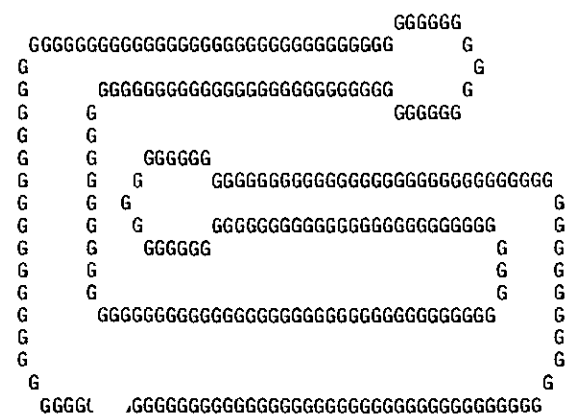
SUPERVISOR SBD User Manual

VOLUME 1 - Chapters 1-3

Introduction, Installation and Programming

Issue: 2.1 (December 1985)
Releases: A 2.1, B 2.1
Part No.: 4055022



GRESHAM LION (PPL) LTD LOWER WAY THATCHAM BERKSHIRE RG13 4RE U.K.
A DOWTY Group Company

The information in this document is subject to change without notice and should not be construed as a commitment by Gresham Lion (PPL) Ltd. Gresham Lion (PPL) Ltd assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished for use on a single Supervisor SBD installation and may not be used on other installations or for other purposes without the express permission of Gresham Lion (PPL) Ltd.

No responsibility is assumed for the use of such software on equipment not supplied by Gresham Lion (PPL) Ltd, or for the use of software not supplied by Gresham Lion (PPL) Ltd on any Supervisor installation.

The following are trademarks of Gresham Lion (PPL) Ltd.:

Supervisor 214	Supervisor 1024	Supervisor SBD
Supervisor PCU	Arcaid	GGS

The following are trademarks of Digital Equipment Corp.:

Q-bus	UNIBUS	RT-11
RSX	VAX	DEC
LSI-11	VMS	VT100
MicroVAX		

UNIX is a trademark of AT & T Bell Labs.
ID and Bit Pad One are trademarks of Summagraphics Corp.

C O N T E N T S - Volume 1

PREFACE

P.1 Objectives of the Manual

This manual describes the use of the Supervisor Single Board Display (SBD), which is an intelligent raster graphics display controller on a single DEC quad card. It contains all you should need to know about how to install, set up and use the device.

P.2 Intended Audience

Although the Guide contains information which is relevant to anyone who comes into contact with an SBD system, some parts of the Guide are aimed at people with specialist knowledge. For instance, section 2.2 on hardware installation requires some familiarity with DEC hardware configuration. The next part of the Preface breaks down the Guide section by section, and describes the assumptions made in each section about who will be reading it and what they should already know.

P.3 How to Find Your Way Around the Manual

This Guide contains nine chapters and four Appendices.

- Chapter 1 gives a brief outline of the SBD including its facilities, interfaces, attachments and the software and hardware options available with it. If you wish to assess the features and usefulness to your application of the SBD, or simply to get a general overview of the device and what it does, read this Chapter. (You may also find the command descriptions in 3.10 of use). Note: section 1.8 is the Specification.

- Chapter 2 tells you how to install your delivered SBD and turn it into a working graphics system. This includes how to physically install the board, connect all attachments and cables, and build the support software. This Chapter is aimed at the System Manager or equivalent, who should be familiar with the configuration of his DEC system and the adjustment of the various peripheral devices, as well as with the operating system of the host processor. Hardware and software installation may be performed by different people but they may be required to work together at times.

- Chapter 3 describes in detail the software library which is available to run in the host processor to support the SBD. This should be read by all system programmers and application programmers who are likely to be involved in designing and

writing applications using the SBD. In particular, Section 3.10 describes the functionality of all the commands recognised by the firmware, irrespective of whether the library is used.

- Chapter 4 describes the other devices which can be attached to the SBD, and their use (in outline). Manufacturer's handbooks should be consulted for specific details of the devices themselves. This chapter also describes the optional microprocessor based Peripheral Controller Unit (PCU), the various configurations in which it is used, and the protocol used by the SBD to communicate with it and with the various peripherals which can be attached to it.

- Chapter 5 specifies the format of the basic data structure used to drive the SBD, which is known as the Command Block (or Display List). This Chapter, and Chapters 6 and 7, are likely to be needed only by those who cannot use the standard Drivers or Libraries provided by Gresham Lion (PPL) Ltd) and who wish to write their own.

- Chapter 6 presents the details of how to control the SBD via its Direct Memory Access (DMA) interface, including details of the special software device drivers available, and contains all the device-specific information necessary to allow the writing of device drivers for operating systems not supported by Gresham Lion (PPL) Ltd.

- Chapter 7 specifies the syntax and protocol of the commands used to drive the SBD via an RS-232 serial line, in the two serial formats supported (7-bit human-readable ASCII and 8-bit compact binary).

- Chapter 8 describes the optional Ethernet interface to the SBD.

- Chapter 9 contains details of the terminal emulation offered by the SBD (DEC VT100). This Chapter will be of use to those intending to make special use of this emulation.

- Appendix A contains a list of error values which can be returned by the SBD or by host software, with their meanings and corresponding messages, if any.

- Appendix B contains FORTRAN program listings illustrating various programming techniques described in 3.7.

- Appendix C is an alphabetical quick-reference guide to the subroutines available with the host library. This is intended to be referred to as a 'memory-jogger' once the programmer is familiar with 3.10.

P.4 Related Documents

DEC VT100 User Guide

Summagraphics Bit Pad One Users Manual (FORM 64)

Summagraphics ID Data Tablet/Digitizer User's Manual (FORM 16)

Integrex Colourjet 132 GL Printer Manual

TDS LC-12 User Manual (TP 1069)

P.5 Conventions Used in the Manual

1. Non-printable ASCII characters (where explicitly required) are represented as follows in character strings:

Representation	Character	ASCII octal value
<CR>	carriage return	15
<LF>	line feed	13
<ESC>	escape	33
<NUL>	null	0
<SP>	space	40
<TAB>	horizontal tab	11

2. Certain keys on the keyboard are also represented in angle brackets:

<CTRL>	Control
<SHIFT>	Shift
<BREAK>	Break
<DEL>	Delete
<ESC>	Escape
<CR>	Return
<LF>	Line Feed
<BS>	Back Space

3. In sections describing user interaction with an operating system, user input is underlined, e.g.:

   MCR>LOA SB:

   In some cases this may obscure the underscore character but this should be obvious from the context.

4. Square brackets - [] - are used to delimit optional arguments in argument lists. Everything inside paired square brackets may be omitted.

5. Ellipsis (...) indicates a range of values or arguments, e.g. 1...4 means 1, 2, 3 and 4; L1,...L4, means "L1,L2,L3,L4,".

6. Where a non-character-generating key is shown immediately followed by a character-generating key, e.g. <SHIFT>S, <CTRL>Z, this implies that the two keys should be pressed simultaneously.

CHAPTER 1 - CONTENTS

CHAPTER 1 - CONTENTS (continued)

Figures

Tables

1.1 Product Concept

The Supervisor Single Board Display (SBD) graphics controller consists of a single quad size board which plugs into a Unibus or Q-bus SPC slot in a PDP-11 or VAX-11 computer. It is available in two versions, Model A and Model B. The board contains:

- Motorola MC68000 microprocessor, 8 or 12.5 MHz
- dual-port video memory (512 Kb dynamic RAM)
- static RAM for processor stack (4 Kb)
- video interface with colour mapping PALs (Model A) or LUT (Model B)
- DMA interface to host (optional)
- host serial interface (RS232)
- peripheral serial interface (RS232)
- highly functional firmware in EPROM (up to 128 Kb)
- non-volatile RAM to store setup parameters
- failsafe output timer (watchdog) - Model B only

The dual port video memory permits high speed graphics operation without visible disturbance of the picture. A 50Hz or 60Hz field rate provides flicker-free non-interlaced pictures.

With a keyboard connected to the peripheral serial interface, the system can function, via the host serial interface, as a DEC VT100 software-compatible VDU with scrolling text displayed on the screen, plus many VT100 features such as absolute cursor positioning, selective erase and so on. This feature, combined with the fast DMA interface, gives the SBD advantages over less intelligent displays which use graphics controller chips to achieve faster drawing rates.

The DMA interface is used to command the 68000 to generate raster graphics with a very small work load on the host and a high speed of response. This interface is only enabled when the board resides in the backplane of a host processor. A serial only version is available which can be incorporated within a monitor chassis; this version can also be supplied in a table-top or rack-mounting box (SBX) containing up to two SBDs (see Figure 1.3).

Gresham Lion (PPL) also provide two ready-to-use serial SBD terminal configurations:

- Supervisor Industrial Terminal (SIT) - a ruggedised free-standing heavy-duty monochrome (16 grey-level) terminal with detachable Cherry keyboard and optional clear or red screen filters, for industrial and military applications. This is also available in a monitor-only version (SIM) for use with DMA boards. See Figure 1.1.

- Supervisor Graphics Terminal (SGT) - a full-colour tilt-and-swivel monitor in attractive styled casing, plus low-profile DIN standard keyboard and full range of optional

peripherals, for office environments. This is also available in a monitor-only version (SGM) for use with DMA boards. See Figure 1.2.

An optional keyboard is available, which provides a VT100 layout with special display control keys. Also supported are several sizes of trackerball, a graphics tablet, a mouse, a digitizing table, a colour ink-jet printer, and a video frame buffer/hardcopier.

Each peripheral (except a mouse or tracker ball) can be driven when connected singly to the SBD's peripheral serial interface. (A serial converter (ISI) is required to connect a mouse or trackerball direct to the SBD). To connect multiple devices, a special purpose control unit (PCU) is available which is housed inside the keyboard. The frame buffer is attached to the video output of the SBD.

The two versions of the board (Model A and Model B) have slightly different hardware. Model B provides greater functionality. In summary, the main differences are:

Model A	Model B
Colour mapping performed by PAL chip - 4 different fixed mappings, 2 different PALs.	Colour mapping performed by look-up table (LUT) - 16 colours from palette of 4096.
Fixed white VDU colour	VDU colour selectable from palette of 4096.
No watchdog timer	Watchdog timer flashes or blanks the screen if no host activity - settable to any interval between 1 second and 8 minutes.
Power-up diagnostics	As Model A with addition of serial port fuse checking.
Processor runs at 8 MHz.	Processor runs at 12.5 MHz, giving an average 50% increase in drawing speed.
No Ethernet.	Serial-only versions can have optional Ethernet interface.

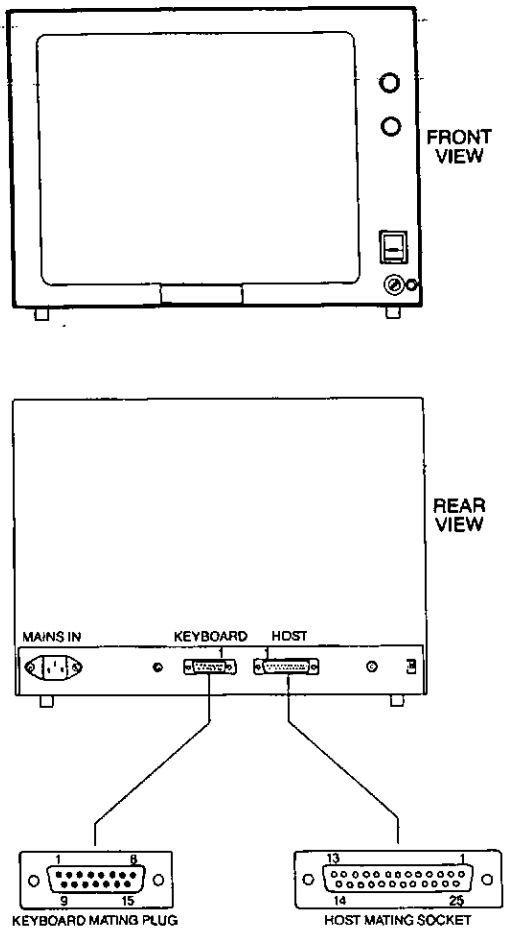


Figure 1.1 Supervisor Industrial Terminal (SIT)

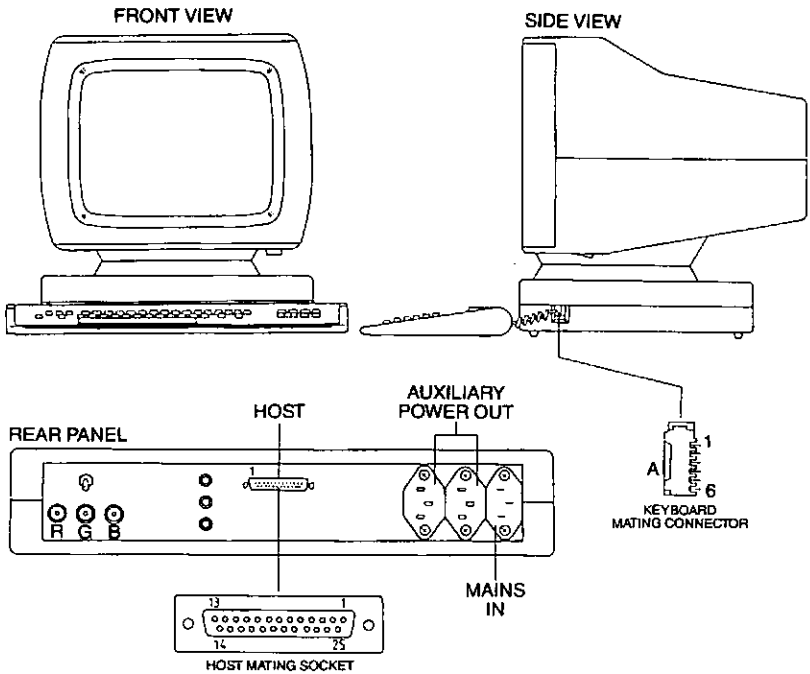


Figure 1.2 Supervisor Graphics Terminal (SGT)

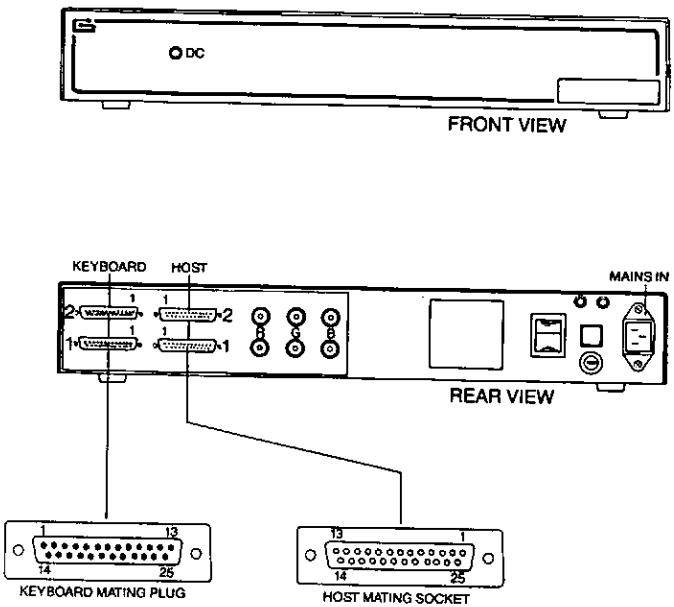


Figure 1.3 Supervisor S8D Serial Box (SBX) - Table-top Version

1.2 Facilities

Most of the functions of the SBD are held in firmware on the board. The host processor can cause these functions to be executed by sending commands via DMA, Ethernet, or one of two serial protocols. The commands are introduced in this section and described in detail in Chapter 3. Appendix C contains a quick command reference guide.

1.2.1 Graphics Resolutions

Four resolutions are supported by the standard SBD. Sufficient RAM memory is present to provide 2 pages each of 768 x 574 pixels resolution, or 4 pages of 512 x 384 pixels resolution (see Table 1.1). All but one of these resolutions require a high-frequency monitor for display.

A special-purpose SBD is available which gives lower resolutions with an increased number of pages (see Table 1.2). All these resolutions can operate on a standard 625 line, 15.6 KHz TV monitor.

Another version of the SBD is available for use with short-persistence monitors and for the American market. This version sacrifices some resolution to operate at 60Hz field rate. One resolution available on this board is compatible with the NTSC television standard.

For further details of selectable resolutions and monitor line frequencies, see 1.8 and the description of the SET command in Chapter 3.10. The commands PAG and SAP allow the use of different pages for display and update, and the ZOO command can be used to transfer pixel data between pages.

SET command base code		30	32	31	33
Horizontal Resolution	pixels	768	768	512	512
Vertical Resolution	lines	586*	574	384	384
Depth (Z axis)	bits	4	4	4	4
Non-interlaced		no	yes	yes	yes
Field Frequency	Hz	50	50	50	60
Line Frequency	KHz	15.6	30.0	20.1	24.2
Display Pages (max)		1/2*	2	4	4

* vertical resolution reduced to 574 with 2 pages.

Table 1.1 Available resolutions - standard SBD

SET command base code		2	4	3	5
Horizontal Resolution	pixels	768	512	384	256
Vertical Resolution	lines	586*	512	293	256
Depth (Z axis)	bits	4	4	4	4
Non-interlaced		no	no	yes	yes
Field Frequency	Hz	50	50	50	50
Line Frequency	KHz	15.6	15.6	15.6	15.6
Display Pages (max)		1/2*	3	8	13

* vertical resolution reduced to 574 with 2 pages.

Table 1.2 Available resolutions - low-resolution SBD

SET command base code		40	41	42*	43
Horizontal Resolution	pixels	768	704	640	480
Vertical Resolution	lines	574	526	480	360
Depth (Z axis)	bits	4	4	4	4
Non-interlaced		yes	yes	no	yes
Field Frequency	Hz	50	60	60	60
Line Frequency	KHz	30.0	33.2	15.75	23.8
Display Pages (max)		2	2	2	5

* NTSC television standard.

Table 1.3 Available resolutions - North American SBD

1.2.2 Output Modes and Channels

Each pixel has four bits of intensity/colour information, allowing up to 16 intensities/colours to be displayed simultaneously. The SBD is thus said to have four display memory "bitplanes".

There are two different models of the SBD, which achieve the mapping of bitplanes to visible colours in different ways.

SBD Model A contains a replaceable PAL (Programmable Array Logic) chip which provides four different mappings, or output modes, selectable by the SOM command. One of the mappings allows area flash, the frequency and mark/space ratio being selectable by a link on the board. The PAL can if necessary be tailored to the individual customer's requirements, but two different standard output PALs are available (see 1.8).

SBD Model B contains two 15-entry look-up tables (LUTs) which allow the user to map the colour values to any 15 colours (colour 0 must always be black) out of a colour palette (range) of 4096 colours, via the MAT command, and also to generate area flash (for all 15 LUT entries) between any two colours in the palette (with fully programmable mark/space ratio) using the FSH command. The SOM command is also available in Model B and provides preset mappings corresponding to those available from both standard types of output PAL in Model A.

SBD Model B also has a fail-safe output timer, or "watchdog". In the event of the 68000 processor ceasing to run, this will blank the screen instantly. It can also be set to automatically flash or blank the screen after a preset time with no commands received from the host. The preset time can be anything from 1 second to 511 seconds (8 minutes 31 seconds).

Individual bitplanes or combinations of bitplanes can be selected for writing. These combinations are referred to as "channels" and are selected by the SEL command, which also defines the foreground and background colour values to be written into the channel when a display primitive is called. The theoretical number of channels available is 15, the most useful 8 being predefined. The DEF command, used to define channels, is therefore available only to provide compatibility with other Supervisor displays.

The concept and use of foreground and background colours is explained in 3.7.4.1.

1.2.3 Graphics Display Commands

1.2.3.1 Line Drawing Primitives

Five line-drawing primitives are supported by the SBD. These are:

- o VEC - draw a straight line (vector).
- o BOX - draw a rectangle with sides parallel to the axes.
- o CIR - draw a circle or any combination of octants of a circle.
- o ELI - draw an ellipse or any combination of quadrants of an ellipse, with semi-axes parallel to the pixel X and Y axes.
- o BIT - draw a single pixel.

All these primitives are clipped correctly at the screen edges. The first four commands support line types (see 1.2.3.5).

The "current position" concept is supported which means that one X-Y coordinate pair can often be omitted, for instance when drawing chained vectors - the vector is drawn from the end-point of the previous vector.

1.2.3.2 Area Drawing and Raster Primitives

Five area-drawing primitives are supported by the SBD. These are:

- o BLK - draw a solid rectangle with sides parallel to the pixel X and Y axes.
- o CIF - draw a solid circle or any combination of solid octants (45 degree segments) of a circle.
- o ELF - draw a solid ellipse or any combination of solid quadrants (90 degree segments) of an ellipse, with semi-axes parallel to the pixel X and Y axes.
- o FLO - fill a polygon or random area with the foreground colour. The SEL command can be used to determine whether the fill boundary is the background colour or any colour which is not the background.
- o RUB - erase a rectangular area, sides parallel to the pixel axes, X start and X width multiples of 32, to the background colour.

The first three of these primitives will support line types (see 1.2.3.5) giving a 45 degree stripe effect. CIF and ELF draw stripes at plus-or-minus 45 degrees depending on the octant / quadrant. FLO also supports line types to a limited extent (the 45 degree stripe is not retained for a random shape, and the boundary must be solid).

In addition to these primitives, the commands WIB and RIB are provided which allow copying of rectangular blocks of pixels (sides parallel to the axes) from an encoded array in the host to the pixel area (WIB) or vice versa (RIB).

The ZOO command allows transfer of any rectangular area of pixels between any locations on any two pages, with optional magnification (by pixel replication) or reduction (by pixel sampling).

All these primitives are correctly clipped at the screen edges, producing a warning message which can be suppressed (see the INI command in 3.6 and the MES command in 3.10).

The current position can be used with BLK to allow the omission of a coordinate pair.

1.2.3.3 Character Strings

Three character fonts are available for use with pixel graphic text commands. These are 5x7, 7x9 and 11x16. Horizontal and vertical spacing are selectable, and any of four string orientations may be used. Single and double-height characters are supported.

All the above character attributes are selected using the SCA command. Actual display of characters is performed using the ALP command. Text may also be displayed magnified by an integer factor, up to 16 using the MAG command.

Strings can be displayed in replacement mode (cell erased to background colour) or additive mode (text only written). See the description of the SEL command.

Where a serial interface to the SBD is not available, it is still possible to make use of the VDU text display via the TXT command, which displays text on the VDU area in the normal VDU font. Attributes may be controlled by the normal VT100 escape sequences embedded in the text.

1.2.3.4 User-Defined Symbols

The SBD provides a facility for defining symbols of any size up to 16 x 16 pixels. Up to 26 such symbols may be defined. This gives in effect a user-definable character set which can be used to display single characters or whole strings.

Symbol attributes analogous to those for pixel text (such as orientation and cell size) can be defined using the SSA command. The SYM command is used for both defining symbols and displaying them.

1.2.3.5 Line Types

Lines do not have to be continuous and unbroken. A facility is provided by which you can define a 16-pixel pattern which repeats along any line, giving dotted lines, dashed lines or any combination. Any 16 bit pattern may be defined using the DLT command. Additionally, 8 useful patterns are predefined (including solid lines ie. no pattern), and these can be selected using the SLT command.

Most of the line and area drawing primitives support line types - in the case of area drawing, the pattern repeats horizontally but each line is shifted one pixel to the left relative to the line above. This gives a pattern of 45 degree stripes. If the repeat is sufficiently small the eye is unable to resolve the stripe and interprets the pattern as a new plain colour or "pseudo-colour".

1.2.4 Peripheral Interaction

Because the SBD allows connection of a variety of peripheral devices, commands have been provided which allow easy control of these devices.

A cross-hair cursor is provided under control of the CUR or TRA commands. The cursor can be moved around the screen by a trackerball or a mouse, and by the direction keys on the keyboard. The TRA and TRO commands allow the use of user-defined cursors and also provide the ability to use trackerballs and the mouse in a "sampling" mode.

The tablets are controlled using two commands: TIN (to set up the tablet transformations) and TAB (to allow control of the cross-hair cursor, or a user-defined cursor, from the tablet).

The ink-jet printer is activated by the DMP command which prints a copy of all or part of the pixel area in a variety of modes. Colour mapping of the printer can be varied using the SDM command - 36 different colours are available.

The frame buffer/hardcopier takes its input from the analogue video signals produced by the SBD and so operates independently of the firmware. The inkjet printer controlled by the frame store is of a similar type to the standard printer, but only provides 8 colours.

The keyboard contains special display control keys, labelled "VDU PIX" and "MIX", which operate the split-screen facility. The host commands PIX, VDU and MIX are available which allow software control of this facility.

1.2.5 System Control Commands

A variety of commands are provided to control the interfaces and functions of the SBD.

The commands SAS and RPC allow a "context save" by which the SBD's current operating parameters (channel, current position etc.) can be saved, and later restored no matter how they have been changed in the meantime. This feature allows independent programs to use the SBD without fear of interference with each other. These commands use archive space to save the context and are therefore included as part of the archiving option (see 1.2.9).

The LIX command controls the index registers, which define the logical screen origin (coordinate [0,0]). On power-up this is either at the top left or bottom left corner of the screen, depending on the power-up SET option specified. LIX can reset the origin, so that effectively the visible pixel area becomes a window on to a much larger virtual coordinate space (up to plus or minus 32767 in both X and Y directions). Primitives drawn outside the "screen window" are simply clipped.

The ASCII introducer character, which precedes all graphics commands using the serial line ASCII protocol, is by default a tilde(~) (ASCII value 176 octal). If this conflicts with user requirements (e.g. it is required to edit a text file containing tildes) the introducer character can be changed using the CHI command. It can also be changed or disabled completely via SETUP - see 2.3.

The SBD operates in several modes of error handling. The mode used depends on whether the host requires status returns, whether the VDU area is to be used for error messages, and the severity of the error. The MES command sets up all these error handling options.

Commands are provided by the host library which allow a task to communicate with more than one SBD unit. The INI command selects the mode of operation of the library, including the unit number or device name, the interface to be used, and the error handling options (INI transparently sends a MES command - see above - to the SBD). The FIN command terminates communication with one unit so that it can then call INI again to dynamically establish communication with another unit.

1.2.6 VDU Control and VT100 Features

The VDU display memory, which is completely separate from and independent of the pixel graphics display memory, gives 24 rows x 80 columns of standard 96 character ASCII plus the VT100 special graphics character set. The low-resolution version gives 64 characters per line at 384x293 and 256x256. The character cell depends on the currently selected resolution; the different cells are described in 1.8.

The VDU display basically emulates a DEC VT100 terminal. There are, however, several blank video lines between each VDU row. This means that the rows are separated vertically. This is only noticeable when reverse video or the VDU line-drawing special graphics characters are used. The VT100 features which are supported and those which are not are described in Chapter 9. It should be borne in mind that the SBD is intended as a powerful graphics display with VT100 features rather than a VT100 look-alike with graphics added on. The SBD will support the standard DEC screen editors, EDT and KED.

The SBD extends the set of VT100 facilities by providing a split screen facility controlled by additional keys on the standard keyboard or by commands (MIX, PIX and VDU) from the host. The screen may be split between pixel graphics in the upper portion of the screen and VDU text at the bottom. Any number of VDU lines may be displayed. The SBD may therefore be used as a VDU even while graphics update is taking place on the pixel area.

The SBD also provides printer port emulation, and a serial command disable facility, if a printer is connected. Escape sequences to control these functions are recognised. There is also an option to provide a second page of VDU memory, which can be used to save the visible page for later restoration.

1.2.7 Configuration Mode (SETUP)

The Supervisor SBD contains a powerful feature called SETUP, designed to assist in configuring the device as part of a computer system. In particular, the addresses of the DMA registers and the host interrupt vector, together with the host serial line characteristics, are configurable via a simple English-language setup menu, eliminating the need for on-board jumpers or Dual In-Line (DIL) switches.

The SETUP dialogue allows the user to configure the hardware and software to suit his application. The following are the parameters that can be configured:-

Host serial line baud rate, parity, data and stop bits.
DMA interface register address and interrupt vector.
Default character font and attributes.
Default symbol size and attributes.
Default video output mode.
Default resolution and memory pages.
Default channel, foreground and background colours.
Default VDU rows in split-screen mode.
Default ASCII introducer character.
Default watchdog timer setting (Model B only).

The specified configuration is saved in non-volatile memory and comes into effect each time the system is powered up or the host bus performs a reset. Most of the parameters can be over-ridden during operation by the normal graphics commands.

SETUP also provides a mode whereby, on entry of the appropriate password, software options and host application packages for which licences have been purchased may be enabled.

SETUP can be entered by pressing the <CTRL>, <SHIFT> and <BREAK> keys simultaneously on the SBD keyboard, or <SETUP> on the Cherry keyboard. If neither keyboard is available, most terminals operating at 9600 baud, which are capable of sending a break condition, can be used to drive the dialogue. The <BREAK> key is then used to activate SETUP.

While the dialogue is active, the SBD can be forced into any of its resolutions by a single key depression; this is useful for setting up monitors. The new parameters may be stored in NVRAM at any time during the dialogue, or the current stored parameters recalled. Exit from the dialogue is by pressing <ESC>. A full user guide to the SETUP dialogue is given in Chapter 2.3.

1.2.8 Power-up Confidence Tests and Diagnostics

On power-up the SBD executes a number of hardware diagnostic tests and displays a message on the bottom VDU line if the checks are satisfactory. If not, a diagnostic LED (light-emitting diode) is made to flash in a coded pattern giving information on the cause of the failure. Once the checks have proceeded far enough for a picture to be generated on the screen, diagnostic messages are displayed on the bottom VDU line, and the LED does not flash.

The checks which are carried out include:

- Static RAM test
- Dynamic RAM test
- NVRAM checksum
- EPROM checksum
- Serial communication tests
- DMA register tests
- Peripheral port fuse test (Model B only)
- Peripheral response tests

For a more detailed description of the diagnostic test sequence, see 2.2.6.1.

1.2.9 Local Segment Storage (Archiving)

The SBD has an optional extra feature whereby any pixel memory not used by the pixel pages can be used to store indexed sequences of graphics commands, called "segments" or "archives". A package of extra commands, called 'Arcaid', is provided to control this feature. Arcaid is a licensed product and must be enabled by means of a password (see 2.3.3) before the pixel memory can be used in this way.

The amount of pixel memory available for segment storage depends on the display resolution in use and on the number of pixel pages defined. This means that all archives are destroyed when these are altered via the SET command.

Archives are referred to by a number (between 0 and 65534) and are accessed through an internal directory structure. The IDI command initialises this directory, and it is also initialised on power-up or when SET is called.

The paired commands BAR (Begin ARchive) and EAR (End ARchive) allow the creation of archives. All commands executed without error between these two commands become part of the archive. The RAR (Recall ARchive) command causes all the commands in an archive to be executed. The RAR command itself may be archived, which allows hierarchical archives to be created. Commands are executed with the attributes (colour, line type, index registers etc) in force when RAR is called rather than those in force when the archive was created. If necessary, these attributes can be overridden by commands embedded within the archive.

Control over the archiving system is provided by the commands DAR (Delete ARchive) and DDI (Display DIrectory). DDI returns information about the number and size of archives.

1.3 Interfaces

1.3.1 Direct Memory Access (DMA) to Host

Using this interface, the on-board MC68000 microprocessor is commanded to generate graphics by a list of graphics instructions (command block) which resides in the memory space of the host. The 68000 accesses the host by becoming bus master and performing a DMA read. This activity takes typically less than 1% of the bus time and so the host performance is degraded by a negligible amount.

The DMA interface appears to the host as four 8 bit registers which respond to host bus addresses in the I/O page. These allow the host either to abort the current operation or to pass a command block pointer to the 68000 and cause it to begin execution of the list. The mechanism of the DMA interface is described in detail in Chapter 6.

The DMA interface operates at a peak speed of 4 words in 8 microseconds, or 1 Mbyte per second. This is, however, an arbitrary figure as read time is in all cases contained within the instruction execution time, and the normal limiting factor is the host operating system Executive I/O facility.

1.3.2 Serial Line to Host

This is a standard RS232/RS423 interface, capable of full duplex operation up to 19200 baud. Data rate control is by XON/XOFF protocol, or EIA modem control signals (CTS/DTR). Transmission format (data bits, stop bits, parity) is selectable using the SETUP configuration dialogue (see 1.2.7, 2.3).

Characters received from the host are displayed in the scrolling text store, and escape sequences interpreted according to normal VT100 conventions.

GGS format graphics commands (see Chapter 3) in both 7-bit ASCII (human-readable) and 8-bit compressed binary protocols can be mixed in with text for the VDU area. The SBD separates the commands from the text and executes the commands without requiring reconfiguration.

1.3.3 Serial Line to Peripheral

The other serial port on the SBD is used to connect one of a range of peripheral devices (keyboard, colour inkjet printer, tablet, trackerball or mouse). The trackerballs and mouse require an Intelligent Serial Interface unit (ISI) to connect directly into the SBD.

