PS 300 runtime system. Qdata blocks are the primary vehicle for communication in the PS 300. When a Qdata message is input to a function, it checks to see if it is a valid message type (Qdtype). When a message is output by a function, it carves a Qdata message of the appropriate type and outputs it.

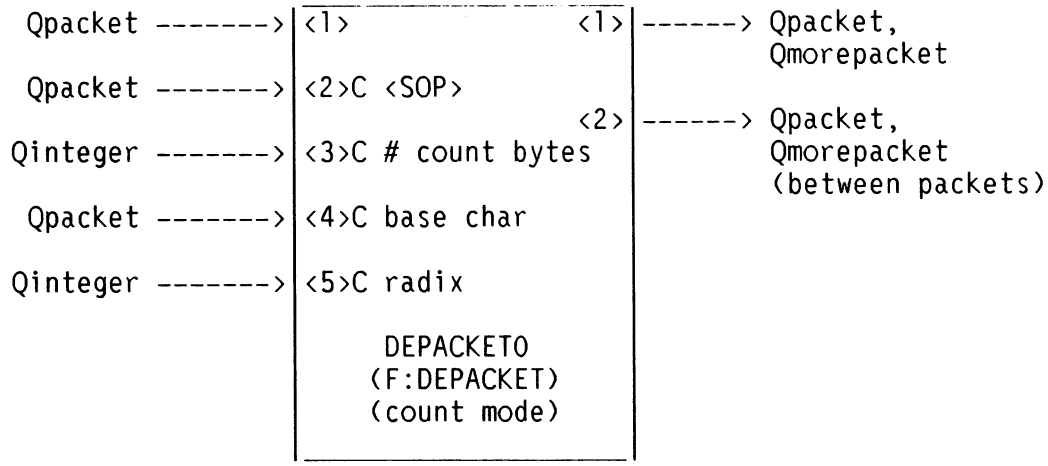
The PS 300 command interpreter expects "tokens". Tokens consists of a size, a data type, and a value. Once given, the type of command is implicit in the type of the token, e.g. "Qsetcontrast" for "Set Contrast". The CI accepts tokens until it has enough to carry out a command. At that point, the CI passes all required data to (and triggers) the appropriate function.

13.1.1 Routing Functions

Data are sent to the PS 300 from the host as a stream of bytes. These bytes must contain information that is intelligible to PS 300 system functions about the nature of the message and where it is to be sent internally in the PS 300. The descriptions that follow describe the data transfer modes used in host/PS 300 communication and briefly describe the system functions that accept, examine, and route data internally in the PS 300.

F:DEPACKET

A system function, F:DEPACKET, accepts data (input to the PS 300 from the host) from receiving functions (BI$, etc.). F:DEPACKET converts a stream of bytes from the host into a stream of Qpacket/Qmorepacket. A Qpacket is a block of character data that can be sent from one PS 300 function to another. When data come from the host through the F:DEPACKET function, they contain a byte for routing control. A Qmorepacket is a Qpacket that when coming from the host through F:DEPACKET, has no routing byte (i.e., a Qmorepacket has the same destination as the previous Qpacket.)



In count mode, F:DEPACKET assumes that a packet is defined as:

<SOP>	count bytes	packet contents

where <SOP> represents the Start of Packet character that is by default the the ASCII ACK character, decimal character code 06 (↑F).

The definition of SOP (one character) is taken from a single character Qpacket on input <2>.

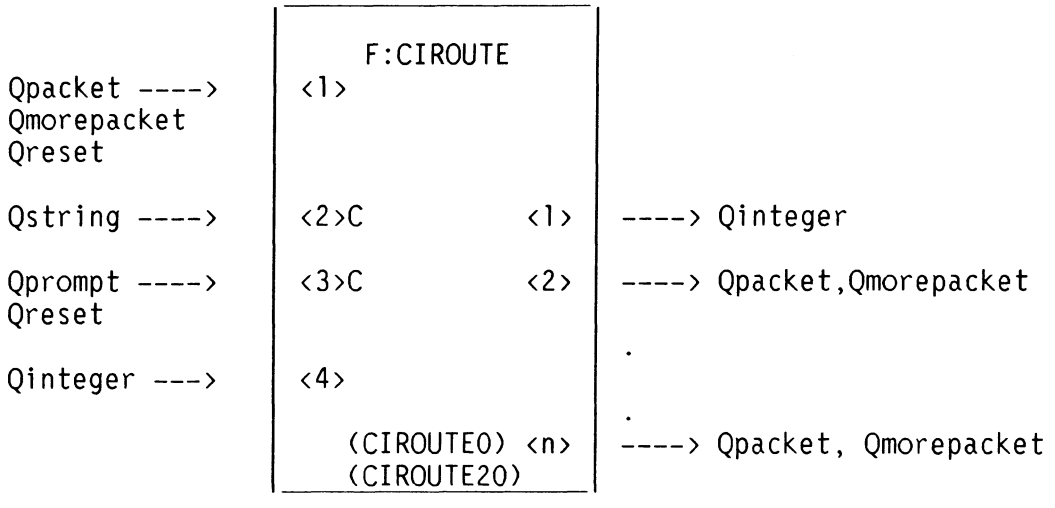
The message count is defined by n bytes (n defined by the Qinteger on input <3>). Each count byte is offset from the base character (the base character is taken from a single character Qpacket on input <4>). After the base character is subtracted, each count byte becomes a digit of the message count whose radix is defined by the Qinteger on input <5>.

Output <1> outputs Qpackets and Qmorepackets of count mode messages. Output <2> outputs Qpackets and Qmorepackets of any messages which are not in count mode.

The <SOP> byte and the count bytes are removed from the start of the packet before the packet is sent to F:CIROUTE, that does the actual routing.

F:CIROUTE

Once data have passed through an instance of F:DEPACKET, the next function to receive them is F:CIROUTE. F:CIROUTE has two instances, one for count mode and one for escape mode. They are functionally very similar, and only the count mode instance, CIROUTE0 will be described. CIROUTE0 examines the first character it receives (the character following the count bytes in count mode and the character following the <FS> character in escape mode) to determine where the packet message is to be sent. These characters are "routing" bytes, and are used to select the appropriate channel for data in the PS 300. Data channels include lines to the terminal emulator, the PS 300 command interpreter (through F:READSTREAM for binary packets), the Disk writing function, the Raster function (for PS 340 systems), and other system functions. A base character (defined on Input <2> of CIROUTE0) is subtracted from this routing character before it is used to select the output channel. The base character defaults to the character zero ("0").



F:CIROUTE demultiplexes a stream of Qpackets/Qmorepackets from input <1> to one of the n output channels. The first byte of an incoming Qpacket is assumed to be the multiplexing byte, equal to the base character (from input <2>) + K, where K is the channel number. If K > (n-3) or K < 0, there is no channel for this output and a pair of messages are sent on outputs <1> and <2>. These can be used to allow for later remultiplexing or further demultiplexing. An integer giving the indicated output port is sent on output <1> and the message for which there was no defined output is sent on output <2>. Whether or not K is within the limits implied by the number of outputs of F:CIROUTE, the multiplexing byte is removed from the start of the packet.

F:DEPACKET passes incoming Qmorepackets out the current channel (as defined by the last Qpacket). Initially, after a Qreset is received, the current channel is -1.

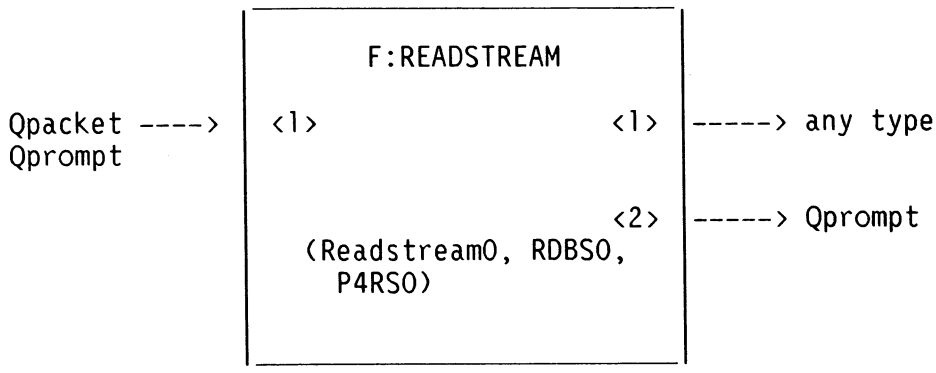
When instancing this function, a parameter is required to specify the number of outputs.

F:CIROUTE is a special version of F:DEMUX. It assumes that it is driving parallel, asynchronous paths to a common destination (the command interpreter). It synchronizes those paths by sending out a Qprompt on a channel at the end of using that channel and waiting for it to come back around before switching to the next channel. This assumes that the common destination can strip Qprompts and send them back (which the CI does). Input <4> gives the maximum channel number, m, for which path flushing is desired. CIROUTE flushes channels 0<=K <= m with Qprompts.

The definitions for the inputs and outputs for F:CIROUTE are described in Chapter 6 of this guide. The definition of the routing bytes used by CIROUTE are also given in Chapter 6.

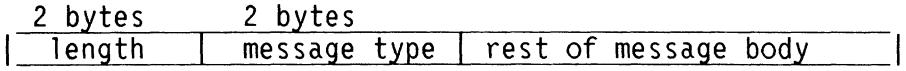
F:READSTREAM

Data that is to be sent in binary packets from F:CIROUTE to the command interpreter (i.e., same path as the GRSs take) is sent through F:READSTREAM.



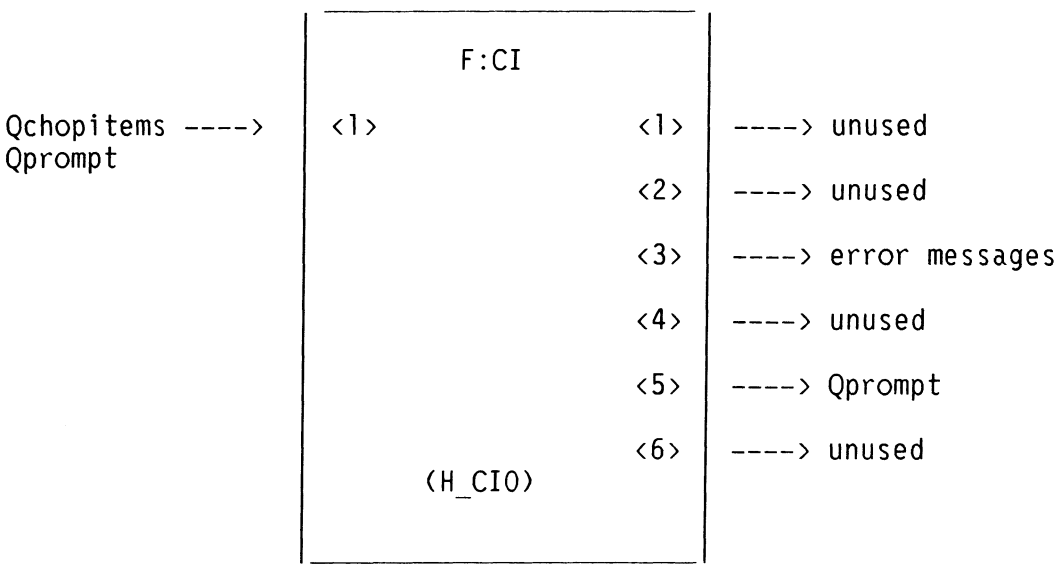
This function converts an 8-bit stream into arbitrary messages. It takes two bytes as the count of information (including message type) and creates a message of that size with the bytes of information that follow it.

The message format on input is:



F:CI

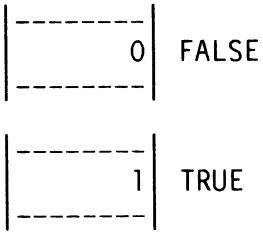
The command interpreter accepts messages through F:READSTREAM.



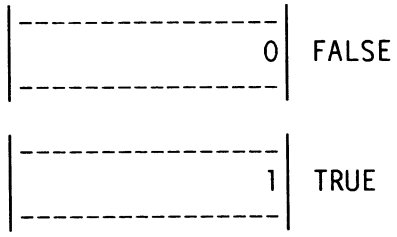
This function interprets commands, creating display structures and function networks. It receives input either from a chop/parse function or a Readstream function (if using the GSRs).

13.1.2 Data Formats for Data Types

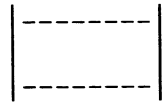
BBOOL - BOOL : 8 BIT BOOLEAN



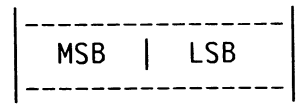
BOOL - BOOL : 16 BIT BOOLEAN



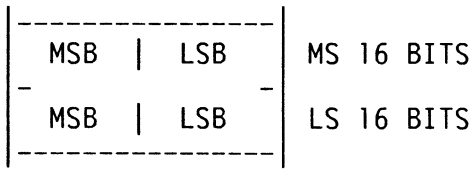
INT8 - BYTE : 8 BIT INTEGER



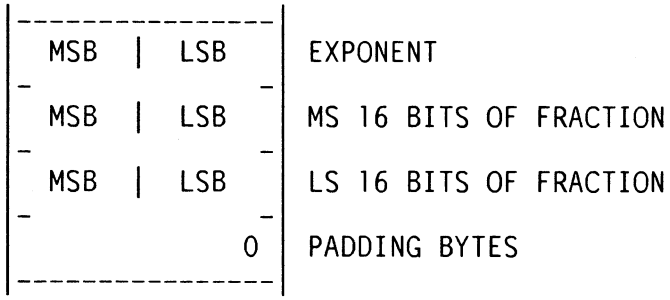
INT16 - WORD : 16 BIT INTEGER



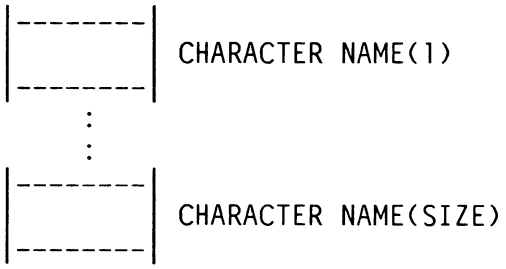
INT32 - LWORD : 32 BIT INTEGER



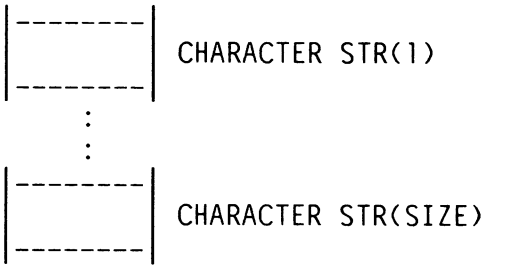
PSREAL - REAL32 : 64 BIT REAL



ID - NAME, SIZE



STRING - STR, SIZE



VECNO - V, POSLIN, DIM, COUNT

COUNT OF

2D VECTOR - MOVE

MSB | LSB	X NORMALIZED FRACTION
MSB | LSB	Y NORMALIZED FRACTION
EXP |INTENS|0	EXPONENT/INTENSITY - MOVE

2D VECTOR - DRAW

MSB | LSB	X NORMALIZED FRACTION
MSB | LSB	Y NORMALIZED FRACTION
EXP |INTENS|1	EXPONENT/INTENSITY - DRAW

OR

3D VECTOR - MOVE

MSB | LSB	X NORMALIZED FRACTION
MSB | LSB	Y NORMALIZED FRACTION
MSB | LSB	Z NORMALIZED FRACTION
EXP |INTENS|0	EXPONENT/INTENSITY - MOVE

3D VECTOR - DRAW

MSB | LSB	X NORMALIZED FRACTION
MSB | LSB	Y NORMALIZED FRACTION
MSB | LSB	Z NORMALIZED FRACTION
EXP |INTENS|1	EXPONENT/INTENSITY - DRAW

VBLNO - V, POSLIN, DIM, COUNT

EXP	INTENS	EXPONENT/INTENSITY

FOLLOWED BY COUNT OF

2D VECTOR - MOVE

MSB	LSB	X NORMALIZED FRACTION
MSB	LSB |0	Y NORMALIZED FRACTION - MOVE

2D VECTOR - DRAW

MSB	LSB	X NORMALIZED FRACTION
MSB	LSB |1	Y NORMALIZED FRACTION - DRAW

OR

3D VECTOR - MOVE

MSB	LSB	X NORMALIZED FRACTION
MSB	LSB	Y NORMALIZED FRACTION
MSB	LSB |0	Z NORMALIZED FRACTION - MOVE

3D VECTOR - DRAW

MSB	LSB	X NORMALIZED FRACTION
MSB	LSB	Y NORMALIZED FRACTION
MSB	LSB |1	Z NORMALIZED FRACTION - DRAW

13.1.3 Error Formatting

This format is used to reset the CI after an error.

ERROR - ERRCOD

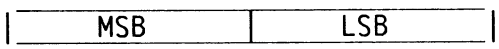
INT16 - 2
INT16 - QERRFL=143

13.2 DESCRIPTION OF DATA PROTOCOL IN THE PS 300

The following sections describe the way the PS 300 stores data, data protocol, and encoding methods used for the packaging of 6-bit binary data.

13.2.1 Data Storage

The PS 300 stores its data as follows:



where the MSB (Most Significant Bit) starts at the low address and the LSB (Least Significant Bit) starts at the high address.