If required, more than one of these peripherals can be attached at once by the use of a Zilog Z80A microprocessor-based control unit housed in the keyboard. This peripheral control unit (PCU) functions as a switching unit to control the flow of data to and from the various peripherals. Any peripheral can be connected or disconnected at will and the SBD will automatically reconfigure itself when reset. However, those peripherals taking their power from the SBD (the PCU or the standard keyboard) cannot be disconnected from the SBD port unless the power is off. Use of the PCU allows the trackerballs and mouse to be used without an ISI.

1.3.4 Ethernet

The Ethernet interface is an optional alternative to the DMA interface, and provides a fast link to a host without the DMA capability (i.e. not a Q-bus or UNIBUS host). The interface is supplied as a piggy-back printed circuit board (PCB) which attaches to four mounting holes on the SBD (available on Model B only) and plugs into the socket for the 68000 processor, which is relocated onto the Ethernet board. The protocol implemented is a series of private messages which are summarised in Chapter 8.

The Ethernet interface will be available in early 1986, when the sections of this manual relating to it may be re-issued.

1.3.5 Video Signal Outputs

The red and blue outputs carry the colour components without sync pulses. The green output carries the green colour component (or the monochrome output) with composite mixed syncs. Video output is via ribbon cable from the 16-way IDC connector.

An extender box or panel is supplied which provides BNC connectors to a monitor but connects via ribbon cable to the SBD.

Model B SBDs can be externally synchronised at a small extra cost.

1.4 Peripheral Devices

The paragraphs in this section give a brief description of the peripherals available for use with the SBD. For a more detailed discussion, refer to Chapter 4. Contact Gresham Lion (PPL) about any further problems.

1.4.1 Keyboard

Connecting the keyboard to the Single Board Display allows it to become a DEC VT100 compatible VDU. The keyboard is a low profile design. It can include a Peripheral Control Unit, which is a switching device used to connect logically any one of 5 possible peripherals to the Single Board Display. A schematic diagram of a complete system which includes keyboard and PCU is shown in figure 1.4.

1.4.2 Trackerballs

The trackerball, like the digitising tablet and mouse, is a device for positioning a cursor on a graphics display. The unit remains static and the ball is rotated, causing the cursor to move in a similar direction to the rotation. One type of trackerball (2.25 inches ball diameter) is supplied in a plastic case which has three push buttons. These can be assigned specific functions. Other sizes of trackerball can be supplied, but without casing or push-buttons. Trackerballs may be connected via a PCU (in combination with other peripherals) or singly via an Intelligent Serial Interface (ISI) which converts the parallel TTL signals to serial RS232.

1.4.3 Mouse

The mouse can be considered as an upside-down trackerball. The mouse casing is held in the hand, and the ball is rotated by sliding the mouse along a suitable smooth surface. The same restrictions on connection to the SBD apply as for the trackerballs, and in fact the electrical (and software) interfaces are identical. The mouse has three buttons corresponding to those on the two-inch trackerball.

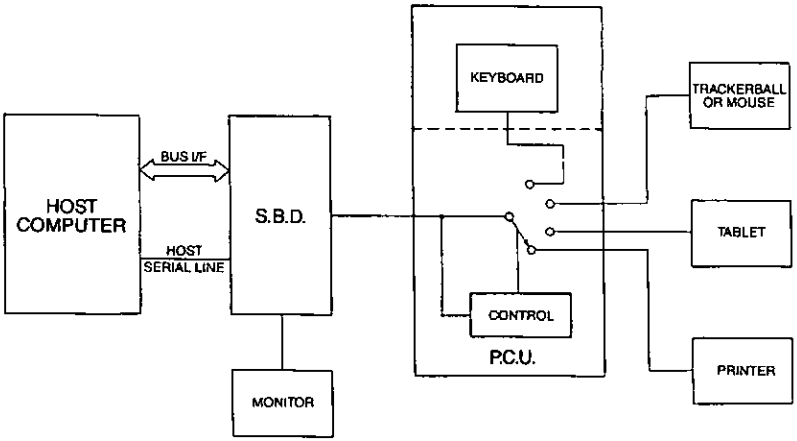


Figure 1.4 Schematic diagram of PCU and peripherals

1.4.4 Printer

The Integrex Colourjet 132GL inkjet printer can be used to produce copies of any information displayed on the screen. This unit has been customised to Gresham Lion specifications. It has a palette of 36 colours and prints onto an A4 width paper roll. It can also be used as a serial text printer; the SBD may be switched into a "transparent" mode in which all characters received from the host are routed to the printer. Printer-specific escape sequences may be embedded in this data to control facilities such as colour, character size and special graphics - refer to the manufacturer's handbook.

1.4.5 Hardcopier

An alternative method of producing hardcopy is to connect the Graftel VP200 video capture unit between the SBD and the video display monitor. This device intercepts the video signals and over a period of about 10 seconds "freezes" the complete picture into its own local frame buffer. The frozen picture may then be printed on the Graftel unit's local printer (which is a Canon PJ1080) while the SBD gets on with something else.

There are two essential differences between connecting a printer directly to the Single Board Display and using a Graftel unit (apart from the extra cost of this unit). The main difference is that, with the printer connected directly, the SBD will suspend execution of any new commands until printing is finished. With the Graftel unit connected, on the other hand, it is the responsibility of the user to leave the picture steady for the 10 or so seconds of capture time after which printing may take place in parallel with further graphics commands.

The other difference is that a printer connected to the SBD will print a representation of the pixel memory whilst a Graftel unit prints from the actual video. Accordingly any of the logical colours may be mapped onto any of the printer's 36 colours whereas the Graftel will always endeavour to represent the display screen as faithfully as possible on paper within the limits of its 8 colour palette. (An example where this is important is in the representation of flashing colours.) Also, if any of the VDU scrolling text is visible at the bottom of the screen at the time the screen is captured, the Graftel unit will print this.

The Graftel unit cannot be used as a serial text printer. The SBD is not aware that the unit is connected, and it therefore does not appear in the power-up message, and the transparency mode escape sequences (see 9.4) are not enabled.

1.4.6 Graphics Tablets

Graphics tablets can be used to control a cursor or they can be used to digitise diagrams and pictures and so enter data into the pixel memory of the SBD.

Two sizes are available for use with SBD. The smaller is 40cm by 40 cm and has an active area of 28cm by 28cm. The other is considerably larger and is normally described as a digitising table. Both have a resolution of 0.1mm.

1.4.7 Monitors

A wide variety of monitors are available from reputable manufacturers for the resolutions provided by the SBD. Gresham Lion PPL will be pleased to advise customers as to which monitors are the most suitable for their application. The monitor can be supplied with the SBD or can be purchased direct from the manufacturer.

1.5 Host Software Support

1.5.1 Hosts and Operating Systems Supported

Gresham Lion (PPL) Ltd can provide assembler-coded subroutine libraries, supporting all interfaces, and special-purpose device drivers for the DMA interface, for the following hosts and operating systems:

- o DEC PDP-11 series running: RSX-11M / RSX-11M-PLUS / MicroRSX RT-11
- o DEC VAX-11 running: VAX/VMS / MicroVMS

A subroutine library coded in FORTRAN-77, which will support the ASCII serial interface only, is available for non-DEC hosts which do not have the appropriate bus structure (Unibus or Q-bus) to support the DMA interface. This can be easily tailored to suit most hosts.

A library and DMA driver written in C are also available which can be ported to run on most Q-bus or UNIBUS based UNIX systems.

This manual provides enough information to allow systems programmers to write their own libraries/drivers for unsupported operating systems if need be.

1.5.2 Host Hardware Requirements

To use the DMA interface to the SBD, the host computer must support one of the following bus architectures:

- o Q-bus (e.g. LSI-11, MicroVAX II - not MicroVAX I)
- o UNIBUS (e.g. PDP-11, VAX-11, VAX 8600)

To use the ASCII serial protocol to the SBD, the host must provide at least one RS232C/RS423 full duplex asynchronous serial line running at one of the baud rates described in Appendix A. The host must support either XON-XOFF or EIA data rate control.

To use the binary serial protocol, the host must not only provide a serial interface as above, but must also be capable of sending 8 data bits (not including parity bit). Receipt of 8-bit data is not required, except for the RIB command (see 3.10).

1.5.3 DMA drivers

For the DMA interface, device drivers are provided for the RSX-11M/M-PLUS, RT-11 and VAX/VMS operating systems. The drivers are easily configured to suit the host environment (see 2.3).

Drivers are also available written in C for certain Q-bus based UNIX systems, which can be easily ported to other such systems.

These drivers provide the standard functions required by the operating system (such as ATTACH, DE-ATTACH and KILL in RSX-11M/M-PLUS). Only one specific Supervisor SBD function is needed which is EXECUTE. This has two specific parameters, "Buffer Address" and "Size". No specific direction of transfer is implied by the Execute function. The specified buffer is, in effect, an area of dual port memory which Supervisor SBD can access randomly for writing and reading. The actual operations carried out depend on function codes contained in the buffer. The host's function is therefore to make available an area of contiguous or contiguously-mapped memory containing graphics commands, after which no further intervention from the host is required.

1.5.4 Serial Line Drivers

The standard host operating system terminal drivers are used for the serial line interface. Certain features may be required which are enabled at system generation time. These are described more fully in 2.4.

1.5.5 Subroutine Libraries

Host Libraries are provided under RSX-11M/M-PLUS, Micro-RSX, RT11, VAX/VMS, MicroVMS and UNIX which support all the available interfaces (and can switch between them at run time). The libraries are coded in assembler (in C for UNIX) for speed and efficiency in execution, and can be simply reconfigured (see 2.4) to make them smaller if any of their features are not required for a particular application.

FORTRAN-77 and C-coded libraries are available which give greater portability where the SBD is being driven serially from a host not supported by the standard libraries.

The libraries provided give a convenient interface between FORTRAN (or other procedural high-level language) and the device or terminal driver.

1.5.6 Tailoring Host Software

A command file is supplied with RT-11 and RSX-11 software kits which can be used to configure and build the host library and driver. Its operation is largely self-explanatory but an outline is given in 2.4. A question-and-answer dialogue is initiated which guides the user through the reconfiguration process, with extensive help facilities. By this means the library can be made as small as necessary to support a particular application environment.

For VMS and UNIX systems, where task size is not such a major consideration, a pre-built library and driver are supplied, together with the source code for modification if required. Reconfiguration is then carried out by editing a configuration macro file.

1.6 Hardware Overview

The Single Board Display is a bit mapped, 4 plane graphics display system which uses a Motorola 68000 microprocessor for local intelligence. The processor operates at either 8 MHz (Model A) or 12.5 MHz (Model B). The 32 bit internal architecture of the 68000 is well suited to graphics generation. This is exploited by providing direct access to half a megabyte of dual port display RAM and by implementation of the host computer bus interface which allows Direct Memory Access (DMA) of host memory using up to 22 address lines.

In addition to these features, the SBD hardware includes the following:

- Up to 64K words of program ROM (using 4 256K EPROMS).
- 2K words of fast static RAM for stack and working storage.
- 2 serial lines, one for communication with the host and one for various peripherals such as a keyboard or a printer.
- 4 registers are present on the host bus for communication between the 2 processors.
- 256 bits of non volatile RAM allow parameters such as baud rates to be stored.

The SBD is available with either Q-Bus or Unibus interfaces. There are two basic models, Model A and Model B. The major difference between the two models is in the method of colour mapping. Model A has four different software-selectable mappings (available as colour or monochrome versions). Model B has colour look-up tables and can therefore display any 15 colours (plus black) from a palette of 4096 colours. The VDU colour is also programmable on Model B.

Other additional features available only on Model B include:

- Detection of failure or blown fuse on peripheral port +12V supply
- Watchdog timer to blank video on failure, or blank or flash the screen after a programmable time without host communication
- Protection of host memory on DMA interface
- DMA I/O address settable on 16-word boundary anywhere in I/O page

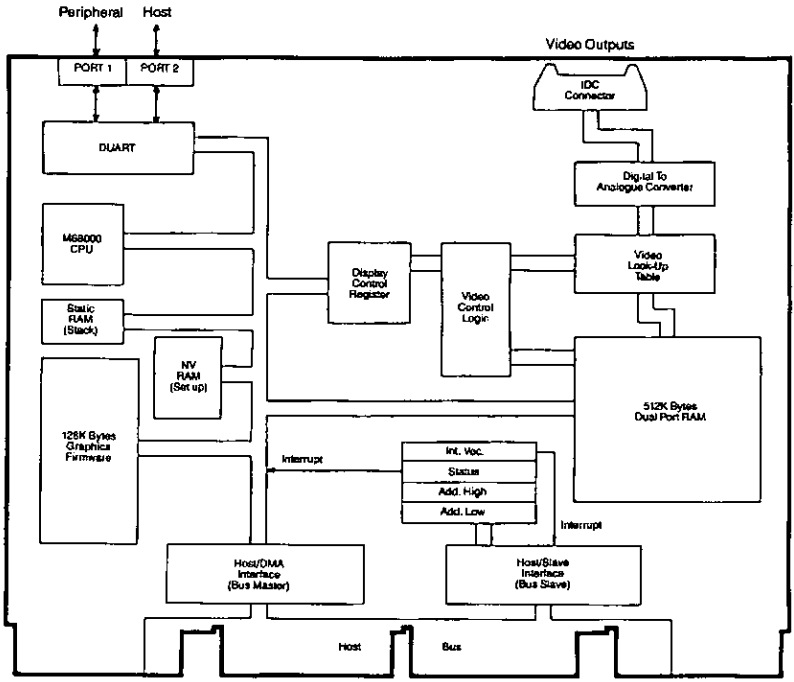


Figure 1.5 Block Diagram of SBD Internal Architecture

1.7 Firmware Overview

1.7.1 General

Software Product References

The following software products are described in this manual. If your system has earlier or later releases of firmware, Product Updates may be obtained from Gresham Lion (PPL) describing the differences. Every attempt is made to maintain upward compatibility between versions of the same monitor, but this cannot be guaranteed.

Model code	Monitor name	Version	Corresponding host software version	Availability
A	SBD2A1	2.1	1.1	November 1985
B	SBD2B1	2.1	1.1	November 1985

The SBD monitor is a program, written in 68000 assembler, to control the local intelligence in the Supervisor SBD single board display. It performs the following functions:-

- Hardware and software configuration dialogue
- Initialisation and hardware confidence tests
- Control of the host serial line, host DMA and peripheral serial line interfaces
- Display of VDU text with VT100 features
- Interpretation of graphics commands from the host DMA interface, in GGS block format (see Chapter 5)
- Local archiving and recall of graphics command blocks (optional)
- Interpretation of graphics commands received via the host serial line, in GGS ASCII or GGS binary formats
- Interpretation of graphics commands received via the optional Ethernet interface, in GGS binary format
- Control of display memory and screen formatting

1.7.2 Summary of operation

The SBD firmware monitor program is largely interrupt-driven. Hardware interrupts can be generated by DMA requests from the host, receipt of an Ethernet packet, receipt and transmission of characters in both serial ports, the receipt of a "break" condition on the peripheral serial port, and the completion of a display scan (field flyback). On Model B an interrupt also occurs as the display changes from pixel to VDU text scanning (split-screen). Interrupt service routines handle all these conditions as well as error conditions such as bus timeouts.

Once power-up diagnostics have been completed, the monitor enters an idle loop which continually checks for command input or other request for action. Sources of requests can be:

- DMA command block ready
- Serial line (ASCII or binary) command block ready
- Ethernet command block ready
- BREAK condition or control-shift-break keycode received on peripheral port (enter SETUP)
- SET UP keycode received on peripheral port (enter Terminal Setup)
- Other data received on peripheral or host ports
- Character transmission from peripheral or host ports complete
- VDU character ready for processing
- Screen scan complete or screen split (Model B)

All other non-interrupt-level activity is as a result of one of the above requests. All graphics command sources are eventually converted internally to a common block format (see Chapter 5) which can be interpreted by the command block execution code. The amount of conversion required depends on the command source; for instance, host command blocks accessed via DMA, and archived command blocks, require no conversion (and therefore no double-handling), whereas serial commands received via the ASCII protocol have to be completely converted from ASCII to binary format.

Serial data is received into fast ring-buffers and for this reason commands and escape sequences must be parsed "on the fly" rather than waiting until a complete command sequence is received.

1.7.3 Internal Addressing Scheme

The 68000 has a 24-bit addressing capability (16 Mbyte) which gives an address range of 0 to FFFFFF (hex). The upper half of this address range (800000 and above) is mapped to the host address space if a DMA interface is present. Table 1.4 gives the mapping of all the items which the 68000 can address. This table is intended as a diagnostic guide which, should a bus timeout occur, may help to localise a fault. (Bus timeouts cause the SBD to display its SETUP page with a message giving the bus address - see 2.2.6.3).

Notes on Table 1.4:

Note (1): On UNIBUS systems, the host address range is 800000-83FFFF (256K bytes), but this may be mapped to a full 4M bytes by the UNIBUS mapping hardware.

Note (2): The DMA register start and end addresses are variable according to the configured value of the DMA address in the I/O page. Ranges of start addresses are:

Model A standard : FFE008,FFE010,...,FFE7F8
Model A modified : FFE808,FFE810,...,FFEFF8
Model B : FFE020,FFE040,...,FFFFE0

Address range Lower (hex)	Upper(hex)	Size in bytes (decimal)	Contents
000000	00FFFF	64K	EPROM code (Low)
080000	08FFFF	64K	EPROM code (High)
100000	10000F	32	Serial line control registers (odd bytes)
102000	102001	2	DMA control register (Model B only)
104000	104001	2	Display control register
108000	10807F	128	NVRAM (odd bytes, low nybbles)
180000	180003	4	Ethernet control registers
184000	187FFF	16K	Ethernet packet queue RAM
200000	27FFFF	512K	Display RAM
380000	380FFF	4K	Static RAM (stack etc).
800000	BFFFFF (1)	4096K	Host memory (22-bit)
FFE008	FFE00F (2)	8	DMA registers (odd bytes)

Table 1.4 SBD Internal Addressing Map

1.8 Hardware Specification

1.8.1 Raster Standard & Pixel Graphics Resolution

STANDARD SBD Models A and B

SET Code	30	32	31	33	
Horizontal Resolution	768	768	512	512	lines
Vertical Resolution	586	574	384	384	lines
Line Frequency	15.6	30	20.8	24.2	KHz
Field Frequency	50	50	50	60	Hz
Line Blanking	12.8	7.73	13.9	7.2	micro-seconds
Field Blanking	1248	867	1584	785	micro-seconds
Scan Mode	I/L	N/I	N/I	N/I	
Maximum Pixel Pages	1/2*	2	4	4	
Pixel Planes	4	4	4	4	

* Vertical resolution is reduced to 574 if 2 pages are selected. Field Blanking then becomes 1632 micro-seconds.

LOW RESOLUTION SBD Models A and B

SET Code	2	4	3	5	
Horizontal Resolution	768	512	384	256	lines
Vertical Resolution	586	512	293	256	lines
Line Frequency	15.6	15.6	15.6	15.6	KHz
Field Frequency	50	50	50	50	Hz
Line Blanking	12.8	21.9	12.8	12.8	micro-seconds
Field Blanking	1248	3648	1248	3648	micro-seconds
Scan Mode	I/L	I/L	N/I	N/I	
Max Pixel Pages	1/2*	3	8	13	
Pixel Planes	4	4	4	4	

NORTH AMERICAN SBD Models A and B

SET Code	40	41	42	43	
Horizontal Resolution	768	704	640	480	lines
Vertical Resolution	574	526	480	360	lines
Line Frequency	30.0	33.2	15.75	23.8	KHz
Field Frequency	50	60	60	60	Hz
Line Blanking	7.50	6.40	20.0	9.60	micro-seconds
Field Blanking	860	810	1400	1500	micro-seconds
Scan Mode	N/I	N/I	I/L	N/I	
Max Pixel Pages	2	2	2	5	
Pixel Planes	4	4	4	4	

1.8.2 Scrolling Text Resolution

STANDARD SBD Models A & B	768x574	512x384	
Character size	5 x 7*	5 x 7	pixels
Character horizontal pitch	8	6	pixels
Characters per row	80	80	
Number of rows	24	24	

LOW RESOLUTION SBD Models A & B	768x574 512x512	384x293 256x256	
Character Size	5 x 7*	5 x 7	pixels
Character horizontal pitch	8	6	pixels
Characters per row	80	64	
Number of rows	24	24	

NORTH AMERICAN SBD Models A & B	768x574 704x526 640x480	480x360	
Character Size	5 x 7*	5 x 7	pixels
Character horizontal pitch	8	6	pixels
Characters per row	80	80	
Number of rows	24	24	

* Each of the 7 pixel rows is displayed twice

1.8.3 DMA Control Registers

Bus Mode	Bus Slave Device
Format	4 x 8-bit registers, alternate bytes
Cycles Supported (Unibus) (Q-bus)	DATI DATO DATOB DATIP DATI DATO DATOB DATIO
Cycle Time (Unibus MSYNC Low to SSYNC Low) (Q-bus BDIN/BDOUT Low to BRPLY Low)	150 ns maximum
Slave Address Model A	Adjustable on 4 word boundaries from 760010 to 763770(8). (Modified vn. 764010 to 767770). Factory setting, 760010 / 764010.
Model B	Adjustable on 16 word boundaries from 760040 to 777740(8). Factory setting, 764000.
Interrupt Vector Address	Adjustable on 2 word boundaries from 0 to 374. Default 270
Interrupt Priority Level	Level 4 Unibus BR4, Q-bus IRQ4

1.8.4 Host DMA Interface

Bus Mode	Bus Master Capability
Addressing Range	18 bits (Unibus), 22 bits (Q-bus)
Data Width	16 bits
Cycles performed	DATI DATO DATOB
Cycle Time:-	(n = delay before SSYNC/RPLY)
Unibus NPGIN high to BSACK high	800+n ns
Q-bus BDMGIN high to BSYNC high	1200+n ns

1 to 4 words transferred per bus mastership with 4 micro-seconds minimum between masterships.

1.8.5 Video Output Modes

SBD MODEL A

One of four output modes which are programmed into a PAL and selected using SOM. There are two standard PALS, colour and mono:-

SOM	Colour Output	Mono Output
0	8 RGB + white overlay	8 grey/green + white overlay
1	8 RGB + flash	8 grey/green + flash
2	16 grey/green	16 grey/green
3	16 colours	16 colours

Table 1.5 Output Modes - Model A

(Grey/green implies grey on a monochrome monitor, green on a colour monitor).

SBD MODEL B

Colour look up tables are provided. Any 16 colours from a palette of 4096 can be selected. Colour zero must always be black.

1.8.6 Video and Timing Outputs

		Model A	Model B
Red and blue outputs:	75 ohms	+1.0 volts	+0.65 volt
Green/monochrome output:	video	+1.0. volt	+0.65 volt
	pedestal	+0.11 volt	+0.05 volt
	sync (composite)	-0.45 volt	-0.29 volt

The following timing outputs are provided at TTL levels:-

Line Sync
Line Blanking
Field Sync
Field Blanking
Mixed Sync
Pixel Clock Input/Output

Pin	Signal	Pin	Signal
1	0V	9	Line blank +
2	Red	10	Field sync -
3	0V	11	Field blank -
4	Green	12	Mixed sync +
5	0V	13	Pixel clock
6	Blue	14	N/C
7	0V	15	N/C
8	Line sync +	16	0V

Table 1.6 IDC connector video and timing outputs

Pin numbering is similar to that on the 10-pin serial connectors (see Figure 2.5), pin 1 being indicated by a small arrow on the connector).

1.8.7 Power Consumption and Bus Loading.

	+5 Volts	4.5 Amps, typical
UNIBUS	+15 Volts	100 mA maximum (1.1 Amps with keyboard)
	-15 Volts	100 mA maximum
Q-BUS	+12 volts	300 mA maximum (1.3 Amps with keyboard)
	Bus Loading	1 AC Load, 1 DC Load

1.8.8 Mechanical

Board Size (Quad SPC)	8.5 x 10.5 inches (229.7 x 265.7 mm)
Serial Line Host	10 way ID Connector
Serial Line Peripheral	10 way ID Connector
Video & Timing Signals	16 way ID Connector

(Cable harnesses and adaptors provide 25 way male host, 25 way female keyboard and BNC video connectors external to the board - see 1.9).

Board extraction:-	Model A	Non jacking handles/spacers
	Model B	Board stiffener with jacking handles.

Removal of NPGIN/NPGOUT wire-wrap jumper between backplane pins CA1 and CB1 is necessary on Unibus installations.

1.8.9 Host Serial Interface

Electrical Standard	RS423 signal levels - RS232 compatible.
Data Bits	7 or 8 bits plus parity
Parity Mode	Odd/Even/1/0 or none
Start Bits	1
Stop Bits	1 or 2
Transmit and receive Baud Rates	19200, 9600, 4800, 2400, 1200, 600, 300, 110
Data Rate control, Receive	XON/XOFF protocol or EIA - DTR
Data Rate Control, Transmit	XON/XOFF protocol or EIA - CTS

Signal	Direction	10-way pin(1)	25-way pin(2)
Clock	Out	1	N/C
OV	Out	2	7
Transmit data (TX)	Out	3	2
Data Terminal Ready (DTR)	Out	4	4
Clear to Send (CTS) + (RS232 OV)	In	5	N/C
Receive data (RX) + (RS232 OV)	In	7	N/C
Receive data (RX) -	In	8	3
OV	Out	9	N/C
Clear to Send (CTS) -	In	10	5

Table 1.7 Host serial port connector outputs

Note (1): 10-way connector on SBD board. For pin numbering see Figure 2.5.

Note (2): 25-way D (M) connector on SIT, SGT, SBX and MicroPDP/MicroVAX Adaptor Plate (MMAP). For pin numbering see Figures 1.1 to 1.3.

1.8.10 Peripheral Serial Interface

Electrical Standard	RS423 signal levels - RS232 compatible.
Data Bits Format	8 bits, no parity
Start Bits	1
Stop Bits	1
Transmit and Receive Baud Rates	9600
Transmit and Receive Data rate Control	XON/XOFF as required by peripheral.

Signal	Direction	10-way pin(1)	25-way pin(2)	15-way pin(3)	6-way pin(4)
SO decode signal	In	1	25	N/C	N/C
OV (signal)	Out	2	7	7	2
Transmit data (TX)	Out	3	2	2	1
Control line (CON)	Out	4	18	N/C	6
S1 decode signal	In	5	13	N/C	4
Receive data (RX) + (RS232 OV)	In	7	N/C	N/C	N/C
Receive data (RX) -	In	8	3	3	3
OV (power)	Out	9	N/C	15	N/C
+12V	Out	10	20	11	5
-12V	Out	N/C	N/C	9	N/C

Table 1.8 Peripheral serial port connector outputs

Note (1): 10-way connector on SBD board. For pin numbering see Figure 2.5.

Note (2): 25-way D (F) connector on SBX and MMAP. For pin numbering see Figure 1.3.

Note (3): 15-way D (F) connector on SIT. For pin numbering see Figure 1.1. The only peripherals allowed are the Cherry or the simple keyboard.

Note (4): 6-way telephone (W polarisation) on SGT. For pin numbering see Figure 1.2. The only peripherals which may be connected to this port are the PCU or the ISI.

1.8.11 Ethernet Interface Option

This section will be issued in early 1986.

1.9 Ordering Information and Cabling

This section gives the part numbers for all versions and models in the SBD family, together with peripherals, cabling and software.

1.9.1 SBD - Board Assemblies

The SBD is available as a board-level component in various different versions. These are summarised in Table 1.9 below. The differences fall under the following headings:

- Mod - Model (A or B). Defines the two major architectures of the SBD family (see 1.6).
- DMA - DMA Interface (U=UNIBUS, Q=Q-Bus, N=None). Defines the environment (bus architecture) for which the board is intended. "None" implies that the board is not to be installed in a host computer and therefore the DMA interface is removed. (An Ethernet interface board can then be fitted, if Model B).
- Add - Address range. Applies only to Model A boards. Most versions have DMA addresses ranging over the lowest quarter (1K bytes) of the I/O page. Some have been modified to occupy the second quarter. (L = address range 760010-763770(8), H = 764010-767770)
- PAL - Video (C=Colour, M=Monochrome). Applies to Model A only. Defines the video output PAL fitted (see 1.8).
- Res - Resolution (H=High, L=Low, N=North American). The high resolution version provides video line frequencies up to 30 KHz at 768x574 non-interlaced. The low resolution version provides only 15.6 KHz line frequencies at lower resolutions. The North American version provides a range of resolutions at 60Hz line rate. (See 1.2.1).

Part Number	Mod A	Mod B	Bus U	Bus Q	Bus N	Add L	Add H	PAL C	PAL M	Res H	Res L	Res N
439000	X		X			X			X	X		
439001	X		X			X		X		X		
439003	X		X				X	X		X		
439010	X		X			X			X			X
439011	X		X			X		X				X
439100	X			X		X			X	X		
439101	X			X		X		X		X		
439103	X			X			X	X		X		
439110	X			X		X			X			X
439111	X			X		X		X				X
439600	X				X	X			X	X		
439601	X				X	X		X		X		
440300	X			X		X			X		X	
440301	X			X		X		X			X	
440400		X		X		N/A		N/A		X		
440410		X		X		N/A		N/A				X
440500		X	X			N/A		N/A		X		
440510		X	X			N/A		N/A				X
440600		X			X	N/A		N/A		X		
440700		X		X		N/A		N/A			X	

Table 1.9 Available versions of SBD board assemblies

1.9.2 Standard Products

As well as being available as board components, the SBD can be supplied packaged in a range of products - see Table 1.10.

Part Number	Description
530800	Supervisor Industrial Terminal (SIT) - red filter
530801	Supervisor Industrial Terminal (SIT) - clear filter
530810	Supervisor Industrial Monitor (SIM) - red filter
530811	Supervisor Industrial Monitor (SIM) - clear filter
534500	SBD Graphics Terminal (SGT) - 20 KHz monitor
534501	SBD Graphics Terminal (SGT) - 30 KHz monitor
	SBD Graphics Monitor (SGM) - 20 KHz monitor
	SBD Graphics Monitor (SGM) - 30 KHz monitor
533700	SBD Serial Box (SBX) - single board free-standing
533701	SBD Serial Box (SBX) - twin board free-standing
533702	SBD Serial Box (SBX) - single board rack-mounting
533703	SBD Serial Box (SBX) - twin board rack-mounting
533704	SBX Upgrade Kit - single to twin board (free-standing)
533705	SBX Upgrade Kit - single to twin board (rack-mounting)

Table 1.10 Standard Products Associated with SBD

1.9.3 Peripherals

The following range of peripherals (Table 1.11) can be ordered for use with the SBD and its associated terminal packages. Normally Gresham Lion (PPL) will supply these, but where another supplier is indicated spares and repairs can be arranged directly. Where PPL alone is shown, a modification is required to the manufacturer's standard product which must be carried out by PPL.

PPL Part Number	Recommended Supplier	Description
533200	PPL	Key Tronic KB100 keyboard (standard)
531200	PPL	Key Tronic KB100 keyboard (with PCU)
5023058	Integrex	Colourjet 132 Gresham Lion inkjet printer
4044043	Canon	Box of 4 paper rolls for Canon/Integrex printer
4044044	Canon	Replacement ink cartridge (black) for Canon/Integrex printer (Canon J1-25B)
4044045	Canon	Replacement ink cartridge (colour) for Canon/Integrex printer (Canon J1-20C)
532602	PPL	Marconi RB2 2-inch trackerball
5026008	TDS	LC12 digitising tablet - cursor or stylus available
-	Summagr	Bit Pad One (RS232) - 4 cursor styles available
-	Summagr	ID digitising table - 4 cursor styles available
-	Graftel	VP200/3 Video Processor (frame buffer)
-	Canon	PJ1080 inkjet printer for use with Graftel VP200
-	PPL	Intelligent Serial Interface for mouse and trackerballs
-	P&G	3-inch trackerball (other sizes also)
535800	PPL	Mouse (for PCU)
535801	P&G	Mouse (for ISI)
-	PPL	Various monitor types available - PPL can advise

Table 1.11 Peripherals and Consumables

Note: P&G Penny & Giles Potentiometers Ltd, Christchurch, Dorset
TDS Terminal Display Systems Ltd, Blackburn, Lancs
Integrex Integrex Ltd, Burton-on-Trent, Staffs
Summagr Summagraphics UK Ltd, Newbury, Berks
Graftel Graftel UK Ltd, Farnborough, Hants
Canon Canon Electric (GB) Ltd, Basingstoke, Hants

1.9.4 Cables and Connectors

The following charts (Tables 1.12 to 1.15) give the PPL part numbers for the various kinds of cable which may be required for the SBD. Lengths given in brackets after the part number are default lengths, but other lengths may be ordered specially.

Wiring information is available from Gresham Lion (PPL) Ltd. on request.

MMAP stands for MicroPDP/MicroVAX Adaptor Plate (see Table 1.15).