The host must send the MSB first, followed by the LSB. Some hosts store their data in an address order that reverses this sequence. If this is the case, the MSB and LSB must be reversed in the host before being sent to the PS 300.

13.2.2 Six-Bit Binary Data Encoding Method

Binary data must be encoded in the following manner. This encoding process occurs prior to sending the byte count of binary vector data.

NOTE

The byte count of binary data must not include the count of bytes that result when the data is passed through the encoding scheme.

1. The encoding process collects 16-bit words until it has two sets.

WARNING

The carriage control characters must be suppressed when transmitting binary data, or the carriage control characters will be interpreted as binary data.

2. A two-word set is broken up into 5 bytes with 6 significant bits, and 1 byte with 2 significant bits. These bits are extracted from the least significant end to the most significant end of the two-word set.

3. The order that the bytes are sent to the PS 300 reverses the order in which they were extracted. The byte with 2 significant bits is sent first, followed by the the last 6-significant-bit byte, and so on.

4. To make the bytes printable ASCII characters, a zero '0' character (hex 30 or decimal 48) is added to each byte prior to sending them.

This encoding process is illustrated in the example that follows.

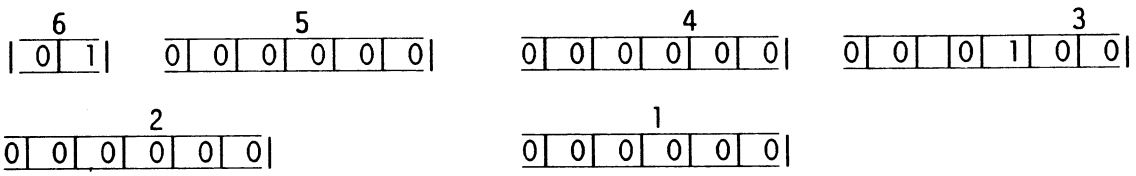
The two-word set of 16-bits are held internally in the host in the following bit sequence.

The first two word set is:

x = | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

y = | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

The two 16-bit words are broken up into 5 bytes with 6 bits, and 1 byte with two bits in the following order:



The zero "0" character is now added to each 6-bit byte:

byte1
000000
+ 110000 (zero)
110000

byte2
000000
+ 110000 (zero)
110000

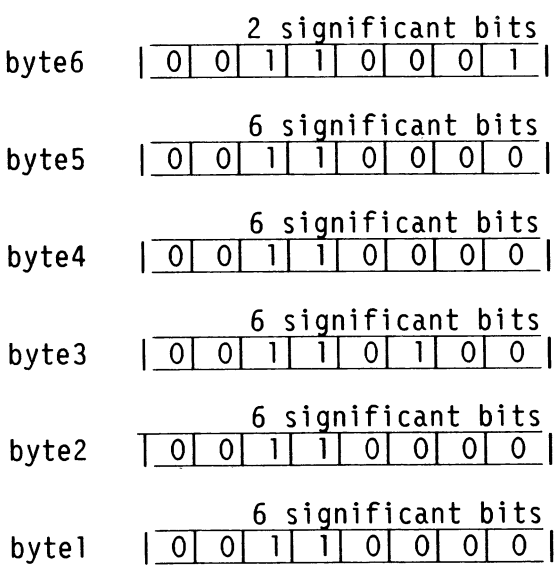
byte3
000100
+ 110000 (zero)
110100

byte4
000000
+ 110000 (zero)
110000

byte5
000000
+ 110000 (zero)
110000

byte6
----01
+ 110000 (zero)
110001

After the encoding procedure, the bits will be sent to the PS 300 in the following sequence of bytes. Note that the sequence order of the bytes has been reversed.



13.2.3 Example of Encoding Binary Data

An example of encoding binary vector data is given in the following section.

Please note that the example assumes escape mode. (Refer to Chapter 4 for a description of escape mode.)

The vector list to be encoded is:

```
AA:=  vec itemized n=4
      P 1,1,0 I=1.0
      L -.25, .75, .5 I= .75
      P 10,5,.001  I=.5
      L -.001, -.002, .003 I= .1
      ;
```

The data in PS 300 8 bit binary format is as follows:

NOTE

The X,Y,Z mantissa's and the Vector exponent are 2's complement numbers.

```
0100000000000000 = 1/2 (x mantissa)
0100000000000000 = 1/2 (y mantissa)
0000000000000000 = 0  (z mantissa)
00000001 1111111 0 exponent =1 intensity= 7F(hex) p/l=p

1110000000000000 = -1x2**-2 (x mantissa) NOTE 2's complement
0110000000000000 = 3x2**-2  (y mantissa)
0100000000000000 = 1x2**-1  (z mantissa)
00000000 1100000 1 exponent=0 inten=60(hex) p/l=l

0101000000000000 = 5x2**-3
0010100000000000 = 5x2**-4
0000000000000010 = 1x2**-14
00000100 1000000 0 exp=4 int=40(hex) p/l=p
```

```
1101111100111100  = -2097x2**-13
1011111001110111  = -16777x2**-15
0110001001001101  =  25145x2**-15
11111000 0001100 1 exp= -8 int=0C(hex) p/l=1

0000000000000000   padding
```

For exercise, check the last vectors in decimal.

```
x= -2097x2**-13x2**-8 = -2097x2**-21 = -9.99928E -4 -> -.001
y= -1677x2**-15x2**-8 = -16777x2**-23 = 1.99997E -3 -> -.002
z= 25145x2**-15x2**-8 = 25145x2**-23  = 2.99752E -3 ->  .003
```

The left column is the binary data re-encoded into the six-bit format. The right column is the 8-bit data.

```
002P0     0000000000001010          = 10 (size of label+8)
          0000000000101100          = 44 (Qlabel)

000P01    0000000000000010          = 2 (size)
          0000000000000001          = 1

11@@00    0100000101000001          = AA (name)
          0000000000000000          = 0

00;@2D    0000000000101101          = 45 (data byte count)
          0000000010010100          = 148 (q3dvhd)

030000    0000001100000000          item= 3/bnorm= .false.
          0000000000000000          = 0

000000    0000000000000000
          0000000000000000          = 0

000000    0000000000000000
          0000000000000000
```

```
000000    0000000000000000
          0000000000000000          = 0

000000    0000000000000000
          0000000000000000

000000    0000000000000000
          0000000000000000          = 0

000000    0000000000000000
          0000000000000000

000000    0000000000000000
          0000000000000000          = 0

000000    0000000000000000
          0000000000000000

000000    0000000000000000
          0000000000000000          = 4 (veccount)

001000    0000000000000100
          00000000                  = .false. (cblend)
          00000000

0T0@X0    00100100                  = 36 (total byte count)
          0000000100001010          = 266 (qbndat)
          00000000

0P@010    00100000                  = 32 (count*(dimen+1)*2)
          0100000000000000          = 1/2(x mantissa)
          01000000

000001    00000000                  = 1/2(y mantissa)
          0000000000000000          = 0 (z mantissa)
          00000001                    exponent=1

3nh01P    1111111 0                   intensity=7F(hex) p/1=p
          1110000000000000          = -1*x**-2 (x mantissa)
    01100000

00@000    00000000                  = 3*2**-1 (y mantissa)
          0100000000000000          = 1*2**-1 (z mantissa)
    00000000                        exponent=0

31D00X    1100000 1                   intensity=60(hex) p/1=1
          0101000000000000          = 5*2**-3 (x mantissa)
          00101000
```

```
000084    00000000                  = 5*2**-4 (y mantissa)
          0000000000000010          = 1*2**-14(z mantissa)
          00000100                    exponent=4

20gcbn    1000000 0                   intensity=40(hex) p/l=p
          1101111100111100          = -2097*2**-13 (x mantissa)
          10111110

1gHTgh    01110111                  = -16777*2**-15 (y mantissa)
          0110001001001101          = 25145*2**-15(z mantissa)
          11111000                    exponent=-8

0I00D0    0001100 1                   intensity=0C(hex) p/l=l
          0000000000000101          = 5 (data byte count - including
                                        padding)
          00000000

1[0000    01101011                  = 107 (qendls)
          00000000                    padding
          0000000000000000
```

```
002P0 000P0111@@0000;@2D0300000000000000000000000000000000000000000000000000000000000000
0000001000OT0@X00P@0100000013nh01P00@00031D00X00008420gcbn1gHTgh0I00D01[0000
```

The vector list is transmitted to the PS 300 using escape mode as follows:

<SOP>2 {route to six bit binary route}
{and send the encoded data}

If the vectors are vector normalized, an arbitrary sized vector list may be transmitted by sending multiple "packets" of data as above, without the terminating semicolon. When all of the vectors have been transmitted, they must be terminated with a semicolon. HOWEVER the "packets" MUST contain complete vectors.

The vectors will be processed faster if the vector estimate (N=vector_estimate) is equal to or greater than the actual number of vectors transmitted. This is because the PS 300 allocates memory for the vectors based on this estimate. If the estimate is low, the PS 300 must find a new block of continuous memory large enough for the total, copy the vectors in the original block into the new block, and write the new vectors into the block.

13.3 COMMAND INTERPRETER DATA FORMAT

The following section gives the data formats for most of the commands that can be sent to the PS 300 command interpreter.

The format shows the data type followed by the data that should be sent. The data will be expressed as a number, a Boolean, an identifier, or an expression. If an identifier begins with the letter Q, it is a subcommand type and the value of the subcommand to be used is shown after the equal sign (=). If the identifier is SIZE it refers to the size or length of the string or ID about to be transferred. All other identifiers are user supplied variables.

```
ALLOCATE PLOTTER PLOT;

    INT16 - 6
    INT16 - QALLPL=310
    INT32 - PLOT


NAME := ATTRIBUTES [COLOR hue[,sat[,intens]]]
     [DIFFUSE diffus]
     [SPECULAR specul];

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 44
    INT16 - QATTR=357
    PSREAL- HUE
    PSREAL- SAT
    PSREAL- INTENS
    PSREAL- 0.
    PSREAL- DIFFUS
    INT16 - SPECUL


NAME := ATTRIBUTES [COLOR hue[,sat[,intens]]]
     [DIFFUSE diffus]
     [SPECULAR specul]
      AND   [COLOR hue2[,sat2[,inten2]]]
     [DIFFUSE diffu2]
     [SPECULAR specu2];

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 86
    INT16 - QOATTR=358
    PSREAL- HUE
    PSREAL- SAT
    PSREAL- INTENS
    PSREAL- 0.
    PSREAL- DIFFUS
    INT16 - SPECUL
```

```
    PSREAL- HUE2
    PSREAL- SAT2
    PSREAL- INTEN2
    PSREAL- 0.
    PSREAL- DIFFU2
    INT16 - SPECU2

BEGIN

    INT16 - 2
    INT16 - QBEGIN=105

NAME := BEGIN_STRUCTURE

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 2
    INT16 - QBEGOB=103

NAME := BSPLINE
            ORDER = ORDER
            OPEN/CLOSED
            NONPERIODIC/PERIODIC
            N = NVERT
            VERTICES        = X(1),   Y(1),   ( Z(1)  )
                              X(2),   Y(2),   ( Z(2)  )
                               :        :         :
                              X(N),   Y(N),   ( Z(N)  )
            KNOTS = KNOTS (1), ... KNOTS (NKNOTS)
            CHORDS = CHORDS;

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 14
    INT16 - QSTRTC=152
    INT8  - 1
```

```
    BBOOL - .FALSE.
    INT8  - ORDER
    BBOOL - .FALSE.
    INT8  - DIMEN
    BBOOL - (.NOT.OPNCLS)
    BBOOL - (.NOT.NONPER)
    BBOOL - .FALSE.
    INT32 - NVERT

      REPEAT NVERT TIMES
    INT16 - 34
    INT16 - QCRVEC=296
    PSREAL- V (1,1)
    PSREAL- V (2,1)
    PSREAL- V (3,1)
    PSREAL- V (4,1)

    (OPTIONAL)
      REPEAT NKNOTS TIMES
    INT16 - 10
    INT16 - QKNOT=295
    PSREAL- KNOTS (I)

    INT16 - 14
    INT16 - QENDCV=153
    INT32 - CHORDS
    PSREAL- 0


NAME := CHARACTER ROTATE ANGLE (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 10
    INT16 - QROTTX=77
    PSREAL- ANGLE
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0
```

```
NAME := CHARACTERS TRANX,TRANY,TRANZ
             STEP STEPX,STEPY 'CHARS';

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 22
    INT16 - QTXTLB=159
    PSREAL- STEPX
    PSREAL- STEPY
    INT32 - 0
    INT16 - SIZE + 6
    INT16 - QDTSTR=305
    INT16 - SIZE
    INT16 - 1
    STRING- CHARS, SIZE
    INT16 - 26
    INT16 - Q3DPCH=306
    PSREAL- TRANX
    PSREAL- TRANY
    PSREAL- TRANZ
    INT16 - 2
    INT16 - QENDCH=304


NAME := CHARACTER SCALE SCALEX, SCALEY
             (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 18
    INT16 - QTXTSC=166
    PSREAL- SCALEX
    PSREAL- SCALEY
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0
```

```
CONN SOURCE <OUT>:<INP> DEST;

    INT16 - SIZE+8
    INT16 - QALOOK=100
    INT16 - SIZE
    INT16 - 1
    ID    - SOURCE, SIZE
    INT16 - 0
    INT16 - 6
    INT16 - QFNOUT=144
    INT32 - OUT
    INT16 - SIZE+8
    INT16 - QALOOK=100
    INT16 - SIZE
    INT16 - 1
    ID    - DEST, SIZE
    INT16 - 0
    INT16 - 6
    INT16 - QINPIN=145
    INT32 - INP
    INT16 - 2
    INT16 - QCON=138

NAME := COPY CPYFRM (START=) START (,) (COUNT=) COUNT;

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - SIZE+8
    INT16 - QALOOK=100
    INT16 - SIZE
    INT16 - 1
    ID    - CPYFRM, SIZE
    INT16 - 0
    INT16 - 6
    INT16 - QCOPY=123
    INT16 - START
    INT16 - COUNT

DEALLOCATE PLOTTER PLOT;

    INT16 - 6
    INT16 - QDALLP=311
    INT32 - PLOT
```

```
NAME1 := PATTERN i (i) [AROUND_CORNERS] [MATCH|NOMATCH]
                   LENGTH 1;

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME1, SIZE
    INT16 - 0
    INT16 - 46
    INT16 - QPATRN=149
    BBOOL - .NOT. CONTIN
    BBOOL - MATCH
    PSREAL- LENGTH
    INT8  - SEGS ( 0<SEGS<=32 )
    INT8  - 0
    INT8  - PATTRN (1 TO SEGS)

     IF SEGS < 32 REPEAT TO EQUAL 32 INT8 VALUES
    INT8  - 0

    INT16 - 2
    INT16 - QENDCH=304


DELETE NAME;

    INT16 - 2
    INT16 - QDELET=237
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0


NAME := DECREMENT LEVEL_OF_DETAIL
          (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 2
    INT16 - QDECLV=134
```

```
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE


DEL NAME*;   (WILD CARD DELETE COMMAND)

    INT16 - 2
    INT16 - QDELW=57
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0


DISCONNECT SOURCE <OUT>:<INP> DEST;

    INT16 - SIZE+8
    INT16 - QALOOK=100
    INT16 - SIZE
    INT16 - 1
    ID    - SOURCE, SIZE
    INT16 - 0
    INT16 - 6
    INT16 - QFNOUT=144
    INT32 - OUT
    INT16 - SIZE+8
    INT16 - QALOOK=100
    INT16 - SIZE
    INT16 - 1
    ID    - DEST, SIZE
    INT16 - 0
    INT16 - 6
    INT16 - QINPIN=145
    INT32 - INP
    INT16 - 2
    INT16 - QDISCN=139


DISCONN SOURCE:ALL;

    INT16 - SIZE+8
    INT16 - QALOOK=100
```

```
    INT16 - SIZE
    INT16 - 1
    ID    - SOURCE, SIZE
    INT16 - 0
    INT16 - 2
    INT16 - QALLDS=219

DISCONNECT SOURCE <OUT>:ALL;

    INT16 - SIZE+8
    INT16 - QALOOK=100
    INT16 - SIZE
    INT16 - 1
    ID    - SOURCE, SIZE
    INT16 - 0
    INT16 - 6
    INT16 - QFNOUT=144
    INT32 - OUT
    INT16 - 2
    INT16 - QALLDS=219

DISPLAY NAME;

    INT16 - 2
    INT16 - QDSPOB=118
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0

END;

    INT16 - 2
    INT16 - QEND=106

END OPTIMIZE;

    INT16 - 4
    INT16 - QOPTIM=162
    BOOL  - .FALSE.
```

```
END_STRUCTURE;

    INT16 - 2
    INT16 - QENDOB=104


ERASE PATTERN FROM NAME;

    INT16 - SIZE+8
    INT16 - QERAPA=332
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0


NAME := EYE BACK DISTB
        LEFT/RIGHT DISTLR
        UP/DOWN DISTUD
        FROM SCREEN AREA WIDTH WIDE
        FRONT BOUNDARY = FRONT
 BACK BOUNDARY  = BACK
 (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 50
    INT16 - QEYE=155
    PSREAL- DISTLR
    PSREAL- DISTUD
    PSREAL- -DISTB
    PSREAL- WIDE
    PSREAL- FRONT
    PSREAL- BACK
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0
```

```
NAME := F:FNNAME;

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - SIZE + 12
    INT16 - QFLOOK=99
    INT16 - 0
    INT32 - 0
    INT16 - SIZE
    ID    - FNNAME, SIZE
    INT16 - 0


NAME := F:FNNAME (INOUTS);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - SIZE + 12
    INT16 - QPARFN=267
    INT16 - INOUTS
    INT32 - 0
    INT16 - SIZE
    ID    - FNNAME, SIZE
    INT16 - 0


FOLLOW NAME WITH TRANSFORMATION-OR-ATTRIBUTE COMMAND;

    INT16 - 2
    INT16 - QFOLLO=115
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
```

```
NAME := CHARACTER FONT FONTNM (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 2
    INT16 - QUFONT=131
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - FONTNM, SIZE
    INT16 - 0
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0


FORGET NAME;

    INT16 - 2
    INT16 - QFORG=113
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0


NAME := FIELD_OF_VIEW ANGLE
 FRONT BOUNDARY = FRONT
 BACK BOUNDARY  = BACK
 (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 24
```

```
    INT16 - QFOV=156
    PSREAL- ANGLE
    PSREAL- FRONT
    PSREAL- BACK
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0

NAME := IF CONDITIONAL_BIT BITNUM IS ONOFF
          (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 8
    INT16 - QCOND=174
    BOOL  - ONOFF
    INT32 - BITNUM
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0

NAME := IF LEVEL_OF_DETAIL COMP LEVEL
          (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 8
    INT16 - QCOND=174
    INT16 - (COMP + 2) * 256
    INT32 - LEVEL
    INT16 - SIZE+8
    INT16 - QNAME=45
```

```
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0


NAME := IF PHASE ONOFF (THEN APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 8
    INT16 - QCOND=174
    BOOL  - ONOFF
    INT32 - 15
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0


NAME := ILLUMINATION x,y,z,
      [COLOR hue[,sat[,intens]]]
      [AMBIENT ambien];

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 66
    INT16 - QLGHTS=355
    PSREAL- X
    PSREAL- Y
    PSREAL- Z
    PSREAL- 1.
    PSREAL- HUE
    PSREAL- SAT
    PSREAL- INTENS
    PSREAL- AMBIEN
```

```
INCLUDE NAME1 IN NAME2;

    INT16 - 2
    INT16 - QSETAD=125
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - NAME1, SIZE
    INT16 - 0
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - NAME2, SIZE
    INT16 - 0

PINIT:  INITIALIZE

    INT16 - 2
    INT16 - QINITN=121
    INT16 - 2
    INT16 - QINITD=122
    INT16 - 2
    INT16 - QINITL=293

INITIALIZE CONNECTIONS;

    INT16 - 2
    INT16 - QINITC=218

INITIALIZE DISPLAYS;

    INT16 - 2
    INT16 - QINITD=122

INITIALIZE NAMES;

    INT16 - 2
    INT16 - QINITN=121
```

```
NAME := INCREMENT LEVEL_OF_DETAIL
                (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 2
    INT16 - QINCLV=133
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0


NAME1 := INSTANCE (OF NAME2);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME1, SIZE
    INT16 - 0
    INT16 - 2
    INT16 - QUSE=120
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - NAME2, SIZE
    INT16 - 0
    INT16 - 2
    INT16 - QENDLS=107


NAME := LABEL X, Y, Z, 'STRING'
              :  :  :      :
              X, Y, Z, 'STRING';

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - LABBLK, SIZE
```

```
    INT16 - 0
    INT16 - 26
    INT16 - QDELTA=308
    PSREAL- STEPX
    PSREAL- STEPY
    PSREAL- 0
     THE NEXT 10 LINES FOR EACH LABEL
    INT16 - SIZE + 6
    INT16 - QDSTR=305
    INT16 - SIZE
    INT16 - 1
    STRING- LABEL, SIZE
    INT16 - 26
    INT16 - Q3DPCH=306
    PSREAL- X
    PSREAL- Y
    PSREAL- Z

    INT16 - 2
    INT16 - QENDCH=304


NAME := LIGHTPEN
              (ON/OFF)
              (CROSS ON/CROSS OFF)
              (BLAST ON/BLAST OFF)
              (SETXY LPX, LPY)
              (DELTA = DELTA)
              (RATE  = RATE)
              (APPLIED TO APPLY)

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 26
    INT16 - QLIGHT=336
    BBOOL - LPONF
    BBOOL - PCONF
    BOOL  - BLAST
    PSREAL- LPX
    PSREAL- LPY
    INT16 - DELTA
    INT16 - RATE
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
```

```
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0

NAME := LOOK AT AT FROM FROM UP UP (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 90
    INT16 - QLKAT=158
    PSREAL- FROM(1)
    PSREAL- FROM(2)
    PSREAL- FROM(3)
    PSREAL- 0
    PSREAL- AT(1)
    PSREAL- AT(2)
    PSREAL- AT(3)
    PSREAL- 0
    PSREAL- UP(1)
    PSREAL- UP(2)
    PSREAL- UP(3)
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0

NAME := MATRIX_2X2 (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 50
    INT16 - QMAT2=17
    PSREAL- MATRIX (1,1)
    PSREAL- MATRIX (1,2)
    PSREAL- 0
    PSREAL- 0
```

```
    PSREAL- MATRIX (2,1)
    PSREAL- MATRIX (2,2)
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0

NAME := MATRIX_3X3 (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 90
    INT16 - QMAT3=18
    PSREAL- MATRIX (1,1)
    PSREAL- MATRIX (1,2)
    PSREAL- MATRIX (1,3)
    PSREAL- 0
    PSREAL- MATRIX (2,1)
    PSREAL- MATRIX (2,2)
    PSREAL- MATRIX (2,3)
    PSREAL- 0
    PSREAL- MATRIX (3,1)
    PSREAL- MATRIX (3,2)
    PSREAL- MATRIX (3,3)
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0

NAME := MATRIX_4X3 MAT VEC (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 122
```

```
    INT16 - QMATRN=206
    PSREAL- MAT(1,1)
    PSREAL- MAT(1,2)
    PSREAL- MAT(1,3)
    PSREAL- 0
    PSREAL- MAT(2,1)
    PSREAL- MAT(2,2)
    PSREAL- MAT(2,3)
    PSREAL- 0
    PSREAL- MAT(3,1)
    PSREAL- MAT(3,2)
    PSREAL- MAT(3,3)
    PSREAL- 0
    PSREAL- VEC(1)
    PSREAL- VEC(2)
    PSREAL- VEC(3)
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0

NAME := MATRIX_4X4 (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 130
    INT16 - QMAT4=19
    PSREAL- MATRIX (1,1)
    PSREAL- MATRIX (1,2)
    PSREAL- MATRIX (1,3)
    PSREAL- MATRIX (1,4)
    PSREAL- MATRIX (2,1)
    PSREAL- MATRIX (2,2)
    PSREAL- MATRIX (2,3)
    PSREAL- MATRIX (2,4)
    PSREAL- MATRIX (3,1)
    PSREAL- MATRIX (3,2)
    PSREAL- MATRIX (3,3)
    PSREAL- MATRIX (3,4)
    PSREAL- MATRIX (4,1)
    PSREAL- MATRIX (4,2)
    PSREAL- MATRIX (4,3)
```

```
    PSREAL- MATRIX (4,4)
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0

NAME := NIL;

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 2
    INT16 - QMKNIL=236

OPTIMIZE STRUCTURE;

    INT16 - 4
    INT16 - QOPTIM=162
    BOOL  - .TRUE.

PATTERN NAME WITH PATNAM;

    INT16 - SIZE+8
    INT16 - QNAMPA=316
    INT16 - SIZE
    INT16 - 1
    ID    - PATNAM, SIZE
    INT16 - 0
    INT16 - SIZE+8
    INT16 - QAPPPA=333
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
```

```
NAME :=  [WITH [ATTRIBUTES attr] [OUTLINE r]]
   POLYGON [COPLANER] ( [S] x,y,z [N x,y,z] ) )
     :          :                :
        [[WITH [ATTRIBUTES attr] [OUTLINE r]]
   POLYGON [COPLANER] ( [S] x,y,z [N x,y,z] ) )];

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - SIZE+8
    INT16 - QWTATT=349
    INT16 - SIZE
    INT16 - 1
    ID    - ATTR, SIZE
    INT16 - 0
    INT16 - NVERTS * 8 + 4
    INT16 - QNORML=354
    INT16 - NVERTS
    VECNO - NORMS, VEDGES, DIMEN, NVERTS
    INT16 - NVERTS * 8 + 4
    INT16 - QPOLYG=318 OR QCOPOL=319
    INT16 - NVERTS
    VECNO - VERTS, VEDGES, DIMEN, NVERTS
    INT16 - 2
    INT16 - QEPOLY=320


NAME := POLYNOMIAL
             ORDER = ORDER
             (DIMEN IMPLIED IN SYNTAX)
             COEFFICIENTS = X(I),   Y(I),   Z(I)
                            X(I-1), Y(I-1), Z(I-1)
                              :       :       :
                            X(0),   Y(0),   Z(0)
             CHORDS = CHORDS;

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 14
    INT16 - QSTRTC=152
    INT8  - 2
    BBOOL - .FALSE.
```

```
    INT8 - ORDER
    BBOOL - .FALSE.
    INT8 - DIMEN
    BBOOL - (.TRUE.)
    BBOOL - (.TRUE.)
    BBOOL - .FALSE.
    INT32 - ORDER+1
     REPEAT ORDER+1 TIMES
    INT16 - 34
    INT16 - QCRVEC=296
    PSREAL- V (1,1)
    PSREAL- V (2,1)
    PSREAL- V (3,1)
    PSREAL- V (4,1)

    INT16 - 14
    INT16 - QENDCV=153
    INT32 - CHORDS
    PSREAL- 0

PREFIX NAME WITH TRANSFORMATION-OR-ATTRIBUTE COMMAND;

    INT16 - 2
    INT16 - QPREFX=114
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID   - NAME, SIZE
    INT16 - 0

NAME := RAWBLOCK SIZE (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID   - NAME, SIZE
    INT16 - 0
    INT16 - 6
    INT16 - QRAWBL=350
    INT32 - SIZE
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID   - APPLY, SIZE
    INT16 - 0
```

```
NAME := RATIONAL BSPLINE
        ORDER = ORDER
        OPEN/CLOSED
        NONPERIODIC/PERIODIC
        N = NVERT
        VERTICES        = X(1), Y(1), ( Z(1), ) W(1)
                          X(2), Y(2), ( Z(2), ) W(2)
                             :     :       :
                          X(N), Y(N), ( Z(N), ) W(N)
        KNOTS = KNOTS (1), ... KNOTS (NKNOTS)
        CHORDS = CHORDS;

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 14
    INT16 - QSTRTC=152
    INT8  - 1
    BBOOL - .TRUE.
    INT8  - ORDER
    BBOOL - .FALSE.
    INT8  - DIMEN+1
    BBOOL - (.NOT.OPNCLS)
    BBOOL - (.NOT.NONPER)
    BBOOL - .FALSE.
    INT32 - NVERT
      REPEAT NVERT TIMES
    INT16 - 34
    INT16 - QCRVEC=296
    PSREAL- V (1,1)
    PSREAL- V (2,1)
    PSREAL- V (3,1)
    PSREAL- V (4,1)

    (OPTIONAL)
      REPEAT NKNOTS TIMES
    INT16 - 10
    INT16 - QKNOT=295
    PSREAL- KNOTS (I)

    INT16 - 14
    INT16 - QENDCV=153
    INT32 - CHORDS
    PSREAL- 0
```

```
REMOVE NAME;

    INT16 - 2
    INT16 - QREMOB=119
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0

REMOVE FOLLOWER OF NAME;

    INT16 - 2
    INT16 - QUNFOL=117
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0

REMOVE NAME1 FROM NAME2;

    INT16 - 2
    INT16 - QSETRM=124
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - NAME1, SIZE
    INT16 - 0
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - NAME2, SIZE
    INT16 - 0

REMOVE PREFIX OF NAME;

    INT16 - 2
    INT16 - QUNPFX=116
    INT16 - SIZE+8
```

```
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0

NAME := ROTATE IN X ANGLE (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 10
    INT16 - QROTX=74
    PSREAL- ANGLE
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0

NAME := ROTATE IN Y ANGLE (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 10
    INT16 - QROTY=75
    PSREAL- ANGLE
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0
```

```
NAME := ROTATE IN Z ANGLE (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 10
    INT16 - QROTZ=76
    PSREAL- ANGLE
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0


NAME := RATIONAL POLYNOMIAL
        ORDER = ORDER
        (DIMENSION IMPLIED IN SYNTAX)
        COEFFICIENTS = X(I),   Y(I),   Z(I),   W(I)
                       X(I-1), Y(I-1), Z(I-1), W(I-1)
                          :       :       :       :
                       X(0),   Y(0),   Z(0),   W(0)
        CHORDS = CHORDS;

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 14
    INT16 - QSTRTC=152
    INT8  - 2
    BBOOL - .TRUE.
    INT8  - ORDER
    BBOOL - .FALSE.
    INT8  - DIMEN+1
    BBOOL - (.TRUE.)
    BBOOL - (.TRUE.)
    BBOOL - .FALSE.
    INT32 - ORDER+1
      REPEAT ORDER+1 TIMES
    INT16 - 34
    INT16 - QCRVEC=296
    PSREAL- V (1,1)
```

```
    PSREAL- V (2,1)
    PSREAL- V (3,1)
    PSREAL- V (4,1)

    INT16 - 14
    INT16 - QENDCV=153
    INT32 - CHORDS
    PSREAL- 0


RESERVE_WORKING_STORAGE Bytes;

    INT16 - 6
    INT16 - QRSVST=314
    INT32 - BYTES


NAME := SCALE BY V (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 26
    INT16 - QSCALE=164
    PSREAL- V(1)
    PSREAL- V(2)
    PSREAL- V(3)
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0


NAME := SET CONDITIONAL_BIT BITNUM ONOFF
        (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
```

```
    INT16 - 6
    INT16 - QSETBT=89 OR QCLRBT=90
    INT32 - BITNUM
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0

NAME := SET CHARACTERS SCREEN_ORIENTED/FIXED
        (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 6
    INT16 - QCHARP=253
    INT32 - 1
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0

NAME := SET CHARACTERS SCREEN_ORIENTED
        (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 6
    INT16 - QCHARP=253
    INT32 - 0
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0
```

```
NAME := SET CHARACTERS WORLD_ORIENTED
        (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 6
    INT16 - QCHARP=253
    INT32 - -1
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0


SETUP CNESS TRUE/FALSE <INP>NAME;

    INT16 - SIZE + 12
    INT16 - QCNESS=330
    INT16 - INP
    INT16 - 0 OR 1
    INT16 - 0
    INT16 - SIZE
    ID    - NAME, SIZE
    INT16 - 0


NAME := SET COLOR HUE,SAT (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 18
    INT16 - Q2COLR=167
    PSREAL- HUE
    PSREAL- SAT
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0
```

```
NAME := SET CONTRAST TO CONTRAST
        (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 10
    INT16 - QCONTR=232
    PSREAL- CONTRA
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0


NAME := SECTIONING_PLANE (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 2
    INT16 - QSECPL=315
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0


NAME := SET CSM ONOFF (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 3
    INT16 - QSECSM=303
```

```
    BBOOL - ONOFF
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0

NAME := SET DISPLAYS ALL ONOFF (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 4
    INT16 - QSCOPS=93
    BOOL  - ONOFF
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0

NAME := SET DEPTH_CLIPPING ONOFF (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 4
    INT16 - QDCLIP=95
    BOOL  - ONOFF
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0
```

```
NAME := SET DISPLAY N ONOFF (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 6
    INT16 - QSTDSP=235
    INT32 - N
    INT16 - 2
    INT16 - QDSCON=233 OR QDSCOF=234
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0

NAME := SET INTENSITY ONOFF IMIN:IMAX
        (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 20
    INT16 - QSTINT=301
    BOOL  - ONOFF
    PSREAL- IMIN
    PSREAL- IMAX
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0

NAME := SET LINE_TEXTURE PATTRN <AROUND> (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
```

```
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 6
    INT16 - QTXTUR=344 OR QCTXTR=345
    INT32 - PATTRN
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0


NAME := SET LEVEL_OF_DETAIL TO LEVEL
        (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 6
    INT16 - QLEVEL=88
    INT32 - LEVEL
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0


NAME := SET PICKING IDENTIFIER = PICKID
 (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 2
    INT16 - QPCKNM=110
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
```

```
    INT16 - 1
    ID    - PICKID, SIZE
    INT16 - 0
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0

NAME := SET PICKING LOCATION = XCENTR, YCENTR
                               XSIZE,  YSIZE
        (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 50
    INT16 - QPCKBX=194
    PSREAL- XCENTR
    PSREAL- YCENTR
    INT32 - 0
    INT32 - 0
    INT32 - 0
    INT32 - 0
    PSREAL- XSIZE
    PSREAL- YSIZE
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0

NAME := SET PLOTTER ONOFF (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 3
```

```
    INT16 - QSETPL=309
    BBOOL - ONOFF
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0


NAME := SET PICKING ONOFF (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 8
    INT16 - QPCKNG=91
    BOOL  - ONOFF
    INT32 - 0
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0


NAME := SET RATE PHAON PHAOFF INIONF DELAY
        (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 34
    INT16 - QBLDEF=205
    PSREAL- PHAON
    PSREAL- PHAOFF
    PSREAL- 1 OR 0
    PSREAL- DELAY
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
```

```
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0

NAME := SET RATE EXTERNAL (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 6
    INT16 - QSETBI=89
    INT32 - 15
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0

NAME := SET COLOR BLENDING BLEND
        (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 10
    INT16 - QCLRBL=321
    PSREAL- SAT
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0
```

```
SEND TRUE/FALSE  TO <INP> DEST;

    INT16 - 4
    INT16 - QBOOL=2
    BOOL  - B
    INT16 - SIZE+8
    INT16 - QALOOK=100
    INT16 - SIZE
    INT16 - 1
    ID    - DEST, SIZE
    INT16 - 0
    INT16 - 6
    INT16 - QINPIN=145
    INT32 - INP
    INT16 - 2
    INT16 - QSTORE=137


SEND FIX (I) TO <INP> DEST;

    INT16 - 6
    INT16 - QINTGR=3
    INT32 - I
    INT16 - SIZE+8
    INT16 - QALOOK=100
    INT16 - SIZE
    INT16 - 1
    ID    - DEST, SIZE
    INT16 - 0
    INT16 - 6
    INT16 - QINPIN=145
    INT32 - INP
    INT16 - 2
    INT16 - QSTORE=137


SEND M2D (MAT) TO <INP> DEST;

    INT16 - 50
    INT16 - QM2BLD=254
    PSREAL- MATRIX (1,1)
    PSREAL- MATRIX (1,2)
    PSREAL- 0
    PSREAL- 0
    PSREAL- MATRIX (2,1)
    PSREAL- MATRIX (2,2)
    INT16 - SIZE+8
    INT16 - QALOOK=100
```

```
    INT16 - SIZE
    INT16 - 1
    ID    - DEST, SIZE
    INT16 - 0
    INT16 - 6
    INT16 - QINPIN=145
    INT32 - INP
    INT16 - 2
    INT16 - QSTORE=137

SEND M3D (MAT) TO <INP> DEST;

    INT16 - 90
    INT16 - QM3BLD=255
    PSREAL- MATRIX (1,1)
    PSREAL- MATRIX (1,2)
    PSREAL- MATRIX (1,3)
    PSREAL- 0
    PSREAL- MATRIX (2,1)
    PSREAL- MATRIX (2,2)
    PSREAL- MATRIX (2,3)
    PSREAL- 0
    PSREAL- MATRIX (3,1)
    PSREAL- MATRIX (3,2)
    PSREAL- MATRIX (3,3)
    INT16 - SIZE+8
    INT16 - QALOOK=100
    INT16 - SIZE
    INT16 - 1
    ID    - DEST, SIZE
    INT16 - 0
    INT16 - 6
    INT16 - QINPIN=145
    INT32 - INP
    INT16 - 2
    INT16 - QSTORE=137

SEND M4D (MAT) TO <INP> DEST;

    INT16 - 130
    INT16 - QM4BLD=256
    PSREAL- MATRIX (1,1)
    PSREAL- MATRIX (1,2)
    PSREAL- MATRIX (1,3)
    PSREAL- MATRIX (1,4)
    PSREAL- MATRIX (2,1)
```

```
    PSREAL- MATRIX (2,2)
    PSREAL- MATRIX (2,3)
    PSREAL- MATRIX (2,4)
    PSREAL- MATRIX (3,1)
    PSREAL- MATRIX (3,2)
    PSREAL- MATRIX (3,3)
    PSREAL- MATRIX (3,4)
    PSREAL- MATRIX (4,1)
    PSREAL- MATRIX (4,2)
    PSREAL- MATRIX (4,3)
    PSREAL- MATRIX (4,4)
    INT16 - SIZE+8
    INT16 - QALOOK=100
    INT16 - SIZE
    INT16 - 1
    ID    - DEST, SIZE
    INT16 - 0
    INT16 - 6
    INT16 - QINPIN=145
    INT32 - INP
    INT16 - 2
    INT16 - QSTORE=137

SEND COUNT*DRAWMV TO <INP> DEST;

    INT16 - 6
    INT16 - QNBOOL=243
    BOOL  - DRAWMV
    INT16 - COUNT
    INT16 - SIZE+8
    INT16 - QALOOK=100
    INT16 - SIZE
    INT16 - 1
    ID    - DEST, SIZE
    INT16 - 0
    INT16 - 6
    INT16 - QINPIN=145
    INT32 - INP
    INT16 - 2
    INT16 - QSTORE=137

SEND REAL-NUMBER TO <INP> DEST;

    INT16 - 10
    INT16 - QREAL=4
    PSREAL- R
```

```
    INT16 - SIZE+8
    INT16 - QALOOK=100
    INT16 - SIZE
    INT16 - 1
    ID    - DEST, SIZE
    INT16 - 0
    INT16 - 6
    INT16 - QINPIN=145
    INT32 - INP
    INT16 - 2
    INT16 - QSTORE=137

SEND 'STR' TO <INP> DEST;

    INT16 - SIZE + 6
    INT16 - QSTR=5
    INT16 - SIZE
    INT16 - 1
    STRING- STR, SIZE
    INT16 - SIZE+8
    INT16 - QALOOK=100
    INT16 - SIZE
    INT16 - 1
    ID    - DEST, SIZE
    INT16 - 0
    INT16 - 6
    INT16 - QINPIN=145
    INT32 - INP
    INT16 - 2
    INT16 - QSTORE=137

SEND V2D (V) TO <INP> DEST;

    INT16 - 34
    INT16 - QVEC2=10
    PSREAL- V (1)
    PSREAL- V (2)
    PSREAL- 0
    PSREAL- 0
    INT16 - SIZE+8
    INT16 - QALOOK=100
    INT16 - SIZE
    INT16 - 1
    ID    - DEST, SIZE
    INT16 - 0
    INT16 - 6
```

```
    INT16 - QINPIN=145
    INT32 - INP
    INT16 - 2
    INT16 - QSTORE=137

SEND V3D (V) TO <INP> DEST;

    INT16 - 34
    INT16 - QVEC3=13
    PSREAL- V (1)
    PSREAL- V (2)
    PSREAL- V (3)
    PSREAL- 0
    INT16 - SIZE+8
    INT16 - QALOOK=100
    INT16 - SIZE
    INT16 - 1
    ID    - DEST, SIZE
    INT16 - 0
    INT16 - 6
    INT16 - QINPIN=145
    INT32 - INP
    INT16 - 2
    INT16 - QSTORE=137

SEND V4D (V) TO <INP> DEST;

    INT16 - 34
    INT16 - QVEC4=16
    PSREAL- V (1)
    PSREAL- V (2)
    PSREAL- V (3)
    PSREAL- V (4)
    INT16 - SIZE+8
    INT16 - QALOOK=100
    INT16 - SIZE
    INT16 - 1
    ID    - DEST, SIZE
    INT16 - 0
    INT16 - 6
    INT16 - QINPIN=145
    INT32 - INP
    INT16 - 2
    INT16 - QSTORE=137
```

```
SEND VALUE (VARNAM) TO <INP> DEST;

    INT16 - SIZE+8
    INT16 - QALOOK=100
    INT16 - SIZE
    INT16 - 1
    ID    - VARNAM, SIZE
    INT16 - 0
    INT16 - 2
    INT16 - QFETCH=186
    INT16 - SIZE+8
    INT16 - QALOOK=100
    INT16 - SIZE
    INT16 - 1
    ID    - DEST, SIZE
    INT16 - 0
    INT16 - 6
    INT16 - QINPIN=145
    INT32 - INP
    INT16 - 2
    INT16 - QSTORE=137

SEND VL (NAME1) TO <INP> NAME2;

    INT16 - SIZE+8
    INT16 - QALOOK=100
    INT16 - SIZE
    INT16 - 1
    ID    - NAME1, SIZE
    INT16 - 0
    INT16 - SIZE+8
    INT16 - QALOOK=100
    INT16 - SIZE
    INT16 - 1
    ID    - NAME2, SIZE
    INT16 - 0
    INT16 - 6
    INT16 - QINPIN=145
    INT32 - INP
    INT16 - 2
    INT16 - QSTORE=137
```

```
NAME := SOLID_RENDERING (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 2
    INT16 - QSOLRE=343
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0

NAME := STANDARD FONT (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 2
    INT16 - QSTDFO=132
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0

NAME := SURFACE_RENDERING (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 2
    INT16 - QSURRE=342
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0
```

```
NAME := TEXT SIZE SIZEX SIZEY (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 18
    INT16 - QTEXTS=339
    PSREAL- SIZEX
    PSREAL- SIZEY
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0

NAME := TRANSLATE BY V (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 26
    INT16 - QTRANS=73
    PSREAL- V(1)
    PSREAL- V(2)
    PSREAL- V(3)
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0

VARIABLE NAME;

    INT16 - SIZE+8
    INT16 - QVARNM=204
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
```

```
NAME := VECTOR_LIST (DOTS, CONNECTED, ITEMIZED, SEPARATE) N=N <VECTORS>;

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 45
    INT16 - Q2DVHD=147 OR Q3DVHD=148
    INT8  - 1 (DOTS)
  OR
    INT8  - 3 (CONNECTED, ITEMIZED)
  OR
    INT8  - 4 (SEPARATE)
    BBOOL - BNORM
    INT32 - 0
    PSREAL- 0
    PSREAL- 0
    PSREAL- 0
    PSREAL- 0
    INT32 - VECCOU
    BBOOL - CBLEND
  BLOCK NORMALIZED
    INT16 - 4+COUNT*2*DIMEN+2
    INT16 - QBNDAT=266
    INT16 - COUNT*2*DIMEN+2
    VBLNO - VECS, POSLIN, DIMEN, COUNT
  VECTOR NORMALIZED
    INT16 - 4+COUNT*(DIMEN+1)*2
    INT16 - QBNDAT=266
    INT16 - COUNT*(DIMEN+1)*2
    VECNO - VECS, POSLIN, DIMEN, COUNT

    INT16 - 2
    INT16 - QENDLS=107


NAME := VIEWPORT HORIZONTAL = XMIN:XMAX
        VERTICAL   = YMIN:YMAX
        INTENSITY  = IMIN:IMAX
        (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
```

```
    INT16 - 54
    INT16 - QVIEW=160
    PSREAL- XMIN
    PSREAL- XMAX
    PSREAL- YMIN
    PSREAL- YMAX
    PSREAL- IMIN
    PSREAL- IMAX
    INT32 - 0
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0

NAME := WINDOW X = XMIN:XMAX
        Y = YMIN:YMAX
        FRONT BOUNDARY = FRONT
        BACK  BOUNDARY = BACK
        (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 50
    INT16 - QWINDO=157
    PSREAL- XMIN
    PSREAL- XMAX
    PSREAL- YMIN
    PSREAL- YMAX
    PSREAL- FRONT
    PSREAL- BACK
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0
```

```
NAME := WRITEBACK (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 2
    INT16 - QWBACK=277
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0

NAME := CANCEL XFORM (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 2
    INT16 - QXFCAN=273
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0

NAME := XFORM MATRIX (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 2
    INT16 - QXFMAT=270
    INT16 - SIZE+8
    INT16 - QNAME=45
```

```
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0

NAME := XFORM VECTOR_LIST (APPLIED TO APPLY);

    INT16 - SIZE+8
    INT16 - QLABEL=44
    INT16 - SIZE
    INT16 - 1
    ID    - NAME, SIZE
    INT16 - 0
    INT16 - 2
    INT16 - QXFVEC=271
    INT16 - SIZE+8
    INT16 - QNAME=45
    INT16 - SIZE
    INT16 - 1
    ID    - APPLY, SIZE
    INT16 - 0
```