NOTE: connector genders given are for the connectors on the cable:

M = male (plug), F = female (socket)

Conn from SBD Host Port	Conn from SBX/SGT Host Port	Conn from MMAP Host Port	Conn to host (external)	Conn to host (external)	Conn to host (internal)
10-way AMP	25-way D type (F)	25-way D type (F)	25-way D type (F) Null Modem	25-way D type (M) Extender	10-way AMP (DLVJ1 etc) Null Modem
			1016180 (0.3m)		
			1016189 (1m)		
			* 1016254 (6m) 1016240 (6m)		
			1016249 (3m) * 1016271 (1.5m flat)		
					* 1016253 (0.3m)

Table 1.12 Host Connection Cables

* = includes EIA modem control lines (CTS/DTR)

Conn from SBD Peripheral Port	Conn from SBX/MMAP Peripheral Port	Conn from SGT Peripheral Port	Conn to VT100 VDU	Conn to PCU keyboard	Conn to standard keyboard	Conn to ISI box
10-pin AMP	25-way D type (M)	Telephone W polarity	25-way D type (F)	9-way D type (F)	Telephone W polarity	15-way D type (F)
		1016243(3m)				
			1016250 (3m)			
	1016208 (6m)			1016260 (35m)		
		1016245 (3m*)		1016269 (5m*)		
			1016268 (1.0m curly)			
		1016209 (6m)				
		1016246 (3m)		1016277 (10m)		
			1016267 (6m)			
			Apply to PPL			
			Apply to PPL			

Table 1.13 Keyboard & Configuration Cables

Notes

1 * = can be extended by 9m using 1016276.

2 The Cherry keyboard for the SIT is supplied with its own cable.

Conn from SBD Peripheral Port	Conn from SBX/MMAP Peripheral Port	Conn from PCU keyboard	Conn to peripheral
10-pin AMP	25-way D type (M)	9-way D type (M) 1016242 (0.7m)	Mouse or Marconi trackerball: flying lead
		1016265 (1m)	Penny & Giles trackerball: 9-way D type (F)
		1016211 (6m)	Tablet: 25-way D type (M)
		1016255 (3m)	
		Telephone A 1016210 (1m)	
		1016267 (6m)	Intelligent Serial I/F: 15-way D type (M)
		1016266 (3m)	
		1016244 (6m)	Printer: 5-pin DIN (M)
		1016256 (3m)	
		Telephone W 1016219 (1m)	

Table 1.14 Peripheral Cables

Part number	Description
533800	MicroPDP/MicroVAX Distribution Panel Insert (MMAP) - standard connectors. Contains Host & Peripheral Ports (25-way D) and 3x75 ohm BNC video. Requires 1 off 1016253 to connect to host.
533801	MicroPDP/MicroVAX Distribution Panel Insert (MMAP) - Transverse Monolith filter connectors.
1016070	Video cable BNC 75 ohm (red) 6 metres
1016068	Video cable BNC 75 ohm (blue) 6 metres
1016069	Video cable BNC 75 ohm (green) 6 metres
1016071	Video cable BNC 75 ohm (black) 6 metres
1016272	Video cable BNC 75 ohm (red) 12 metres
1016274	Video cable BNC 75 ohm (blue) 12 metres
1016273	Video cable BNC 75 ohm (green) 12 metres
1016275	Video cable BNC 75 ohm (black) 12 metres
533000	16-way IDC ribbon cable to 3xBNC adaptor (box)
532300	16-way IDC ribbon cable to 3xBNC adaptor (Klippon)

Table 1.15 Other Cables etc.

1.9.5 Software and Documentation

Table 1.16 (below) gives the part numbers for ordering host software supplied by Gresham Lion (PPL) Ltd. The number given is a suffix; full part numbers are 020-5-xxx.

Example: SB driver for RT-11: 020-5-256.

Note: the operating system and media must be specified when ordering - they are not implied by the part number. The standard distribution media are RX01/RX02 (8") and RX50 (5.25") floppy diskettes. There will be a material and transcription charge for other media (see 2.1.1).

The SBD firmware is supplied as standard with each board delivered. There is a charge for firmware upgrades, which can be carried out on either a self-maintenance or a return-to-factory basis (see 2.2.7.2).

Documents available:

4055022 SBD User Manual (this document).
4055031 SIT Manual
4055032 SBX Manual
4055033 MicroVAX/MicroPDP Adaptor Plate (MMAP) Manual
4055035 PCU Technical Manual
4055036 PCU Diagnostic Firmware Manual

Software Product		Part no 020-5-	Media: Number of volumes required Programs or Programs/Images: TU58	RX01	RX50	RX02	RL01 RL02 TK50 TAPE
Host Library, Device Driver, & standard Fortran test	RSX	256	2	2	2	1	1
	RT11	256	2	2	2	1	1
	VMS	257	2	2	2	1	1
	Unix	319	-	-	-	-	1*
Fortran Demos, Set "A".	RSX	312	5/3	5/3	5/3	5/2	1
	RT11	311	3/1	3/1	3/1	3/1	1
	VMS	318	5/3	3/1	3/1	3/1	1
	Unix	-	-	-	-	-	-
DRAW	RSX	279	1	1	1	1	1
	RT11	-	-	-	-	-	-
	VMS	-	-	-	-	-	-
	Unix	-	-	-	-	-	-
GLIDA Objects only, image for VMS	RSX	313	1	1	1	1	1
	RT11	320	1	1	1	1	1
	VMS	321	1	1	1	1	1
	Unix	-	-	-	-	-	-
TEK	RSX	316	1	1	1	1	1
	RT11	-	-	-	-	-	-
	VMS	-	-	-	-	-	-
	Unix	316	-	-	-	-	1*
FORTRAN 77 SERIAL LIBRARY. Format:- Files 11, RT11, Unix on TAPE, or listing on paper.		302	1	1	1	1	1

Table 1.16 Host Software

TAPE = 1600/3200 bpi, 9 track, 600 ft x 1/2 in
- = not currently released
* = TAPE only
"Images" are large bit-map files used by one of the demonstration programs.

CHAPTER 2 - CONTENTS

Figures

Tables

2 Installation Guide

2.1 Distribution Kit

2.1.1 Contents

Your Supervisor SBD Distribution Kit should contain a subset of the following and a check list with the relevant items marked. See 1.9 for part numbers and ordering information.

- Supervisor Single Board Display(s) - board only, or packaged as ordered (e.g. as Supervisor Industrial Terminal (SIT), Supervisor SBD serial box (SBX), Supervisor Graphics Terminal (SGT), etc.) containing firmware as specified.

- SBD User Manual (this document).

- Host Support Software as ordered for the appropriate host/operating system. This can be delivered on:

  - RL01/RL02 disk cartridge (media/transcription charge)
  - TK50 CompacTape cartridge (media/transcription charge)
  - TU58 DECtape cassette (media/transcription charge)
  - 1600/3200 BPI 1/2" magnetic tape (media/transcription charge)
  - RX01/RX02/RX50 floppy diskette (no added charge)

  The support software is a subset of:

  - SB DMA device driver
  - GGS Subroutine library
  - GGSBLD Customise and build command file
  - SBDTST Confidence test program (FORTRAN)
  - DRAW Mimic generation utility (RSX-11M only)
  - GLIDA Interactive drawing package

  Also available are FORTRAN demonstration programs to aid in familiarising yourself with the SBD. Hosts and operating systems supported are specified in 1.5, and part numbers are given in 1.9.

- Cabling as required, selected from:

  Adapters:

  - IDC 16-way to BNC 3x75 ohm (box or Klippon)
  - SBD to MicroPDP or MicroVAX rear distribution panel

Video Cables:

- 75 ohm BNC (Red, Green, Blue or Black)

Serial Cables:
(PP = peripheral port, HP = host port)

- SBD PP to VDU (supplied if no K/B or PCU ordered)
- SBD PP to keyboard
- SBD PP to PCU
- SBD PP to tablet
- SBD PP to printer
- SBD PP to mouse
- SBD PP to Intelligent Serial Interface
- PCU to tablet
- PCU to printer
- PCU to 2" trackerball (integral to trackerball)
- PCU to other trackerball
- PCU to mouse (integral to mouse)
- SBD HP to host 10-way AMP
- SBD HP to host 25-way D (F)
- SBX (or MicroPDP/MicroVAX adapter) to host 25-way D (F)
- SBX (or MicroPDP/MicroVAX adapter) to keyboard
- SBX (or MicroPDP/MicroVAX adapter) to PCU
- SBX (or MicroPDP/MicroVAX adapter) to tablet
- SBX (or MicroPDP/MicroVAX adapter) to printer
- SBX (or MicroPDP/MicroVAX adapter) to 3" trackerball
- SBX (or MicroPDP/MicroVAX adapter) to mouse
- SGT to PCU (curly cable)

o Peripherals as ordered, selected from:

- Simple keyboard
- Peripheral control unit (PCU) in keyboard
- Intelligent Serial Interface
- Marconi Trackerball
- Penny & Giles Trackerball(s)
- Mouse
- Inkjet printer
- Graphics tablet including power supply
- Digitizing table and control unit
- Colour frame buffer with printer
- Monitor

2.1.2 Unpacking

Care should be taken at all times when unpacking and installing the equipment, as several of the items are delicate. Follow the guidelines given in 2.2.7 for care and maintenance. When handling the SBD itself try to avoid areas where static buildup is possible, and preferably take precautions against static (wrist earthing straps, antistatic mats etc.).

Check the delivered equipment against the delivery schedule and the original order to ensure that all the required parts of the kit are present.

Follow the guidelines in the rest of Chapter 2 for installing and configuring the hardware and software.

2.2 Hardware Installation

2.2.1 Patchable Jumper Settings

The SBD contains very few patchable settings, since most of its configurable features are modifiable by means of SETUP (see 2.3) and Terminal Setup (see 9.5). There are seven patchable links on the Model A board and six on the Model B, which the user will rarely have occasion to change. The links are shown in Figures 2.1 (Model A) and 2.2 (Model B).

Model A Link Table

L1 - L4 (MEMORY CONFIGURATION) - These should only need changing if EPROM sizes change for future software releases.

b to a EPROM - 27256 (32 Kbyte).
b to c EPROM - 2764 (8 Kbyte) or 27128 (16 Kbyte).
b to d Static RAM - 6264 (8 Kbyte) (L2 & L3 only).

Factory settings:

Model A - (version 2.1) - L1 and L4 : b-c
L2 and L3 : b-a
Model B - (version 2.1) - L1 to L4 : b-a

L5 (RESET)

b to a 68000 reset caused by SBD Reset Signal and Bus Init.
b to c 68000 reset caused by SBD Reset (power-up) only.

All serial line only boards are factory set b-c and must not be changed. All other boards are set to b-a, but this can be changed if Bus Initialisations are frequent or inconvenient (as in RT-11SJ systems).

L6 (TEST)

Not to be removed. For factory internal test only.

L7 (FLASH RATE) - controls flash rate in Video Output Mode 1.

b to a 1.6 Hz 50% Mark to Space Ratio.
b to c 0.8 Hz 50% Mark to Space Ratio (factory set).
a to b to c 0.8 Hz 75% Mark to Space Ratio

On Q bus boards, two further links (LK1 and LK2) are provided. These are linked so that if Interrupt Levels 5 and 6 are not wired in on the backplane, the SBD will still operate.

In normal use, it should only be necessary to change L5 and L7. A wire wrap tool should be used to do this. Boards can be supplied with a non standard configuration if specified when ordering, and changes to the patching can be carried out if the board is returned to the factory.

A simple 'reset' button can be wired up by taking wires from pins b and c on link L5 to a push-button which breaks the circuit when pressed. This is the equivalent of removing and replacing the link.

Model B Link Table

L1 - L4 As Model A.

L5 As Model A. Positioned between L1 and L2.

L6 Does not exist.

L7 WATCHDOG

b to a Enable fail-safe blanking. Screen will blank after a software selectable time without host activity or if the 68000 halts. This over-rides the selectable blanking or flashing on failure which is provided in the configuration dialogue.

b to c Disable hardware fail-safe blanking. Action will depend on the configuration dialogue failure option selected.

Note: There are various CT and WP positions marked for factory setting. These should not be changed without prior consultation with Gresham Lion PPL. The Q-bus links LK1 and LK2 on Model A are replaced on Model B by cuttable tracks CT1 and CT2.

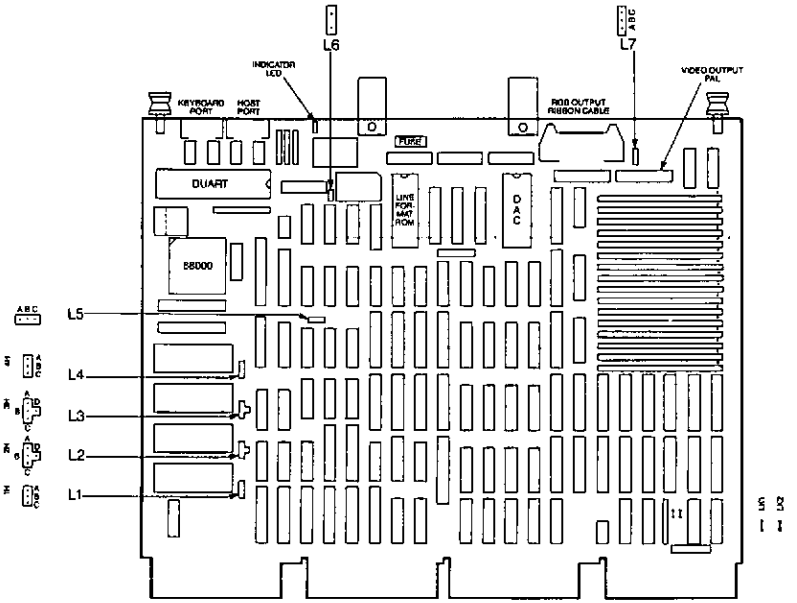


Figure 2.1 SBD (Model A) link positions

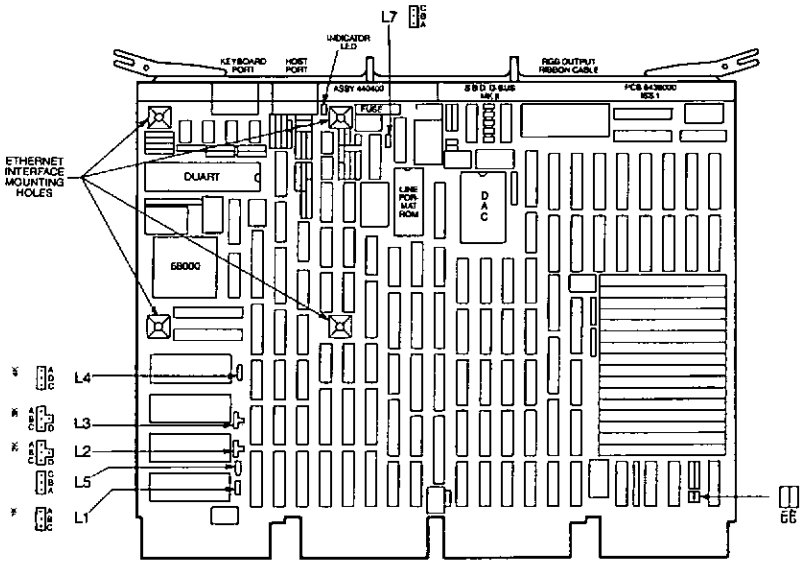


Figure 2.2 SBD (Model B) link positions

2.2.2 Installation

2.2.2.1 DMA Configurations - Q-bus Backplane

No modifications are required to Q-Bus backplanes. However, since the SBD makes use of interrupts and can become bus master, bus grant continuity must be maintained between the processor and the SBD. This means that there must be no empty positions between the two boards, unless they are fitted with a grant continuity module.

Since the RQDX1 disk controller board fitted to the MicroPDP-11 does not pass on all of the grant lines, this board should always be the last one in the backplane.

If the SBD is installed too far away from the processor, and especially where there are intervening block DMA controllers (e.g. disk controllers, Ethernet interfaces), the SBD's bus mastership requests may time out, causing a bus error. You should therefore install the SBD as close as possible to the processor, and other DMA controllers as far away as possible (as recommended by DEC).

2.2.2.2 DMA Configurations - UNIBUS Backplane

Because the SBD has an NPG (Non-Processor Grant) capability, that is it sometimes needs to become bus master, special action must be taken when it is installed in a UNIBUS backplane. This section outlines the procedure involved, but you should also consult your processor's System Manual or User's Guide for orocessor-specific details.

The special action involves removing a wire-wrap link between two of the pins on the UNIBUS backplane, for each SBD you are using.

For this you will need:

- a wire-wrap tool (capable of unwrapping also). Also useful would be tweezers for removing unwrapped wire, and a wire stripper and cutter.
- appropriate tools for gaining access to the backplane. Usually a screwdriver is all that is required; sometimes also an Allen key for opening the cabinet door.

This procedure must be carried out with all power off and the processor isolated from the mains.

1. Expose the backplane

The method of gaining access to the backplane varies with the processor, but usually involves removing a cover plate or panel, sliding out the card cage (mounting box) on runners, rotating the card cage on its hinges to a vertical position and locking it, and sometimes removing a backplane cover plate. DEC system manuals do not give much guidance on this procedure, but a few minutes' study followed by trial and error usually work. On larger machines, such as the VAX 11/750, the UNIBUS backplane can be examined in situ. (Make sure you know which of the backplanes is the UNIBUS!)

2. Examine the backplane

Once you have exposed the backplane, you will see an array of pins arranged in groups or blocks. These blocks are arranged in vertical "stripes" consisting of six blocks (Figure 2.3). The number of stripes depends on your processor and configuration. Each stripe contains four card slots on the other side. For printed circuit backplanes the division into stripes is not obvious and it may be necessary to count pins.

3. Orient the backplane

Looking at the backplane, determine which way the PCBs face (i.e. which is the component side). The six blocks, or to use DEC terminology, rows, are designated with the letters A to F. With the backplane vertical, if the component side is on the right (as would be the case with a PDP-11/44) then row A is at the bottom and row F at the top. If the component side is on the left (e.g. VAX-11/750) then row A is at the top. If in doubt, look for any quad or dual height cards in the UNIBUS - dual cards are likely to be in rows A-B and quad cards (e.g. the SBD) in rows C-F.

The procedure described here assumes that row A is at the bottom (Figure 2.3).

4. Locate a spare SPC slot

Each 'stripe' contains four PCB slots. The low numbered slots are on the processor side of the cabinet. Often there is space on one side of the cabinet to allow for expansion by addition of further 4-slot backplanes.

By consulting your System Manual for processor backplane assignments, and by studying the PCB's installed in your system, you should be able to locate the SPC (Small Peripheral Controller) slots. Some of these are hex (6 row) size and some are quad (4 row) size.

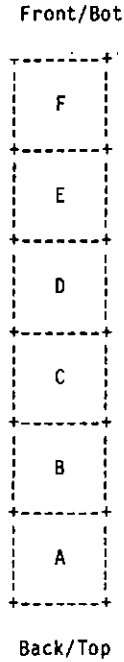


Figure 2.3 UNIBUS Backplane Row Nomenclature

The quad SPC slots occupy only rows C to F (see Figure 2.3), leaving A and B free for other dual modules. The last A-B dual slot on the backplane will normally contain a terminator or bus extension module. All empty SPC slots will contain a single size quarter height bus grant continuity card (G727), or "flip chip", in row D. SPC slots with the jumper already removed will contain a dual height bus grant continuity module (G7273) in rows C and D.

The SBD, being quad size, can go in either of the two types of SPC slot. However, it must be in rows C to F. This means that the flip chip must be removed before inserting the SBD.

5. Locate Row C in the slot

The slots are sometimes numbered. Having located the slot into which the SBD will be installed, look at the backplane again, locating the stripe which contains the slot. Look closely at row C which is third from the bottom in Figure 2.3, and compare it with the pin numbering diagram (Figure 2.4).

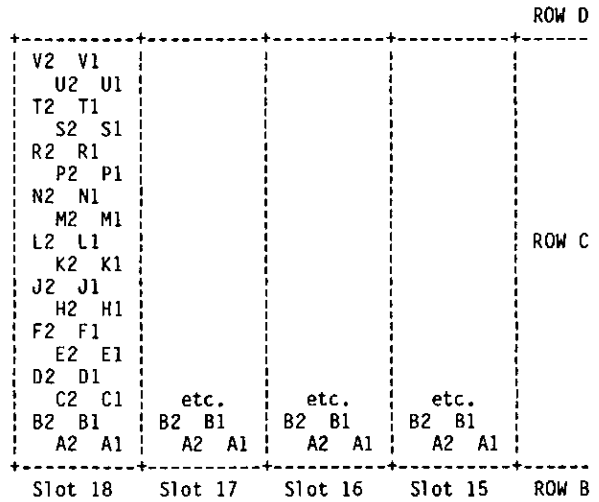


Figure 2.4 Layout of pins in a UNIBUS 4-slot row

Let us assume that the backplane block we are concerned with houses slots 15 to 18 and we have decided to install the SBD in slot 17. This is the second slot from the left as you look at the backplane. It is not very easy to pick out the slots, due to the staggering of alternate pin rows, but each horizontal line of pins (there are 18 lines in the row) contains 8 pins, 2 pins for each slot. The rows are, confusingly, lettered A to V (omitting G,I,O,Q), so the pins are designated A1 and A2, B1 and B2 (offset to the left), C1 and C2 (above A1/A2) and so on. The link we wish to remove connects pins A1 and B1 on the slot in question, and you now have enough information to locate these pins and find the wire link. Remember that in some processors, e.g. VAX 11/750, the "A" pins are at the top of the row.

6. Remove the link

Use the wire wrap tool to remove the link (push the remover end onto each pin and turn anticlockwise until the wire can be removed from the pin). Take care that the wire does not break and shed loose pieces into the backplane.

Insert the SBD into its designated slot, not forgetting to remove the flip chip first.

7. Test installation

Now try switching the power on and attempting to boot the machine. If the processor will not boot, check very carefully that you have removed the link from the correct slot. If all seems to be correct, there may be an address conflict between one of the SBDs and another device on the system; if so, check and reconfigure the SBD (see 2.3). If the processor still will not boot then it is best to replace all the links removed and call in a qualified engineer.

If the processor boots successfully, switch off the power again and repeat the process for the next SBD, until all SBD slots are modified and the boards installed.

8. Finish up

Now replace the backplane cover plate (if any), hinge the mounting box back to horizontal (if required), and close the cabinet.

Complete the installation of the SBDs as described in chapters 2.2.3 to 2.2.6.

9. Replacing links

If you remove an SBD for any reason, such as maintenance, you must reinstall the wire-wrap link and replace the G727 module in row C, or else insert a G7273 bus grant continuity module in rows C and D, before powering up the processor. BE VERY CAREFUL THAT YOU RELINK THE CORRECT PINS (CA1 to CB1). If you have not, there may be a short-circuit and an expensive repair will be necessary. It may be worth marking the pins (with a small piece of tape or sleeving for instance) when the link is removed. Use an easily distinguishable piece of wire when re-linking to facilitate removal when re-installing the SBD.

2.2.2.3 Serial Configurations

Installation of serial configurations (which can be serial-only, or else combined with the DMA interface) consists largely of connecting the supplied box or terminal (if any) to the power supply, establishing an RS232 link to the host computer with an appropriate cable (see Table 1.12, section 1.9.4), and switching on. Appropriate mounting kits and instructions are supplied with rack-mounting SBX systems.

Figures 1.1 to 1.3 (section 1.1) give the positions of power input and host RS232 connections on SIT, SGT and SBX respectively. Signals and connections for the host RS232 connectors are given in 1.8.9.

If the SBD is serial-only but is installed inside the host computer, you should ensure:

- link L5 is set b-c (see 2.2.1)
- the SBD is configured to NO DMA (see 2.3.2)
- on UNIBUS systems, the NPG link CA1-CB1 (see 2.2.2.2) is NOT removed.

2.2.2.4 Ethernet Configurations

The Ethernet interface will be available in early 1986 when this section will be issued.

2.2.3 Attaching Peripherals

When the SBD powers up, experiences a bus reset or exits from SETUP, it loads information from the keyboard connector about the peripheral connected (see 2.2.6 and 4.2.1). Therefore, if a peripheral is to be changed, one of these three events must occur before the system will function.

WARNING: if you connect or disconnect any peripheral (especially one taking its power from the SBD) while the SBD is powered, it is liable to generate a break which will put the SBD into SETUP mode (see 2.3). There should be no risk of damage to the SBD or peripheral in the event of accidental disconnection, but it is good engineering practice to switch off the power before attempting reconfiguration.

Tablets, if not purchased through Gresham Lion (PPL), may need to be set up by altering internal switch settings. Refer to 4.5 for details. Note that tablets set to manufacturer's standard settings will prevent the SBD from operating if connected directly (i.e. not via a PCU). They must be configured for remote control first.

Any peripheral, including the PCU, which is connected directly to the SBD, is attached via the keyboard port 10 pin connector shown in Figure 2.5. If a PCU is supplied, then other peripherals are connected to the rear of the keyboard case via the connectors shown in Figure 2.6. A schematic diagram of the complete system is shown in Figure 1.4, Section 1.4. Note that trackerballs and mouse require either a PCU or an Intelligent Serial Interface (ISI) in order to connect to the SBD, since they do not generate RS232 signals.

A description of how to link each peripheral to the PCU follows (Table 2.1). The cable connector and the location of its mating connector are given. If a single peripheral is to be connected to the SBD, the same description applies to the peripheral.

PERIPHERAL	PCU
MARCONI TRACKERBALL, P & G MOUSE Flying Lead	9 pin D type plug
P & G TRACKERBALL 9-pin D type On angled side of housing	9 pin D type plug
TABLE 25-pin D type At rear of control module (separate power supply)	Telephone - A polarisation
TABLET 25-pin D type Under ledge at rear of tablet (separate power supply)	Telephone - A polarisation
PRINTER 5 pin DIN Right hand side - next to ON/OFF switch (separate power supply)	Telephone - W polarisation

Table 2.1 Locations and types of PCU/peripheral connections

Notes on Table 2.1:

1. The 9 pin D type socket on the rear of the PCU is for connection to the SBD's peripheral port.

2. Only one out of tablet or digitizing table can be connected simultaneously.

3. Only one out of Marconi trackerball, Penny & Giles trackerball, or mouse can be connected simultaneously.

4. Part numbers for the various cables required are given in 1.9.

5. The mouse and the 3" trackerball connect to the ISI module by means of the 9-way D connector. The ISI is connected to the SBD's peripheral port by means of the 15-way D connector on the other end of the ISI.

6. Connections and signals for the peripheral ports on SBD, SGT and SBX/MMAP are given in 1.8.10.

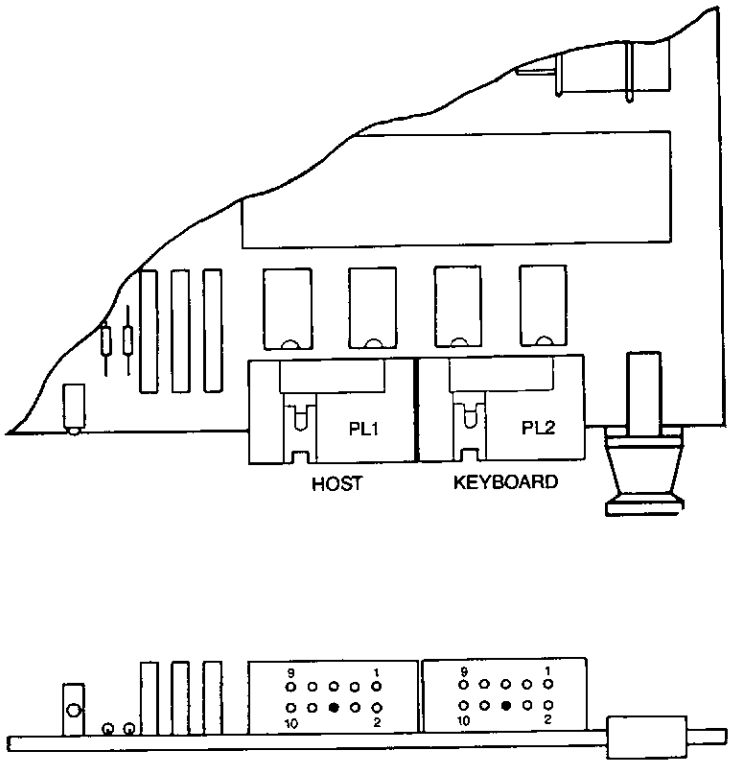


Figure 2.5 Positions of SBD 10-pin serial connectors

For signal connections, see Table 1.7, section 1.8.9.

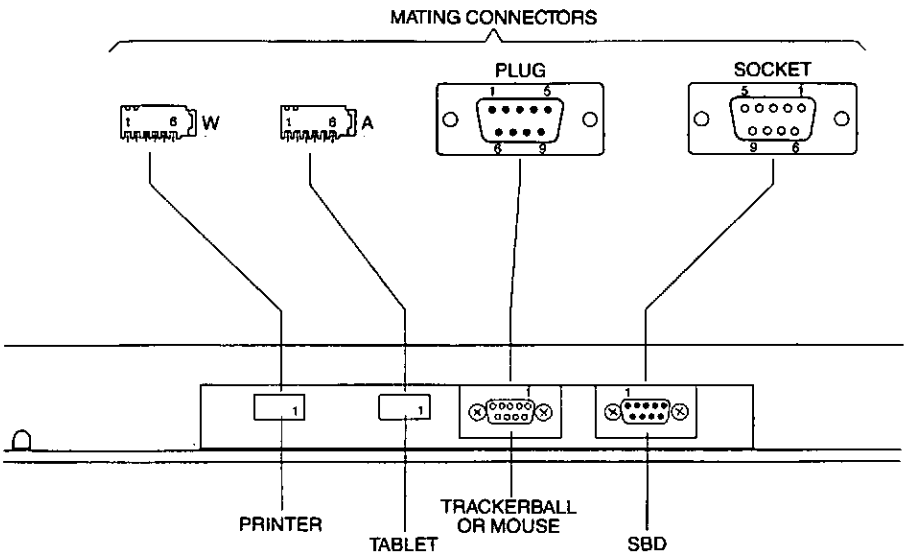


Figure 2.6 Rear of PCU showing peripheral connectors

For connections, see 4.2.2.

2.2.4 Starting Up

Now that your SBD(s) are installed, you are ready to switch on the power, either on the host computer, or (in the case of a serial-only display) on the box or terminal containing the SBD. When you do this, you should see a message similar, though not necessarily identical, to the following at the bottom of the display:

POWER UP CHECK -- OK : K/B TAB 01234/A2.1

For an explanation of this message see 2.2.6. If you do not see the message there may be several reasons:

1. Monitor not correctly adjusted (N.B. unless there has been damage in transit, this should not be the case when both board and monitor are supplied by Gresham Lion (PPL), since they will have been tested together prior to shipping). See 2.2.5 for instructions on setting up monitors.

2. SBD not set to correct resolution for monitor. See 2.2.5.

3. Missing or faulty video cable(s). Also ensure that the IDC connector on the end of the ribbon cable is inserted correctly into the SBD (match the arrowheads).

4. Monitor not switched on or faulty.

5. Fault detected on power-up: attempt to trace fault using information in 2.2.6.

6. Power supply failure - inspect any power indication LEDs. On serial line boxes, the +5V and +12V LEDs should be lit.

7. SBD configured to power up with no VDU lines visible (see 2.3).

2.2.5 Setting Up Monitors

The SBD specification, which is in 1.8, contains the information required to set up video monitors at any of the resolutions provided. The monitor handbook should be referred to for instructions on how to do this.

The standard SBD powers up in 768x574 non-interlaced as the default set at the factory, the low resolution SBD in 512x512, and the Northa American SBD in 640x480. To change these to the resolution required, enter SETUP (see 2.3) and type the appropriate KEY NUMBER shown below (Table 2.2). (N.B.: don't use the keypad number keys).

STANDARD : SET code	Resolution	NORTH AMERICAN: SET code	Resolution	LOW RESOLUTION : SET code	Resolution	KEY NUMBER
30	768x574 I	40	768x574 NI	2	768x574 I	1
31	512x384 NI	41	704x526 NI 60Hz	3	384x293 NI	2
32	768x574 NI	42	640x480 I 60Hz	4	512x512 I	3
33	512x384 NI 60Hz	43	480x360 NI 60Hz	5	256x256 NI	4

Table 2.2 Key codes to force SBD into different resolution formats

The SETUP menu (see Figure 2.7, section 2.3.1) should then appear correctly on the screen. If the monitor is not correctly set up for your desired resolution, the SETUP menu will not be correctly synchronised. Adjust the monitor until the text is clearly displayed. Then change parameters as required, and save the resulting configuration, as described in 2.3.

If you cannot obtain a picture and you think that the monitor is working and is correctly adjusted, follow the fault-finding procedure in 2.2.6.2.

On exit from SETUP (by pressing <ESC>) the power-up message as described in 2.2.6.1 should appear at the bottom of the screen. For example:

```
POWER UP CHECK -- OK : K/B T/B  01796/B2.1
```

If the monitor desynchronizes on exit from SETUP, you have probably forgotten to change the default resolution parameter (SET code). (Simply pressing a key as described above does not permanently save that resolution). Re-enter SETUP, change this to the correct value (Table 2.2), save it - <SHIFT>S - and exit.

2.2.6 Fault Finding

2.2.6.1 In-Built Diagnostics

The SBD firmware contains a powerful self-diagnostic capability which is exercised every time the power to the board is switched on, and following bus resets if the link L5 is set appropriately (see 2.2.1). A subset of the diagnostics is executed on exit from SETUP (see 2.3).