CUSTOMER INSTALLATION AND USER MANUAL

PS 300 ASYNCHRONOUS INTERFACE

EVANS & SUTHERLAND

June 1986
E&S #901194-078 A1

PREFACE

This manual contains two types of customer information: customer installation requirements and user information specific to the PS 300 asynchronous interface.

The asynchronous interface is the standard interface provided with the PS 300. No additional hardware installation is required to interface the host with the PS 300. Any E&S hardware information given in this manual is for reference only and should not become the basis for any unauthorized installation on your part.

Part I of this manual lists the installation requirements that you, as the customer, must meet before an Evans & Sutherland customer engineer can test the asynchronous interface as part of system installation. These requirements are summarized in the Customer Installation Checklist, which is at the beginning of Part I.

Part I also includes Chapters 1 and 2. Chapter 1 introduces the asynchronous interface and details the cabling and software requirements for host-to-PS 300 communication.

Chapter 2 provides installation instructions for the host-resident PS 300 I/O Subroutines (PSIOs) and the Graphics Support Routines (GSRs). It is intended for PS 300 customers using the DEC RSX-11M, DEC VAX/VMS, and DEC VAX/UNIX operating systems.

Part II of this manual contains user information for the PS 300 asynchronous interface. This is supplemental information which is helpful, but not essential, for installation. The **PS 300 Document Set** contains all the user information for the PS 300 systems.

Chapter 3 describes the asynchronous serial data characteristics for the PS 300. This includes port defaults and information on how to change port status and PS 300/host interface values.

Chapter 4 provides support information on the asynchronous interface, specifically transmission protocol and error detection. This information is needed for orderly data communication between the PS 300 and the host.

Chapter 5 describes the two ways in which data can be transported over an asynchronous line: in escape mode or count mode. The mode used is dependent on application and can be selected by the user.

Chapter 6 contains information on setting up a SITE.DAT file. The SITE.DAT file resides on the PS 300 graphics firmware and is used to configure the PS 300 for site-specific needs.

RELATED DOCUMENTS

PS 300 Document Set

The PS 300 Document Set contains system operation, programming, and system-management information. The information is organized into five volumes. Volume 3B of this set describes Pascal V2 and FORTRAN-77 Graphics Support Routines under the VAX/VMS operating system. C Graphic Support Routines are described in the on-line MAN subdirectory of the magnetic distribution tape.

Getting Started with the Unix™/PS 300 Software (E&S #901172-090)

This manual is intended for PS 300 customers using the VAX/4.2 BSD UNIX operating system. It provides a general description of the UNIX/PS 300 software distribution and installation procedures for the software.

PS 350 User's Manual (E&S #901172-092)

This manual provides information specific to the operation of the PS 350 and notes the differences, both hardware and software, between the PS 330 and the PS 350. This document is a supplement to the **PS 300 Document Set**.

CONTENTS

PART I

INSTALLATION INFORMATION

PART II

USER INFORMATION

TABLES AND FIGURES

PART I

CUSTOMER INSTALLATION INFORMATION

CUSTOMER INSTALLATION CHECKLIST

PS 300 ASYNCHRONOUS INTERFACE

The user of the PS 300 Graphics System is the supplier of all cabling and connectors to be used in the interface between the PS 300 and the host system. RS-232-C or RS-449 cables and the connectors are available through most major computer product supply centers.

Before the PS 300 asynchronous interface can be tested during system installation, you, as the customer, should be able to appropriately answer the following questions:

1. Do I have the correct RS-232-C cable with the appropriate 25-pin connectors or the correct RS-449 cable with the appropriate 37-pin connectors?

2. Is a null-modem cable configuration necessary to connect the pin signals correctly on either the RS-232-C or RS-449 interface?

3. Have I determined the appropriate cable length?

4. Do the cables running from the host to the PS 300 processor terminate with a female connector?

5. If I am operating under a DEC RSX-11M, DEC VAX/VMS, or DEC VAX/UNIX operating system, have I loaded the host-resident software supplied by E&S?

6. Does the host respond to X-ON/X-OFF characters sent from the PS 300? (If this requirement is not met, baud rate adjustment or buffer management may be required so that PS 300 buffers are not overflowed.)

7. Is the host able to send any 8-bit byte? (If this requirement is not met, the GSRs cannot be used, and applications are limited to sending ASCII commands that must be parsed.)

CUSTOMER INSTALLATION CHECKLIST

The decision to use shielded or unshielded cable is left to the user. Shielded cable is highly recommended in noisy environments, even though its higher capacitance per foot may reduce operating speed.

NOTE

To comply with FCC regulations, all RS-232-C and RS-449 cabling must be shielded and properly grounded.

1. INSTALLATION REQUIREMENTS

The asynchronous interface is the standard interface provided with the PS 300 Graphics System. With the exception of interface cabling and connectors, no additional hardware is required to interface the host with the PS 300.

As part of the PS 300 Graphics System installation procedure, an Evans & Sutherland Customer Engineer will be attaching interface cabling to the PS 300 and testing the asynchronous interface. Before the Customer Engineer arrives at your site, you, as the customer, must meet several requirements.

As the customer, you are responsible for running the host interface cabling from the host to the PS 300. This chapter details the site hardware and software requirements to do this. If you are operating under a DEC RSX-11M, DEC VAX/VMS, or DEC VAX/UNIX operating system, you are also responsible for the installation of the host-resident software (provided by E&S). This is detailed in Chapter 2.

Please read this manual carefully and follow the procedures outlined in the first two chapters BEFORE any installation procedures begin. If you have any questions regarding installation procedures or requirements, call the E&S Customer Service Center at: (800) 582-4375.

The first part of Chapter 1 also introduces some of the basic concepts of data communication which directly affect the interface to be set up between the PS 300 and the host. The standards for the EIA RS-232-C and the EIA RS-449 interfaces are provided.

Complete definitions of the system functions referred to in this chapter are found in Volume 5 of the **PS 300 Document Set.**

INTRODUCTION TO THE INTERFACE

One of the most important considerations in setting up the configuration characteristics of a PS 300 Graphics System is the interface between the host computer system and the PS 300. The standard data communication interface to the PS 300 is an asynchronous serial line. The terms "asynchronous" and "serial" refer to two important communication characteristics.

Binary data may be transferred between electronic devices in "serial," over a single line, or in "parallel," over several lines at once, by changes in current or voltage. In serial transmission, the bits that represent a character are sent down a single wire, one after the other. These serial signals are converted to parallel form at the reception end by shift registers.

Data transfers may be of a "synchronous" nature, where the exact bit framing of each byte of information is coordinated for the entire message by the transmission of two or more synchronization characters at the beginning of the message. All characters that follow these characters occur within a specific time frame called a "character time."

In data transfers of an "asynchronous" nature, each byte is self-defined by the use of a start bit and one or more stop bits. The start and stop bits occur before and after the byte of data. For this reason, this mode of transmission is referred to as "Start/Stop Transmission." In this mode, the arrival time of each character is random. Each end of the transmission line must know what the transmission rate is to sample the line at correct intervals following the receipt of a start bit.

Under PS 300 Graphic System protocol, RS-232-C and RS-449 are the standard interfaces used for serial asynchronous communication.

Two basic factors should be considered when you are deciding which standard of interface to use.

- Distance between the host and the PS 300
- Level of outside electrical interference

The RS-232-C is the standard of interface communication set by the Electronic Industries Association (EIA). Most interfaces are designed to this standard.

RS-449 was designed to replace RS-232-C as the standard. It supports the higher line integrity, and should be used as necessary (in conjunction with balanced RS-422) when noise levels or distance dictate.

The standard for both RS-232-C and RS-449 contains:

- The electrical signal characteristics.
- The interface mechanical characteristics.
- A functional description of the interchange circuits.
- A list of standard subsets of specific interchange circuits.

SITE HARDWARE REQUIREMENTS

RS-232-C Specifications

The physical connection between the PS 300 and the host is made through plug-in, 25-pin connectors (Cannon or Cinch DB Series). These connectors are keyed for 13 pins on the top row and for 12 pins on the bottom row. The PS 300 ports on the Communication Connector panel provide the male element for the interface. The pin assignments and signal definitions supported by the PS 300 are detailed in Appendix A of this manual.

RS-232-C standard states that the cable run distance between the data communications equipment should be no longer than 50 feet. However, longer cabling distances have been successfully used.

For the PS 300 EIA-RS-232-C communication ports, a Control-ON (logical 0), or "SPACE" condition exists if the voltage present is greater than +5 volts and less than +25 volts with respect to signal ground. A Control-OFF (logical 1), or "MARK" condition exists if the voltage present is less than -5 volts and greater than -25 volts with respect to signal ground. This assumes that the PS 300 signal ground and the communication data device signal ground are at the same potential.

RS-449 Specifications

RS-449 is a general purpose interface using a 37-pin connector. In conjunction with RS-422 (balanced) or RS-423 (unbalanced) communications, RS-449 incorporates new electrical characteristics that permit higher data signaling rates, allow greater distance between equipment, and reduce outside noise interference.

RS-449 and RS-232-C are highly compatible, but RS-232-C restrictions apply when used in conjunction with RS-449. Adapter connectors for modifying to modify the 37-pin RS-449 connector to the 25-pin RS-232-C connector are available through major computer product supply centers.

The connector pin definitions for EIA-RS-449 interface supported by Port 1 of the PS 300 Communication Connector panel are listed in Appendix A at the end of this manual.

RS-449 Port 1 on the PS 300 is capable of operating as RS-422 balanced communications, or as RS-423 unbalanced communications. Of the two standards, RS-422 allows the greatest distance and highest noise immunity, while maintaining high data transmission speed between the host and the PS 300. The RS-422 balanced voltage-circuit maintains two digital signals (A and B) as opposed to the single-digital signal used in the RS-423. For a "MARK" or OFF condition, the "A" terminal of the generator is negative with respect to the "B" terminal. For a "SPACE" or ON condition, the "A" terminal of the generator is positive with respect to the "B" terminal. As one signal in RS-422 goes to a SPACE state (ON condition), the corresponding signal goes to a MARK (OFF condition). The data is read by the receiver by comparing both signals.

The actual receivers used on the PS 300 have differential inputs. Their positive (+) inputs correspond to A terminals, and their negative (-) inputs correspond to B terminals. These receivers will accept a wide voltage range from +25 volts to -25 volts and meet RS-449 specifications.

The drivers used by the PS 300 have differential outputs with positive (+) corresponding to A terminals and negative (-) corresponding to B terminals. The output voltages range from +0.5 volts to +2.5 volts minimum. These meet RS-449 specifications.

SITE SOFTWARE REQUIREMENTS

The PS 300 Asynchronous Package contains the host-resident software you are required to install if you are operating under a DEC RSX-IIM, VAX/VMS, or VAX/UNIX operating system. It also contains the PS 300 firmware, Diagnostic Utility diskette, demonstration diskettes, and the Performance Verification Test. A packing list shipped with the software provides a complete list of the contents.

The PS 300 Asynchronous Package is included in the purchase price of the PS 300 and is shipped to the address indicated on the PS 300 sales checklist. Following is a list of the PS 300 Asynchronous Package numbers for ANSI host-resident software and UNIX tar host-resident software:

For PS 330 systems:

PS 330 Asynchronous Package
E&S #904150-001

PS 330 UNIX Asynchronous Package
E&S #904150-007

For PS 340 systems:	PS 340 Asynchronous Package E&S #904150-010
	PS 340 UNIX Asynchronous Package E&S #904150-017
For PS 350 systems:	PS 350 Asynchronous Package E&S #904150-020
	PS 350 UNIX Asynchronous Package E&S #904150-029

E&S INSTALLATION PROCEDURES

When you have completed all the customer installation requirements, as shown on the customer installation checklist, the Evans & Sutherland Customer Engineer can attach the interface cabling to Port 1 of the PS 300 and run the Interface Performance Verification Test (IPVT) for the PS 300 Asynchronous Interface. For systems that support the GSRs, this test verifies the communication link between the PS 300 and the host system.

The communication link is tested by:

1. Sending PS 300 binary commands via the asynchronous interface that will manipulate a surface defined by a grid using the dials and the function keys.

2. Sending a binary vector list to the PS 300 via the asynchronous interface.

3. Performing a data recirculation test by sending a variable-sized buffer, alternating between even and odd amounts of data, to HOST-MESSAGE and then routing it back to the PS 300. The text is compared after recirculation.

The IPVT has run successfully when the object (defined surface) can be manipulated using the control dials and function keys and there are no errors logged during the data recirculation test.

Upon successful completion of all tests, you, as the customer, will sign an Interface Performance Verification Test acknowledgement that initiates the 60-day product warranty.

2. HOST-RESIDENT SOFTWARE

This chapter contains the installation instructions for the host-resident PS 300 I/O Subroutines (PSIOs) and Graphics Support Routines (GSRs). It is intended for those PS 300 customers who use DEC RSX-11M, DEC VAX/VMS, and DEC VAX/UNIX operating systems.

PSIO INSTALLATION UNDER DEC RSX-11M

This section describes the installation of the host-resident PS 300 I/O Subroutines under the DEC RSX-11M operating system.

The magtape enclosed with the PS 300 Graphics Firmware contains the RSX-11 FORTRAN and MACRO source files necessary to Task Build with the user application programs.

The relevant file names and contents are listed below.

- PSIO.FTN - contains the subroutines PSSEND, PSREAD, PSPOLL, PSVECS, PSEXIT, PSETUP, PSCHAR, and PSFIXI.
- RSXPSIO.FTN - contains the second level subroutines that are machine dependent.
- RSXPSSER.MAC - contains the routines for communication to the serial line.

The process of compiling, task building, and running the FORTRAN and ASSEMBLER routines for running with the user programs is outlined below.

Compiling the FORTRAN Programs

The FORTRAN files should be compiled in the following manner:

```
>F4P PSIO=PSIO
>F4P RSXPSIO=RSXPSIO
```

or

```
>FOR PSIO=PSIO
>FOR RSXPSIO=RSXPSIO
```

using the format

```
user_program_object=user_program_source
```

Assembling the MACRO Code

The MACRO files should be assembled in the following manner:

```
>MAC RSXPSSER=RSXPSSER
```

using the format

```
user_program_object=user_program_source
```

Task Building (Linking) the Programs

The programs should be task built with the user application program in the following manner:

```
TKB>user_program_task_image=user_program_object,PSIO,
   RSXPSIO,RSXPSSER,
TKB>LB:[1,1]F4POTS/LB or LB:[1,1]FOROTS/LB
TKB>/
ENTER OPTIONS:
TKB>UNITS=10
TKB>ASG=serialline_id:m
TKB>//
```

where m is the logical unit number passed as the first argument in PSETUP. Refer to the description of the subroutine PSETUP in Volume 3b of the **PS 300 Document Set** for a description of the arguments.

NOTES

1. The explicit statement of a FORTRAN library is not required if the library is part of the system library SYSLIB.

2. F4P under RSX-11M has a facility for virtual array declaration. Do not declare the vector passed to PSVECS as a VIRTUAL array.

3. If the user is running RSX-11M on a system with floating point processor, the /FP switch must be included when task building.

   TKB> TEST/FP = TEST, ...

Example:

```
TKB>CUBES=CUBES,PSIO,RSXPSIO,RSXPSSER, LB:[1,1]F4POTS/LB
TKB>/
ENTER OPTIONS:
TKB>UNITS=10
TKB>ASG=TT2:7 (serialline_id)
TKB>//
```

In this example terminal port TT2: is assigned to logical unit number 7. It is assumed that the PS 300 is connected to TT2:.

Execution

To execute a user program task image that has been created by the task builder, submit a RUN command to the RSX-11M operating system as follows:

RUN user_program_task_image

Example: RUN CUBES

INSTALLATION UNDER DEC VAX/VMS

This section describes the installation of the PSIOs under VMS with the user programs that call the subroutines.

The magtape enclosed with the PS 300 Graphics Firmware contains the VMS FORTRAN and MACRO source files necessary to LINK with the user application programs.

The relevant file names and contents are listed below.

- PSIO.FOR - contains the subroutines PSSEND, PSREAD, PSPOLL, PSVECS, PSEXIT, PSETUP, PSCHAR, and PSFIXI.
- VAXPSIO.FOR - contains the second level subroutines that are machine dependent.
- VAXPSSER.MAR - contains the routines for communication to the serial line.

The process of compiling, linking, and executing the FORTRAN and ASSEMBLER routines for running with user programs is outlined below.

Compiling the FORTRAN Code

The FORTRAN files should be compiled in the following manner:

```
$FOR PSIO
$FOR VAXPSIO
```

Assembling the MACRO Codes

The MACRO code should be assembled in the following manner:

```
$MACRO VAXPSSER
```

Linking the Routines

The routines should be linked with the user program in the following manner:

```
$LINK user_program_object,PSIO,VAXPSIO,VAXPSSER
```

Example:

```
$LINK CUBES,PSIO,VAXPSIO,VAXPSSER
```

Allocating and Assigning the Serial Line

The serial line has the logical name PS. To run the program, the physical name must be assigned to the logical name.

Example:

```
$ALL TTA0:
$ASSIGN TTA0: PS
```

where TTA0: is the RS-232 serial line connected to Port 1 of the PS 300.

Refer to the description of the PSETUP routine in Volume 3b of the PS 300 Document Set on selecting the proper communication line at the time of program execution.

Program Execution

To execute a user program that has been created by the linking, submit the following command to the VMS operation system:

```
$RUN user_program_executable_image
```

Example:

```
$RUN CUBES
```

INSTALLATION OF THE FORTRAN GSRs UNDER DEC VAX/VMS

This section contains brief installation instructions for the DEC/VAX FORTRAN-77 version of the PS 300 Graphics Support Routines (GSRs). The GSRs will compile only under a FORTRAN-77 compiler and are supported under VMS Version 3.2 and higher. PS 300 Graphics Firmware Version P5.V03 or higher is required to run the Graphics Support Routines.

The GSRs are distributed on 1600-bpi magtape. The source files for the DEC VAX/VMS FORTRAN-77 version of the Graphics Support Routines are listed below along with a description of each file.

File Name	Description
GSRF.FOR	Source file for the GSRs.
PROFORLIB.FOR	Source file for the intermediate code between GSRF.FOR AND PROLIB.MAR.
PROIOLIB.MAR	Macro Source file for the low-level I/O subroutines used by the GSRs to communicate with the PS 300.
PROCOMF.FOR	Contains the global definitions of the FORTRAN-77 VAX GSR's and is INCLUDED by GSRF.FOR.
PROFORCOM.FOR	Contains the global definitions for the PROFORLIB.FOR.
PROCONST.FOR	Contains file that may be INCLUDed by the user in an application program. Contains the constant definitions.

The object module library containing all of the DEC VAX/VMS FORTRAN-77 GSR subroutines is contained in the file:

GSRF.OLB

To link your program with the DEC VAX/VMS FORTRAN-77 GSRs, enter the following command:

$ LINK <pgm>,<...any additional user object modules...>,GSRF/LIB

The files:

CIRCLEF.FOR

BLKLEVF.FOR

respectively contain the source code of two VAX FORTRAN-77 GSR programs demonstrating some of the subroutine calls. These will need to be compiled and linked with the GSR library.

To recreate the DEC VAX/VMS FORTRAN-77 GSRs from the original source files, the following DCL command sequence should be used:

```
$ FORTRAN GSRF
$ FORTRAN PROFORLIB
$ MACRO  PROIOLIB
$ LIBRARY/CREATE GSRF GSRF,PROFORLIB,PROIOLIB
$
```

General Notes

To read the GSR files off of the accompanying tape, first create a subdirectory for the PS 300 software and set your default to this directory. Using the VMS Backup Utility, enter the following commands:

```
$ ALLOCATE MTNN:
$ Mount/Foreign MTNN:
$ Backup MTNN:PSDIST.BCK[...]*.*
$ Dismount MTNN:
$ Deallocate MTNN:
$
```

where MTNN: is the physical device name of the tape drive being used. This will create the subdirectory A2V01.DIR which is the parent directory of the PS 300 host software.

READFOR.GSR contains a short description of each of the FORTRAN GSR files. The magtape contains many other files that are not related to the Graphics Support Routines.

INSTALLATION OF THE PASCAL GSRs UNDER DEC VAX/VMS

This section contains brief installation instructions for the DEC VAX PASCAL V2 version of the Graphics Support Routines (GSRs). The GSRs will compile only under a VAX PASCAL V2 compiler and are supported under VMS Version 3.2 and higher. PS 300 Graphics Firmware Version P5.V03 or higher is required to run the Graphics Support Routines.

The GSRs are distributed on 1600-bpi magtape (E&S #904015-004). The source files for the DEC VAX/VMS PASCAL V2 version of the Graphics Support Routines are listed below along with a description of each file.

File Name	Description
GSRP.PAS	Source file for the GSRs
PROPASLIB.PAS	Source file for the intermediate I/O procedures conceptually residing between GSRP.PAS AND PROIOLIB.MAR
PROIOLIB.MAR	Macro Source file for the low-level I/O procedures used by the GSRs to communicate with the PS 300
PROCONST.PAS	File that should be INCLUDEd by the user in an application program. Contains CONSTant definitions.
PROTYPES.PAS	File that should be INCLUDEd by the user in an application program. Contains TYPE definitions.
PROEXTRN.PAS	File that should be INCLUDEd by the user in an application program. Contains EXTERNal definitions.

The object module library containing all of the DEC VAX/VMS PASCAL V2 GSR procedures is contained in the file:

GSRP.OLB

To link your program with the DEC VAX/VMS PASCAL V2 GSRs, enter the following command:

$ LINK <pgm>,<...any additional user object modules...>,GSRP/LIB

The files:

CIRCLEP.PAS

BLKLEVP.PAS

contain the source code of two VAX PASCAL V2 GSR programs demonstrating some of the procedures. These will need to be compiled and linked with the GSR library.

To recreate the DEC VAX/VMS PASCAL V2 GSRs from the original source files, the following DCL command sequence should be used:

```
$ PAS GSRP
$ PAS PROPASLIB
$ MACRO  PROIOLIB
$ LIBRARY/CREATE GSRP GSRP,PROPASLIB,PROIOLIB
$
```

General Notes

To read the GSR files off of the accompanying tape, first create a subdirectory for the PS 300 software and set your default to this directory. Using the VMS Backup Utility, enter the following commands:

```
$ ALLOCATE MTNN:
$ Mount/Foreign MTNN:
$ Backup MTNN:PSDIST.BCK[...]*.*
$ Dismount MTNN:
$ Deallocate MTNN:
$
```

where MTNN: is the physical device name of the tape drive being used. This will create the subdirectory A2V01.DIR which is the parent directory of the PS 300 host software.

The file READPAS.GSR contains a short description of the Pascal GSR files. The magtape contains many other files that are not related to the Graphics Support Routines.

INSTALLATION OF THE C GSRs UNDER DEC VAX/UNIX

This section contains brief installation instructions for the DEC VAX/4.2 BSD UNIX and ULTRIX32 versions of the C Graphic Support Routines. Though dependence on 4.2 BSD UNIX features has been minimized, there is no guarantee that the PS 300 GSRs for VAX 4.2 BSD UNIX will be portable with other UNIX environments such as System V. PS 300 Graphics Firmware Version A1.V01 or higher is required to run the Graphics Support Routines. For more information on UNIX and the PS 300, refer to **Getting Started with the Unix™/PS 300 Software.**

The UNIX/PS 300 software is distributed on a 9-track, 1600-bpi magnetic tape in **tar** format. The software is organized into a hierarchy of directories.