Table 2.3 below lists the sequence followed on power-up and the observed results if a failure is detected.

Error Indications

Until the point at which the SBD has ascertained that sufficient function is present to generate a message on the monitor screen, the single LED on the edge of the board is used to flash a diagnostic code. After this point, any diagnostic information is displayed on the bottom line of the VDU area. If the SBD is forced to enter SETUP mode due to an error on power-up, a message on the screen indicates the cause of the failure, combined with a LED flash code (in case the screen is not formatted).

Notes on Table 2.3

1. Unless otherwise stated below, all fatal errors during power-up or at any other time will cause the SBD to enter the configuration dialogue (SETUP). A message at the bottom of the VDU screen will explain the reason for entry, e.g.:

No.	Message	Reason for entry
1	<none>	Keyboard or VDU requested entry
2	BUS ERROR AT 00FFE009 PC = 83A00	Non-existent memory or register, or timeout on bus grant*
3	NVRAM CHECKSUM ERROR	NVRAM fault
4	UNEXPECTED INTERRUPT	hardware or NVRAM fault
5	DMA SELF-INTERRUPT TEST FAILED	DMA interrupt failure

* Further diagnosis of message no. 2 is possible by referring to section 1.7.3 for the location of the address specified in the message. This may help to identify a failed component, for instance. The SBD times out after 1 millisecond (Model A), 640 microseconds (Model B), when attempting to gain bus mastership. Therefore, devices which disobey the rules for bus mastership, and retain the bus for longer periods than this (e.g. DEQNA Ethernet interface), may cause a bus error on the SBD. If this is a problem, Gresham Lion (PPL) can carry out a modification to

increase the timeout.

Since in some circumstances (particularly bus errors) it is possible to enter SETUP before the screen is formatted, LED flash codes will be displayed while in SETUP (a unique code depending on the reason for entry) as follows:

```
Message no. 1 = LED off
   "      "  2 = LED code 2
   "      "  3 = LED code 3
   "      "  4 = LED code 4
   "      "  5 = LED code 5
```

2. LED codes are as follows:

   Code N = indefinite repeat of (2 sec off, followed by N repeats of (1/3 sec off, 1/3 sec on)).

3. The power up message (PUM) on the bottom line of the VDU area will report the peripheral configuration, serial number, and firmware version number. It is in two parts, and will have the general form:

   <message> <peripherals> <serial no.>/<firmware version>

   ```
   ¦-PUM1--¦ ¦----------------PUM2----------------------¦
   ```

   Examples:

   ```
   EPROM CHECKSUM AB32F019 K/B T/B PRT TAB  01027/A2.1
   POWER UP CHECK -- OK :  FUSE             01088/B2.1

                           ¦____________¦   ¦_______¦
                            Any combination  Serial &
                            depending on     version
                            configuration    number
   ```

   The peripheral codes are:

   K/B Keyboard, VDU or PCU (must be PCU if other peripherals in configuration)

   T/B Trackerball or mouse (via PCU)

   PRT Inkjet printer

   TAB Graphics tablet (Summagraphics or TDS) or table (Summagraphics)

   MSE Trackerball or mouse (via ISI). NOTE: it is impossible for both TAB and MSE to appear since they imply different peripheral configurations. MSE always appears on its own.

FUSE Only appears on Model B, if the +12V power supply fails or the fuse blows. Always in reverse video (see 2.2.6.3, B7e).

The version number indicates by its first letter whether the board is Model A or Model B.

Table 2.3 Power up sequence

Action	Action on error detect
Hardware reset causes PC and supervisor stack to be set up.	
Turn LED on.	
Perform Static RAM test.	LED flash code 6. Abandon further tests.
Set up user stack and enter User mode.	
Read NVRAM into volatile area. Set up default values including serial line parameters.	
Enable host SLU Tx and Rx, but not BREAK.	
Enable peripheral SLU Tx and Rx, but not BREAK.	
Perform Dynamic RAM address line number test.	LED flash code 7. Abandon further tests.
---------- Re-entry point on exit from SETUP ---------- (exit from SETUP resets Supervisor and User stacks)	
Set up selected defaults by calling SET, SEL, MIX, SOM, SCA, SSA, CHI and redefine serial line parameters. Screen should now be formatted.	
Perform EPROM checksum. If OK, set first part of power-up message (PUM1) to "POWER UP CHECK -- OK :"	Note error & continue. Set PUM1 to "EPROM CHECKSUM xxxxxxxx" where xxxxxxxx is hex value of checksum.
Display PUM1.	
Perform NVRAM checksum.	Set NO DMA, calculate new checksum & update NVRAM. Then enter SETUP (LED flash code 3).

Table 2.3 Power up sequence (continued)

Action	Action on error detect
If DMA required, set up DMA registers and perform read/write tests. Continue test at 30ms interval until successful. After 15 sec, timeout.	On timeout enter SETUP with bus error at <DMA register address> in hex. (LED flash code 2). e.g. 760010(8) = 00FFE009(16)
Test DMA self-interrupt.	If interrupt fails, set NO DMA, calculate new checksum & update NVRAM. Then enter SETUP (LED flash code 5).
Determine what is connected to peripheral port. Initialise power-up message part 2 (PUM2).	
If PCU connected, request configuration. If no reply after 1 sec, timeout.	
Check peripheral port fuse (Model B only)	Add "FUSE" (reverse video) to PUM2.
Add "K/B" to PUM2 if PCU, simple keyboard or VDU connected.	If timeout on PCU, set "K/B" to reverse video in PUM2.
If keyboard or PCU connected, sound bell, set click on/off as appropriate, light "online" LED.	
Enable BREAK on peripheral port if simple keyboard or VDU connected (not PCU).	
If trackerball or mouse connected via PCU, add "T/B" to PUM2.	
If tablet connected, add "TAB" to PUM2, then:	
Attempt to communicate with tablet. Timeout after 1 sec if no reply.	If timeout, set "TAB" to reverse video in PUM2.
If trackerball or mouse connected via ISI, add "MSE" to PUM2.	

Table 2.3 Power up sequence (continued)

Action	Action on error detect
Attempt to communicate with ISI. Timeout after 1 sec if no reply.	If timeout, set "MSE" to reverse video in PUM2.
If printer connected, add "PRT" to PUM2, then:	
Check whether printer powered on.	If printer off, set "PRT" to reverse video in PUM2.
Add board serial number and firmware version number to PUM2.	
Display PUM2 on VDU.	
Turn LED off.	
Set DTR signal high on host port. If EIA not in use, send XON.	
Start normal idle loop.	

2.2.6.2 First-line Fault Finding

Below is a table which describes the states in which faults, either internal or external to the SBD, can affect the working of the SBD.

A) Fault during graphics generation or peripheral handling

1. Screen deformats.

Possible causes :

a. Memory failure causing loss of format word(s) - if intermittent, may be able to recover by:

(i) entering SETUP (<CTRL><SHIFT><BREAK>) and exiting (<ESC>)
(ii) sending SET, IDI or RST commands
(iii) resetting (<SETUP> followed by 0)

If hard, should be detected on power-up - return board to factory for repair.

b. 68000 failure. Board will no longer function at all, even on power-up, when LED remains permanently off. Return board to factory for repair. On Model B, this fault causes a blank screen if the watchdog is enabled (see 2.2.1).

c. Firmware error. Depending on nature of error, may be recoverable as in (a). Report exact circumstances of failure to Gresham Lion (PPL) Customer Support Desk. Attempt to reproduce conditions of failure if possible.

2. SETUP menu appears unexpectedly.

A message at the bottom of the screen should give information on the reason for entry to SETUP, and the LED should be flashing a code (see Notes on Table 2.3).

Possible causes:

a. Hardware fault e.g. loss of DMA register, NVRAM corruption. Attempt to exit from SETUP. If board immediately enters SETUP again, try resetting or powering down. If bus error, consult Table 1.4, section 1.7.3, to try to identify area of failure. Report to Gresham Lion (PPL).

b. Host bus timeout. Another DMA controller in the system is gaining bus mastership and not releasing it in the time frame specified by DEC (8 microseconds). The SBD's own bus mastership request times out (after 1 millisecond on Model A, 640 microseconds on Model B), causing a bus error. The address given may be host memory or a DMA register (see Table 1.4, section 1.7.3). There are two possible situations:

(i) A block DMA controller is closer to the processor than the SBD and thus has priority. You should try to put the SBD closer to the processor than such controllers as MSCP disk controllers and Ethernet controllers. As the SBD does not perform block DMA, this should not adversely affect system performance.

(ii) A controller is violating DEC rules and retaining bus mastership too long. The DEQNA is particularly bad in this respect if heavily loaded. A modification to the SBD may be required to increase its timeout, unless the system can be tuned to reduce the bus load.

3. Screen blanks on running PPL-supplied host package

Possible causes:

a. Option not enabled via password in SETUP (see 2.3.3).

B) Fault on power-up

1. Screen unformatted, LED off.

Possible causes:

a. 68000 has failed. Screen appearance depends on type of monitor in use: may be bright jagged lines, dim vertical bars, or completely black (if Model B). Perform the following checks before calling PPL Service Dept.

- Check that the host computer is still working; if not, switch off.

- Check the +5 volt supply to the backplane. In a heavily loaded system, the power supply may be unable to maintain the voltage.

- Check that the board is inserted into the backplane correctly.

- If the board is in an SBD box, check power supply lights.

- If the board is being used for serial line only or if there is no processor in the computer, ensure that link L5 is b-c, and that the DMA register address is set to "NO DMA".

- If the board has recently had a new software release, check that EPROMs are correctly inserted in sockets. Confirm upward compatibility with Gresham Lion (PPL) Customer Support. (NVRAM may require reformatting).

b. Power-up resolution setting not compatible with monitor frequency, or monitor out of adjustment. Screen will appear mostly black, but the power-up message should be discernible at the bottom even if unsynchronised and unreadable. Adjust monitor or set correct resolution as described in 2.2.5.

c. Firmware is operating normally but a soft error in dynamic RAM has corrupted one or more format words. Attempt to recover as in A)1.a. If the problem recurs, contact PPL Service Dept.

2. Screen unformatted, LED flashing or permanently on.

Possible causes:

a. See 2.2.6.1. If LED code is 6, static RAM has failed. If LED code is 7, dynamic RAM has failed. Otherwise the dynamic RAM may have a soft error which is affecting format words, but the real cause of the error is given by the LED code. Note: failure of Static RAM compromises the ability of the firmware to execute properly, and so the LED flash code may not be working - the LED will be on, however. Furthermore, if the hardware is failing to generate screen interrupts (possible if the Dynamic RAM fails), the timer facility for LED flashing will not be working, and the LED would again be permanently on.

3. Screen formatted, SETUP menu appears immediately, LED flash.

Possible causes:

a. See 2.2.6.1. A message will be displayed on the screen giving the reason for entry to SETUP. Contact PPL Customer Support desk if you cannot ascertain the cause.

4. Screen formatted, SETUP menu appears after 15 sec.

   Initially the first part of the power-up message (POWER UP CHECK -- OK) appears. After about 12 seconds (Model A) or 8 seconds (Model B), the SETUP menu appears with the message indicating a bus error at the DMA register address (and LED flash code 2).

   Possible causes:

   a. The DMA interface has failed, or the SBD was not correctly inserted into the backplane, or the NPG link was not removed (in a UNIBUS system).

   b. (More likely). The SBD cannot access the bus to read the DMA registers, probably because the host processor is not running. (Some processors take longer than the SBD to power up, hence the built-in delay). This can also be caused by failure to maintain bus grant continuity signals to the SBD, either by leaving an empty slot in the backplane, or by having a controller such as the RQDX1 (MicroPDP disk controller with Version 8 or earlier firmware) which does not pass on the signals and is not the last card in the backplane. Another possible cause is no host processor at all, e.g. in the serial-only configurations. The board should be set to "NO DMA" (see 2.3) in this case.

5. SBD continually resets.

   Possible causes:

   a. At intervals of approximately 1 second the monitor screen reformats and the keyboard (if any) beeps. Some processors continually reset the bus on booting when the system disk is not ready. No action is necessary; the resets will cease when the system disk is ready.

   b. The SBD continually enters SETUP and then resets. May be caused by the above in combination with another fault (see B above) or, if the message at the bottom of the screen is "UNEXPECTED INTERRUPT", this may be caused by an attempted field upgrade where the board should have been returned to the factory (i.e. incompatible NVRAM formats).

6. SBD O.K. but host fails to boot.

Possible causes:

a. CSR address set to "NO DMA" - in Model A this causes the CSR to appear at an illegal address (e.g. 160000).

b. NPG link removed on UNIBUS backplane and not replaced when SBD removed, or else removed when serial-only board installed.

c. Insertion of SBD has disturbed other cards which no longer make good contact in the backplane.

d. On some newer DEC systems (particularly VMS and MicroRSX) an Autoconfigure operation is performed at system boot. This may involve examining all devices appearing in the floating address space (760010-763776 octal) and attempting to make them interrupt. This technique does not work with the SBD as it does not have a DEC standard register format, and so the host may halt or hang at this point. You may be able to circumvent the problem by setting the SBD's DMA address as high as possible, at least above all the other devices in the floating address space. If this fails, you should have the modification to SBD(A) to put its address range up to 764010-767770. All versions of Model B can be set outside the floating address space.

7. Peripheral Mnemonic in reverse video in power-up message.

Possible causes:

a. If "K/B" in reverse video: cannot communicate with keyboard. No beep heard on power-up. Check:

- cable plugged into SBD but not keyboard
- PCU failed (is ONLINE LED lit? Do PIX/VDU and MIX keys work? Does <CTRL><SHIFT><BREAK> enter SETUP?)
- fuse on +12V supply to peripheral has blown. (For Model B, an explicit message indicates this - see below).

b. If "TAB" in reverse video, tablet is not returning data. Check:

- auxiliary power supply not switched on, or power lead plug loose

- stylus or cursor not in proximity to active surface
- cable not properly connected

c. If "PRT" in reverse video, printer is switched off or cable not properly connected.

d. If "MSE" in reverse video, the ISI, mouse or trackerball is not working. Check:

- ISI unit (Intelligent Serial Interface) failed
- cable not properly connected.

e. If "FUSE" in reverse video, the peripheral port fuse has failed (Model B only). May mean other peripherals are in reverse video also (e.g. "K/B"). This can also imply that no +12 volt supply (+/- 15V on UNIBUS) is present on the backplane. Check fuse integrity and backplane power supply.

2.2.7 User Maintenance and Upgrade Procedures

2.2.7.1 Maintenance

No maintenance of the SBD need be carried out by the user, other than normal procedures of handling and cleanliness which would apply to a sophisticated computer component. For example:

- o Take care when unpacking not to damage the board.
- o Avoid exposing the board to static discharges, e.g. when handling.
- o Take care when inserting the board into a computer backplane.
- o Ensure that the computer's fans do not become blocked by a build up of dust in the filters or by any external object.

If the board is sold as part of a larger system such as an Industrial Terminal or Q Box, then maintenance of the system may be required and directions for this will be included with the relevant documentation.

If the board fails, then it should be returned for repair. Please contact Gresham Lion (PPL) Service Dept. to arrange for this to happen. Also contact PPL if you would like details about Maintenance Agreements to cover our products.

2.2.7.2 Firmware Upgrades

As part of Gresham Lion (PPL)'s policy of continued improvement, new releases of the SBD controlling firmware may be issued from time to time, and these will be available as upgrades to existing systems at a small additional cost (depending on whether it is carried out by the customer or a Service Engineer, and whether the replaced components are returned to PPL).

The upgraded firmware is packaged in four Erasable Programmable Read Only Memory (EPROM) chips. Upgrading is normally a simple replacement, detailed in the following sections. In some cases an upgrade may require the board to be returned to the factory because of hardware modifications: this will be stated in the release note.

a. Note: directions used are when looking at the SBD's component side, with the ribbon cable (IDC) connector at the top. At all times avoid touching the pins of the EPROMs with fingers or any non-conducting material which may contain a static charge.

b. The EPROMs are inserted in the four sockets at the lower left, labelled 1H to 4H. (Note:—on some early Q-bus boards, 1H is erroneously labelled H1). The EPROMs are labelled 1H to 4H. (Do not confuse HI, short for HIGH BYTES, with 1H the socket number on the label.) On Model B, EPROM sockets are labelled 1K to 4K.

c. Carefully lever out the existing EPROMs with a suitable tool such as a wide-bladed screwdriver. Try to avoid bending the legs. Please return these to PPL to avoid being invoiced for the new EPROMs.

d. Consult Figure 2.1 or 2.2 showing the positions of the patching links on the board. Each EPROM position has an associated link. Using a wire-wrap tool, or jumpers, make links as shown in the table below (last column).

Socket	Link	EPROM Number	EPROM Name	EPROM Type	Link
4H/4K	L4	4H	ODD LO	27128 27256	b-c (Model A) b-a (Model B)
3H/3K	L3	3H	ODD HI	27256	b-a
2H/2K	L2	2H	EVEN HI	27256	b-a
1H/1K	L1	1H	EVEN LO	27128 27256	b-c (Model A) b-a (Model B)

e. Install the new EPROMs, with the notch on the right, in the four sockets according to their labels. Ensure that all pins are engaged in the sockets before pushing home with minimum necessary force.

f. Re-install the SBD and switch on power. The standard power-up message (see 2.2.6.1) should appear on the last VDU line of the display. If any problems occur contact the Customer Service Desk at Gresham Lion (PPL) Ltd.

g. EPROM types 27128 each contain 16K bytes of firmware, and types 27256 contain 32K bytes, giving a total of 96K bytes for the SBD firmware. Model B uses 4 x 27256 EPROMS since it runs at 12.5 MHz and 27128 EPROMS of the right speed are unavailable. Earlier releases of Model A firmware (pre-2.1) were contained in four 27128s (64K bytes total). If you are upgrading from these, you will have to change links L2 and L3 as shown above. If necessary, return the SBD to the factory for upgrade.

2.3 SBD Firmware Configuration (SETUP)

SETUP is a means by which you can configure the way in which the firmware on board the SBD behaves, without the need for hardware jumpers on the board. It is simple and self-explanatory and is operated by single-key depressions from the SBD's keyboard or any external VDU with the required operating characteristics.

Once you have made your selections, you may then store them in non-volatile memory (NVRAM) so that they are preserved even when the power to the board is switched off.

The next few sections are a brief guide to SETUP, explaining how to operate it, what the various options mean and their effects on the behaviour of the SBD, and how to install the optional extras.

Note: SETUP is sometimes referred to as "setup mode", "configuration mode", or "the configuration dialogue". All these terms are synonymous.

SETUP is a destructive operation, since on exit from it the SET command is executed, all archives and SAS context records are lost, all pixel and VDU pages are erased, and the power-up message is displayed. All the options in SETUP are designed to be chosen once when the SBD is installed, and then left alone. (They should be considered as replacements for patches and links).

This is in contrast to the operator comfort VDU features selectable via Terminal Setup (see 9.5) which can be entered only by pressing the <SETUP> key on the SBD keyboard. Terminal Setup does not affect the pixel area or archives, and it is possible to set up the board so that the VDU data are restored as well. (See the SET command.)

Requirements for SETUP operating device

SETUP can always be operated from the SBD keyboard, whether or not it contains a PCU. However, if this is not available, most VDU terminals can also operate SETUP. All you need is a configuration cable connecting the SBD peripheral port to the VDU. If you have not ordered a keyboard with your system, a configuration cable suitable for use with DEC VT100 or compatibles will normally be supplied. Ordering information for this cable is given in 1.8, and a wiring diagram is shown in Figure 2.7.

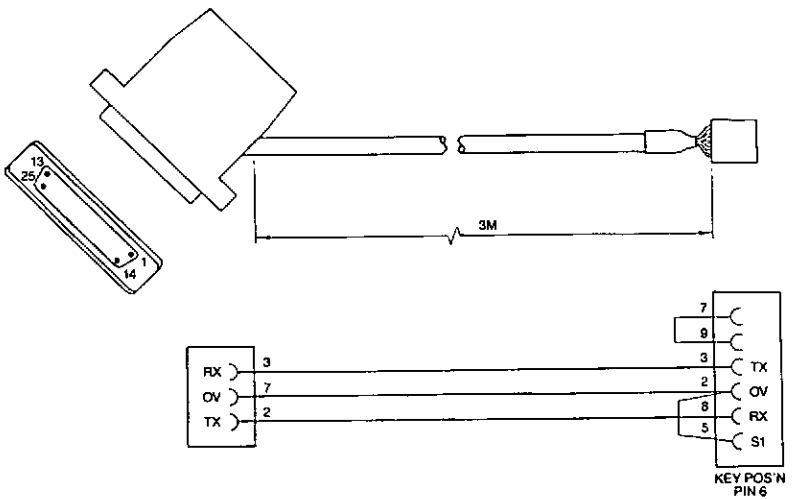


Figure 2.7 Wiring for SETUP configuration cable

This shows the wiring for the SBD's 10-pin connector. For corresponding SGT/SBX/SIT/MMAP connections, see Table 1.8, section 1.8.10 (N.B. the connection shown between 7 and 9 is not required for these connectors).

The VDU must be set to the following characteristics:

- RS232/423 electrical format (data leads only)
- transmit speed 9600 baud
- 8 data bits with no parity OR 7 bits plus any parity (8th bit is ignored once in SETUP)
- 1 start bit, 1 stop bit
- capable of generating a BREAK condition on the serial line (usually by pressing the BREAK key)

An unorthodox way of operating SETUP, if no keyboard and no suitable terminal are available, would be to connect an RS232 serial line from a host port to the SBD's keyboard port, then write a utility to send the appropriate codes to the SBD. The host terminal driver must be capable of sending 8-bit data in order to send the code which enters the dialogue. The codes which such a utility would be required to send to the SBD are given in Table 2.4.

Character values				Meaning of character
Octal	Dec	Hex	ASCII	
(354	236	EC		Enter SETUP
(BREAK on line				
033	27	1B	<ESC>	Exit from SETUP
(161	105	71	q	Move to previous menu option
(361	241	F1		(up arrow on SBD keyboard)
(162	106	72	r	Move to next menu option
(362	242	F2		(down arrow on SBD keyboard)
061	049	31	1	*Set resolution to 768x574 50Hz interlace
062	050	32	2	*Set resolution to 512x384 50Hz N/I
063	051	33	3	*Set resolution to 768x574 50Hz N/I
064	052	34	4	*Set resolution to 512x384 60Hz N/I
040	032	20	<SP>	Change current option forwards
012	010	0A	<LF>	Change current option backwards
123	083	53	S	Save current values in NVRAM
122	082	52	R	Restore NVRAM values

Table 2.4 Character values used to operate SETUP

Note: receipt of these codes alone is enough. There is no need to append <CR> or any other terminator.

* = Different resolutions on low resolution and North American versions (see Table 2.5).

2.3.1 Using SETUP

2.3.1.1 Entering SETUP

Using the SBD keyboard: hold down <CTRL> and <SHIFT> and press <BREAK>.

Using a VDU: press the <BREAK> key to enter SETUP. The cable must have been plugged into the SBD when it was powered up. Nothing is echoed to the screen of the VDU; the whole procedure takes place on the monitor driven by the SBD.

You should see the screen clear and then display a menu headed "SINGLE BOARD DISPLAY SETUP". An example of the menu is shown in Figure 2.8.

If something did obviously happen but the screen is not correctly formatted, it may be that your monitor is not correctly adjusted or else the SBD has powered up with a different screen resolution from the monitor. You can test this by pressing keys from 1 to 4, which alter the displayed resolution of the SBD to one of the four available values, as given by Table 2.5. If none of the resolutions will synchronise the screen then try adjusting the monitor, with the SBD set to the resolution you wish to use. Note: use of these keys does not affect NVRAM or any of the displayed options.

When you can see the menu you will notice that it consists of a series of headed options in two columns, with the VDU cursor placed on the first option (see Figure 2.8).

```
+-------------------------------------------------------------+
|        S I N G L E   B O A R D   D I S P L A Y   S E T U P   |
|                                                             |
|    HOST SERIAL LINE                  VIDEO OUTPUT           |
|       Baud Rate        : 9600           Mode       : 0      |
|       Parity/data bits : NONE / 8                           |
|       Stop Bits        : 1 / XOFF    SET COMMAND            |
|                                         Resolution : 32     |
|    DMA INTERFACE                        Pages      : 2      |
|       Register Address : 764000                             |
|       Vector Address   : 270         SEL COMMAND            |
|                                         Channel    : 0      |
|    PIXEL CHARACTER SET                  Foreground : 7      |
|       Size             : 7 x 9 x 1      Background : 0      |
|       Horizontal Pitch : 12                                 |
|       Vertical Pitch   : 17          MIX COMMAND            |
|       Orientation      : 0              Rows       : 3      |
|                                                             |
|    PIXEL SYMBOL SET                  PROTECTION     :       |
|       Horizontal Size  : 16                                 |
|       Vertical Size    : 16          INTRODUCER     : ~     |
|       Orientation      : 0                                  |
|                                      WATCHDOG       : 8 5:00|
|                                                             |
|q-UP r-DOWN <SP>-OPT(+) <LF>-OPT(-) S-SAVE R-RESTORE <ESC>-EXIT|
+-------------------------------------------------------------+
```

Figure 2.8 Appearance of SBD SETUP menu

Note 1: the WATCHDOG option has no effect on Model A.

Note 2: the screen boundary shown is for a 64-character wide display. Two of the resolutions on the low-resolution board, and all on the high-resolution board, have an 80-character wide display (the last 16 positions on each line are not used by SETUP).

KEY	High-resolution board				Low-resolution board				North American board			
	Resoln (X x Y)	Field freq (Hz)		Line freq (KHz)	Resoln (X x Y)	Field freq (Hz)		Line freq (KHz)	Resoln (X x Y)	Field freq (Hz)		Line freq (KHz)
1	768x574	50	I	15.6	768x574	50	I	15.6	768x574	50	NI	30.0
2	512x384	50	NI	20.1	512x512	50	I	15.6	704x526	60	NI	33.2
3	768x574	50	NI	30.0	384x293	50	NI	15.6	640x480	60	I	15.8
4	512x384	60	NI	24.2	256x256	50	NI	15.6	480x360	60	NI	23.8

Table 2.5 Operation of Resolution Selection Keys in SETUP

2.3.1.2 Moving the Cursor

The VDU cursor identifies the option that may be changed. When using the SBD keyboard, it can be moved to the option immediately below the currently selected one by pressing the "down arrow" key, or moved to the option above by pressing "up arrow". The cursor moves from the bottom of one column to the top of the other and vice versa.

When using a VDU to drive SETUP, you cannot use the direction keys (if any) - these will exit you from the menu, since they generate character sequences beginning with <ESCAPE>. Instead, use "q" (lower case) to move to the previous option and "r" (lower case) to move to the next, as directed by the prompt line at the bottom of the screen. Make sure you do not have the (CAPS LOCK) key on.

If the keyboard you are using has autorepeat you will find that this is operational, and is sometimes helpful in moving quickly to the desired option and selecting the appropriate value.

2.3.1.3 Changing an Option

Each configurable option has a list of possible values associated with it, the current active value (not necessarily that stored in NVRAM) being displayed on entry to SETUP.

While the cursor is positioned on any particular option, pressing the space bar of the keyboard or VDU will step forward through the list of values for that option, erasing the previous value and displaying the next in its place. Automatic wraparound occurs between the first and last values in the list, in either direction.

Pressing the <LINE FEED> key will step backwards through the list of values, which can be useful for options with a large list of values. The comments above on the use of autorepeat apply particularly here.

The range of possible values for each option is given in 2.3.2.

2.3.1.4 Saving and Restoring the Configuration

All alterations to configurable options made in SETUP become operational when you exit from it, until you enter SETUP again and make further changes. The current operational configuration is always the one displayed just before exit from SETUP. However, if the power to the SBD is switched off, or a bus reset occurs, all changes made and not saved in NVRAM are lost, and the NVRAM values are restored.

To save a configuration in NVRAM, press "S" (upper case, i.e. with the SHIFT key depressed) while in SETUP after making the desired changes. The screen is cleared and redisplayed to confirm that the configuration has been saved. The SBD remains in SETUP and further changes can be made if necessary.

You should be aware that, if your currently active SETUP option values differ from those stored in NVRAM, and you then enter Terminal Setup (see 9.5) and press <SHIFT>S, the SETUP values will be saved in NVRAM at the same time as the Terminal Setup values. (This also applies in reverse, if you change Terminal Setup values temporarily.) For this reason it is not a good idea to operate the SBD for long periods with temporary option values selected.

For Model B, the VDU text colour is also held in NVRAM. It is set to white at the factory and there is no option in SETUP to change it. However, if it is modified during operation (using the MAT command) and then SETUP is entered and <SHIFT>S pressed, the current VDU colour is saved and will be operative next time the SBD is powered up or reset.

The configuration currently stored in NVRAM can be restored by pressing <SHIFT>R. This overwrites any SETUP or Terminal Setup values which may have been changed but not saved. Again, the SBD remains in SETUP mode.

2.3.1.5 Automatic Option Changing

Technically it is possible to create a configuration with conflicting options. For instance, a character font could be selected which would not fit in the current character cell, or altering the default channel number could cause the current foreground and background colours to be invalid.

Where such situations occur, SETUP will automatically select appropriate values for the invalidated options and will display them. They may still be changed if necessary, although SETUP will not let you change them back to the invalid values.

2.3.1.6 Exiting from SETUP

Once you are happy with the configuration, and have saved it if necessary, you can exit from SETUP by pressing escape (ESC). When this is done the SBD goes through its power-up checks and displays the power-up message after clearing the VDU screen. On exiting, the options displayed are used to configure the SBD, not those saved in NVRAM.

2.3.1.7 Summary of Commands

Note: it is never necessary to terminate a command with a carriage return - single key depressions (or simultaneous multiple ones) are all that is required. In the following table, brackets delimit the key to be pressed, and plus signs indicate simultaneous depression. Unless otherwise stated, all keys are unshifted.

Key depressions	Action
<BREAK>	enter SETUP from VDU
<CTRL>+<SHIFT>+<BREAK>	enter SETUP from SBD keyboard
<1> <2> <3> <4>	alter displayed resolution of the SBD. Does not affect NVRAM or the currently selected SET option.
<UPARROW> <r>	move to previous option
<DOWNARROW> <q>	move to next option
<SPACE>	select next value in list
<LINEFEED>	select previous value in list
<SHIFT>+<S>	save new values in NVRAM
<SHIFT>+<R>	restore old values from NVRAM
<ESC>	exit from SETUP

Table 2.6 SETUP command keys

2.3.2 Power-up Initialization Settings

HOST SERIAL LINE

These options determine the parameters used to set up the RS232 serial line to the host computer.

Baud Rate

This defines the speed at which the host serial line operates, in bits per second. Selectable values are 110, 300, 600, 1200, 2400, 4800, 9600, 19200. The factory setting is 9600 which is equivalent to about 1000 characters per second. The SBD is technically capable of being set to 38400 baud using RS423 signal levels. If you require this, consult Gresham Lion (PPL) Ltd.

Parity / data bits

This option defines both the type of parity (on the left of the slash) and the number of data bits in the character (on the right). Parity may take the following values:

None - no parity bit (factory set)
Mark - parity bit always 1
Space - parity bit always 0
Even - parity bit depends on character
Odd - parity bit depends on character

There may be 7 or 8 data bits (factory set to 8).

Stop Bits

This option defines both the number of stop bits in the character (on the left of the slash) and the data rate control method (on the right). There may be 1 (factory setting) or 2 stop bits. Data rate control is either XOFF (XON/XOFF - factory setting) or EIA (via CTS and DTR).

When EIA protocol is used, the SBD sets DTR low whenever it is unable to accept further characters into its receive buufer. Similarly, the SBD will cease transmitting to the host whenever the host sets CTS low.

If the keyboard (transmit-to-host) buffer fills, the "K/B LOCK" LED on the keyboard is lit, and further key depressions simply cause the keyboard to beep.

DMA INTERFACE

These options set up the parameters for operating the DMA interface, which are the address of the first DMA register and the interrupt vector address. When using a DMA driver it should be configured or set to use these parameter values.

Register Address

This defines the address of the first of the four DMA control registers. The registers appear at successive word addresses in the I/O page but occupy only the least significant 8 bits of each address. The address is given in SETUP in 18-bit octal; however, the address at which the registers actually appear depends on the system architecture - for example:

16-bit system	18-bit system	22-bit system	
160010	760010	17760010	(octal)
E008	3E008	3FE008	(hex)

The possible range of values differs between Model A and Model B.

Model A standard	: 760010, 760020, 760030.... 763770	default 760010
Model A modified	: 764010, 764020, 764030.... 767770	default 764010
Model B	: 760040, 760100, 760140.... 777740	default 764000

Between the largest and the smallest address the option appears as "NO DMA". This is intended for use in situations where there is no host processor to provide bus arbitration, or else no bus (as in the serial-only configurations). It prevents the SBD from attempting to access the DMA registers which would cause a bus error.

WARNING: do not attempt to set NO DMA with a Model A SBD when it is in a backplane with a host, since the register address is then undefined and the host may be prevented from booting. For Model A (standard) and Model B, on VAX computers, the DMA address should be set above all other devices in the first quarter of the I/O page.