To load the software, mount the distribution tape on the tape drive and create a directory in which to load the software. In the examples that follow, it is assumed that this directory name is: **/usr/ps300/dist.**

To create the directory and load the software, enter the following UNIX commands:

```
mkdir /usr/ps300/dist
cd /usr/ps300/dist
tar xv
```

At this point, all files are extracted from the tape and stored in the hierarchy of directories shown below:

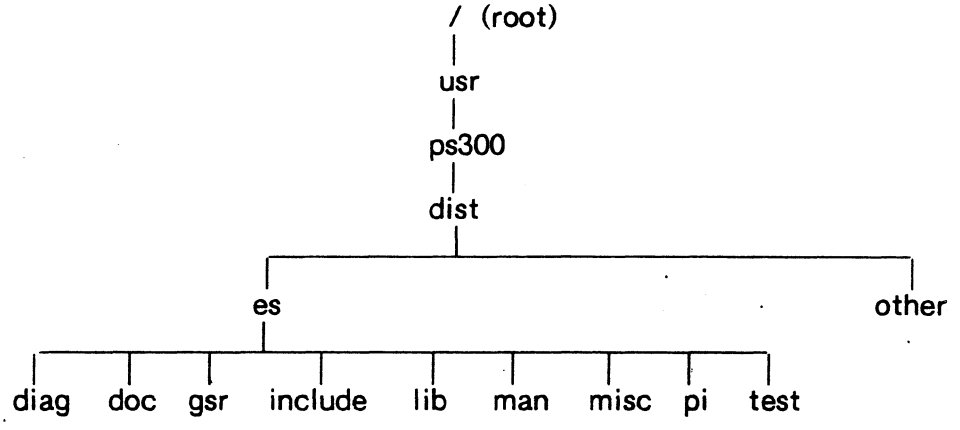


Figure 2-1. Hierarchy of Directories

The magtape contains many files not related to the Graphics Support Routines. README files exist in most of the directories which contain notes on the software contained in them.

A list of the GSR calls is available in the on-line MAN subdirectory. The C source files themselves reside in the GSR subdirectory. Refer to the TEST subdirectory for sample programs and a makefile demonstrating use of the C GSR library.

The object module library containing ALL of the DEC VAX/UNIX C GSR subroutines is contained in the file:

libgsr.a

in the LIB subdirectory.

To compile and link your program with the C GSRs, enter the following command:

%cc <pgm> <any additional user object modules...> /usr/ps300/dist/es/lib/libgsr.a

To recreate the C GSRs from the original source files, use the following commands:

```
%cd usr/ps300/dist/es/gsr
%make
```

PART II

USER INFORMATION

3. ASYNCHRONOUS SERIAL COMMUNICATION CHARACTERISTICS FOR THE PS 300

This section describes the serial I/O parameters the PS 300 Graphics System has defined for each port. The defaults (values assigned to each port when the system is powered on in standard configuration) for the data characteristics are listed in this section. For information on how these values can be configured in a bootable file on the PS 300 Graphics Firmware diskette, refer to Chapter 6 of this manual.

- The baud rates available on Ports 1, 3, and 4 on the PS 300 are: 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 9600, and 19200. Port 5, which is connected to the data concentrator, runs at 19200. Port 5 is not available for customer use.

- The PS 300 may be configured for 5, 6, 7, or 8 bits per character, although the host port must pass all characters of the 7-bit ASCII character set (i.e., 7 or 8 bits per character).

- Only one start bit will be accepted (and generated) by the PS 300.

- The PS 300 will accept (and generate) 1 or 2 stop bits.

- The PS 300 and the host can communicate using an X_ON - X_OFF protocol. In this protocol, control sequences are generated that tell the sender (either the PS 300 or the host) when to start (X_ON), or stop (X OFF) data transmission. These control sequence values default to Control-S (DEC 17 character) for X_ON, and Control-Q (DEC 19 character) for X_OFF. Under X_ON - X_OFF, bit stripping is controlled by the /MASK TO_7 BITS option.

  Additionally, there are available values for data characteristics that are unique to the X_ON X_OFF protocol. These values and their definitions are shown in the SETUP INTERFACE command section of Chapter 6.

- The PS 300 will run with even, odd, or no parity. Parity is a character checking device that operates by adding non-information bits to data, making the total number of ones in each grouping of bits either odd, for odd parity, or even for even parity. This permits error detection for an odd number of incorrect bits in each group.
- Each port may be configured to cause a trap to the PS 300 Debugger in the event a break is detected on that port.

- The PS 300 may be set to a maximum number of 127 buffers to hold data transmitted from the host. The default is eight buffers. Each buffer may be set to a maximum of 32,767 bytes, with the default at 48 bytes per buffer. This option allows the user to specify the amount of memory space to be allocated for data reception from the host.

  The user may specify the number free input buffers below which the host will be sent an X_OFF to suspend transmission. The number of free buffers above which the host will be sent an X_ON to resume transmission may also be specified.

Asynchronous Port Defaults

The defaults for Ports 1, 3, 4 and 5 are:

Port 1 - 9,600 baud, 8 bits per character, 1 stop bit, no parity, no_mask_to 7_bits, transparent mode. Sends all X_ON and X_OFF protocol characters, ignores incoming X_ON - X_OFF (no_hear_XON), 8 48-byte buffers with 0 STOP buffers and 1 GO buffer, and debug break disabled.

Port 3 - (debug port) 9,600 baud, 8 bits per character, 1 stop bit, no parity, non-transparent mode that accepts all X_ON and X_OFF protocol characters, 8 48-byte buffers with 0 STOP buffers and 1 GO buffer, and debug break enabled.

Port 4 - 300 baud, 8 bits per character, 1 stop bit, no parity, non-transparent mode that accepts all X_ON and X_OFF protocol characters, 8 48-byte buffers with 0 STOP buffers and 1 GO buffer, and debug break disabled.

Port 5 - 19,200 baud, 8 bits per character, 1 stop bit, no parity, transparent mode that does not recognize the X_ON X_OFF protocol characters, 8 48-byte buffers, and debug break disabled.

Port 5 is connected to the data concentrator and is not available for customer use.

The status of all the ports may be verified by using the SHOW INTERFACE command.

Changing Port Status

The following command sequence can be used to change any of the default values on Ports 1, 3, 4, and 5. These new values must be within the acceptable values for data characteristics as given in the previous section.

The port values are changed by entering the command:

SETUP INTERFACE <name>/<options>;

where name is the port being reconfigured, options refers to the option setting the communications interface. The command:

SHOW INTERFACE <name>;

where <name> is the port, can be used to check the values of a given port.

In using these commands, the ports names are as follows:

Port 1 is designated port10
Port 3 is designated port30
Port 4 is designated port40
Port 5 is designated port50 (not available for customer use)

The available options for SETUP INTERFACE are:

/SPEED=<baud rate> - input and output communications speed between 50 and 19200.

/EVEN_PARITY - establishes monitoring of parity on input and generation of parity on output, using EVEN parity.

/ODD_PARITY - establishes monitoring of parity on input and generation of parity on output, using ODD parity.

/NO_PARITY (default) - terminates the monitoring of parity on input and generation of parity on output.

/BITS_PER_CHARACTER=<number of bits per char> - sets the width of a character in bits (normally 8, even for 7-bit ASCII).

/STOP_BITS_PER_CHARACTER=<number of stop bits per char> - sets the number of stop bits for each character (normally 1).

/XON_XOFF - enables the PS 300 to use X_ON and X_OFF protocol to tell the host (or device) on this port to resume or suspend transmission. Default is to this protocol.

/NO_XON_XOFF - disables the use of XON and XOFF protocol from the PS 300 to the host (or device) on this port to resume or suspend transmission.

/HEAR_XON - enables the use of XON_XOFF protocol for the host (or device) on this port to tell the PS 300 to resume or suspend transmission. Default enables HEAR_XON.

/NO_HEAR_XON - disables the use of XON_XOFF protocol for the host (or device) on this port to tell the PS 300 to resume or suspend transmission.

/BREAK - enables the receipt of a BREAK on this port to call the ROM debugger.

/NO_BREAK - disables the receipt of a BREAK on this port to call the ROM debugger. Default is NO_BREAK.

/SPEED_EXTERNAL - sets the port speed to that of an attached modem, rather than from an internal clock. (This applies only to those ports with full modem support.)

/NO_SPEED_EXTERNAL - tells this port to use its internal clock, at the speed set by /SPEED=. Default is NO_SPEED EXTERNAL.

/BUFFERS=<number of buffers> - specifies the number of buffers in the input pool. Default is 8 buffers.

/BUFFER_SIZE=<number of bytes> - specifies the size of each buffer in the input pool. Default is 48 bytes.

NOTE

If input is received continuously, buffers will be filled until they are full. The buffer size will, in this case, specify the quantum of input being processed by subsequent functions.

If input is received at much less than the maximum baud rate, buffers will be released to waiting functions after 2 character times without receipt of a byte. In this case, the strict product of <buffer size> and <number of buffers> will not be the true amount of input buffering.

/N_STOP_BUFFERS=<number of buffers> - specifies the number of free input buffers below which the sender is told to suspend transmission. This has no effect unless the port is in /XON_XOFF mode. Default is 1 Stop Buffers. This is for host to PS 300 communication only.

/N_GO_BUFFERS=<number of buffers> - specifies the number of free input buffers above which the sender is told to resume transmission. This has no effect unless the port is in /XON_XOFF mode. Default is 2 Go Buffers.

The following four commands allow the user to specify non-standard X_ON-X OFF characters:

/SEND_XON_CHAR=<char code> - specifies the character code as an integer (defaults to decimal 17) to be sent out from the PS 300 to tell the sender to resume transmission. This has no effect unless the port is in /XON_XOFF mode.

/SEND_XOFF_CHAR=<char code> - specifies the character code as an integer (defaults to decimal 19) to be sent out from the PS 300 to tell the sender to suspend transmission. This has no effect unless the port is in /XON_XOFF mode.

/OBEY_XON_CHAR=<char code> - specifies the character code as an integer (defaults to decimal 17) that, when received by the PS 300, allows the PS 300 to transmit.

/OBEY_XOFF_CHAR=<char code> - specifies the character code as an integer (defaults to decimal 19) that, when received by the PS 300, stops the PS 300 from transmitting.

/MASK_TO_7_BITS - specifies that incoming bytes are to have their 8th bit, normally the parity bit, stripped off.

/NO_MASK_TO_7_BITS - (default) specifies that incoming bytes are not to be masked.

/BREAK_TIME=<break time> - specifies the length of time in centiseconds that an outgoing BREAK is to be held. This defaults to 10. Maximum = 127.

/ASYNCHRONOUS - normal mode of operation.

All commands are terminated with a semicolon (;) and a carriage return.

The menu available with the SHOW INTERFACE command lists only those parameters that are relevant to the interface; for example, in synchronous mode, the X_ON/X_OFF parameter would not be listed.

4. PS 300 TRANSMISSION PROTOCOL AND ERROR DETECTION

This chapter details the transmission protocol necessary to receive and transmit data over the asynchronous interface. It also provides a brief description of the three types of errors detected by the Enhanced Programmable Communications Interface (EPCI) status register.

PS 300 TRANSMISSION PROTOCOL

The PS 300 Graphics System uses an X_ON-X_OFF handshaking protocol to maintain orderly data communication over a full duplex, asynchronous, serial line between itself and the host computer. The receiver of X_OFF (dec 19) is to suspend transmission as soon as possible. The receiver of X_ON (dec 17) is being prompted to resume transmission until the next reception of X_OFF. The PS 300 will suspend transmission within one character time and can accept up to one buffer full of characters after X_OFF is sent.

The following equation shows how many bytes of an empty buffer are left when an X_OFF is sent. An X_OFF will be sent to the host that many bytes before input buffering is exhausted.

((Number of STOP buffers +1) * Number of bytes/buffer) - 1

Data Reception

The PS 300 defaults to eight 48-byte buffers available to receive data from the host computer. Transmitted characters are placed in the first free buffer starting in the first position and continue to the end of the buffer. When the buffer is full, the next available buffer is used. If all allocated buffers are full, the PS 300 will drop everything off the line until a buffer is free.

When the X_ON - X_OFF protocol is used, the PS 300 will send an X_OFF to the host (sender), when the number of free buffers is equal to the number of STOP buffers. The PS 300 will send X_ON to the host when the number of free buffers is equal to the number of GO buffers.

X_OFF received on the host input port disables data transmission from the host to the PS 300 until the PS 300 sends X_ON. If a host transmission aborts before X_ON is transmitted, or if the host transmits X_OFF as part of the LOGOFF message, it is necessary to manually clear the X_OFF condition. X_OFF is cleared, and the port re-enabled for transmission whenever a SETUP or SHOW INTERFACE command is executed.

Rebooting the PS 300 will also clear the X_OFF condition.

The default for Port 1 of the PS 300 is NO_HEAR_XON_XOFF when transmitting data.

Host to PS 300 Transmission Without X ON-X OFF

Operation without support of the X_ON - X_OFF protocol is discouraged. If X ON - X_OFF protocol is not available on the host, it is up to the user to ensure that an adequate number of buffers are allocated for data reception on the PS 300.

Transmission Errors

If the X_ON-X_OFF protocol is not used, and the number of available buffers is not large enough to hold the incoming data from the host (sender), data characters will be lost. These lost characters are detected and counted by the input routines. The SHOW INTERFACE command will give the current error counts for each port.

Messages characteristic of lost input characters are:

- PARSER SYNTAX ERROR due to bad syntax generated by the lost characters
- ERROR E 12 *** Message which function cannot handle

TRANSMISSION ERROR DETECTION

The Enhanced Programmable Communications Interface (EPCI) used on PS 300 Ports 1, 3, 4, and 5 is able to detect three types of transmission errors. When one of these transmission errors occurs, a bit is set in the EPCI status register where it can be read by the Graphics Control Processor. The errors detected are:

- Parity errors (if parity is enabled)
- Framing errors
- Overrun errors

The SHOW INTERFACE command will display all errors detected from the last PS 300 boot.

Parity Errors

The parity bit follows the character bits in data transmission. If there are 7 bits/characters, and parity is enabled, the total number of bits is 8 with the parity bit being the last transmitted bit. Ignoring the start bit and stop bit(s), the letter "A" when transmitted with EVEN parity would appear as follows:

lsb						msb	
1	2	3	4	5	6	7	parity
1	0	0	0	0	0	1	0

where "lsb" is the least significant bit and "msb" is the most significant bit.

The same character transmitted with ODD parity would look like this:

lsb						msb	
1	2	3	4	5	6	7	parity
1	0	0	0	0	0	1	1

EVEN parity sets the state of the parity bit such that the number of 1s in the 8 bits is an even number.

ODD parity sets the state of the parity bit such that the number of 1s in the 8 bits is an odd number.

If parity is enabled, the EPCI determines the parity of the received character and compares this parity with the parity bit transmitted. If they do not agree, the parity error flag is set in the EPCI status register.

From the example of the character "A", it can be seen that if the host and the PS 300 do not agree on the parity they are using, every character received or transmitted will generate a parity error.

This vertical error detection scheme can only discern an odd number of bit errors. For example, if bits 2 and 3 are erroneously changed to 1s, so that the character transmitted appears to be:

lsb						msb	
1	2	3	4	5	6	7	parity
1	1	1	0	0	0	1	0

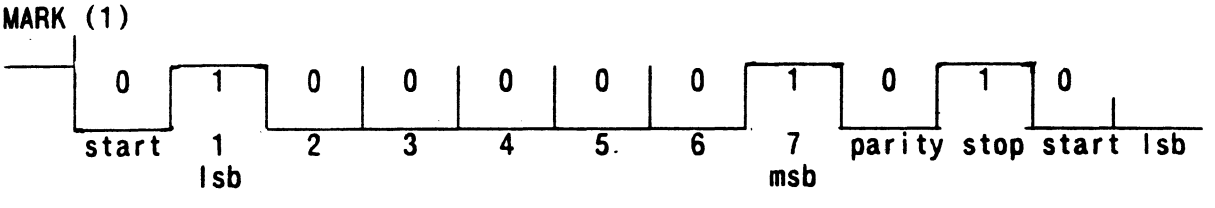
EVEN parity

the parity bit is correct for the character received ("G") but incorrect for the letter sent ("A").

The PS 300 supports ODD and EVEN parity, or NO parity.

Framing Errors

"Framing" is the process of determining which group of bits constitute a character. An error in this process is called a "framing error". Characters are framed by the start bit and the stop bit(s). Looking at the character "A" again (assume one stop bit):



The line is held in a MARK condition with current flowing when characters are not being transmitted. If for some reason the EPCI failed to detect the start bit when the signal goes to an ON, or SPACE condition, it is possible that it would assume bit 2 was the start bit, and bit 3 was the lsb, etc. At the time EPCI expected to see a stop bit, it would instead see the lsb of the next character, and a framing error would occur. When a framing error does occur, the EPCI sets the framing-error flag in the status register.

Overrun Errors

An overrun error occurs when the Graphics Control Processor (GCP) fails to read the characters in the holding register of the EPCI before the next character received is placed in the holding register. When this happens, the EPCI will overwrite the contents of the holding register with the next character. This overwrite causes the overrun error flag to be set in the EPCI status register.