Vector Address

This option defines the address of the interrupt vector to which the SBD will interrupt when completing a DMA transaction (with interrupts enabled). This is specified in octal on 2-word boundaries and has the same range for all versions of the board. If the Register Address is set to "NO DMA" this value is not used since the DMA interface is disabled.

Range 0, 4, 10, ... 374 (Factory set 270)

PIXEL CHARACTER SET

These options define the default parameters for display primitives which use the pixel character fonts (ALP and MAG). They can be changed by means of the SCA command.

Size

This gives the size of the font selected together with its height (single or double). The range of options is:

Option	SCA font code	Meaning
5 x 7 x 1	0	5x7 font, single height
5 x 7 x 2	1	5x7 font, double height
7 x 9 x 1	2	7x9 font, single height
7 x 9 x 2	3	7x9 font, double height
11 x 16 x 1	4	11x16 font, single height
11 x 16 x 2	5	11x16 font, double height

Horizontal Pitch

This gives the width of the background cell for the font, in pixels. The possible range of values is 6 to 20, but the larger fonts force a larger minimum:

Font 5x7 - minimum 6
Font 7x9 - minimum 8
Font 11x16 - minimum 12

Vertical Pitch

This gives the height of the background cell for the font, in pixels. The possible range is 8 to 32, but the larger fonts force a larger minimum:

Font 5x7 - minimum 8
Font 7x9 - minimum 10
Font 11x16 - minimum 17

The minimum value (one pixel clearance) will cause lowercase characters with descenders to be out of position and is therefore not necessarily the best choice. See the SCA command for a discussion of this.

Orientation

This is in the range 0 to 3 and corresponds to the orientation codes described for the SCA command:

0 = normal (character right, string right)
1 = hotel-sign (character right, string down)
2 = up (character up, string up)
3 = down (character down, string down)

PIXEL SYMBOL SET

These options define the default parameters for the SYM command. They can be changed during operation using the SSA command. NOTE: symbol definitions are not preserved if SETUP is entered.

Horizontal Size

This gives the width of the symbol cell and therefore defines the number of bits per word used when defining symbols. The permitted range is 1 to 16.

Vertical size

This gives the height of the symbol cell and therefore defines the number of words required to define a symbol. The permitted range is 1 to 16.

Orientation

This defines the default orientation for strings of symbols. It has the same range (0 to 3) and meanings as for the pixel character set orientation, described above.

VIDEO OUTPUT

Mode

This gives the default value for the video colour mapping. The values used are the same as the argument for the SOM command. Mode zero is factory set.

Mode Value	SBD(A) Mono	SBD(A) Colour	SBD(B)
0	8 grey levels plus white overlay	8 colours plus white overlay	As SBD(A) colour
1	8 grey levels plus 8 flashing grey levels Rate/MSR link L7	8 colours plus 8 flashing colours Rate/MSR link L7	As SBD(A) colour Flash rate 0.8 Hz(*) MSR 1:1
2	16 grey levels	16 grey levels	As SBD(A)
3	16 fixed colours	16 fixed colours	As SBD(A)
4	Illegal	Illegal	As for SBD(A) mono mode 0
5	Illegal	Illegal	As for SBD(A) mono mode 1

Table 2.7 Colour mappings corresponding to Output Mode values

(*) 0.67 Hz for 60Hz frame rate.

You will note from Table 2.7 that the SBD Model B, which allows a much wider range of colour mappings, is limited in its defaults to an emulation of the two types of Model A board. The colour mapping can be changed during operation by the SOM command (Model A or B) or by the MAT command (Model B only). See 3.7.3 for the actual RGB values corresponding to these colours/levels.

The flash rate is alterable by a hardware link on Model A (see 2.2.1) and by the FSH command on Model B.

NOTE: although the VDU text colour for Model B is fully programmable by the MAT command, there is no corresponding default specification in SETUP. The VDU colour is, however, held in NVRAM, and the value at the time the SETUP configuration is saved becomes the new power-up default. The value is factory set to white.

SET COMMAND

These options set up the default operating resolution. They are alterable in use by means of the SET command.

Resolution

This corresponds to the first argument of the SET command (q.v.) and sets up the screen resolution (e.g. 32 gives 768x574 non-interlaced). The range of values is x = 30, 31, 32, 33 and (50+x), (200+x), (250+x). Note that values involving the addition of 100 (100+x, 150+x, 300+x, 350+x) although permitted by the SET command, are not allowed as SETUP values for this option. This is because it is not sensible to power up without erasing the pixel memory. The factory setting is 32 unless the SBD is supplied as part of a system requiring a different resolution.

On the low-resolution version, the base range of x is 2...5. On the North American version, it is 40...43. See the SET command in 3.10.

Pages

This sets up the default number of pixel pages. Although the theoretical range is 1 to 13, in practice the maximum is constrained by the resolution code selected. Maximum values for the different resolutions are given in 3.3.3, and also under the SET comand in 3.10.

NOTE: If the resolution code is 30 (or 80, 230 or 280), then if the number of pages requested is 1, the SBD will power up with resolution 768x586 (interlaced). If the number of pages is 2, the SBD will power up as 768x574 (interlaced). This applies to both high- and low-resolution versions. (SET code is 2 for low-resolution boards).

SEL COMMAND

As the heading suggests, these options set the default values which the SEL command can modify during operation. Most application programs would ignore the defaults as good programming practice.

Channel

Selects the default channel, in the range 0 to 7. The option selects both foreground and background channels.

Foreground

Selects foreground colour (pixel value to be written to bitplanes as foreground). The range is 0 to 15 depending on channel selected, e.g. for a channel with only one plane such as channel 5, the permitted range would be 0 or 1. Note that you cannot select complement colour (-1).

Background

Selects the background colour with the same range as the foreground colour. Note that you cannot select additive mode (-1), or any of the fill inversion values (100+n). The background channel is the same as the foreground channel on power-up.

MIX COMMAND

Rows

This option defines the number of rows of VDU scrolling text that will be visible at the bottom of the screen on power-up. This can be overridden during use by the MIX command. The range is 0 to 23. There is also the option to select "VDU" which means that the screen is wholly VDU on power-up (equivalent to executing the VDU command). This is provided for use when the SBD is being used as a system console and must therefore display whole screens of text when the system is booted (as on the MicroPDP). When "VDU" is selected, pressing the MIX key on the keyboard will generate an implicit MIX(3) unless the MIX command (with a non-zero argument) is used to change this.

If zero MIX lines are selected, this will cause the SBD to power up with only pixel data visible, and text written to the VDU will not cause it to enter MIX. Since this mode of power-up will obscure the diagnostic message on the bottom VDU line, PPL strongly recommend that you not use it. If you do, the MIX key on the keyboard has the same effect as for "VDU" above (an implicit MIX(3)).

PROTECTION

This is a special option. If you position the cursor on this option and press <SP> or <LF>, you will enter a mode which allows you to enable software options. This is described more fully in 2.3.3.

INTRODUCER

This option allows you to define the default introducer character used by the ASCII human-readable serial line command format (see 7.2). It is factory set to tilde (~). It can be altered during operation by means of the CHI command. The option can also be set to "NONE" which disables the ASCII protocol and means that all 7-bit characters are treated as VDU text. This is useful if the ASCII protocol is not used and you wish to use the SBD to edit files which may contain the introducer character.

The range of valid introducers is given in the description of the CHI command in 3.10. The binary protocol is not affected by this option; it cannot be disabled.

WARNING: Changing this option may invalidate programs which use the ASCII protocol to communicate with the display. The host library can be configured to use a different introducer by default.

WATCHDOG

This option is used by Model B only. It allows you to set a maximum time for which the SBD will wait for commands or text from the host before blanking or flashing the screen. It is specified in minutes and seconds, in the range 0:01 to 8:31 (i.e. maximum of 511 seconds). There is also an "OFF" option which disables the watchdog, so that the interval between communications can be indefinite without causing screen blanking or flashing. "OFF" is the factory setting.

The letter in front of the time value indicates the action to be taken when the watchdog timer expires. (When the watchdog is set to "OFF" the letter does not appear). The letter has two possible values:

- B (Blank) indicates that the screen is to be blanked to black.
- F (Flash) indicates that the screen is to flash to black at a rate of 1 second on, 0.67 seconds off (in time with the VDU cursor).

The look-up tables are specially loaded to achieve this. When a command is subsequently received by the SBD, the previous look-up table values and flash rate are restored.

There are two points at which the "OFF" option appears, as F changes to B and vice versa. The factory setting is the "OFF" value between B 8:31 and F 0:01.

NOTE: this feature, which is implemented by firmware, should not be confused with the hardware-implemented automatic watchdog which blanks the screen if the 68000 microprocessor fails. The hardware watchdog can only be disabled by inserting link L7 (b-c).

2.3.3 Installation of Options.

Explanation

When you position the cursor on the PROTECTION field of the SETUP menu and press <SPACE> or <LINE FEED>, you will find that the screen is cleared and some different text is displayed. You are now in a special phase of SETUP which allows you to install and enable optional software. "Optional" software is software for which you have paid separately and for which you have been supplied a licence.

NOTE: when acquiring an SBD system for the first time, you should find that all options purchased have already been enabled. It is normally only necessary to carry out this procedure where options are installed on an existing SBD installation.

Currently the optional software includes:

- the local segment storage or archiving package ('Arcaid') which can be added to the on-board firmware to allow access to RAM not used for screen display. Arcaid is standard for serial-only boards.

- the following interactive host applications packages:

  - DRAW, a mimic diagram generation package - requires Arcaid, keyboard and/or trackerball (RSX-11M or VAX/VMS)

  - GLIDA (Gresham Lion Interactive Drawing Aid), a freehand drawing and diagram utility - requires Arcaid and an input device (in order of preference: tablet, mouse, trackerball, keyboard). (RSX-11M, RT-11 and VAX/VMS).

  On the licence you will find:

- the serial number of the SBD to which the licence applies. This number should correspond with the serial number displayed at the top of the Installation screen of SETUP. If it does not, you will not be able to install any of the options.

- sections labelled 1 to 17, each referring to a single "program number". The first 12 entries correspond to the positions in the 12-digit binary "protection code" which you will see displayed on the screen. Program number 1 corresponds to the digit at the extreme right of this code, program number 12 at the extreme left. All digits are either zero (program disabled) or 1 (enabled). When you first enter this phase of SETUP all digits should be zero. The correspondence between optional software packages and program numbers is given in Table 2.8.

- For each package (option) which you have actually purchased, you will find with the corresponding program number on the licence a 6-character sequence of ASCII numbers, symbols and upper-case letters in the range 0 (zero) to Z. This is the password which will allow you to enable the option for that particular SBD.

An example of the upper part of a licence is shown in Figure 2.9. This example covers options ARCAID and GLIDA for SBD number 4660.

LICENCED USER : Acme Leisure Products

CUSTOMER PURCHASE ORDER : 80299

LICENCE NUMBER : 1072

LICENCE COMMENCEMENT DATE : 12 August 85

AUTHORISED SYSTEM SERIAL NO : SBD-Q-4660

This Certificate confirms that the User is licensed by GRESHAM LION (PPL) LTD. to use the following Software Products in accordance with the Conditions stated below.

SOFTWARE PRODUCTS LICENCED Issue Date : 1 July 85

1	ARCAID-KO7DKN	2		3	
4	GLIDA -X2>@7B	5		6	
7		8		9	
10		11		12	
13	GGS	14	SB	15		16		17	

Figure 2.9 Example licence for SBD software products (upper part)

Operation

The Installation phase of SETUP is largely self-explanatory. However, a brief guide is given here.

On entry to the Installation screen you will be prompted for a program number (see above). Consult your licence and enter the program number of the first purchased option. (In our example this would be 1). Invalid input causes an error message and reprompt.

When a valid program number is entered you will be prompted for a password. Enter the 6-digit sequence of characters opposite the program number on the licence (in our example, KO7DKN). Ensure the letters are upper case. Note that zero would have a slash through it to distinguish it from letter "O".

You should then see a message informing you that the option has been enabled, and you will be reprompted for a program number. At this point you may choose to exit back to the normal SETUP menu by pressing <ESC>, or you may continue with the next program number on your licence.

IMPORTANT

After exiting back to SETUP, you MUST type <SHIFT>S to save the new protection code in NVRAM, otherwise the enabled options will be lost when power is switched off and the process must be repeated.

Failure to Enable Options

If you do not enable Arcaid, attempts to use archiving commands (and SAS, RPC) will generate errors stating that there is no archiving memory available or the archive directory is full.

If you do not enable a host package, attempts to run it will result in the screen going black, and the SBD will no longer function until reset.

Program Number	Package Name	Use	Enable required
1	Arcaid	Archiving (local segment storage)	Yes
2	DRAW	Mimic generation	Yes
3	-		
4	GLIDA	Diagram generation and freehand drawing	Yes
5-12	-		
13	GGS	Host library	No
14	SB	DMA driver	No
15-17	-		

Table 2.8 Correspondence between program numbers and packages

2.4 Software Installation

This section describes how to install the host software which will have been supplied with your SBD, according to your order.

First of all, you should physically mount the medium containing the software distribution on your system and make a backup copy of it. It is assumed that you know how to do this for your particular operating system. If not, consult the operating system documentation.

Next, copy the software from the distribution medium to your working disk. The software is in directory [200,200] (RSX), or [200200] (VMS). For RSX and RT-11, the build command file (described later) allows you to build and install software direct from the distribution medium, but this may not be advisable since such a medium (e.g. floppy disk) is likely to be slow and have limited free space, and where possible will be write-protected.

Further steps are described in terms of the specific operating systems.

2.4.1 RSX-11M/M-PLUS and Micro-RSX Software Installation

Load the distribution medium into a suitable drive and mount it. For instance, if it is an RX50 floppy diskette:

MCR>MOU DU1: DCL>MOUNT DU1:

You should then make a backup copy of the medium. Where possible, the medium will be supplied write-protected. If it is a magnetic tape, remove the write-enable ring first.

Next (optionally) copy the distributed software into a suitable directory, e.g. if the current default:

MCR>PIP /NV/CD=DU1:[200,200]*.*

DCL>COPY/OWN DU1:[200,200]*.* SY:

One of the files in the distribution kit is an indirect command file called GGSBLD.CMD. This command file allows the user to configure and build the RSX host software for the SBD. (It also supports the host software for another product, Supervisor 1024). The command file will run under either MCR or DCL.

GGSBLD executes in three phases. The first phase allows the user to specify the location of the source files and the eventual destination of the built software. It also performs a number of checks to ensure that the build can complete successfully. The second phase allows the user to specify a number of configuration details so that the software can be built correctly for the environment it is to be run in, and also so that unnecessary features can be excluded to save space. By default the library will contain all features, and if you select the default configuration, the only questions asked will be those requiring a selection from alternatives (e.g. host processor type, DMA address etc.). The third phase actually builds the software.

To run the command file (assuming that it is in the default device and directory), type:

>@GGSBLD

under either DCL or MCR.

The command file will ask a number of questions, which are largely self-explanatory. Online help is available by pressing <ESC> in response to any question (there may be a slight delay); the question is then repeated after the help text is printed.

The first two questions ask whether you intend to build the library (GGS) and/or the DMA driver (SB). Your response to these questions affects the operation of the configuration phase.

Therefore, if you wish the library to support the DMA interface, you should answer "Y" to the question concerning the DMA driver. If you do not wish to build the driver at this time, carry on to perform the configuration and then skip over the build phase by answering "N" to the first question in that phase. Then invoke the command file again, tell it that you only wish to build the library, skip the configuration phase and execute the build phase. As you can see, it is quicker to build both library and driver together if this is possible.

If you already have some knowledge of the SBD, you can omit the configuration phase by editing the configuration file directly beforehand. (This file is called CONFIG.MAC and is created by the configuration phase of GGSBLD. A preconfigured version is supplied in the distribution kit.) There are enough comments in this file to allow easy reconfiguration.

When building the SBD DMA driver, you must have write access to the system disk and directory, which normally means that you must be working in a privileged account when on a multiuser protected system.

NOTE: Certain system files created during the SYSGEN process are required to be present on your system disk in order to build both the driver and the library. The system checking phase of GGSBLD verifies the presence of all required files and informs you if they are not all present.

The command file does not install the DMA driver. The method of doing this is:

```
>LOA SB:
```

On RSX-11M systems, the DMA register and vector addresses are built into the driver (they are among the questions asked by the GGSBLD command file). They can only be changed by rebuilding the driver and rebooting the system.

To check that the driver has installed correctly under RSX-11M, type the following:

```
MCR>DEV SB:                 | DCL>SHOW DEVICES SB:
SB0: Loaded                 | SB0: Loaded
SB1: Offline Loaded         | SB1: Offline Loaded
MCR>                        | DCL>
```

In this example, the driver has been built to support two devices, but when it was loaded, the Executive found that the second device was not at the CSR address specified when the driver was built. The Executive therefore assumes that the second device is offline.

On RSX-11M-PLUS and MicroRSX systems, the DMA register and vector addresses used by the driver can be changed dynamically. Only the number of units supported is fixed when the driver is built. You can change these addresses as in the following example, provided you are in a privileged account:

```
>CON SET SBA CSR=160010 VEC=270
>CON SET SBB CSR=160200 VEC=170
>CON ONLINE SBA,SB0:                    (or CON ONL ALL)
>CON ONLINE SBB,SB1:
>CON DIS ALL ATT FOR SB
SBA     CPA     Online,Accpath,Driver
                CSR=160010, Vector=000270, Pri=000005, Urm=000001
SB0:    SBA0:   Online,Accpath,Driver
 Port SB0A SBA0:        Online,Current,Accpath
SBB     CPA     Offline,Driver
                CSR=160200, Vector=000170, Pri=000005, Urm=000001
SB1:    SBB0:   Offline,Driver
 Port SB1B SBB0:        Offline

>
```

In the above example the Executive could not find the second device at the specified address and so assumed that it was offline.

Note that the SBD is a single-unit-per-controller device, and so different units (SB0:, SB1: etc.) have different controller names (SBA, SBB etc.).

It is advisable to put the commands for loading the driver in the startup command file (LB:[1,2]STARTUP.CMD).

NOTE: if you are using the serial interface to the SBD, you should set the following options for the serial port (TTnn:). These commands are also privileged and could be performed on startup.

```
MCR>SET /BUF=TTnn:132.          DCL>SET TERMINAL:TTnn:/WIDTH:132.
```

If you are using the binary serial protocol, you must in addition set the following (these prevent certain control characters from being expanded or inserted by the terminal driver):

```
MCR>SET /NOWRAP=TTnn:           DCL>SET TERMINAL:TTnn:/NOWRAP
MCR>SET /HHT=TTnn:              DCL>SET TERMINAL:TTnn:/TAB
MCR>SET /FORMFEED=TTnn:         DCL>SET TERMINAL:TTnn:/FORMFEED
MCR>SET /HFILL=TTnn:0           DCL>SET TERMINAL:TTnn:/CRFILL:0
MCR>SET /NOVFILL=TTnn:          DCL>SET TERMINAL:TTnn:/NOLFFILL
```

None of these options should prevent the SBD from being used as a VT100-compatible VDU as well as a serially-driven display. It may be advisable to set the TT unit to NOBROADCAST in addition while it is

in intensive use, though broadcast messages should not normally interfere with serial commands.

Once the software is built, the driver (if any) successfully installed, and the system options set up, you can compile, taskbuild and run the test program (SBDTST.FTN). The commands used are somewhat system-dependent, but an example would be:

```
MCR>FOR SBDTST=SBDTST
MCR>TKB SBDTST/FP=SBDTST,GGS/LB,LB:[1,1]FOROTS/LB
MCR>RUN SBDTST
```

```
DCL>FORTRAN/NOLIST SBDTST
DCL>LINK/FLOAT SBDTST,GGS/LIBRARY,LB:[1,1]FOROTS/LIBRARY
DCL>RUN SBDTST
```

This program generates a test card via any of the available interfaces, and serves to verify that the SBD and the host software are functioning correctly. If you do not have a FORTRAN compiler, it should not be too difficult to translate the program into your usual high-level language.

2.4.2 RT-11 and TSX-PLUS Software Installation

For RT-11 systems, the host software will operate under SJ, FB and XM monitors. Under TSX-PLUS, each terminal attached to the system appears as a stand-alone RT-11SJ system to the user.

Mount the distribution medium in a drive on your system and (optionally) copy the distributed software to your working disk. For example:

```
.COPY DU1:*.* DK:
```

A command file (GGSBLD.COM) is supplied with the distribution kit which is functionally identical to that supplied for RSX-11M systems. This command file will execute under RT-11 V5.0 onwards only. If you have an earlier version, contact the PPL Customer Support Desk to obtain a build command file for that version (which will not support interactive configuration).

The command file is invoked from the normal KMON prompt by typing "IND GGSBLD", or "IND ddn:GGSBLD" if the command file is in a different device from DK:. See the operational description above under RSX-11M systems. Note that when you type <ESC> to obtain help, you must also type <RETURN>.

Once the library and DMA driver (called a "handler" under RT-11) have been built, you will need to install the handler. To do this, type the following:

```
.INSTALL SB:
```

This command need only be entered once; after that the RT-11 system tables contain the handler information even if the system is rebooted, provided the device is at the correct address (specified during driver configuration) when the INSTALL command is entered. (If not, the error "Invalid device" is produced). If the system is booted with the SBD not at the correct address, the driver must be reinstalled before loading.

```
.LOAD SB:
```

This command must be entered each time the system is booted. If the SBD(s) are not at the address(es) built into the driver, the error "Invalid device" is produced.

```
.SET SBn: CSR=mmmmmm

.SET SBn: VEC=mmm
```

These commands are used if the SBD(s) have different CSR or vector addresses form those specified when the driver was built. They must be entered when the driver is not loaded, e.g.:

.UNLOAD SB:

.SET SB1: CSR=160200

.SET SB1: VEC=170

.LOAD SB:

Addresses must be specified in octal, 16-bit format.

.SET SBn: TIMON=xxxx

This command allows you to specify a timeout value for any DMA I/O to the SBD. The timeout is in decimal seconds, and the maximum value permitted depends on the system clock frequency (determine by SHOW ALL KMON). At 50 Hz the maximum is 1310, at 60 Hz it is 1092.

NOTE: the timeout facility is only enabled if your RT-11 system has been generated with device timeout support.

.SET SBn: TIMOFF

This command disables the timeout option. The driver will wait indefinitely for an I/O to complete. This is advisable if you use the SBD's local cursor regularly for long periods (see the CUR and TAB commands in 3.10).

It is a good idea to put the necessary commands in your startup command file (STARTS.COM, STARTF.COM or STARTX.COM).

You can examine the handler status with the command:

.SHOW DEV

Device	Status	CSR	Vector(s)
DX	Not installed	177170	264
DY	Resident	177170	264
DL	Installed	174400	160
LS	Installed	176500	300 304
NL	Installed	000000	000
LP	Not installed	177514	200
SB	134274	160010	270 170

The line giving SB: device status shows, from left to right, the address of the loaded handler, the CSR address of SB0:, and the vector addresses for SB0: and SB1: (if any).

NOTE: when using the serial line interface to the SBD under RT-11, there are a number of different hardware configurations possible which affect the coding of host applications programs using the host

library. See the description of the INI command in 3.6.

Once the software is built and installed, you can compile, link and run the test program SBDTST.FOR. For example:

```
.FORTRA SBDTST

.LINK SBDTST,GGS

.RUN SBDTST
```

Operation of this test program is as for RSX-11M systems (see above).

NOTE: for XM systems, when using the DMA interface, you may need to link the program twice, as described in 3.8.

2.4.3 VAX/VMS and MicroVMS Software Installation

Before starting installation, you should mount the distribution medium in a suitable drive, e.g. if a TU58 drive 1:

```
$ MOUNT CSA1:/OVER=ID
```

Then copy the files from the medium to a suitable directory, presumably the current default:

```
$ COPY CSA1:[200200]*.* []
$ DISMOUNT CSA1:
```

To install the driver you will need CMKRNL and CMEXEC privileges, or use a system account. The driver should be installed after the SBD(s) are physically installed, and their CSR addresses and vectors have been set to suitable values using SETUP. (NOTE: if the SBD's DMA registers appear in the first quarter of the I/O page (address range 760010 to 763770 octal), they should be set higher than the CSRs of any other device in that range.

Build the driver as follows:

```
$ @SBBLD
```

To install the driver, first copy it to the system directory:

```
$ COPY SBDRIVER.EXE SYS$SYSROOT:[SYSEXE]
```

Then run the SYSGEN program:

```
$ RUN SYS$SYSTEM:SYSGEN          (or MC SYSGEN)
SYSGEN>CONNECT SBA0/ADAPT=n/CSR=%Oaaaaaa/VEC=%Ovvv
```

where:

n = UNIBUS adapter to which SBD is connected (see below). Always zero for Q-bus machines (MicroVAX).

aaaaaa = DMA register address of SBD, in 18-bit octal (SBD factory set at 760010 or 764010 (Model A), 764000 (Model B)).

vvv = interrupt vector address of SBD in octal - (SBD factory set at 270)

Connect further SBD units in a similar way. They will be named SBB0, SBC0 etc. The first CONNECT command automatically loads the driver. It is a good idea to include these commands in the system startup file, SYS$MANAGER:SYSTARTUP.COM.

Other useful commands include:

```
SYSGEN>SHOW/DEVICE=SBDRIVER      (verify driver loaded)
SYSGEN>SHOW/UNIBUS               (list I/O page addresses)
SYSGEN>SHOW/ADAPT                (list UNIBUS adapters)
SYSGEN>SHOW/CONFIG/ADAPT=n       (list devices on adapter)
SYSGEN><CTRL>Z                   (or <CTRL>Y to exit)
```

In VMS systems, because of the 32-bit addressing capability, task address space is not such a critical factor as it is in PDP-11 based systems. The host library is therefore supplied pre-built and ready for use. Source code is also supplied in case users wish to reconfigure the library, and there is a simple command file (GGS.COM) which will rebuild the library. Reconfiguration must be done by editing CONFIG.MAR before the build.

The host library is supplied in two different forms:

- An object library (GGS.OLB). Linking to this makes processes larger and increases link time, but allows symbolic debugging and slightly reduces overhead when running. The object library can reside in any directory.

- A shareable image (GGS.EXE) together with an error message image (GGSMES.EXE). More than one program can link to such a shareable image without the library code becoming part of the program image itself. It is possible to avoid specifying the image in the link command at all, by inserting it into the default image library, viz.:

  ```
  $ LIBRARY/SHARE SYS$SHARE:IMAGELIB.OLB GGS.EXE
  ```

  The library is then linked automatically to any program which calls any of the library subroutines.

  If a number of processes are going to use the image, additional benefit can be obtained by INSTALLing it. This will mean that the code and read-only data are shared by all processes linked to the image, but each process has its own copy of the read/write data, thus saving physical memory and reducing overall image size. To install the shareable image (you must have CMKRNL privilege) type the following:

  ```
  $ COPY GGS.EXE SYS$SHARE
  $ RUN SYS$SYSTEM:INSTALL          (or MC INSTALL)
  INSTALL>SYS$SHARE:GGS/OPEN/SHARE/HEADER RESIDENT
  INSTALL><CTRL>Y
  ```

  Again, it is useful to insert these commands in the system startup command file.

The normal test program supplied is called SBDTST.FOR. This displays a dynamic test card at 768x574 resolution. Other demonstration programs may be purchased separately. All programs are supplied in FORTRAN source format (*.FOR), and must be compiled, then linked as follows. (A command file, BLD.COM, is provided to do this: it links to the library GGS.OLB which it assumes is in the current directory.)

```
$ @BLD SBDTST
```

or

```
$ FORTRAN/OBJECT SBDTST
$ LINK/EXE SBDTST,GGS/LIBRARY/INCLUDE=GGSMES  (if using GGS.OLB)
```

If using the shareable image, the link command becomes:

```
$ LINK/EXE SBDTST,SYS$INPUT/OPT
GGS/SHARE       (specify dev/dir if neither default nor installed)
<CTRL>Z
$
```

If the image is in the default image library, then just "LINK SBDTST" will do. To run the test program, you will have to set up the logical names "ggs_dma" and "ggs_serial" as follows:

```
$ DEFINE ggs dma SBA0           (or desired SBD)
$ DEFINE ggs serial TXB2        (or other serial port)
$ RUN SBDTST
```

A message file, GGSMES.MSG, is provided which contains all the error messages used by the host library. This file conforms to the standard VMS format for message files and can be used by the VMS message mechanism. This means that application specific messages can be added to those already defined for the library, by editing GGSMES.MSG. Care should be taken not to change those messages already defined.

The message file is supplied as a prebuilt version, GGSMES.EXE. If you add messages, compile and link the message file as follows:

```
$ MESSAGE/OBJECT GGSMES
$ LINK/SHARE GGSMES
```

The image can be treated in the same way as GGS.EXE for installing and linking (see above). You can also set it as the temporary default message file (which means that your applications programs need not link to it) as follows:

```
$ SET MESSAGE GGSMES
```

Note: when using the serial interface to the SBD, you may have to set the port so that it is shareable. This is a privileged command. See the VAX/VMS Operator's Guide. For example:

```
$ SET PROTECTION=(WORLD:RWLP)/DEVICE TXB2:
```

If you are using the binary serial protocol, you must in addition set the following terminal characteristics (these prevent certain control characters from being expanded or inserted by the terminal driver). For example, if port TXA7 is to be used:

```
$ SET TERMINAL /WIDTH=255. TXA7:
$ SET TERMINAL /NOWRAP TXA7:
$ SET TERMINAL /TAB TXA7:
$ SET TERMINAL /FORM TXA7:
$ SET TERMINAL /CRFILL=0 TXA7:
$ SET TERMINAL /LFFILL=0 TXA7:
```

None of these options should prevent the SBD from being used as a VT100-compatible VDU as well as a serially-driven display. It may be advisable to set the port to NOBROADCAST as well when not being used as a VDU.

2.4.4 UNIX Software Installation

The UNIX host software will be released in late 1985 when this section will be issued.

2.5 Use of SPR's

At the back of this manual you will find a sample Software Problem Report (SPR) form. This can be used to report any problems you may encounter in using the SBD and its associated host support software. Copies of this form are acceptable, or further copies may be obtained by applying to Gresham Lion (PPL) Ltd.

Note: before completing an SPR you should attempt to reduce the problem to its basics, and supply enough information to allow PPL to reproduce the problem.

How to fill in an SPR

An example completed SPR is shown in Figure 2.10.

(a) Please type or print neatly to ensure legibility.

(b) Leave the first two fields (Title and SPR Number) blank.

(c) Fill in your name (SPR raised by), company name, company address and telephone number (with extension), and the date on which you identified the problem.

(d) In the section headed "Installation Details", enter the information requested as follows:

Host processor - e.g. DEC MicroVAX II, DEC PDP-11/70, etc.

Operating system - including version/revision, e.g. RSX-11M-PLUS 2.2C, MicroVMS 4.2 etc.

SBD Type (Part No) - see Table 1.3 in section 1.9 for the part number of your SBD. (E.g. Model A, Q-bus, colour PAL, high resolution, first quarter of I/O page = 439101).

Firmware version - this is printed on the labels of the EPROMs on the board and also appears at the end of the power-up message (see 2.2.6.1). (E.g. A1.10)

Serial No. - this is printed on the board itself (e.g. SBD-Q-1567). The numeric part is displayed in the PROTECTION page of SETUP (see 2.3.3), and also in the power-up message on the bottom VDU line (see 2.2.6).

Host Serial Line - Enter characteristics of the host serial line if this is used:

Speed - baud rate, e.g. 9600

Data - number of data bits (7 or 8)

Parity - type of parity (None, Mark, Space, Even, Odd)

Stop - number of stop bits (1 or 2)

DRC - Data Rate Control method (EIA or XOFF)

Peripherals used - delete peripherals which are not used in your configuration:

PCU - keyboard plus Peripheral Control Unit (4 connectors in rear)

KBD - standard keyboard (1 connector in rear)

ISI - Intelligent Serial Interface for trackerball/mouse

TB2 - Marconi 2" trackerball (with 3 buttons)

TB3 - Penny & Giles 3" trackerball (or other P&G trackerball)

PRT - inkjet printer (connected to SBD or PCU)

TAB - graphics tablet (TDS or Summagraphics)

TID - digitiser table (Summagraphics ID)

MSE - Penny & Giles mouse

Other - e.g. frame buffer, Cherry keyboard

(e) In the section headed "Application details", give the following information:

Interface(s) used - delete any interface not used.

Special features - please supply any information which may help PPL to recreate your application environment.

(f) In the section headed "Problem details", give the following information:

Likely source - delete all options except the ones where you think the problem lies (note: GGS = host library, SB = host DMA driver).

Brief description of problem - use this space to describe the nature of the problem. Please try to give sufficient information to allow PPL to recreate the problem.

Urgency - delete as required. Please do not specify "Immediate" unless you are unable to use the system as it stands.

(g) Leave the rest of the form blank and send to:

Customer Support Desk
Gresham Lion (PPL) Ltd.
Lower Way
Thatcham
Berks RG13 4RE
U.K.

A Support Engineer will contact you as soon as possible with the results of our investigation.