5. METHODS OF COMMUNICATION

Volume 1 of the PS 300 Document Set describes the various methods of data communication that can be used over the PS 300/host line. These include standard ASCII transmission or E&S supplied host-resident software packages, the PSIOs and the GSRs. For those users whose host system cannot support the GSRs, refer to Chapter 5 of Volume 5 for a description of the routing bytes and their channels and to Chapter 13 for information on writing your own programs.

The host-resident software routines package data prior to sending it in binary format to the PS 300. These routines use count mode (described in the next section) and attach the routing bytes required to channel the data to the proper PS 300 system function . The data 'packets' built by the software routines include all the necessary information to process the data, and are in a form that is immediately acceptable to the PS 300 system function F:CIROUTE.

When the software packages are not used for host/PS 300 communication, F:CIROUTE still expects to receive data in a specific packet format . The PS 300 system functions that interface between the system and the hardware interface, (data reception functions, such as F:DEPACKET for asynchronous, ASCII communication) are responsible for building the data packets. These packets may be in either ASCII or binary, and for asynchronous communication, may be in either count or escape mode. The following section describes the count and escape modes and is of particular interest to those programmers not using the GSRs for data formatting.

DATA COMMUNICATIONS - ESCAPE AND COUNT MODE

Data is sent to the PS 300 from the host as a stream of bytes. These bytes must contain information that is intelligible to PS 300 system functions about the nature of the message and where it is to be sent internally in the PS 300. The following section explains the data transfer modes used in host/PS 300 communication and briefly describe the system functions that accept, examine, and route data internally in the PS 300.

Data may be transported over an asynchronous line in two modes: escape mode or count mode. The mode used is dependent on application and can be selected by the user. Count mode is the faster mode, as the system function, F:DEPACKET, that converts a stream of bytes into a stream of packets does not have to check the identity of each byte.

A system function, F:DEPACKET, accepts data input to the PS 300 from the host. F:DEPACKET converts a stream of bytes from the host into a stream of Qpacket/Qmorepacket. A Qpacket is a block of character data that can be sent from one PS 300 function to another. When data comes from the host through the F:DEPACKET function, it contains a byte for routing control. A Qmorepacket is a Qpacket that when coming from the host through F:DEPACKET, has no routing byte (i.e., a Qmorepacket has the same destination as the previous Qpacket.)

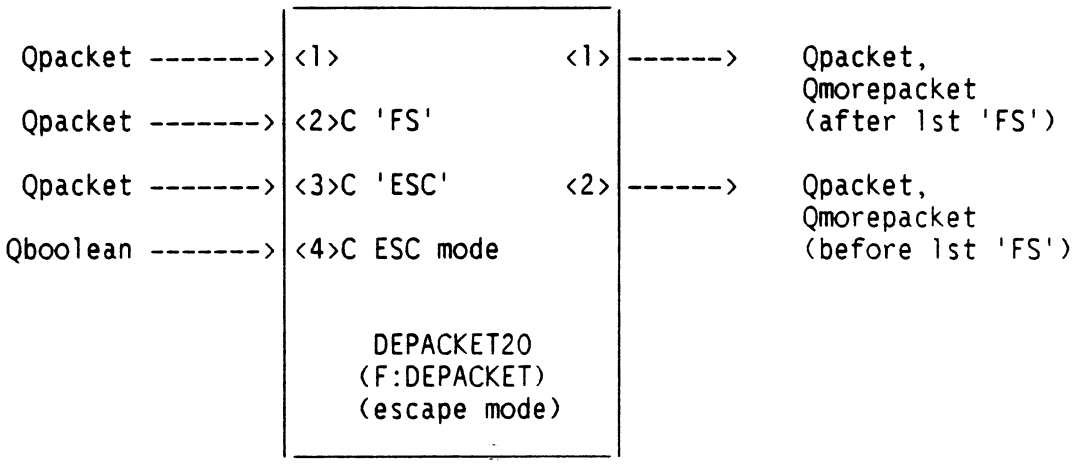
There are two instances of the F:DEPACKET function. The first, DEPACKET0, accepts all incoming bytes from the host on input <1>. It channels all incoming data through to output<2> until it sees the Start of Packet (SOP) character <ACK> (ACKNOWLEDGE - decimal character code 06 - ↑F) that signifies the start of a count mode packet.

All the data sent through to output<2> of DEPACKET0 are sent to input<1> of the second DEPACKET function, DEPACKET20, which then checks all incoming data for the SOP character <FS> (Field Separator - decimal character code 28 - ↑\) that signifies the start of an escape mode packet. It will also route all incoming bytes out output<2> until it sees the <FS> character. Output <2> of DEPACKET20 is connected to ES_TE1 (the screen).

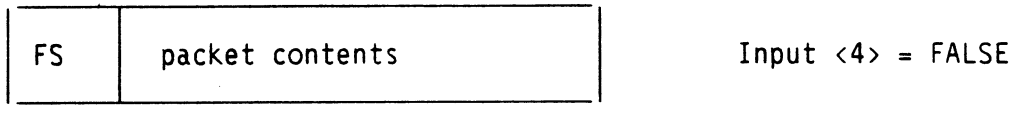
These instances of F:DEPACKET are described below. The characters that are used to signify SOP (<FS> and <ESC> characters) may be changed by the user by sending the new characters to the correct inputs of F:DEPACKET.

Escape Mode

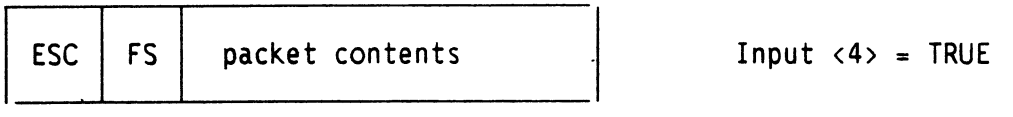
In escape mode, F:DEPACKET looks at every byte to see if it is a Start of Packet (SOP) character, which by default in escape mode is the ASCII Field Separator <FS> character, or an <ESC> character.



In escape mode, F:DEPACKET assumes that a packet is defined as either:



or



where <FS> represents the Start of Packet character that is by default the decimal character code 28 (↑\).

The definition of FS (one character) is taken from a single character Qpacket on input <2>.

In the first mode (input <4> = FALSE), any FS or ESC characters within the message packet must be escaped by prefixing them with an ESC character (i.e. the <ESC> character, decimal character code 16 (↑P)). Thus <ESC><x> becomes <x> for all values of x.

In the second mode (input <4> = TRUE), only ESC characters within the message packet must be escaped by prefixing them with an ESC character. The ESC character is defined by a single character Qpacket on Input <3>.

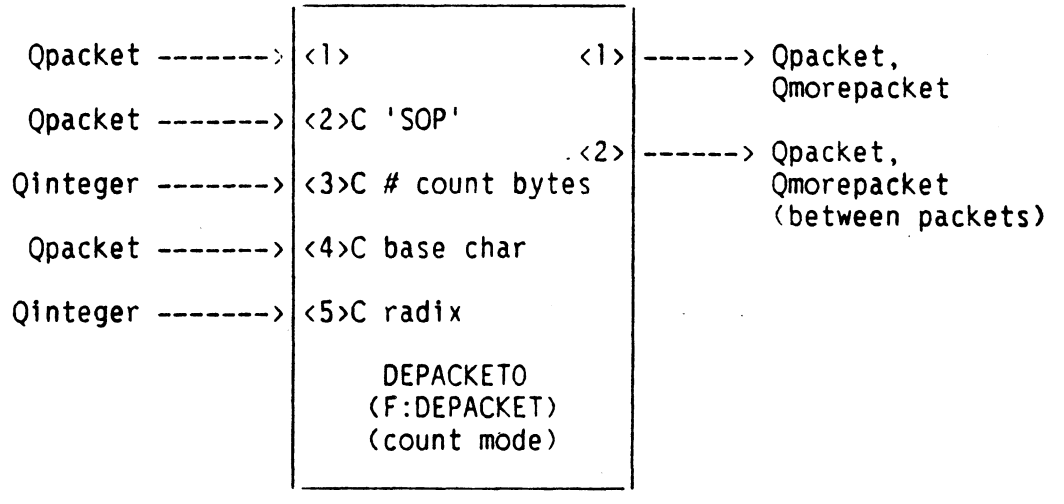
Output <1> outputs Qpacket and Qmorepackets of any messages after the first SOP control character is received. Output <2> outputs Qpackets and Qmorepackets of any messages before the first SOP control character is received. A Qpacket is output on Output <1> each time a SOP control character is received. Otherwise Qmorepackets are output.

Output <2> is normally connected to the Terminal Emulator Input and Output <1> is connected to F:CIROUTE for both Count and Escape Modes.

The routing path will be used for data transfer until the multiplexing function sees another <SOP> character, and a packet with another routing byte.

Count Mode

In count mode, once the SOP <ACK> character is seen, F:DEPACKET merely counts the bytes until the count is reached. No attempt is made to decode any bytes until the count is reached. Because F:DEPACKET does not examine the data, it is faster than escape mode, where all bytes are checked by F:DEPACKET to see if they are <FS> or <ESC> characters. Also, count mode allows for the use of any <SOP> or <ESC> sequences as part of the data.



In count mode, F:DEPACKET assumes that a packet is defined as:

ACK	count bytes	packet contents

where <SOP> represents the Start of Packet character that is by default the the ASCII ACK character, decimal character code 06 (↑F).

The definition of SOP (one character) is taken from a single character Qpacket on input <2>.

The message count is defined by n bytes (n defined by the Qinteger on input <3>). Each count byte is offset from the base character (the base character is taken from a single character Qpacket on input <4>). After the base character is subtracted, each count byte becomes a digit of the message count whose radix is defined by the Qinteger on input <5>.

Output <1> outputs Qpackets and Qmorepackets of count mode messages. Output <2> outputs Qpackets and Qmorepackets of any messages which are not in count mode.

The <SOP> byte and the count bytes are removed from the start of the packet before the packet is sent to F:CIROUTE, that does the actual routing.

6. THE SITE.DAT FILE

The final file in the PS 300 Graphics Firmware is the SITE.DAT file. Though it is not required, the SITE.DAT file enables the user to change default features for the PS 300 in a bootable file and, thus, to store information across power-up sequences. The SITE.DAT file is assumed to contain a stream of ASCII commands.

This chapter details the steps to be followed to create and download the SITE.DAT file. It also details the actual values and parameters that may be set or changed using SITE.DAT, including:

- port values
- <ESC> and/or <SOP> sequence characters
- keyboard features
- display features.

For details on how to change certain data packet and routing characters, refer to Volume 5, Chapter 5, of the **PS 300 Document Set**.

You may want to include a SITE.DAT file for the following reasons:

- To change Host-PS 300 communication defualt values
- To change Terminal Emulator features
- To implement specific PS 300 features that are not part of the standard firmware package.

CREATING AND DOWNLOADING THE SITE.DAT FILE

The SITE.DAT file can be written from the host system to the Graphics Firmware diskette, or it can be entered locally on the PS 300. To write to the diskette from the host follow the procedures below, or emulate the examples provided in this chapter that use the PS 300 Graphics Support Routines (GSRs). Refer to Volume 3 of the **PS 300 Document Set** for instructions on using the GSRs, and to Chapter 2 of this manual for instructions on installing the host-resident GSRs.

To create the SITE.DAT file:

1. Make a copy of the PS 300 Graphics Firmware on a scratch diskettte by following the procedures in Volume 5, Chapter 10, of the PS 300 Document Set.

2. Create a file on the host called SITE.DAT that contains PS 300 commands. These commands can be used to modify the default values of the Graphics Control Processor ports on the PS 300, as well as any data communication protocol procedures, so that the new values reflect site-specific requirements.

3. The file created must begin with the demuxing character and routing byte:

   ↑\:

   (↑ represents the CONTROL value of the \ key)

   that will route the file to the PS 300 Firmware diskette.

4. Next include the PS 300 commands you are using to configure your system.

5. The file must then include the demuxing character and routing byte:

   ↑\;

   followed by the command CLOSE SITE; that will close the file.

6. The file must end with the routing bytes to the TE:

   ↑\>

   The last command in the SITE.DAT file should cause some action on the PS 300 Display or attached terminal to signal the operator that the SITE.DAT file has been successfully loaded (e.g., "SYSTEM READY").

   Because of limited disk space, make the file as compressed as possible.

7. Mount the backup copy of the Firmware diskette and boot the system.

8. Copy the host-resident SITE.DAT file to the PS 300 backup diskette using the host copy utilities:

COPY	on VAX/VMS
PIP	on RSX-11
stty raw -echo; cat file_spec; stty cooked echo	on VAX/UNIX.

Refer to the manuals on the host system utilities for help in copying files from the host. Some examples have been provided in the following section.

NOTE

There may be special cases, such as loading a user-written function with a SITE.DAT file or copying mouse firmware to the PS 300 diskette, when a newly-created SITE.DAT can supercede the SITE.DAT file you already have on system diskette. For help on combining two SITE.DAT files, please call the Customer Service Hotline at: 1-800-582-4375.

Be sure the Write Protect tab is removed from the diskette. If the tab is in place and an attempt is made to write to the diskette, an error message will be displayed.

All commands in the SITE.DAT file must be terminated with a semicolon.

Name suffixing conventions must be used when system functions are referenced in the SITE.DAT file. Refer to Volume 5, Chapter 7, of the PS 300 Document Set for name suffixing conventions.

The following SITE.DAT example illustrate Steps 2-6 above:

```
↑\:
SEND CHAR(65) to <1>MESSAGE_DISPLAY;
↑\;
CLOSE SITE;
↑\>
```

Using the GSRs to Create the SITE.DAT File

Before using the GSRs, software source files must be loaded on the host and linked to the user application programs.

The following example shows how to create a SITE.DAT file using the IBM VS FORTRAN GSRs. Comments about the code appear in braces.

Two substitutions can be made in the following example for the VAX FORTRAN GSRs:

1. CALL PATTCH('LOGDEVNAM=TT:/PHYDEVTYP=ASYNC')
2. Use PPUTG instead of PPUTGX

Example 1

```
EXTERNAL ERR

CALL PATTCH('',ERR) {Enable PS 300/host communications - IBM ONLY}
CALL PMUXG(11,ERR)  {Define the channel to memory}
CALL PPUTGX('SEND CHAR(65) TO <1>MESSAGE_DISPLAY;',37,ERR)
   {Send characters out demuxing channel. The message between single
   quotes is a sample command to be included in the SITE.DAT file. This call
   must be repeated for each command that will be included in the SITE.DAT.}
CALL PMUXG(12,ERR)  {Define the channel to the diskette}
CALL PPUTGX('CLOSE SITE;'11,ERR)          {Close the file}
CALL PDTACH(ERR)    {Detach PS 300 host communications}

END

SUBROUTINE ERR(ERROR)

INTEGER*4 ERROR

WRITE(6,1) ERROR
FORMAT(' ERROR = ',15)
STOP

END
```

The next example illustrates how to create a SITE.DAT file using the IBM Pascal VS GSRs. Comments about the code appear in braces.

Two substitutions can be made in the following example for the VAX Pascal GSRs:

1. CALL PATTACH('LOGDEVNAM=TT:/PHYDEVTYP=ASYNC')
2. PPUTG instead of PPUTGX

Example 2

```
PROGRAM SITE(INPUT,OUTPUT)

CONST
%INCLUDE PROCONST

TYPE
%INCLUDE PROTYPES
%INCLUDE PROEXTRN

PROCEDURE ERR;

BEGIN;
WRITELN('ERROR IS: ', ERROR)
HALT
END;


BEGIN;
PATTACH(' ',ERR); {Enable PS 300 host communications-IBM only}
PMUXG(11,ERR)';  {Define the channel to memory}
PPUTGX('SEND CHAR(65) TO <1>MESSAGE_DISPLAY;'ERR);
   {Send characters out demuxing channel. The message between the single
   quotes is a sample command to be included in the SITE.DAT file. This
   procedure must be repeated for each command that will be included in the
   SITE.DAT.}
PMUXG(12,ERR);   {Define the channel to the diskette}
PPUTGX('CLOSE SITE;',ERR);              {Close the file}
PDETACH(ERR);    {Detach PS300 from host communication}

END.
```

The final example illustrates how to create a SITE.DAT file using the C GSRs.

Example 3

```
#include <sys/file.h>
#include <sys/types.h>
#include "../include/gsrext.h"

main(argc, argv)
```

```
int argc;
char *argv[];

{

int dd,cc,nbytes;
char buffer[80];

if( --argc <2 ){
  printf("Usage: site ps300device site_file/n");
            exit(1);
      }

PAttach(argv[1]);
PMuxG(11);

dd = open(argv[2],O_RDONLY,0);
if (dd < 1){
  printf("error trying to open file/n");
  PDetach();
  exit(-1);
}
cc = read(dd,bnuffer,80);

while(cc>0){
PPutG(buffer,cc);
cc = read(dd,buffer,80);
}

PMuxG(12);
PPutG("close site;",11);

PDetach();

}
```

CHANGING PS 300 PORT VALUES USING THE SITE.DAT

PS 300 port values may be changed to suit specific site requirements in two ways: the default values can be changed by using the SETUP INTERFACE commands in configuration mode, or the SETUP INTERFACE commands can be entered into the SITE.DAT file. If the value needs to be changed for just one session, so that the port will go back to its default values during the next boot-up, the SETUP INTERFACE command can be entered during a PS 300 session.