SUPERVISOR SINGLE BOARD DISPLAY SOFTWARE PROBLEM REPORT

Title :
GL (PPL) use only SPR Number :

SPR raised by : A.I.USER Date : 05 /JUL / 1985
Company : ACME LEISURE SERVICES INC.
Address : 999 DOLDRUM WAY Tel : (0123) 778877
DEADEND
GLOS GL12 1SB U.K.

Installation details (if appropriate) :
Host processor : DEC MicroVAX II Op.Sys : MicroVMS 4.1
SBD Type (Part No) : 439101 Firmware version : A1.10
Serial No. : 1765
Host Serial Line : Speed 19200 Data 8 Parity None Stop 1 DRC XOFF
*Peripherals used : PCU/~~KBD/ISI/TB2/TB3~~/PRT/TAB/~~TID~~/MSE/~~Other~~(state):

Application details (if appropriate) :
*~~Interface(s) used~~ : DMA / ~~serial-ASCII-/-serial-binary-/-Ethernet~~
Special features : Use as VDU

Problem details :
*~~Likely source~~ : ~~GGS-/-SB-~~/ SBD firmware / ~~Unknown-/Other~~:
Brief description of problem (please pay particular attention to the state of the SBD e.g. SET, SEL, SOM, etc.). Continue on other sheet if necessary.

The lowercase "j" character does not descend in either the 7x9 or 11x16 fonts, no matter what the character cell height.

*Urgency : ~~Immediate~~ / ASAP

* = delete where not applicable
GL (PPL) use only Associated SCN number :

Figure 2.10 Example completed Software Problem Report form

2.6 Getting Started

By the time you reach this point you should have installed your SBD(s), seen the correct power-up message, set up your configuration, built the host software (if supplied), and run the test program to verify the correct operation of the SBD. You now wish to familiarise yourself with the system.

You may wish to move straight on to Chapter 3 and begin coding test programs. Section 3.8 describes how to build and run your application programs. However, there are one or two ways in which you can experiment with the facilities available without actually having to create programs. All of these methods require the connection of a serial line from an RS232 port on the host to the SBD's host serial connector, and involve the sending of ASCII format commands (see 7.2) to the SBD.

RSX-11M/M-PLUS and MicroRSX

The MCR PIP utility (COPY under DCL) allows you to send data to a serial device. You can either create a file of ASCII format commands and send it to the SBD, or send commands direct as they are typed in at the terminal. Note: the SBD should not be logged-in if it is being used as a slave terminal.

NOTE: In the examples which follow the ASCII introducer is assumed to be set to query, (?).

To send a file of commands (assuming the SBD is connected to, say, TT3: and the file of commands is called SBD.DAT) :-

```
MCR>PIP TT3:=SBD.DAT    ¦  DCL>COPY SBD.DAT TT3:
```

To send commands direct to the SBD (on TT3:) :-

```
MCR>PIP TT3:=TI:        ¦  DCL>COPY TI: TT3:
?SEL 0 1 0              ¦  ?SEL 0 1 0
?VEC 0 0 767 573        ¦  ?VEC 0 0 767 573
.                       ¦  .
. etc                   ¦  . etc
.                       ¦  .
<CTRL>Z                 ¦  <CTRL>Z
>                       ¦  >
```

Type in the commands in the ASCII syntax (see 7.2). After you type the <RETURN> at the end of each line, the command should be executed by the SBD. In the above example, you would see a red line from upper left to lower right of the screen after typing the VEC command. Type <CTRL>Z to return to the monitor prompt.

RT-11

You can only use the first method in RT-11 (creating a file of graphics commands) since the RT-11 monitor does not support multiple terminals.

```
.TYPE SBD.DAT
```

This causes the monitor to send the contents of the file to the SBD (which would normally be the console terminal). Because of the introducer at the start of each record, the SBD interprets it as a graphics command and executes it.

VAX/VMS

Both of the above methods can be used. Assume the SBD is connected to port TXA12:, which must have been made shareable (see 2.4.3). For a file of commands:

```
$ COPY SBD.DAT TXA12:
```

For direct command entry:

```
$ COPY SYS$INPUT TXA12:
?SEL 0 1 0
?VEC 0 0 767 573
<CTRL>Z
$
```

UNIX

Both the above methods can be used. Assume the SBD is connected to port tty10 (you must have write access to the device). For a file of commands:

```
% cat sbd.dat > /dev/tty10
```

For direct command entry:

```
% cat > /dev/tty10
?SEL 0 1 0
?VEC 0 0 767 573
<CTRL>D
%
```

CHAPTER 3 - CONTENTS

CHAPTER 3 - CONTENTS (continued)

CHAPTER 3 - CONTENTS (continued)

Figures

Tables

3 Host Subroutine Library (GGS)

3.1 Introduction

The GGS library is a software package intended to link to applications programs in the host computer, to provide an easy command interface to the SBD. Essentially it is designed to hide the details of the hardware interface from the applications programmer, allowing him to concentrate on the graphical aspects.

GGS versions are available for most DEC operating systems (RSX-11M, RSX-11M-PLUS, MicroRSX, RT-11, VAX/VMS, MicroVMS) and also for UNIX-based systems. A portable library written in FORTRAN-77 (GGSF77) is available which can run on any host with a suitable compiler, and will drive an SBD using the ASCII serial protocol; this library does not use the command block concept (see below) and does not allow command blocking (host segment storage).

GGS is coded in native assembly language (MACRO-11 or MACRO-32) and is configurable, by conditional assembly, to adjust its size and facilities to the minimum necessary (see 2.4). The UNIX version is written in C.

GGS is not a conventional collection of independent subroutines, since most of its processing is handled by common code. A block diagram of its structure is given in Figure 3.1. It consists basically of two parts: a front-end and a display interface. The data structure which links these two parts is the Command Block (CB), sometimes called a Display List. The function of the front-end is to produce a CB and the function of the display interface is to send the CB to the SBD via the chosen method (DMA, serial ASCII, or serial binary).

An application program may generate its own CBs (the format is given in Chapter 5) and call the library routine OUTPUT to send them through the display interface, bypassing the front-end. (OUTPUT is only available through the assembler-coded libraries for RSX, RT-11 and VMS). This can make programs smaller since they include less of the GGS code, as long as:

- the program does not call any other library routines
- the program does not use the serial ASCII interface
- the library is not configured to provide full error messages.

The call to OUTPUT (as used by FORTRAN) is as follows:

1. CALL OUTPUT (CB)
2. CALL OUTPUT (CB,NR,RA)

where

CB is an array containing a command block in the correct format (see Chapter 5). Its first word must contain the total length of the block in bytes (including the count).

NR is the number of return values expected for the CB - if omitted, this is assumed to be zero.

RA is the location in array CB of the first return value. All return values must occupy consecutive bytes, which normally means that only one command in the block may have return values. Note: if the DMA interface is used, call format 1 may be used and this restriction does not apply.

The repertoire of callable routines in the library is given in 3.4, and the detailed call formats and command descriptions are given in 3.6 (for the INI command) and 3.10 (for all others).

Most of the entry points in the GGS library map onto commands in the SBD firmware (e.g. VEC, BLK etc.). There are a few commands which do not map directly to SBD functions. These are:

INI Initialise library (this sends a MES command to the SBD)

FIN Close library (no effect on SBD)

BBK Start command blocking (prevents subsequent commands from being sent to the SBD)

EBK Stop command blocking (either no effect on SBD, or sends commands contained in the block)

EGB Execute command block (this causes the commands in the block to be sent to the SBD).

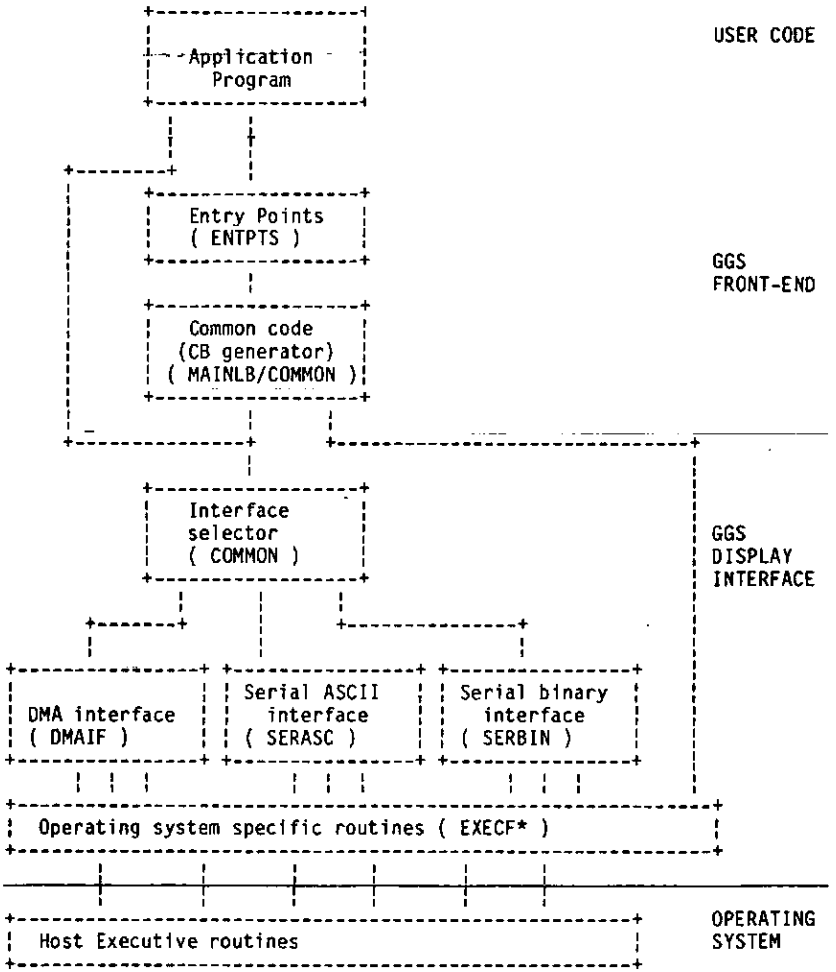


Figure 3.1 Block diagram of GGS library

* = X (RSX-11M), R (RT-11), V (VMS).

The names in brackets are source code modules.

3.2 Interfaces Supported

The SBD can be driven from GGS via two physical interfaces, DMA (Direct Memory Access) and serial (RS232). The serial interface may use one of two protocols, ASCII (7-bit) human-readable, and binary (8-bit) compact. A comparison of the two serial protocols is given in 7.1. All libraries except GGSF77 support all three interfacing methods; GGSF77 supports only the 7-bit ASCII serial interface.

The interface to be used is selected by the INI command (see 3.6) but can be predefined for a particular library configuration, without using INI. (You must do this if you wish to call OUTPUT only.) If INI is used (and FIN) it is possible to switch between interfaces at run time.

GGS does not support the Ethernet interface to the SBD.

3.3 Operating Modes

3.3.1 Synchronous/Asynchronous

Supervisor SBD has a 68000 microprocessor as local intelligence and the Asynchronous mode of operation is provided to allow the microprocessor to operate in parallel with the host. This feature is implemented by the host library.

By default the library operates synchronously, which means the library waits for a driver call to complete before returning control to the application program. In asynchronous mode, the library initiates a driver function and immediately returns control to the calling program. When the library next wishes to call the driver, it waits if the previous driver call has not completed (although the drivers themselves can, in fact, handle any number of queued I/O requests, subject to any restrictions imposed by the host operating system).

The advantage of synchronous mode is that any status values from the Supervisor SBD will be available when FORTRAN returns after the library call, and so any errors resulting from the call can be examined, and will be reported in their proper context. (Note that calls involving return values are always processed in synchronous mode regardless of the mode selected). The disadvantage is that Supervisor SBD and FORTRAN operations cannot take place in parallel. Of course, in a multi-tasking system other tasks can run while the library is waiting for driver completion.

Asynchronous mode therefore provides parallel processing for the FORTRAN and Supervisor SBD operations at the expense of having errors occuring asynchronously.

In practice the synchronous mode may be used during debugging so that errors can be easily interpreted. When program execution is error-free, asynchronous mode can be used to speed up execution.

The INI library call (see 3.6) is used to establish the desired mode before operation. The asynchronous mode is not supported by the serial protocols under RT-11.

3.3.2 Blocking

This function is implemented in the host library. It permits a sequence of commands (a Command Block) to be assembled in a user specified array. This array can then be executed with the EGB or OUTPUT commands (see 3.1), or stored in a file for later use. This provides a mechanism for generating fixed pictures or parts of pictures rapidly from a compact data base. In effect it provides an archiving system using host main memory or backing storage.

The library does not support virtual arrays. Use of these will normally lead to the program crashing at run time, or else an error such as (83) "No command block terminator found".

3.3.3 Archiving Option ("Arcaid")

The ARCAID package is an optional extra which resides on board the SBD. It is purchased separately and must be enabled by means of a password (see 2.3.3). It allows you to store, recall and examine lists of graphics commands within the SBD's own on-board memory. All memory not used for pixel or VDU display is usable for archiving, up to the maximum of 512K bytes. Another term for archiving is "local segment storage".

The amount of archiving memory freed for use by the Arcaid package depends on the current resolution and number of pixel and VDU pages. Table 3.1 gives the amounts in bytes and in 512-byte "blocks" (see DDI in 3.10). For the purpose of estimating useful space, a 4-argument VEC call occupies 10 bytes.

Naturally, executing graphics commands from archive is not only faster (since it does not involve host bus accesses) but imposes no load on the host processor. (In a system which is not heavily used the difference in execution time between DMA and local command execution is not great.) Since an archive can be recalled with a single command, it is useful for implementing pop-up menus (see 3.7.7), user-defined cursors (3.7.5) and for storing mimic diagram background information (for example in process control systems) particularly when the SBD is driven serially.

Archives can be nested (one archive can recall another). Also, the LIX command allows segments in an archive to be shifted to any starting location in the virtual coordinate space. In this way, quite complex pictures may be built up very quickly from a few simple components. Appendix B contains an example program which illustrates this. The limit of nesting is 10 levels of archive.

No of pixel pages	Resolution (Horizontal x Vertical) 768x 586	768x 574	704x 526	640x 480	512x 512	512x 384	480x 360	384x 293	256x 256
1	272384 (532)	276480 (540)			365568 (714)	404992 (791)		451072 (881)	474624 (927)
2	-	51200 (100)			230400 (450)	303616 (593)		392192 (766)	439808 (859)
3	-	-	-	-	95232 (186)	202240 (395)		333824 (652)	404992 (791)
4	-	-	-	-	-	100864 (197)		274944 (537)	370176 (723)
5	-	-	-	-	-	-		216576 (423)	335360 (655)
6	-	-	-	-	-	-	-	157696 (308)	300544 (587)
7	-	-	-	-	-	-	-	99328 (194)	265728 (519)
8	-	-	-	-	-	-	-	40448 (79)	230912 (451)
9	-	-	-	-	-	-	-	-	196096 (383)
10	-	-	-	-	-	-	-	-	161280 (315)
11	-	-	-	-	-	-	-	-	126464 (247)
12	-	-	-	-	-	-	-	-	91648 (179)
13	-	-	-	-	-	-	-	-	56838 (111)

Table 3.1 Archive space available at different resolutions/pages

Values without brackets in bytes, with brackets in 512-byte blocks.

Use of a second VDU page (see SET) reduces available space by the following amounts (bytes):

Resolution (Horizontal x Vertical)								
768x 586	768x 574	704x 526	640x 480	512x 512	512x 384	480x 360	384x 293	256x 256
17408 (34)	17408 (34)			16896 (33)	12800 (25)		10240 (20)	10240 (20)

3.3.4 Error Handling Modes

The INI command may be used to select any combination of the following methods of error reporting. The MES command can be used to enable or disable option 3 and to suppress or permit error reporting by the SBD to the host.

1 The application program terminates. Any available trace-back mechanism is activated, and a message indicates, by statement numbers, the path from the main program to the subroutine statement causing the error. If this option is disabled the program will continue. This traceback message is meaningful only when operating in synchronous mode.

2 A message is printed on the host terminal (e.g. LUN 5 in RSX-11M systems) giving details of the error. If this option is disabled no message will appear.

3 A message is sent to the VDU area of the SBD itself, which will cause the display to automatically switch to MIX if no scrolling lines are visible and the current number of MIX lines is greater than zero. If this option is disabled no message appears.

For errors detected by the host library, option 3 is not available, and any errors occurring in host operating system directives immediately take option 1 (and 2 if enabled).

Error Severities

Error conditions detected by the SBD firmware have varying degrees of severity, classed as: success, informational, warning, error, fatal (success is not an error condition and never generates a message). In some cases it might be advantageous to suppress all messages of a given severity. An example is when using the "screen window onto virtual coordinate space" technique (see 3.7.10), in which graphical primitives are defined in the coordinate range (-32768 to 32767) and all primitives which do not appear within the current screen "window" (defined by the LIX command) are simply clipped. Under normal circumstances, all clipped primitives generate informational message 13 ("Coordinate out of range"). If the informational severity has been suppressed, these messages are not generated, which means:

(a) the primitives take less time to execute

(b) the primitives can be archived and recalled from archive.

The INI command is used to select the severities to suppress (see 3.6). This communicates the information to the SBD via the MES command, which could also be used.

Error Handling in Archives

Errors are handled slightly differently when archiving. When in archive mode (after a call to BAR), any non-success status terminates the archive UNLESS the corresponding severity has been suppressed as described above. A subsequent call to EAR would then generate an error and be ignored. An exception to this is commands of severity F (Fatal) which invariably terminate the archive even if suppressed. However, even if you force a command into an archive in this way, this does not guarantee that the command will be executed (if error severity is E or F) either when archived or on recall

If an archive has been created with commands which are faulty (by suppressing the errors), then on recall (using RAR) the archived commands either generate the same errors or no errors depending on the error suppression in force at the time of recall. The first command encountered which actually generates an error terminates the recall; no further commands in the archive are executed.

Note that a command can still generate a non-success status even if you have prevented error reporting both to the VDU and to the host (options 2 and 3 above). The error occurs, even if you don't see it, because it has not been suppressed.

Any command requiring return values to the host is forbidden while archiving, since its recall would be pointless (the host could not obtain the returned values). The command is still executed despite the error, and the appropriate number of values is always returned to the host. Such commands produce a severity E (error) and so can be forced into the archive as described above. On recall, the command produces an error when executed. However, return values are NOT returned to the host, but written back into the archive in SBD memory.

3.4 Command Repertoire.

The following commands are available under the host library to drive the SBD:-

System Control:

* INI INItialise library, define current unit
FIN FINish with current unit
SET SET up display resolution and pages
DEF DEFine channel
SEL SELect channel
SOM Select Output Mode
LIX Load IndeX registers
PIX display PIXel data
VDU display VDU scrolling text
MIX display MIXed vdu/pixel data
PAG select visible PAGe
SAP Select Access Page
MES control error reporting and MESsages
CHI CHange serial ascii command Introducer
SCA Set Character Attributes
SSA Set Symbol Attributes
DLT Define Line Type
SLT Select Line Type
SYN SYNchronise to field flyback
MAT load MApping Tables (model B only)
FSH set FlaSH rate (model B only)

Peripheral Control:

CUR activate pixel CURsor
DMP DuMP pixel block to inkjet printer
SDM Set printer Dump colour Mapping
TIN digitizing Tablet INitialisation
TAB activate digitizing TABlet
TRA activate TRAckerball/mouse sample mode
TRO turn TRackerball/mouse sample mode Off

Graphics Primitives:

ALP draw ALPha (ascii) string
MAG draw MAGnified string
VEC draw VECtor
CIR draw CIRcle
BIT write single BIT (pixel)
TXT write vdu TeXT
ELI draw ELlIpse
BOX draw rectangular BOX outline
SYM define/draw SYMbol

Area Fills & Bit-Block Transfer

BLK draw solid rectangular BLocK
FLO FLOod random bounded area
CIF draw CIrcle Filled (solid circle)
ELF draw ELlipse Filled (solid ellipse)
* RIB Read Image Block to host
RUB RUB out (erase) block
* WIB Write Image Block from host
* ZOO pixel block ZOOm/reduce

Command Blocking:

* BBK Begin BlocKing
* EBK End BlocKing
* EGB Execute Graphics Block

Archiving (optional):

IDI Initialise archive DIrectory
DDI Display archive DIrectory
BAR Begin ARchiving
EAR End ARchiving
RAR Recall (execute) ARchive
DAR Delete ARchive
SAS Save context in Archive Space
RPC Restore Previous Context

Note: primitives marked with "*" indicate that some or all features of the command are not implemented by the FORTRAN-77 library (GGSF77).

3.5 Calling Subroutines From High-level Languages

For simplicity of presentation, the graphics commands described in Chapter 3.10 are defined in terms of FORTRAN calls to the GGS host library. The user may bypass the library and pass command blocks (as defined in Chapter 5) direct to the DMA driver (see Chapter 6), or else send commands serially to the SBD using the binary or ASCII protocols (see Chapter 7). Whichever method is used, the arguments are the same in value and effect.

A summary of graphics commands is given in Appendix C for quick reference.

3.5.1 Calling Conventions

Note that there are two calling conventions possible in FORTRAN for all the subroutines described :

1. Call as subroutine e.g. CALL SOM(0)

   This is the format used throughout this document.

2. Call as function e.g. STATUS = SOM(0)

where STATUS is of type INTEGER or BYTE (LOGICAL*1). The variable STATUS is loaded with a numeric value indicating the success or failure of the operation (1 = success, other values are defined in Appendix A). NOTE: all subroutines called in this way must be declared as type INTEGER or BYTE (LOGICAL*1), either explicitly or via the IMPLICIT statement.

Call format 2 can be used in combination with the error handling mode switches in the INI call (or the default ones specified during library configuration, if INI is not called) to give any desired mode of error handling. If commands are blocked (see 3.3.2) or the asynchronous mode is used (3.3.1), the only errors that can be returned in STATUS are those detectable by the host library, such as an invalid number of arguments.

For subroutines called with no arguments (e.g. PIX, FIN etc.), FORTRAN calls of format 2 must have parentheses, as in:

```
STATUS = PIX()
```

If this is not done FORTRAN does not realise that PIX is a function. Some versions of FORTRAN, e.g. DEC PDP-11 FORTRAN-77, behave differently from others in that they pass one null argument rather than none. If this is the case you should configure the host library (see 2.4) to strip nulls from the argument list.

A list of error codes and their meanings is given in Appendix A. Errors associated with particular commands are also given in the sections of Chapter 3.10 dealing with the commands.

3.5.2 Optional Arguments

Arguments which are optional (i.e. can be defaulted) are shown enclosed in square brackets in 3.10. If compulsory arguments are omitted, the program will probably crash with an illegal or odd address error.

e.g. CALL INI (,2,,)

under RSX-11M, will cause commands to be sent to unit SB2: (DMA interface) or TT2: (serial interface) - all other arguments are taken to be the defaults defined during library configuration (see 2.4). Note that where an optional argument is omitted, the separating commas must remain, otherwise an incorrect number of arguments will be passed to the subroutine. See the comments above on the passing of a single null argument.

It is important to note the difference between defaulting and omitting arguments. Where more than one version of a subroutine call is given in 3.10, the library (and the SBD) depend on the number of arguments supplied to determine the action taken. For instance:

```
CALL VEC(X2,Y2)
```

succeeds (this is a valid form of VEC with two arguments), but:

```
CALL VEC(,,X2,Y2)
```

will probably crash because four arguments are supplied but two compulsory ones have been defaulted. You can, however, configure the assembler-coded host library (see 2.4) so that it can strip off null arguments (if the high-level language you are using cannot call a subroutine with different numbers of arguments).

If you are using a high-level language (such as most implementations of PASCAL) which not only requires that a subroutine (procedure) be called with the same number of arguments every time, but also is unable to pass null arguments, you must resort to the approach taken by the FORTRAN-coded serial line library GGSF77, which is to give each version of the call a different name by appending the number of arguments present, e.g.:

```
CALL VEC2(X2,Y2)
CALL VEC4(X1,Y1,X2,Y2)
```

NOTE: as SET is a reserved word in PASCAL, call the SET routine using the name SETP (or SET1/SET2 if using the scheme above).

3.5.3 Argument Data Types

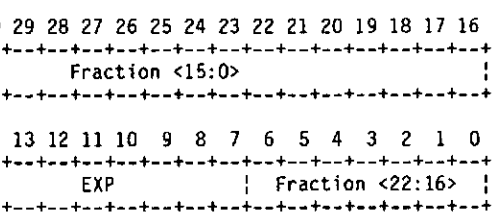
The following description applies only to the assembler-coded libraries for RSX, RT-11 and VMS.

Under RSX-11M and RT-11, all arguments, including word arrays but not strings, are of type INTEGER*2 unless otherwise stated. (A notable exception is TIN). However, under VMS:

- single integer arguments should be longwords, but only the least significant 16 bits of each argument are read.
- any return value arguments must be INTEGER*4 - the top 16 bits are set to zero.
- word arrays used for WIB, RIB, BBK and EGB must be INTEGER*2. (The library could use an INTEGER*4 array, but its size must be specified in words).
- word arrays used for SDM, MAT and SYM must be INTEGER*4.

All values and ranges are given in decimal, unless preceded by a double quote (e.g "377) which implies octal radix in DEC FORTRAN. All arguments must be passed by reference, that is the argument list contains only pointers to argument values and not the arguments themselves (see below). FORTRAN does this automatically, even where constants are specified in the argument list, but other languages, particularly PASCAL, do not.

For the TIN command, all six arguments are REAL*4 (DEC single-precision floating-point format). Considered as a doubleword (32 bits), bits <0:22> constitute the fractional part, f, normalized in the range (1/2 <= f < 1), so that its most significant bit is always 1 - this bit is assumed ("hidden") and does not appear (it would be bit 23). Bits <23:30> are the exponent, e, in excess 128 notation, meaning that exponents from -128 to 127 are represented by values of 0 to 255 in this bit field. Bit 31 is the sign bit (1 if negative). The value 0.0 is represented by all bits clear.



Floating Point numbers can be considered as (2**E)*F where E and F are the actual exponent and fraction respectively (rather than their representations here, e and f). For instance, the number 1.0 is (2**1)*(1/2) and so would be represented as zero in the fraction and 129(10) in the exponent - thus the first word would be 40200(8) (i.e. 16512(10)), and the second zero.

Note: using the ASCII serial protocol (see 7.2) the arguments to TIN are sent as two decimal numbers each (so TIN(1.0, ...) would be sent as "TIN 16512 0 ... etc. They can also be sent as octal numbers by preceding them with a double quote(").

3.5.4 Use of strings

Where a string argument is required, FORTRAN allows the use of quoted strings, such as:

```
CALL ALP('This is a string')
```

If a single quote is required in the string it must be doubled, as in:

```
CALL MAG(100,100,2,'What''s that?')
```

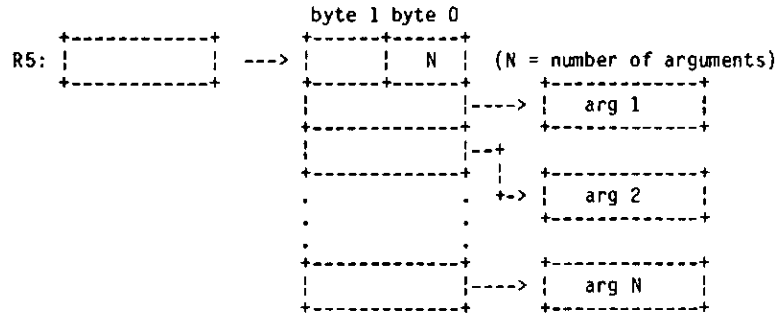
PDP-11 FORTRAN passes quoted strings as null-terminated ASCII. However, VMS FORTRAN passes them as string descriptors. Under VMS, variable string arrays should be passed in the form of byte arrays (e.g. LOGICAL*1) with the length specified using the optional NC argument (i.e. a Hollerith). CHAR arrays should NOT be used as these are passed using a VAX array descriptor which the GGS library does not recognise. Note that if a string literal is passed between FORTRAN subroutines before being passed to the GGS library, it will be passed as a byte array so that a character count (NC argument) must also be specified. A byte array, or any other data type containing an ASCII character value in each byte, terminated by a null (zero) byte, can therefore only be used with PDP-11 FORTRAN unless the Hollerith method is used.

If a Hollerith string is specified, the byte array must be preceded by an argument (NC) containing a character count (not including the trailing null if present). You cannot use this with a quoted string in VMS FORTRAN.

The maximum number of characters which can be passed by the string commands (ALP, MAG, SYM and TXT) is configurable, subject to an absolute maximum of 132. Strings larger than the maximum are truncated.

3.5.5 Argument list structure

All assembler-coded versions of the host library adhere to the standard FORTRAN calling convention for their host machine. For PDP-11 hosts, the convention is:



The address list is contiguous, but the actual arguments may be scattered about in memory. The library's function is to gather all the scattered arguments into a contiguous buffer (either a user-supplied array or an internal single-command buffer) to pass to the SBD.

NOTE: On PDP's the words in the address list are 16-bit, on VAX they are 32-bit. VMS passes the (32-bit) address of the argument list in AP, not R5.

3.6 Operating System-Specific Initialisation (INI)

The INI command is one of the routines provided by the GGS host library. It specifies the operating modes for the library and identifies the physical unit with which the library is to communicate.

The INI command is described here, rather than in 3.10 with the rest of the host library commands, because:

- o It varies slightly in format and meaning between different operating systems
- o It is critical to the correct operation of applications programs.

3.6.1 The INI command under RSX-11M/M-PLUS and Micro-RSX

Call Format

1. CALL INI ([L],[U],[E],[M])

Arguments

L = Logical Unit Number (LUN) for the library to use (1-8). Note that one LUN (usually LUN 5) is used for error reporting and is therefore not allowed. You can configure which LUN is reserved (see below).

U = Supervisor physical unit number (0-15). Note that serially driven devices have a different device type (TT:) from DMA driven ones (SB:) so the actual unit identified by this number depends on the interface selected by the fourth argument. NOTE: this argument can also take the value -1 (implying that no LUN to physical unit association will take place at run time - see 3.8), or it can be a filename (see bit 6 of the M argument below).

E = Local Event Flag number for the library to use for communication with the slave (1-64). In addition, one local event flag is reserved for the library's internal use : you can configure which one (see below).

M = Mode of operation (14-bit pattern - see below).

All parameters may be defaulted. The defaults are specified by editing the library source module CONFIG.MAC before building the library, or by using the configuration command file GGSBLD supplied with the distribution kit (see 2.4).

Mode Argument

The parameter M is a bit pattern defining the mode of operation of the library. The bits have the following meaning (the value of the bit when set, in decimal, is given in brackets) :-

Bit 0. (1) If zero, errors are not returned from the SBD to the host. If 1, errors are returned.

Bit 1. (2) If zero, errors are displayed in the scrolling text store of the SBD. If 1, errors are not displayed locally.

Note: bits 0 and 1 are sent to the SBD as bits <0:1> of a MES command, when INI is called (see also M bits 10:13)

Bit 2. (4) If zero, the host program continues on detection of a non-fatal error. If 1, execution terminates. Fatal errors (e.g. in executive calls) always cause termination.

Bit 3. (8) If zero, a message is printed on the assigned device when an error occurs. If 1, no message is printed.

Bit 4. (16) If zero, the serial line is used to send commands to the SBD. If 1, the DMA interface (if available) is used. If the interface selected is not available, an error is generated (subject to the options selected by bits 2 and 3 of the argument).

Bit 5. (32) (Ignored if bit 4 is 1). If zero, the compressed binary format (see 7.1) is used on the serial line. If 1, the human-readable ASCII format (see 7.2) is used.

Bit 6. (64) (Ignored if bit 4 is 1). If zero, the serial protocol selected by bit 5 is sent down the serial line to the SBD. If 1, then the U argument (see above) is interpreted as a filename (null-terminated character array) and all commands are written to the specified file in the selected format. The files thus created can be sent directly to the SBD via the usual utilities (e.g. PIP) to generate pictures.

Bit 7. (128) If zero, the synchronous mode is used by the library (see 3.3.1). If 1, the asynchronous mode is used, which may cause a meaningless traceback message to be generated if bit 2 is non-zero and an error occurs. Note that only one I/O may be outstanding at any one time even in asynchronous mode. If a command requires numeric values returned from the SBD (not status values) the asynchronous mode, if selected, is overridden and the library waits for the response from the SBD. This ensures that the library is able to write the values to the correct destinations in the host program.

Bit 8. (256) If zero, blocking is enabled (see 3.3.2). If 1, blocking is disabled; and the BBK, EBK and EGB commands are ignored; all commands are thus sent as they are called, one by one (i.e. in immediate mode). This feature is a debugging aid in that it provides immediate detection of all errors in a command block (normally only the first error in a block is reported to the host). Once the block is error free only one minor code change is needed to enable blocking mode. Disabling blocking mode means that commands encountered which would be illegal in blocking mode do not generate an error to say so. NOTE: enabling blocking mode does not put the library in blocking mode - you must call BBK to do this.

Bit 9. (512) If 1, the library 'attaches' the physical unit specified by the U argument for its exclusive use, locking out other programs (and the host Executive) from access to the device. If zero, the device is not attached. This generally means:

(a) that the device can be used as a VDU and as a serially-driven graphics display at the same time

(b) multiple programs can access the display (with care!)

(c) an unsolicited input AST may be set up by the user's task to intercept keyboard input. See RSX Executive Reference and I/O User's Guide.

Bits 10-13 determine the SBD's response to errors in addition to bits 0 and 1. For each of the four bits, if it is set, the corresponding severity of error is suppressed by the slave and is not displayed on the VDU area or returned to the host. Thus if all bits are zero, all messages are displayed / returned. An error which is suppressed in this way does not cause a command to be rejected when archiving (unless fatal).

The correspondence is:

Bit no.	Severity
10 (1024)	I - Informational
11 (2048)	W - Warning
12 (4096)	E - Error
13 (8192)	F - Fatal (note: these errors are always displayed on the VDU area, and cannot be archived).

These bits are sent to the slave as bits <2:5> of the MES argument.

Examples

```
CALL INI (1,0,2,3089)          ! bits 0,4,10 and 11 set
```

This associates the library with DMA unit SB0: using LUN 1. Local event flag 2 is used to synchronise output. Status values will be returned to the library, but errors of severity 'informational' or 'warning' will be suppressed. This call would send a command equivalent to CALL MES (13) to the SBD.

```
CALL INI (,'SY:[100,1]OUT.DAT',,"140)  ! bits 5 & 6 set
```

This tells the library to write commands in the ASCII protocol into the file OUT.DAT in directory [100,1] on the default device. The first command in the file would be "^MES 0". The default LUN and local event flag set up when the library was configured (or defined in an earlier INI command) will be used.

3.6.2 The INI command under RT-11

Under RT-11, the INI command is broadly similar to that for RSX-11M, with the following exceptions:

1. The LUN (Argument 1) and event flag (Argument 3) are ignored.

2. The Unit number (Argument 2) is not used in the case of a single-terminal serial system, a TSX-PLUS system, or a multi-terminal serial system in which the SBD terminal is not attached (see exception 3, bit 9 below).

   Where communication with an SBD is by DMA, the physical unit number must be specified as for RSX (0 for SB0:, 1 for SB1: etc.) A value of -1 is not allowed.

   Where communication with an SBD is serial, in a multiterminal system with the SBD as a private (attached) device, the physical unit number must be specified as the LUN (e.g. console = LUN 0, terminal 1 = LUN 1, etc.). Do not confuse this LUN with the RSX FORTRAN LUN which is the first argument, and is not used in RT-11 systems. To use a serial SBD as a private terminal, bit 9 of the Mode argument must be set to attach the device. If this isn't done, the console will be used.

3. The following restrictions apply to the Mode (Argument 4):

   Bit 5 - 1 = ASCII protocol, 0 = binary (if bit 4 = 0). The binary serial protocol is only supported in the Multiterminal private device situation (i.e. bit 9 must also be set and the system must have been SYSGENed for Multiterminal support).

   Bit 6 - set to 1 to write commands to a file. Either binary or ASCII protocols can be used, irrespective of the system configuration. (Note: this option is scheduled for future release but is not available under the current release.)

   Bit 7 - 1 = asynchronous mode, 0 = synchronous. Asynchronous mode is not supported under any serial line protocol.

   Bit 9 - 1=attach, 0=don't attach. This bit distinguishes between the two possible Multiterminal serial situations (private, which supports the binary protocol, and non-private, which does not - see above). If an SBD with a keyboard is used as a private device (bit 9 = 1) the keyboard is disabled. In the other Multiterminal situation, where a program is run with the command "FRUN/TERMINAL:n prog", the SBD terminal which is LUN n becomes the "console" for the program and any terminal I/O outside host library calls will go to that SBD.

3.6.3 The INI command under VAX/VMS and MicroVMS

The INI command used in VMS systems is broadly similar to that used in RSX-11M systems. However, the method of specifying the device to be communicated with is different. The general format of the call is:

```
CALL INI ([string_length],[device_mnemonic_string],[E],[M])
```

If string_length is undefined or zero then device_mnemonic_string must be passed by string descriptor (i.e. string literal in FORTRAN). Otherwise string_length is the length of device_mnemonic_string, which is passed as the address of the string (i.e. a byte or LOGICAL*1 array in FORTRAN). NOTE: do NOT use a CHAR array or datatype as this uses a type of descriptor not recognised by the library. This method is used because VMS device names include the controller letter as well as the physical unit number.

If device_mnemonic_string is undefined then the configurable default name is used. In the preconfigured library it is 'SBA0'.

In general the use of logical names (e.g. 'ggs_dma' or 'ggs_serial') provides the simplest solution.

Naturally, the device type specified must correspond to the interface selected by the M argument (e.g. an SBxnn: device if using DMA).

The E and M arguments are as for RSX-11M. Writing of serial data to a file (M bit 6) is not supported by the current release.

Example

```
CALL INI (,'ggs_serial',,0)
```

This uses the binary protocol to communicate with the serial SBD device identified by the logical name "ggs_serial". This must be defined before the program is run, e.g.:

```
$ DEFINE ggs_serial TXB2
```

3.6.4 The INI command under UNIX

The UNIX host software will be available in late 1985, at which time this section will be issued.

3.6.5 The INI command for GGSF77

The FORTRAN-77 code host library GGSF77 uses an INI command similar to that for VMS:

```
CALL INI4 (LEN,'DEVICE',,MODE)
```

The LEN argument is compulsory and must be the number of characters in the string DEVICE. DEVICE may be a physical or logical name depending on the features of the host operating system. (GGSF77 tries to OPEN the specified name). Other restrictions apply to the MODE argument due to the limited facilities of GGSF77:

Bit 4 - ignored since only the serial interface is supported.

Bit 5 - must be 1 to select the ASCII interface, since the binary protocol is not supported.

Bit 6 - ignored. If DEVICE refers to a file, commands will be written to that file.

Bit 7 - ignored. GGSF77 operates only in synchronous mode.

Bit 8 - ignored. GGSF77 does not support blocking.

Bit 9 - ignored. GGSF77 does not explicitly attach the device for exclusive use. Some implementations of FORTRAN may do this.

3.7 Concepts and Programming Techniques

3.7.1 Bit Planes

The SBD's display memory (dynamic RAM) is made up of 4 memory planes - each plane, at the highest resolution, consisting of 768 x 574 bits. These planes are grouped together (conceptually, one on top of another) to give four bits of information for each pixel, allowing a maximum of 2 raised to the 4th power of colours - that is, 16. Another way of saying this is that the SBD has a "depth", or Z axis, of four bits.

For most applications all the planes are combined and the picture is drawn using all 16 colours (ie. the combination of the four bits). At times, however, it is useful to separate the planes and draw two pictures, one being made up of three planes (8 colours) and the other being a one bit overlay. These pictures can be drawn and changed independently but they are displayed simultaneously.

Example

```
        ...
        Draw picture using planes 0, 1 and 2 (e.g. channel 1)
        ...
        CALL SEL(5,1,0)            ; Select plane 3 only (channel 5)
        CALL BLK(100,100,200,200) ; Write a block in plane 3
        ...
        CALL SEL(1,1,0)            ; Select a colour in planes 0-2
        ...                        ; for further drawing
```

Suppose that the pixels before BLK(100,100,200,200) had the following values (hexadecimal notation):

```
+---> X    100
|           v
|     112222 444462225553333
V                          -< 100 >-
      112222 244622225533333
Y     111222 246222225333333
      111122 462222225333333
```

After BLK(100,100,200,200), they would have the following values (in hex):

```
          100
           v
      112222 444462225553333
                           -< 100 >-
      112222 ACCEAAAADDBBBBB    (A=10,  B=11 etc).  A value of 8  has
      111222 ACEAAAAADBBBBBB    been added to all pixel values within
      111122 CEAAAAAADBBBBBB    the block.
```

3.7.2 Channels

In order to write to individual memory planes as described in the previous section it is necessary to select them. The planes which are selected for writing are collectively known as a channel.

On power up there are eight predefined channels. These are:

Channel number	Planes selected
0	0, 1, 2, 3
1	0, 1, 2
2	0
3	1
4	2
5	3
6	0, 1
7	2, 3

The convention adopted is that plane 0 is the least significant bit of the pixel value (i.e. bit 0, value 1 when set) and plane 3 is the most significant bit (i.e. bit 3, value 8 when set).

Further channels may be defined with a DEF command. (Alternatively the preset channels may be re-defined.) The only allowable channels are the pre-defined ones and a DEF command has the effect of re-naming them. DEF cannot define a channel which has combinations of planes other than the above, such as (1,2) or (1,2,3).

The SEL command selects channels. It is possible to select independent channels for writing the selected foreground and background colour values. Unless explicitly specified, the background channel is the same as the foreground channel.

A background channel is most useful in a system with lookup tables, as in Model B, since it means that the background data can be written in different planes from the foreground, and remapped at will (e.g. made invisible by mapping to black) without changing the foreground mapping.

A clear distinction should be made between the channel, which determines what values can be written to the pixel planes, and the colour mapping (see next section) which determines how the four planes are represented on the screen.

All primitives use the current channel for writing and reading pixels, with the exception of WIB, RIB, DMP and ZOO. These four always affect all four pixel planes irrespective of the currently selected channel.

3.7.3 Colour Mapping

The SBD has colour mapping tables to translate the pixel values in memory into displayed colours on the screen. As a result a given pixel may be displayed as any one of several different colours without changing its value. In order to make the examples easy to follow, we will assume that the colour mapping tables have been set by a SOM 3 command unless we state otherwise.

3.7.3.1 Colour Mapping - SBD A

The SBD A tables may be loaded with four series of preset values. These are selected by setting the output mode (SOM) function. The values are given below for both the colour version (Table 3.2) and the monochrome version (Table 3.3) of SBD(A).

There are 16 grey levels ranging from grey 0, which is black, to grey 15, or white. The grey levels of SOM 2 and, on the monochrome board, SOM 0 and SOM 1 use only the green gun on the monitor. They appear, correctly, as grey levels on a monochrome monitor but as various shades of green on a colour monitor. The two greys (and black and white) on SOM 3 are true greys and as such are correctly reproduced on a colour monitor.

In the tables, * denotes a flashing colour (flash to black, with duty cycle and mark/space ratio determined by a hardware link - see 2.2.1).

Pixel value	SOM 0	SOM 1	SOM 2	SOM 3
0	black	black	black	black
1	red	red	grey 1	red
2	green	green	grey 2	green
3	yellow	yellow	grey 3	yellow
4	blue	blue	grey 4	blue
5	magenta	magenta	grey 5	magenta
6	cyan	cyan	grey 6	cyan
7	white	white	grey 7	white
8	white	black	grey 8	dark grey
9	white	*red	grey 9	dark green
10	white	*green	grey 10	orange
11	white	*yellow	grey 11	light blue
12	white	*blue	grey 12	dark blue
13	white	*magenta	grey 13	dark brown
14	white	*cyan	grey 14	light grey
15	white	*white	white	purple

Table 3.2 Colour mapping for SBD(A) colour version

Pixel value	SOM 0	SOM 1	SOM 2	SOM 3
0	black	black	black	black
1	grey 3	grey 3	grey 1	red
2	grey 5	grey 5	grey 2	green
3	grey 7	grey 7	grey 3	yellow
4	grey 9	grey 9	grey 4	blue
5	grey 11	grey 11	grey 5	magenta
6	grey 13	grey 13	grey 6	cyan
7	white	white	grey 7	white
8	white	black	grey 8	dark grey
9	white	*grey 3	grey 9	dark green
10	white	*grey 5	grey 10	orange
11	white	*grey 7	grey 11	light blue
12	white	*grey 9	grey 12	dark blue
13	white	*grey 11	grey 13	dark brown
14	white	*grey 13	grey 14	light grey
15	white	*white	white	purple

Table 3.3 Colour mapping for SBD(A) monochrome version

3.7.3.2 Colour Mapping - SBD B

The SBD B has a set of variably loadable look-up tables. The colour corresponding to pixel value 0 must be black but the the other 15 values may be matched with any colour from a palette of 4096.

In order to provide compatibility with SBD A and also to make SBD B easy to use, the SOM instructions have been implemented. These load the look up tables with one of the sets of predefined values which emulate the colour output PAL of SBD A. The SOMs 2 and 3 are the same in both the monochrome SBD A and colour SBD A. On the SBD B SOM 0 and SOM 1 correspond to the colour SBD A and a new pair of instructions (SOM 4 and SOM 5) give the equivalents to the monochrome SBD A SOM 0 and SOM 1. On SBD B all grey levels are displayed as grey on both monochrome and colour monitors. (SBD(A) grey levels appear on the green output only.)

A more flexible method of setting the colours is provided by the MAT command. In this command each colour is described by an integer which is made up from the red, green and blue components. Each component is in the range 0 to 15 and the colour number is made by calculating:

$$256*B + 16*G + R$$

where R is the red component
G is the green component
and B is the blue component

The syntax of the MAT command is given in 3.10.

The components and colour number which make up the SOM 3 colours are given in Table 3.4 below.

Flashing colours are achieved by dynamically changing the look-up tables. There are two tables, numbered 0 and 1, and the SBD firmware automatically switches between them at regular intervals. The FSH command sets the switch intervals - see 3.10. This flash rate is independent of the watchdog timer (fixed) flash rate.

Colour	Red	Green	Blue	Combined
0 Black	0	0	0	0
1 Red	15	0	0	3840
2 Green	0	15	0	240
3 Yellow	15	15	0	4080
4 Blue	0	0	15	15
5 Magenta	15	0	15	3855
6 Cyan	0	15	15	255
7 White	15	15	15	4095
8 Grey 5	5	5	5	1365
9 D. green	0	5	0	80
10 Orange	10	5	0	2640
11 Slate	0	5	10	90
12 D. blue	0	0	5	5
13 Brown	5	0	0	1280
14 Grey 10	10	10	10	2730
15 Purple	5	0	5	1285

Table 3.4 Colour Values for SOM 3 on Model B

Note: all values are decimal.

3.7.4 Colour Selection

This section describes the way in which colours are selected on the SBD and the facilities arising from this. Sub-section headings relate to:

- foreground and background colours
- complement colour
- additive mode
- line types.

3.7.4.1 Foreground and Background Colours

On the SBD, colours are selected using the SEL command. This command names two colours; a foreground colour, and a background colour. In general, the foreground colour is used to draw primitives which require only one colour.

Example:

```
CALL SEL(0,3,4)
CALL BIT(100,100)
```

This draws one dot in colour value three.

An exception to this is RUB which draws in background colour so that:

```
CALL SEL(0,3,4)
CALL RUB
```

clears the screen to colour value four.

Some primitives require two colours. An example of this is ALP which draws the text string in foreground colour in a cell which is drawn in the background colour. MAG and SYM also put their patterns in foreground on a a cell of background colour.

FLO uses the colours in an entirely different way. FLO fills an area, which is completely bounded by background colour, with foreground colour. The way to remember this is by F for foreground, F for filler; B for background, B for border.

Example:

Take a triangle made from three lines in colour 4 (blue) which encloses the pixel [100,100].

After the commands:

```
CALL SEL(0,3,4)
CALL FLO(100,100)
```

the triangle will be filled with colour 3 (yellow).

It should be noted, in passing, that the FLO routine uses the foreground colour to determine which areas have already been filled. Accordingly, it is assumed that the area to be filled contains no foreground colour. If any enclosed pixels are already in foreground colour then it is possible that not all the area will be filled. The area may contain 'islands' bounded by background colour - these will not be filled over.

3.7.4.2 Complement Colour

This is a special foreground colour. Rather than the various pixels being set to a given colour, they are set to the inverse (one's complement) of what they already contain.

For example, if a pixel has value 0, it is set to 15 when written in complement colour using channel 0; if it contains 5 then it is set to 10 and so on. The channel determines which bits are complemented, so that if channel 1 is selected (planes 0,1 and 2) then colour 0 goes to 7 and 1 goes to 6. Table 3.5 gives the effect of complementing with the various channels available.

There are two useful properties of complement colour. One of these is visibility. With SBD A mapping and channel 1 selected the complement colour is always easily distinguished from the original colour. Similarly most useful colour mappings on SBD B will have a visible complement in some channel. This allows the possibility of putting an overlay onto a picture which stands out no matter what the underlying pattern is.

The other property of complement colour is that the information already on the screen has not been lost. This has two effects. One, the user can still see the the original pattern through the overlay and two, if the overlay is drawn a second time it disappears leaving the original picture unchanged.

Channel	0	1	2	3	4	5	6	7
Planes	0,1,2,3	0,1,2	0	1	2	3	0,1	2,3
Source colour	Final colours							
0	15	7	1	2	4	8	3	12
1	14	6	0	3	5	9	2	13
2	13	5	3	0	6	10	1	14
3	12	4	2	1	7	11	0	15
4	11	3	5	6	0	12	7	8
5	10	2	4	7	1	13	6	9
6	9	1	7	4	2	14	5	10
7	8	0	6	5	3	15	4	11
8	7	15	9	10	12	0	11	4
9	6	14	8	11	13	1	10	5
10	5	13	11	8	14	2	9	6
11	4	12	10	9	15	3	8	7
12	3	11	13	14	8	4	15	0
13	2	10	12	15	9	5	14	1
14	1	9	15	12	10	6	13	2
15	0	8	14	13	11	7	12	3

Table 3.5 Effect of complement colour in different channels

These features are used by the SBD firmware for its cursor. The hardware cursor is either a box or a cross, drawn in complement in the current channel. In order to move the cursor, the SBD first deletes it by redrawing it in the current position and then drawing it in a new position. On the SBD A it is recommended that channel 0 is used for output modes 0 and 1, channel 5 for output mode 2, and channel 1 for output mode 3. On SBD B the correct channel to use depends on how the mapping tables are set up, but if the mapping tables are based on the default settings then it is probable that channel 1 is the best one to use.

In order to select a complement colour for application purposes, a negative foreground colour is selected. Note that the firmware cursor is always in complement colour on the current channel and that it neither alters nor depends upon the current selected colour.

It should be noted that only the foreground colour may be set as complement. Any attempt to set the background as complement will result in additive mode being selected. This means that complement colour cannot be used by RUB.

Finally, it is impossible to use complement colour in any sensible way with the FLO primitive (q.v.).

3.7.4.3 Additive Mode

Under normal circumstances there are two colours defined on the SBD, namely the foreground and background colours. This is called "replacement mode" because characters drawn with ALP, MAG and SYM put down a rectangular cell in background colour, so replacing what was previously on the screen at that point.

If a negative background colour is selected then the null colour is used - i.e. the colour which is the same as the existing colour. This is "additive mode", so called because ALP, MAG and SYM add their patterns to the current picture without first putting down a cell.

When additive mode is selected the border colour for FLO becomes anything other than black, i.e. "not-black". This can be shown by an example. Imagine a triangle made up of a red line, a green line and a blue line. Now fill it. Since there is no one colour forming the border the normal FLO cannot be used. If, however, additive mode is selected then FLO will fill the black interior and will take any other colour as the border.

This idea can be extended to include all the negative border colours - ie. not-red, not-green etc., effectively giving the ability to flood over a particular colour without affecting other colours. This is achieved by adding 100 to the background colour argument in SEL. This does not select additive mode - the background colour is assumed to be (100-n) by all primitives except FLO.

For example: Select a not-black border

```
CALL SEL(0,2,-1)      ! also sets additive mode
     or
CALL SEL(0,2,100)     ! border is "not colour 0 (black)"
```

When a not-border colour is selected then, for all purposes other than FLO, the background colour is set to 100 minus the specified value.

For example:

```
CALL SEL(0,3,104)
```

This will fill with colour 3 to a border colour of "not 4", but will draw colour 3 letters on a colour 4 background with ALP.

The RUB command does nothing when in additive mode, but still takes the normal amount of time to do it, because it is writing pixels to the null colour.

3.7.4.4 Line Types

The use of line types permits you to draw lines and solids which are partially foreground and partially background colour. A pattern is defined which repeats every 16 pixels in the direction of the line being drawn. (For details of the interaction of line types with the area drawing primitives, refer to the relevant command descriptions.) The pattern is defined (using DLT) as a 16-bit binary value, each bit corresponding to a pixel, with 1 representing foreground colour and zero representing background colour. Eight of the most useful patterns (including solid foreground) are predefined and can be selected by means of the SLT command.

Among the uses of line types are:

- distinguishing between lines when not enough colours are available

- defining pseudo-colours. With on-off alternating patterns, i.e. SLT(1), the individual pixels are too small for the human eye to resolve, and the foreground and background colours merge to become a new "pseudo-colour". Because you can use line types with FLO, this technique gives you a larger range of possible colours for area fill also. Any other line type used with FLO would not produce sensible pictures.

In Additive Mode, the pixels which would have become background colour are left as they are. This allows you to perform a "semi-overlay" in which the image overwritten is not completely obscured.

Using DLT(0) allows you to write your primitives in background colour should you wish to do this.

3.7.5 Controlling Your Own Software Cursor

In some applications it may be desirable to use a cursor of your own design to indicate a display screen position. This section contains an example of one method of controlling a user defined cursor using a sampled input device. A sampled input device may be either a tablet (using the TAB command) or a trackerball or mouse (using the TRA command).

In this example the system operator must move the software cursor about the screen by using the sampled input device. The subroutine CURPOS is assumed to return the current input device coordinates (corresponding to valid screen coordinates). To optimise the cursor generation, the commands used to draw the cursor shape should be stored in a local SBD archive. The FORTRAN code to do this would be:

```
    CALL BAR(CURSOR)              ! Line 1
    CALL VEC(-5,0,5,0)            ! Line 2
    CALL VEC(0,-5,0,5)            ! Line 3
    CALL EAR                      ! Line 4
```

Line 1 begins the creation of the archive identified by the integer number CURSOR.

Lines 2..3 define a cross cursor with arms eleven pixels in length. The archive's "origin" is at the point of crossing of the arms (0,0). These lines could be replaced with other primitive calls to generate a cursor with a different shape (an arrow, for instance).

Line 4 ends archive CURSOR.

It should be noted that the cursor will be drawn on the current access page with the current attributes (colour, LIX offsets etc.) while the archive is being created. The next piece of FORTRAN code will draw the cursor and enter a continuous loop that ensures it is always drawn at the screen position corresponding to the current input device position.

```
    CALL SEL(0,-1,0)              ! Line 5
    OLDX = 0                      ! Line 6
    OLDY = 0                      ! Line 7
    CALL LIX(OLDX,OLDY)           ! Line 8
    CALL RAR(CURSOR)              ! Line 9
10  CALL CURPOS(X,Y)              ! Line 10
    IF ((X.EQ.OLDX).AND.
   +    (Y.EQ.OLDY)) GOTO 10      ! Line 11
    CALL RAR(CURSOR)              ! Line 12
    CALL LIX(X,Y)                 ! Line 13
    CALL RAR(CURSOR)              ! Line 14
    OLDX = X                      ! Line 15
    OLDY = Y                      ! Line 16
    GOTO 10                       ! Line 17
```

Line 5 selects complement colour. Anything drawn after this SEL call will complement the pixel value for pLanes 0..3.

Lines 6..7 initialise the cursor position.

Lines 8..9 will draw the cursor at position (OLDX,OLDY).

Line 10 will put the current input device/screen position in X and Y.

Line 11 ensures that the cursor is only erased and redrawn if the current position returned by CURPOS differs from that returned by the previous call to CURPOS.

Line 12 erases the cursor currently drawn on the screen by complementing the pixels back to their original values. Note that if a graphics primitive had been drawn through the cursor since it was last drawn, the cursor would not disappear entirely - a "shadow" would be left.

Lines 13..14 redraw the cursor at its new position (X,Y).

Lines 15..16 set the previous position (OLDX, OLDY).

The only addition necessary to this code is a means of terminating the loop, e.g. when the user presses a button on the keyboard or tablet puck. This could be done in a variety of ways depending on the operating system in use.

The user should make sure that the Access Page, where the cursor is drawn, and the Display Page, which is visible on the screen, are the same when using this technique. The SBD's own local cursor is always drawn on the Display Page for CUR and TAB, but the Access Page for TRA.

Complement colour is used to avoid assigning the overlay plane to cursor only, therefore the full sixteen colours are still available for use. If archiving is not enabled, command blocking in host memory (commands BBK, EBK, EGB etc.) could be used instead. However, this isn't quite as efficient as the method described above, because of the increased host/SBD communication.

3.7.6 Multiple Pages and Hidden Update

The SBD's dynamic RAM is made up of at least one pixel page (consisting of four memory planes), plus one or two VDU pages for scrolling text (each VDU page has only one memory plane), and the space remaining may be used for archive storage space. More pixel pages may be defined at the expense of archive storage space. The maximum number of pixel pages allowed is dependent on the current screen resolution - the lower the resolution, the more pixel pages allowed. It is left to the user to make a trade off between the amount of archiving space needed by the application and the number of pages required. The dynamic RAM configuration is done on power-up, and by the SET command which defines the resolution and the number of pixel pages to be used (see the description in 3.10 for details). Table 3.1 (section 3.3.3) gives the correspondence between pixel pages and archive space.

When more than one pixel page is in use, the SBD uses a concept of "current Display Page" and "current Access Page". Pages are numbered from 0 to "n-1" for "n" pages. The PAG command selects the current Display Page which is used to provide video output to the SBD's monitor screen. The SAP command selects the page (not necessarily the Display Page) which will be used for read/write operations by the SBD's display primitives. For example, a call to VEC will cause a line to be drawn in the current Access Page. The SET command always sets both the Access and Display Pages to be page 0.

There are many uses for multiple pages, but as a simple example consider the problem of moving a solid circle across the display screen. This could be done by the following FORTRAN code.

```
    X = 0                          ! Initialise X
    CALL SEL (0,1,0)               ! Select red
10  CALL RUB                       ! Clear to black
    CALL CIF(X,100,25)             ! Draw solid disc
    X = X+1                        ! Increment X position
    IF  (X.LE.XMAX) GO TO 10       ! Repeat until done
    CONTINUE                       ! (XMAX = Terminating valve)
```

This code will perform the task but will be 'untidy' in appearance because of the continual erase/redraw of the disc. The technique of hidden update provides a means of drawing without the viewer seeing the redraw, which makes screen update appear instantaneous, and animation becomes possible. The FORTRAN code below moves the red disc from the left side of the screen to the right side of the screen using hidden update.

```
      PAGE = 1                      ! PAGE is Access Page
      X = 0                         ! Initialise X
      CALL SEL(0,1,0)               ! Select red
      CALL PAG(0)                   ! Page 0 for Display
      CALL SAP(1)                   ! Page 1 for Access
10    CALL RUB(0,0,XMAX,YMAX)       ! Clear Access Page
      CALL CIF(X,100,25)            ! Draw disc in Access
      X = X+1                       ! Increment X coord
      CALL PAG(PAGE)                ! Swap Access to Display
      PAGE = PAGE-1                 ! Swap PAGE variable
      IF (PAGE.LT.0) PAGE = 1       !
      CALL SAP(PAGE)                ! Swap Display to Access
      IF (X.LE.XMAX) GO TO 10       ! Repeat until done
      CONTINUE                      !
```

Note that to erase the Access Page, it is necessary to use the 4-argument form of RUB (the zero-argument form only clears the Display Page).

3.7.7 Pop-Up Menus

Pop-Up menus are becoming increasingly popular in interactive computer graphics, and so it becomes more important that graphics devices can generate such menus apparently instantaneously. The following example describes how the SBD can support pop-up menus that are rectangular in shape, with constant size and with the added constraint that only one such pop-up menu can be displayed on the screen at a time. The method uses archives to store the commands that generate the menus and also uses a hidden page to save the pixel values that will be covered when a menu "pops up". A typical menu could be stored in an archive as follows:

```
CALL BAR (MENUID)
CALL SEL (0,4,3)
CALL SCA (WIDTH, HEIGHT, 0, CHRSET)
CALL ALP (0,0, 'ITEM 1')
CALL ALP ('ITEM 2')
CALL ALP ('ITEM 3')
CALL EAR
```

If the constant menu size is MXS by MYS pixels (in the above case, 6*WIDTH and 3*HEIGHT respectively, assuming a single-height font) then "calling up" this menu at (X,Y) could be done by

```
CALL ZOO (X,Y,X+MXS,Y+MYS,X,Y,X+MXS,Y+MYS,0,1)
CALL LIX (X,Y)
CALL RAR (MENUID)
CALL LIX (0,0)
```

This assumes that page 0 is the display page and page 1 has been defined on power-up or by an appropriate SET. The ZOO primitive copies the pixels about to be covered by the pop-up menu to the hidden page for temporary storage. When the time comes to remove the pop-up menu, simply use ZOO to copy the original pixels back over the menu area.

```
CALL ZOO (X,Y,X+MXS,Y+MYS,X,Y,X+MXS,Y+MYS,1,0)
```

An alternative method, if archiving is not available, is to store all menus on the hidden page and use the "swap" mechanism of ZOO to call up menus and restore data afterwards.

You could support multiple pop-up menus by allocation of particular "blocks" of the hidden page combined with a stack technique. Overlaid menus would add a further complication.

3.7.8 Raster Processing and Window Management

A raster, by definition, is a rectangular array of pixel values. Certain applications require a facility to read rasters from the graphic device's frame buffer, and also a facility to write rasters into the device's frame buffer. As the SBD is an intelligent device interfaced to a host computer used to run the application program (rather than a workstation), the host cannot access the SBD's display memory directly. The primitives RIB (read image block) and WIB (write image block) are provided to transfer raster data between the host memory and the SBD's frame buffer via whatever interface is in use. For detailed information on the use of RIB and WIB see the primitive descriptions.

All of the standard raster operations can therefore be performed by reading a raster into host memory using RIB, processing the raster data if necessary, and then writing it back to the frame buffer using WIB. Although RIB and WIB can also be used to perform the Copy Raster function it can be done more efficiently using the ZOO primitive, which is performed locally by the SBD MC68000 micro-processor with very little communication overhead and a minimal amount of host processing time.

The ZOO primitive is a powerful tool for window management applications. It can be used to save those parts of windows overwritten by other windows. The SAS and RPC commands allow each window to have its own attributes without interfering with other windows.

3.7.9 User-Defined Symbols (Icons)

The SBD allows the application programmer to define graphic items used repetitively as symbols that are stored locally in the SBD and displayed by issuing a single graphics command. The symbols are defined as a rectangular bit pattern. Up to 26 symbols are defined at any time and all symbols are contained in a standard size matrix which is defined before any symbols may be created.

The maximum matrix size for a single symbol is 16 x 16 pixels. Larger symbols can be made up by concatenating several single symbols. The constituent symbols are defined independently and concatenated when the command to display the symbol is given.

As an example, consider defining the Greek letter "beta" as an SBD symbol. The symbol is to be used with the SBD's 7x9 character set (single height) with a 9x15 cell size. The bit pattern octal numbers are (.=0, *=1):

Bit Patterns	Octal Numbers	Decimal
....	000	0
...**....	060	48
..*..*...	110	72
.*...*...	210	136
.*..*....	220	144
.****....	360	240
.*...*...	210	136
.*....*..	204	132
.*****...	370	248
.*.....	200	128
.*.....	200	128
....	000	0
....	000	0
....	000	0
....	000	0

The following FORTRAN code will define this dot pattern as symbol 'B' and will display it as position (X,Y) on the display screen.

```
IMPLICIT INTEGER (A-Z)      ! All variables are integer
DIMENSION DOTS (16)         ! Beta dot pattern array
DATA DOTS /  0, 48, 72,136,144,240,136,132
+          248,128,128,  0,  0,  0,  0,  0/
CALL SSA (9,15,0)           ! Define symbol matrix size
CALL SYM (DOTS,'B' )        ! Define beta symbol
CALL SYM (X,Y,'B')          ! Display beta symbol
```

It should be noted that when the symbol matrix size is defined by SSA, all symbols defined prior to the SSA call will be deleted. You can call SSA with only one argument to change the symbol orientation

without deleting existing symbols. The SET primitive also deletes all currently defined symbols.

Note also that although only 15 words are required in this example to define the symbol (and only 10 of these are non-zero), the "array" form of SYM always requires 16 words in the definition array. This is a constraint of the host library.

3.7.10 Virtual Coordinate Space

The actual coordinate space of the SBD is defined by the current resolution (see SET in 3.10) - e.g. 0...767 in X, 0...573 in Y. Any attempt to read or write a pixel outside this physical range is meaningless, since the pixel does not exist in the display memory. Pixel pages are not physically contiguous, and so coordinates cannot "wrap round" or overflow from one page to another. The SBD therefore "clips" any primitive which attempts to draw outside pixel page boundaries (producing an informational severity message, "Coordinate out of range"). Only those parts of a primitive which actually lie within the pixel page area are drawn.

For example:

```
CALL BLK (-10,-10,9,9)
```

The theoretical size of this block is 20x20 pixels. However, only the part from (0,0) to (9,9) is seen on the screen (10x10 pixels), so three-quarters of the block is clipped.

The LIX primitive (see 3.10) allows you to "remap" the physical coordinate space onto a much larger "virtual coordinate space" (VCS). The VCS coordinates are 16-bit and therefore range from -32768 to +32767 in both X and Y. The LIX command defines a pair of "index registers", whose value is added to any coordinates used as arguments to most output primitives, in order to derive the eventual physical coordinates to be used.