Should the new port value need to be installed more permanently, with the new value booted instead of the default, the SETUP INTERFACE commands should be entered into the SITE.DAT file.

Any of the SETUP INTERFACE commands can be entered in the SITE.DAT file, using the following forms:

SETUP INTERFACE portn/value;

where n is the port name and value is the name of the feature being set.

Example:

SETUP INTERFACE port10/XON_XOFF;

would enable Port 1 to use X_ON and X_OFF protocol to tell the host (or device) on this port to resume or suspend transmission.

SETUP INTERFACE portn/value=<p>;

where n is the port name, and <p> is the specified parameter.

Example:

SETUP INTERFACE port40/SPEED=2400/XON_XOFF;

would set Port 4 to a baud rate of 2400 and enable Port to use X_ON and X_OFF.

CHANGING THE <ESC>, AND/OR <SOP> SEQUENCE CHARACTERS IN THE SITE.DAT FILE

If the <ESC>, and/or <SOP> sequence characters used by E&S are incompatible with the host, or have another site-specific value, these characters can be changed by sending new values for these sequences to an instance of F:DEPACKET in the PS 300.

These new values MUST be included as PS 300 commands in the SITE.DAT file that is loaded during the system power-up. These commands should never be sent down from the host or entered in from the PS 300 keyboard during host transmission.

NOTE

If the <ESC> or <SOP> characters are changed in the SITE.DAT file, this change must be incorporated in the I/O Host-Resident Subroutines code for the PS 300, as the Subroutines use these same sequences for routing.

The PS 300 command for changing the escape mode <SOP> (default is <FS>, decimal character code 28, ASCII character '↑\') character is as follows:

SEND CHAR(I) to <2>DEPACKET20;

where I is the integer value corresponding to the new <SOP> character in escape mode.

The PS 300 command for changing the escape mode <ESC> character is as follows:

SEND CHAR(I) TO <3>DEPACKET20;

where I is the integer value corresponding to the new <ESC> sequence.

The count mode <SOP> character, (ASCII <ACK>, decimal character code 06 or ↑F), can be changed by sending the new integer value to <2>DEPACKET0:

SEND CHAR(I) TO <2>DEPACKET0;

CHANGING KEYBOARD AND DISPLAY FEATURES IN THE SITE.DAT FILE

The SITE.DAT can be used to set bootable values for the SETUP features of the Terminal Emulator (TE). This produces the same effect as changing the SETUP mode of the keyboard and display. For details on the Terminal Emulator, refer to Volume 5, Chapter 8 of the PS 300 Document Set.

TE characteristics are changed by sending a sequence of PS 300 commands to <1>KBhandler1. There are four groups of commands: toggles, breakkey, mode, and displays. Each group is handled differently.

Toggles

These TE options have two values: true and false, or on and off. In SETUP, you change a value to its opposite value by pressing a single Function Key.

The following chart gives the SETUP name, the definition, the default setting, and the PS 300 command sequence to change the default.

Setup Name	Definition	Default Setting	Sequence to Toggle
SRM	Local Echo	OFF	CHAR(22) & 'b'
Awrp	Automatic line wrap	OFF	CHAR(22) & 'c'
ANSI	ANSI sequences obeyed	ON	CHAR(22) & 'd'
VT52	VT52 mode	OFF	CHAR(22) & 'e'
KPM	Keypad Application Mode	OFF	CHAR(22) & 'f'
CKM	Cursor Key Mode	OFF	CHAR(22) & 'g'
Cnum	Keypad Numeric CI Mode	ON	CHAR(22) & 'h'
Knum	Keypad Numeric KB Mode	ON	CHAR(22) & 'i'

To put a command in the SITE.DAT file so that the TE feature is booted up with the desired value, you must sandwich the toggling sequence between a header and trailer sequence that represents the pressing of the SETUP key. The header and trailer sequence is: CHAR(22) & 'o'.

For example, to setup the TE for local echo (host is noecho) and for automatic line-wrap, the following command would be placed in the SITE.DAT file:

```
SEND CHAR(22) & 'o' & CHAR(22) & 'b' & CHAR(22) & 'c' & CHAR(22) &
'o' to <1>KBhandler1;
```

We recommend you use the E&S private escape sequences to set/reset the various TE modes in the SITE.DAT file. These commands are more compact and take less space on the diskette. For example, to set up the TE for local echo (host is noecho) and for automatic line-wrap, the folowing commands can be sent to ES_TE1:

```
Send CHAR(27) & '[>1;2h' to <1>ES_TE1;
```

Break Key

The BREAK key, like the toggles, must be sandwiched between sequences representing the SETUP key. It also has an inner sandwich, telling SETUP to enter and exit BREAK key. The important sequence inside these two sets of sequences represents the special key you designate to be the BREAK key.

For example, to set the HARDCOPY key as the BREAK key, place the following command in the SITE.DAT file:

```
SEND CHAR(22) & 'o' & CHAR(22) & 'j' & CHAR(22) & 'n' & CHAR(22) &
'a' & CHAR(22) & 'o' to <1>KBhandler1;
```

where:

CHAR(22) & 'o' is the SETUP header sequence (to enter SETUP)

CHAR(22) & 'j' is the sequence for Function Key #10 (to enter set/BREAK key mode)

CHAR(22) & 'n' is the sequence designating the HARDCOPY key as BREAK key

CHAR (22) & 'a' is the sequence for Function Key #1 (to exit set/BREAK key)
CHAR(22) & 'o' is the SETUP trailer sequence (to exit SETUP)

Mode

The PS 300 normally comes up in Terminal Emulator (TE) mode; that is, the keyboard outputs to the initial instance of ES_TE. Two other modes are available: Command Interpreter (CI) mode or Keyboard (KB) mode. To put the keyboard in one of these modes, enter one of the following sequences in your SITE.DAT file. Note that these do not have to be sandwiched between SETUP key sequences.

MODE	SEQUENCE
CI	CHAR(22) & CHAR(18)
KB	CHAR(22) & 'R'
TE	CHAR(22) & 'r'

To put the keyboard into interactive mode on bootup, enter the following in your SITE.DAT:

```
Send CHAR(22) & 'R' to <1>KBhandler1;
```

Displays

The two possible displays are the TE display and the Graphics display. These displays are normally on. They are toggled, respectively, by the TERM and GRAPHICS keys.

To turn the display off at boot time, use one of the following sequences in your SITE.DAT file:

DISPLAY	SEQUENCE
TE	CHAR(22) & 's'
Graphic	CHAR(22) & 'p'

For example, to turn the TE display off at boot time, place the following command in your SITE.DAT:

```
SEND CHAR(22) & 's' to <1>KBhandler1;
```

The only TE characteristic that cannot be conveniently set by a SITE.DAT file is the size and placement of the TE display.

SENDING CONTROL SEQUENCES TO THE TERMINAL USING THE SITE.DAT FILE

You may place control sequences that affect the screen display (as well as any other escape sequences) in the SITE.DAT file as ASCII sequences. The TE function ES_TE1 accepts and translates these sequences. Enter the escape sequence in the SITE.DAT in the following form:

```
SEND <char> n &'[P1;P2;...Pn' to ES_TE1;
```

where "[" is the control sequence introducer and P1 through Pn are the parameters that may or may not be present.

APPENDIX A. CONNECTOR PIN DEFINITIONS

Table A-1. RS-232-C Connector Pin Definitions

PIN NO	EIA LABEL	ABBREV. NAME	SIGNAL NAME	DIRECTION
1	AA	GND	Protective ground	N/A
2	BA	TXD	Transmit data	To DCE*
3	BB	RXD	Receive data	From DCE
4	CA	RTS	Request to send	To DCE
5	CB	CTS	Clear to send	From DCE
6	CC	DSR	Data set ready	From DCE
7	AB	GND	Signal ground	N/A
8	CF	DCD	Data carrier detect	From DCE
15	DB	TXCA	Transmit clock	From DCE
17	DD	RXC	Receive clock	From DCE
20	CD	DTR	Data terminal ready	To DCE
24	DA	TXCB	External transmit clock	To DCE

*Data Communication Equipment

RS-232 Signal Definitions

The following are definitions of the RS-232-C signals shown in Table A-1.

- AA, AB (Protective Ground and Signal Ground) - These two grounds are electrically independent. Protective Ground connects to the power ground. Signal Ground connects to the logic ground. No direct frame grounding occurs at the connector. Strict EIA RS-232-C standard definitions are not directly applicable.

- BA (Transmit Data) - Data from the PS 300 are transmitted on this line. The signal is generated by the PS 300 processor.

- BB (Receive Data) - Data are sent to the PS 300 on this line. The signal is passed to the PS 300 via the data communications equipment.

- CA (Request to Send) - This signal is generated by the PS 300 processor. The output may be programmed to conform with EIA-RS-232-C protocol. Generally, an "ON" CA (request to send) signal indicates the PS 300 processor is ready to transmit information.

- CB (Clear to Send) - This signal may be generated by data communication equipment. An OFF condition will terminate data transmission. An ON condition allows data transmission to resume. If no connection is made, an internal pull-up resistor will assert this line to an ON condition (+12V) for non-standard RS-232-C communication.

- CC (Data Set Ready) - This signal may be generated by data communication equipment. The function of this signal is controlled by software within the PS 300 processor. Usually, an 'ON' CC (data set ready) is sent by data communication equipment to indicate that data communications equipment is ready to transmit.

- CF (Data Carrier Detect) - This signal may be generated by a data communication equipment. ON assertion of this signal allows BB (receive data) to be accepted by the PS 300 processor. If no connection is made, this line will be pulled to an ON condition (+12V) to allow non-standard EIA-RS-232-C communication. To disable the BB (receive data) communication, an OFF condition must exist. Definition of this pin is software controlled for Port 1.

- CD (Data Terminal Ready) - This signal is generated by the PS 300 processor and is under software control. When asserted to an ON level, CD indicates that the PS 300 processor is ready to communicate.

- DA TXCB (Transmit Clock B) - This signal is generated by the PS 300 processor. DA provides a timing clock to indicate the center of each element of data. This timing clock can either be equal to the transmitted data frequency, or equal to 16 times the data frequency. DA TXCB is under software control . Port 1 of the PS 300 processor does not directly generate this signal. It relies on TXCA (transmit clock A) to generate this clock.

- DB TXCA (Transmit Clock BA) - This input signal is generated by external transmitting data communications equipment. This clocking signal input can control the rate at which the PS 300 processor transmits data out. The ability to use this clock input is software controlled.

- DD RXC (Receive Clock) - This input signal is generated by external transmitting data communications equipment. This clock determines the rate at which the PS 300 processor receives data. The ability to use this clock is software controlled.

Table A-2. RS-449 Connector Pin Definitions

PIN NO	TERMINAL	EIA LABEL	E&S LABEL	SIGNAL NAME	DIRECTION
1	A	Shield	GND	Signal Ground	N/A
4	A	SD	SDA	Send Data	To DCE*
5	A	ST	STA	Send Timing	From DCE
6	A	RD	RDA	Receive Data	From DCE
7	A	RS	RSA	Request to Send	To DCE
8	A	RT	RTA	Receive Timing	From DCE
9	A	CS	CSA	Clear to Send	From DCE
11	A	DM	DMA	Data Mode	From DCE
12	A	TR	TRA	Terminal Ready	To DCE
13	A	RR	RRA	Receiver Ready	From DCE
17	A	TT	TTA	Terminal Timing	To DCE
19		SG	GND	Signal Ground	N/A
20		RC	GND	Receive Common	N/A
22	B	SD	SDB	Send Data	To DCE
23	B	ST	STB	Send Timing	From DCE
24	B	RD	RDB	Receive Data	From DCE
25	B	RS	RSB	Request to Send	To DCE
26	B	RT	RTB	Receive Timing	From DCE
27	B	CS	CSB	Clear to Send	From DCE
29	B	DM	DMB	Data Mode	From DCE
30	B	TR	TRB	Terminal Ready	To DCE
31	B	RR	RRB	Receiver Ready	From DCE
35	B	TT	TTB	Terminal Timing	To DCE
37		SC	GND	Send Common	N/A

*Data Communication Equipment

RS-449 Signal Definitions

The following are definitions of the RS-449 connector pin signals shown in Table A-2.

- Shield, SG, SC, RC (Receive Common) - These four signals are electrically identical. They connect the PS 300 processor ground with the data communication equipment ground.
- SD (Send Data) - This signal is generated by the PS 300 processor as serial data. The following signals, when connected, need to be "ON" for transmission of serial data:

  - RS - Request to send
  - CS - Clear to send
  - DM - Data mode
  - TR - Terminal ready

Refer to the signal definitions of RS, CS, TR, and DM. If no external connection is made to CS and DM, internal pull-ups will allow SD to function.

- RD (Receive Data) - This signal is generated by data communications equipment as serial data. Receiver circuits are disabled if RR (receiver ready) is connected and in an OFF condition. If no connection is made, internal pull-ups will enable RR and allow reception of data, RD.

- RS (Request to Send) - This signal is generated by the PS 300 processor and is under software control. The RS (request to send) signal when ON indicates information is ready to send.

- RT (Receiver Timing) - This signal is generated by data communication equipment. The clock used is the rate at which the PS 300 processor receives data, signal RD. The use of this clock input is software controlled.

- CS (Clear to Send) - This signal may be generated by data communication equipment. An OFF state will terminate data transmission. An ON state allows data transmission to resume. If no connection is made, internal pull-up resistors will assert an ON state.

- DM (Data Mode) - This signal may be generated by data communications equipment. The function of this signal is controlled by software within the PS 300 processor. Usually, an ON DM signal indicates data communication equipment is ready to send.

- TR (Terminal Ready) - This signal is generated by the PS 300 processor and is under software control. When asserted to an ON condition, TR indicates that the PS 300 processor is ready to communicate, and that the data communication equipment should prepare for data reception.

- TT (Terminal Timing) - This signal is generated by the PS 300 processor and is under software control. TT provides the data communication equipment with transmitted data element timing that corresponds to the SD signal of the PS 300 processor. Port 1 of the PS 300 processor does not directly generate this signal. It relies on STB (Send Timing) to generate the clock.

- ST (Send Timing) - This signal is generated by external data communication equipment. The input clock can control the rate that the PS 300 transmits data (the SD signal) to the data communication equipment generating ST. The use of this input is software controlled.

E&S CUSTOMER SERVICE TELEPHONE INFORMATION LIST

Evans & Sutherland Customer Engineering provides a central service number staffed by CE representatives who are available to take requests from 9:00 a.m. Eastern Time to 5:00 p.m. Pacific Time (7:00 a.m. to 6:00 p.m. Mountain Time). All calls concerning customer service should be made to one of the following numbers during these hours. Before you call, please have available your customer site number and system tag number. These numbers are on the label attached to your PS 300 display or control unit.

Customers in the continental United States should call toll-free:

1 + 800 + 582-4375

Customers within Utah or outside the continental United States should call:

(801) 582-5847, Extension 4848

If problems arise during product installation or you have a question that has not been answered adequately by the customer engineer or the customer service center, contact the regional manager at one of the following Customer Engineering offices:

Eastern Regional Manager
(for Eastern and Central Time Zones)
(518) 885-4639

Western Regional Manager
(for Mountain and Pacific Time Zones)
(916) 448-0355

If the regional office is unable to resolve the problem, you may want to call the appropriate department manager at corporate headquarters:

National Field Operations
(for field service issues)
(801) 582-5847, ext 4843

Software Support
(for sofware issues)
(801) 582-5847, ext 4810

Technical Support
(for hardware issues)
(801) 582-5847, ext 4868

Director of Customer Engineering
(for any unresolved problem)
(801) 582-5847, ext 4840

Company _____

Department _____

Title _____

Name _____

_____

_____

_

READER COMMENT FORM **Publication Number** _____

Title _____

Your comments will help us provide you with more accurate, complete, and useful documentation. After making your comments in the space below, cut and fold this form as indicated, and tape to secure (please do not staple). This form may be mailed free within the United States. Thank you for your help.

How did you use this publication?

- [ ] General information
- [ ] Guide to operating instructions
- [ ] As a reference manual
- [ ] Other _____

Please rate the quality of this publication in each of the following areas.

	EXCELLENT	GOOD	FAIR	POOR
Technical Accuracy Is the manual technically accurate?	[ ]	[ ]	[ ]	[ ]
Completeness Does the manual contain enough information?	[ ]	[ ]	[ ]	[ ]
Readability Is the manual easy to read and understand?	[ ]	[ ]	[ ]	[ ]
Clarity Are the instructions easy to follow?	[ ]	[ ]	[ ]	[ ]
Organization Is it easy to find needed information?	[ ]	[ ]	[ ]	[ ]
Illustrations and Examples Are they clear and useful?	[ ]	[ ]	[ ]	[ ]
Physical Attractiveness What do you think of the overall appearance?	[ ]	[ ]	[ ]	[ ]

What errors did you find in the manual? (Please include page numbers)_____

_____

_____

_____

_____

_____

_____

_____

_____

Name _____ **Street** _____

Title _____ **City** _____

Department _____ **State** _____

Company _____ **Zip Code** _____

All comments and suggestions become the property of Evans & Sutherland.

Fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 4632 SALT LAKE CITY, UTAH

POSTAGE WILL BE PAID BY ADDRESSEE

EVANS & SUTHERLAND
580 Arapeen Drive
Salt Lake City, Utah 84108

ATTN: IAS TECHNICAL PUBLICATIONS

Fold

Cut along dotted line