For example:

```
CALL LIX(10,10)
CALL BLK(-10,-10,9,9)
```

This sequence remaps the BLK corner coordinates to (0,0) and (19,19), so that the whole block is now written to the Access Page in display memory. The visual effect is that the screen origin has moved to (-10,-10).

Using this technique, it is possible to define large 'virtual pictures' with coordinates in the VCS range. Depending on the current value of the index registers (i.e. the last LIX command) some of the primitives in the VCS picture may be mapped wholly or partially onto the physical screen, and thus are seen when the commands are executed. This allows you to define a "window" onto the VCS, whose size is equal to the display resolution. By calling LIX and re-executing all the commands in the picture, the effect is to move this "screen window". Using the hidden update technique described in 3.7.6, it is possible to continually remap the screen window, giving the effect of "panning" across the VCS. (If there is a large number of commands in the picture, the effect may be jerky, since all commands must be executed in order to determine whether

they are clipped).

When using this technique, it is best to suppress all I severity errors (see the MES command in 3.10 and the INI command in 3.6). This prevents "Coordinate out of range" messages from being generated when a primitive is clipped. It also allows clipped primitives to be archived (this is often the best way to store a picture which is to be executed repeatedly as described above).

NOTE: if a supplied coordinate, when the index register is added to it, gives a value outside the range -32768...+32767, the E severity error "Coordinate invalid" is generated, and no part of the primitive is displayed. For this reason, and for reasons described in 7.1.5, it is best to limit yourself to index values and coordinates in the range -16384...+16383. If you do this, such overflow can never occur.

When LIX is called from within an archive, the Index Registers operate relatively rather than absolutely. See the LIX command in 3.10 for an explanation of this.

3.8 Building & Running Programs

3.8.1 Building and Running Programs under RSX-11M/M-PLUS

Use your usual compiler to compile the source code of your applications program. For example:

```
MCR>FOR APPLIC=APPLIC          DCL>FORTRAN APPLIC
```

Then taskbuild the resulting object code with the GGS host library, e.g.:

```
MCR>TKB APPLIC/FP=APPLIC,GGS/LB  DCL>LINK/FLOAT APPLIC,GGS/LIBRARY
```

You need the /FP (or /FLOAT) qualifier if your compiler has been built to use PDP-11 floating point instructions. If you omit the switch, your task will fail to initialise on running. You may also need to specify a run-time library for the compiler (e.g. LB:[1,1]F4POTS.OLB) unless this has been included in a system library (LB:[1,1]SYSLIB.OLB).

If your application has several modules they should all be included in the command line. When using overlays, you need only include the GGS library at the root level. Run the task in the usual way:

```
>RUN APPLIC
```

Note: If your application program does not assign a physical device unit at run time (see 3.6), you can assign one at taskbuild time. This can then be changed by the privileged REA command as long as the task is installed first. For example, if the program (we will assume it is FORTRAN) contains the following initialisation:

```
    CALL INI(1,-1,2,17)          ! Use LUN 1 for DMA
```

then if you wish the task to communicate with SB0: by default, taskbuild it in the following way:

```
MCR>TKB                          DCL>LINK/OPTIONS APPLIC,GGS/LIBRARY
TKB>APPLIC=APPLIC,GGS/LB         Option? ASG=SB0:1
TKB>/                            Option? <CR>
Enter Options:                   DCL>
TKB>ASG=SB0:1
TKB>//
MCR>
```

(You can also do this using an options file.) When the program is run by RUN APPLIC, it will send its commands to device SB0:.

You can change this as long as you install the task first, for example:

```
MCR>INS APPLIC/TASK=...APP      DCL>INSTALL/TASK:...APP APPLIC
```

You can examine the LUN assignments of the installed task as follows:

```
MCR>LUN APP                     DCL>SHOW TASKS:APP/LOGICAL_UNITS
SB0:  1.                        SB0:  1.
SY0:  2.                        SY0:  2.
SY0:  3.                        SY0:  3.
SY0:  4.                        SY0:  4.
TI0:  5.                        TI0:  5.
CL0:  6.                        CL0:  6.
TI0:  7.                        TI0:  7.
TI0:  8.                        TI0:  8.
CL0:  9.                        CL0:  9.
```

You can then reassign the chosen LUN to another SB: unit, say SB1:, (this is a privileged command) and run the program as follows:

```
MCR>REA APP 1 SB1:              DCL>ASSIGN/TASK:APP SB1: 1
MCR>APP                         DCL>APP
```

The program will then communicate with SB1: instead. This reassignment affects only the pool-resident data for the installed task, not the disk copy, and will not affect a running task. To use multiple copies of the task, install it several times with different names, and assign LUNs for each installed version separately.

3.8.2 Building & Running Programs under RT-11

Use your usual compiler to compile the source code of your applications program. For example:

```
.FORTRA APPLIC
```

Then link the resulting object code with the GGS host library, e.g.:

```
.LINK APPLIC,GGS
```

NOTE: If you are using the RT-11 XM monitor with a memory management unit (MMU),you are required to observe a software restriction imposed by Digital, as follows:

The nature of the problem involves the RT-11 XM monitor's use of MMU Page Address Register 1 (PAR1). PAR1 controls the mapping of virtual addresses 20000(8) through to 37776(8). When XM is first bootstrapped with 'Kernel' mapping, the VIRTUAL addresses within computer memory map directly to the PHYSICAL addresses within computer memory. However, the XM monitor itself uses PAR1 to map to EMT (emulator trap) blocks and user data buffers. When the RT-11 system is running, the Kernel virtual addresses within the PAR1 range can be mapped anywhere within computer memory, and the user has no way of controlling the mapping.

This restriction dictates that any data required by the SBD RT-11 DMA Handler that is initialised or allocated within the GGS host library linked to your application program, must not be located between address limits 20000(8) to 37776(8). (If your system uses the MQ handler to communicate among system jobs, all of the restrictions described for PAR1 also apply to PAR2, which controls the range of virtual addresses 40000(8) through to 57776(8).)

The RT-11 host library contains channel definition data used by the XM DMA handler, and so programs linked to it are subject to the restriction.

The GGS host library is designed to give maximum assistance in overcoming this restriction. Unfortunately, the solution to the problem involves some additional work for the user as described below.

a) Compile / assemble your SBD application software as normal.

b) Link your application program with the GGS host library, producing a link map as a by-product of the program linkage operation:

```
.LINK/MAP:DK: APPLIC,GGS
```

c) Examine the link map produced (in this case, DK:APPLIC.MAP). Locate PSECT (Program Section) CHADAT, and read off the virtual address assigned to the start of this PSECT.

Example section of link map:

```
        .
        .
        .
CHADAT  050506 000252 = 85. words  (RW,D,LCL,REL,CON)
  !       !
Virtual start address of PSECT = 50506(8).
```

d) If the virtual address of the CHADAT PSECT is not within the virtual address range 20000(8) to 37776(8) (as in the above example), you DO NOT have to perform any further special program linking procedures for your applications software to execute successfully (unless the PAR2 restriction is also applicable to your system,in which case the virtual address of the CHADAT PSECT must also be outside the virtual address range 40000(8) to 57776(8)).

e) If the CHADAT PSECT falls within the address range(s) specified in d), you must link your applications software with a special link option, to place the CHADAT PSECT outside this/these address range(s).

Use the LINK utility /BOUNDARY switch (or /Y:n option) to relocate the CHADAT PSECT to an acceptable address, for example:

```
.LINK/BOUNDARY:nnnnnn APPLIC,GGS
Boundary section? CHADAT
.
```

The value nnnnnn is an octal number (which must be a power of 2) that represents the VIRTUAL address at which the CHADAT PSECT will be relocated. The value of nnnnnn must be assigned to place the CHADAT PSECT outside the PAR1 and PAR2 (if applicable) virtual address range; this numeric assignment is left to the user's discretion, but would normally be above the highest address in the program. See the RT-11 System Utilities Manual and Software Support Manual.

Where programs linked to the GGS library are overlaid, GGS should normally be linked in the root segment so that it is always in memory.

3.8.3 Building & Running Programs under VAX/VMS

Compile your applications program in the usual way.

To link to the object library, use a command such as:

```
$ LINK/EXE APPLIC,[GLION]GGS/LIBRARY/INCLUDE=GGSMES
```

If you omit the /INCLUDE qualifier, your image will not include any of the GGS error messages. They can still be used, if the message file has been linked separately and installed (see 2.4.3).

To link to the shareable image, specify the option

```
[dir]GGS/SHARE
```

in an options file. Omit the location if it is the current directory.

For example:

```
$ LINK APPLIC,SYS$INPUT/OPT
[GLION]GGS/SHARE
<CTRL>Z
$
```

Linking to the installed image is similar, but you may omit the device/directory specification.

You can spawn or run the process in the usual ways.

3.8.4 Building & Running Programs Under UNIX

The UNIX host software will be available in late 1985, at which time this section will be issued.

3.9 Error Reporting and Debugging

The SBD can be set to return a status code for each command sent to it. The values are in the range 0 (for success) to 63 decimal. As described in 3.6, some or all severities of error can be suppressed. Appendix A details all error codes and their corresponding severities.

The status value returned by the host library, when using the call format STATUS = xxx(ARG1,ARG2...) can be in the range 1 (for success) to 255 (decimal). Note that to obtain compatibility with the host operating systems, the success code from the SBD is converted from 0 to 1. Status values in the range 2 to 63 (decimal) correspond to codes returned by the SBD. Status values in the range 64 to 127 (decimal) are generated by the library itself without reference to the SBD. Status values in the range 128 to 255 (decimal) are operating system dependent; these are often quoted in the relevant operating system documentation as negative decimal numbers, i.e. 255 = -1, 128 = -128.

For RSX-11M systems, codes in the range 128-255 correspond to system directive status codes and can be found in the relevant RSX documentation rather than in Appendix A.

For RT-11 systems, codes in the range 128-255 are generated by the library in response to system directive (programmed request) errors. See Appendix A for their meanings.

For VMS systems, the status returned by the library is 32 bits rather than 8, and is in the standard VMS message format - severity code in bits 0:2, message number in bits 3:14 etc. Message numbers in the range 1 to 127 correspond to those in Appendix A. Errors in system directives generate the VMS-specific codes for those errors.

NOTE: system service errors in all operating systems are generally considered fatal and will cause the program to terminate. For example, under RSX, if a requested SBD is offline, the following might occur:

```
>RUN APPLIC/TASK=FUNNY
GGS-F-<-65.> INI Device offline
FUNNY -- Exiting due to ERROR 9
TRAP instruction trap (SST6)
at PC = 011036
  in "APPLIC" at or after 10
```

The error message would not be quite so informative if the library had been configured without error message text included, to save space. It would then look something like:

GGS-F-ERR=-65. FUNC=1.

You would have to look in the RSX documentation to find out what error corresponded to -65, and in 5.5 to find out which subroutine was being called at the time.

Debugging

There are a number of ways of debugging programs linked to the GGS library. These are merely listed here.

1. Insert debug lines in the code. These can often help by printing out the values of useful variables at critical places in the program flow. Many languages offer compiler switches to remove these lines at compile time when the program is debugged (e.g. D-lines in DEC FORTRAN).

2. If using blocking mode, you can use the INI command (see 3.6) to disable blocking temporarily. All commands which would normally be blocked into an array are then sent direct to the SBD instead. The library normally reports only the first error in a block, so this technique enables you to track down all errors at an early stage.

3. Use the ASCII protocol and redirect the output to a handy VDU or printer nearby. (You can use the SBD itself if you disable the ASCII introducer in SETUP). Make sure the library is configured to send a linefeed as well as a carriage return at the end of each command; this is not one of the normal configuration options and must be set up by editing the CONFIG file (see 2.4). Alternatively, use the feature of the INI command which allows you to write the commands into a file, and print or type that file. In this way you can examine the actual commands that are being produced by your program via the host library.

4. Use a proprietary debug feature to trace program flow (e.g. VAX/VMS DEBUG; PDP-11 FORTRAN-77 DEBUG). You can configure the library so that important labels within it are made into global symbols, if required.

3.10 Subroutine Descriptions

The commands in this section are described in alphabetic order and are not functionally grouped.

These are the full specifications of all commands available under the GGS host library (except for INI which is described in 3.6); for a quick reference see Appendix C. Note that the commands are described in terms of FORTRAN calls; see 3.5 for details of calling routines from other languages. These specifications also serve as a means of describing the range and effect of the command arguments sent to the SBD in case you are not using GGS (see Chapter 5).

Error messages given (with error codes in decimal, and severities) are those specific to a single command or group of commands. A number of errors are general and can be produced by any command, since they are normally caused by factors not relating to a particular command, such as data corruption or interface failures. These errors are:

3.(F) Illegal number of arguments.

A command was recognised but the number of arguments was incorrect.

14.(F) Invalid function code.

This can occur when a DMA command block has incorrect format, causing the SBD to read off the end of it. (Normally this would cause error 43.) Alternatively, a spurious binary introducer character has been received on the serial line - this can happen on power-up in some systems.

23.(E) Insufficient archive memory.

The archive space is full, or the archiving option not enabled.

43.(F) Invalid command block.

The -1 terminator (see Chapter 5) at the end of the command block was not found.

Errors 14 and 43 do not normally produce a sensible command mnemonic in front of the error message on the VDU display of the SBD. "HJI" or "???" are often seen instead.

Glossary of Terms

In the following command descriptions, some terms are used which require explanation. These are:

1. Output Primitives

An output primitive is a command which can cause an actual modification to pixel values in the Access Page (q.v.). In other words, it produces visible output. Examples are RUB, VEC. Note that the primitives may not actually cause a visible change at the time of execution, e.g. if a vector lies completely outside the page, or the Access Page and Display Page are different.

2. Index Registers

These are X and Y offsets which are added to all specified (X,Y) screen coordinates to produce an indexed display position. Note that this operation may cause an output primitive to be clipped even if its specified coordinates would have put it wholly within the pixel page. The Index Registers are set by the LIX command. They are both zero on power-up.

3. Current Position

This is the default position used by certain forms of the output primitives (e.g. VEC with 2 arguments). It is set by the output primitives as specified in the command descriptions, irrespective of whether clipping occurs. On power-up it is (0,0). No command exists to explicitly set the Current Position only; however this can be simulated. The Current Position only applies to the Access Page (q.v.). It may be anywhere in the Virtual Coordinate Space (see 3.7.10).

4. Access Page and Display Page

The Access Page is the area of pixel memory in which output primitives are written. The Display Page is the area of pixel memory which is visible on the monitor screen. Both are the same size, equal to the currently set screen resolution. They may of course be the same page number.

5. Additive Mode, Complement Colour, Line Type, Foreground, Background

See 3.7.4 for an explanation of these terms.

Assumptions

Because the SBD is so flexible in its output capabilities, certain assumptions are made throughout the Command Descriptions, unless explicitly stated otherwise. You should interpret the examples given with this in mind. The assumptions made are:

- Screen resolution is set to 768x574 non-interlaced, with two pages - an implicit SET (32,2) command.

- The Y coordinates increase from top to bottom of the screen, i.e. the screen origin is at upper left. Bear in mind that positional expressions such as "upper left" are invalidated if the Y axis is inverted (increasing up the screen).

- Output Mode is 3 on both SBD (A) and SBD (B).

3.10.1 ALP

Draws an alphanumeric string in the Access Page from the current or a specified position with the current character attributes.

Call Formats

1. CALL ALP (STR)
2. CALL ALP (NC,STR)
3. CALL ALP (X,Y,STR)
4. CALL ALP (X,Y,NC,STR)

Arguments

STR is an ASCII string or a Hollerith. NC is the number of characters in the Hollerith. ASCII strings must be terminated with a null character. Apart from <SPACE>, non-printing characters in the string are ignored.

X and Y are the minimum coordinates of the rectangle which the string cells will occupy. (This is always the point closest to (0,0) and is therefore at the end of the string, rather than the beginning, for orientation code 2 (Up) with Y increasing downward, or orientation 3 (Down) with Y increasing upward). This allows strings to appear in the same place relative to the rest of the picture, irrespective of the direction of the Y axis.

The Index Registers are added to X and Y to determine the position at which the string is drawn.

When orientation mode 0 is selected by SCA, the Current Position is updated to facilitate the drawing of another string immediately below. (This is irrespective of the Y axis direction; if Y increases downwards, Cy is incremented, and if Y increases upwards it is decremented.) The cells of the first characters in the strings will be horizontally aligned and vertically separated by the vertical pitch.

For the remaining orientation modes, the Current Position is unchanged or becomes (X,Y) if these arguments are supplied.

Examples

```
CALL ALP (100,100,'This is a string')

BYTE CHRS(10)
DATA CHRS /'H','e','l','l','o',' ','g','u','y','s'/
CALL ALP (10,CHRS)
```

Errors

13.(I) Coordinate out of range.

All or part of the string was outside the screen boundary. Only those characters whose background cell fits completely on the screen are displayed.

19.(W) String too long.

The string is more than 132 characters long; it is truncated.

38.(E) Coordinate invalid.

The specified coordinates, when added to the Index Registers, generated a value outside the range -32768..32767. The string is not drawn.

3.10.2 BAR BAR

Switches on archiving. Subsequent commands are archived (if successful) after being actioned, until an EAR command is encountered.

Call formats

1. CALL BAR (N)

Arguments

N is the file number by which the archive is to be known, in the range 0 to 65534.

If an archive file already exists with this number, the BAR command (and subsequent EAR) generate errors and are ignored, and intervening commands are not archived.

When archiving, commands which generate a non-successful status are not archived, and also terminate archiving, unless errors of that particular severity have been disabled (see MES and 3.6). Commands which are fatal always terminate archives, even if fatal errors have been suppressed. The SET command is prevented from executing when archiving.

A number of commands are illegal when archiving because they require the return of data to the host. These commands are identified throughout this section. The error is returned and the archive terminated, but the command itself may still execute successfully.

Example

```
CALL BAR (1)
```

Opens archive number 1. Subsequent commands, until EAR is called, will be stored in the archive after they have been executed. It is not possible to disable command execution while archiving.

Errors

20.(E) Invalid archive code.

The archive number was -1 (65535).

21.(E) Archive already open.

Currently in archiving mode (unmatched BAR).

23.(E) Insufficient archive memory.

Archive memory is full, or the archiving option has not been enabled.

27.(E) Archive directory full.

There is insufficient memory left to spawn a subdirectory (which requires 192 words).

3.10.3 BBK BBK

Begin generating a command block in a user-defined array. Following BBK, all commands which are legal in blocking are coded into the array instead of being sent to the SBD. (EBK optionally allows these commands to be executed.)

Call Formats

1. CALL BBK (A,S)

Arguments

A is an INTEGER*2 array of size S words (elements). The maximum value of S is 32767. It is the users responsibility to ensure that the array is large enough to receive all the commands that are to be blocked. On PDP-11 systems, the practical maximum size is much less than 32767 words unless separated I and D-space are used. Virtual arrays are not supported by the library.

Commands which require the return of data to the user program, commands which require an indirect array, and commands which affect the library's operational mode are illegal in blocking. The following are the illegal commands:-

BAR BBK CHI CUR DDI EAR EGB FIN INI RIB TAB TRA WIB

The library protects against blocking into an array which is being accessed asynchronously. However, it cannot detect overlaid arrays which do not start at the same location.

BBK and EBK are not illegal while archiving. The commands after BBK are not included in the archive unless EBK executes them.

Example

```
INTEGER*2 CB(2000)
CALL BBK(CB,1000)          ! use first half of array only
```

Errors

65.(E) Command not blockable.

BBK has already been called without an intervening EBK.

68.(E) Display list locked.

This array, or one occupying the same memory locations, is currently being executed (asynchronous mode only).

3.10.4 BIT BIT

Draws a single pixel in the current Access Page in the foreground colour.

Call Format

1. CALL BIT (X,Y)

Arguments

(X,Y) are coordinates in the current Access Page. The contents of the Index Registers are added to the co-ordinates to determine the position at which the single pixel will be drawn.

The Current Position becomes (X,Y).

Example

```
CALL BIT (0,0)
```

Errors

13.(I) Coordinate out of range.

The specified position was outside the screen boundary.

38.(E) Coordinate invalid.

The specified coordinates, when added to the Index Registers, generated a value outside the range -32768..32767.

3.10.5 BLK BLI

Draws a solid rectangle in the Access Page using the current line type. The diagonal corners of the rectangle are the Current Position and a specified position, or two specified positions.

Call Formats

1. CALL BLK (X2,Y2)
2. CALL BLK (X1,Y1,X2,Y2)

Arguments

(X1,Y1) and (X2,Y2) are co-ordinates in the current Access Page. The contents of the Index Registers are added to the co-ordinates to determine the position at which the block is drawn. The use of line types produces a 45 degree stripe from lower left to upper right.

The Current Position becomes (X2,Y2).

Example

```
CALL BLK (IX,IY+20,IX+20,IY)
```

This draws a 21-pixel square block whose upper left corner is (IX,IY) - though note that the other two opposing corners were specified. The Current Position is set to (IX+20,IY+20).

Errors

13.(I) Coordinate out of range.

The specified position was outside the screen boundary.

38.(E) Coordinate invalid.

The specified coordinates, when added to the Index Registers, generated a value outside the range -32768..32767.

3.10.6 BOX BOX

Draws an outline rectangle in the Access Page, using the current line type. The diagonal corners of the rectangle are the current position and a specified position, or two specified positions.

Call Formats

1. CALL BOX (X2,Y2)
2. CALL BOX (X1,Y1,X2,Y2)

Arguments

(X1,Y1) and (X2,Y2) are coordinates in the current Access Page. The contents of the Index Registers are added to the co-ordinates to determine the position at which the rectangle will be drawn.

The Current Position becomes X2,Y2.

Example

CALL BOX (0,0,767,573)

Draws a box round the edge of the screen (assuming resolution is 768x574).

Errors

13.(I) Coordinate out of range.

The specified rectangle is not wholly inside the screen boundary. It is clipped.

38.(E) Coordinate invalid.

The specified coordinates, when added to the Index Registers, generated a value outside the range -32768..32767.

3.10.7 CHI CHI

Changes the introducer character used by the ASCII serial line protocol. (Also changes that generated by the host library).

Call Format

1. CALL CHI (CHAR)

Arguments

CHAR is a word containing the ASCII value of the new introducer character in the bottom byte, and zero in the top byte. (It can be a quoted string, and in VMS systems it must be.)

Valid ASCII introducers are:

Char	Name	ASCII code (octal)
!	Exclamation	41
#	Pound or hash	43
$	Dollar	44
%	Percent	45
&	Ampersand	46
*	Asterisk	52
/	Slash	57
<	Less than	74
>	Greater than	76
?	Query	77
@	At-sign	100
\	Backslash	134
_	Underscore	137
~	Tilde	176

This command is designed to be used only via the DMA interface, or else sent via the serial line without involving the library (e.g. at system startup). Apart from the fact that it will not work when sent via the ASCII protocol (since GGS sends the command with the new introducer preceding it), there are obvious dangers in changing the introducer in the middle of a program, especially where multiple tasks are using the ASCII protocol to interface with the display. You can configure both the library and the SBD to use any of the above values by default and it is advised that you do this if the PPL default (tilde) is not suitable.

Note: the SBD itself allows any value for the introducer, but the library only allows those above (the same ones which can be selected by SETUP for the SBD).

Example

```
CALL CHI('?')
```

Resets the introducer to query.

Errors

26.(E) Command illegal in archive.

65.(E) Command illegal while blocking.

3.10.8 CIF CIF

Draws a solid circle (or 'pie' segments) in the current line type, in the Access Page.

Call Formats

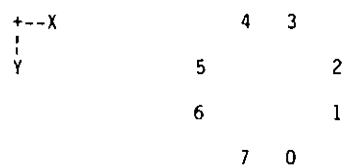
1. CALL CIF (X,Y,R)
2. CALL CIF (X,Y,R,OCT)

Arguments

(X,Y) are the coordinates of the centre of the circle. The Index registers are added to X and Y to determine the position at which the circle will be drawn.

R is the radius of the circle and the maximum value is 4095.

OCT is an optional 8-bit binary number specifying the octants of the circle which are to be drawn. As for the CIR command, each bit represents an octant as follows:



When three arguments are specified, a complete disc is drawn. When four arguments are supplied, the ends of each discrete arc are joined to the centre by straight vectors and the resulting 'pie' segments filled in the foreground colour and current line type. The 45 degree "stripes" caused by the line type are lower left to upper right in octants 2, 3, 6, 7 and upper left to lower right in octants 0, 1, 4, and 5.

The Current Position is unchanged by this command.

Example

```
CALL CIF (100,100,50,"125)
```

This draws alternate octants 0, 2, 4 and 6. Note that the double quote signifies an octal number in DEC FORTRAN.

Errors

13.(I) Coordinate out of range.

The whole of the circle was not inside the screen boundary.

32.(E) Bad radius.

The radius was negative or greater than 4095.

38.(E) Coordinate invalid.

The specified coordinates, when added to the Index Registers, generated a value outside the range -32768..32767.

3.10.9 CIR CIR

Draws a circle or a partial circle in the current Access Page, using the current line type.

Call Formats

1. CALL CIR (X,Y,R)
2. CALL CIR (X,Y,R,OCT)

Arguments

(X,Y) are the co-ordinates of the centre of the circle. The Index Registers are added to X and Y to determine the position at which the circle will be drawn.

R is the radius of the circle and the maximum value is 4095.

OCT is an optional 8 bit binary number specifying the octants of the circle which are to be drawn. Each bit represents an octant as follows:

When only three arguments are supplied a complete circle is drawn.

The Current Position is unchanged by this command.

Example

CALL CIR (200,200,100,15)

Draws the right-hand semicircle (octants 0, 1, 2, 3).

Errors

13.(I) Coordinate out of range.

One or more of the required octants was not completely inside the screen boundary.

38.(E) Coordinate invalid.

The specified coordinates, when added to the Index Registers, generated a value outside the range -32768..32767.

3.10.10 CUR

Activates the local pixel cursor in the Display Page, under control of trackerball or mouse (via PCU) and the Direction Keys. ~~NOTE: CUR does not support trackerballs and mouse connected via an~~ ISI.

Call Formats

1. CALL CUR (X,Y)
2. CALL CUR (X,Y,K)
3. CALL CUR (X,Y,XSIZE,YSIZE)
4. CALL CUR (X,Y,K,XSIZE,YSIZE)

Arguments

(X,Y) are the co-ordinates of the initial position where the cursor is to appear.

XSIZE and YSIZE are the dimensions when a box cursor is required instead of a small cross. X and Y refer to the top left-hand corner of the box when the Y co-ordinate increases downwards. They refer to the bottom left-hand corner when Y increases upwards.

K is the value of the Transmit Key which was pressed. (see below).

The Direction Keys ('Arrow Keys' - UP, DOWN, LEFT, RIGHT) are available on the standard keyboard and allow a method of cursor control (albeit less versatile) if a trackerball or mouse is not available. The command terminates, and the cursor co-ordinates are returned to the host, when any of the Transmit Keys, defined below, is pressed. Keys not defined as Transmit Keys have no action while the cursor is active.

The Transmit Keys are those in the Numeric Keypad, numbered as in Table 3.6. The three buttons on the 2-inch trackerball also function as Transmit Keys, as do the buttons on the mouse.

The cursor is written by complementing the pixel data already present in the current display channel. This ensures that the cursor contrasts with the surrounding data and that the data can be restored when the cursor moves on. In practice, plane 3 (channel 5) and Video Output Mode 0 will often be used for the cursor and the plane will have been erased for the purpose. The cursor will then appear in white. For advice on the selection of complement colours see 3.7.4.2.

The cursor moves in single-pixel increments in response to single depressions of the Direction Keys, but accelerates if the keys are held down for more than half a second or so (with Auto-Repeat

enabled). The Direction Keys can only move the cursor in one of the four directions at one time; angled movement is not possible.

The Current Position is unaffected by the CUR command.

Because this command requires the return of data, it is always performed in Synchronous Mode, and is illegal while blocking.

Keyboard (keypad):

Key engraving	Value returned in K
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
.	10
PF1	11
PF2	12
PF3	13
PF4	14
- (minus)	15
, (comma)	16
ENTER	17

Trackerball and Mouse

Button(s)	Value returned in K
Right	1
Middle	2
Middle + Right	3
Left	4
Left + Right	5
Left + Middle	6
Left + Middle + Right	7

Table 3.6 Return Key Values Corresponding to Send Buttons

Example

```
X = 0
Y = 0
CALL CUR (X,Y,K)
IF (K.EQ.17) STOP
```

Activates the pixel cursor at (0,0) and returns the new coordinates in X and Y when a Transmit key is pressed, together with the value of the transmit key. If the ENTER key is pressed the program terminates, otherwise it continues.

Errors

13.(I) Coordinate out of range.

The specified position was outside the screen boundary.

26.(E) Command illegal in archive.

34.(E) Peripheral not connected.

Neither the keyboard nor a trackerball was present in the configuration.

38.(E) Coordinate invalid.

The specified position was outside the range -32768...32767.

65.(E) Command illegal while blocking.

3.10.11 DAR DAR

Deletes an archive file and releases the space for use by other archives.

Call Format

```
CALL DAR (N)
```

Arguments

N is the file number of the archive to be deleted, in the range 0...65534.

Note: the archives are not 'squeezed' and therefore if a lot of deletion and insertion of archives is taking place, it is good policy to initialise and completely regenerate the archive directory from time to time, to prevent excessive fragmentation and slowing down of recall.

Example

```
CALL DAR (23)
```

Deletes archive number 23.

Errors

20.(E) Invalid archive code.

An archive number of -1 (65535) was specified.

22.(E) Archive not found.

The specified archive did not exist.

26.(E) Command illegal in archive.

You cannnot delete an archive from within another archive. This is to prevent an archive from accidentally deleting itself, with disastrous consequences.

3.10.12 DDI DDI

Returns information allowing the user to generate a directory display. Appendix B gives an example of a program which does this.

Call Format

```
CALL DDI (L,N1,S1,N2,S2,N3,S3,N4,S4)
```

Arguments

L is the 8-word directory line to be returned (in N1...S4).

The directory is structured as a series of "lines" of four directory entries each. Each entry consists of two 16-bit words: the first word (N) is the archive number, the second (S) is the size (in words). If an entry is not allocated it consists of two -1 words; if a line is not allocated then the value -1 is returned in L (which must not, therefore, be a constant).

All nine arguments are returned (including L, even when unchanged). This command cannot, therefore, be archived or blocked. It is performed in synchronous mode.

Line 0 of the directory has a special format:-

N1 number of directory lines
S1 number of archives
N2 -1 (not used)
S2 -1 (not used)
N3 size of archive storage in 256-word blocks (including directory)
S3 number of blocks of archive used (including directory)
N4 number of blocks free
S4 -1 (not used)

Given this information, it is an easy matter to write a program to format a readable directory listing on the host terminal (see Appendix B).

Example

See Appendix B.

Errors

26.(E) Command illegal in archive

65.(E) Command illegal while blocking.

3.10.13 DEF

Assigns memory planes to Display Channels. This command is provided only for compatibility with earlier Supervisor products. The default assignments allow the full capability of Supervisor SBD to be used without calling DEF.

Call Format

1. CALL DEF (C,M,W,G)

Arguments

C is a channel number in the range 0-15. Not more than 8 channels may be defined at one time. If you define new channels, you lose the old corresponding default channel.

M, W, and G give the Mode, Width and Group combination. Mode must be 1 (its value is ignored); Width is the number of planes in the channel (1-4); Group is the lowest numbered plane in the channel (0-3). Planes in a channel must be contiguous. Allowable combinations of M, W, and G are:

Default Channel Number	M	W	G	Planes in Channel
0	1	4	0	0 1 2 3
1	1	3	0	0 1 2
2	1	1	0	0
3	1	1	1	1
4	1	1	2	2
5	1	1	3	3
6	1	2	0	0 1
7	1	2	2	2 3

Example

```
CALL DEF (0,1,4,0)
```

Sets up the default Channel 0 definition.

Errors

8.(E) Invalid channel number.

Channel number not in range 0-15.

9.(E) Invalid channel width.

Channel width not in range 1-4.

10.(E) Incompatible width and group.

Attempt to define an illegal channel.

3.10.14 DLT DLT

This command allows any 16 bit pattern to be specified and selected for use with primitives that support line types.

Call Format

1. CALL DLT (P)

Argument

P is a 16 bit pattern which may be expressed in any convenient way, such as an Octal constant. Note: a value of zero will cause all primitives which support line types to appear in the background colour.

The command SLT (q.v.) allows you to select from a range of predefined useful line types. SLT(0) means all foreground, and is the same as DLT(-1). SLT and DLT override each other in operation.

See 3.7.4.4 for information on the use of line types. The interaction of line types with the area fill primitives is described under the various command headings in this section.

Example

CALL DLT ("52525)

This defines a pattern of alternating foreground and background pixels. This pattern is useful for generating false colours by "dithering" (see 3.7). Equivalent to SLT(1).

Errors

None specific.

3.10.15 DMP

Causes the current Viewing Page to be printed on the inkjet printer. No further graphics commands are processed until the print has completed, but the command can be aborted if the DMA interface is being used.

Call Formats

1. CALL DMP
2. CALL DMP (TIM)
3. CALL DMP (TIM,FLAG)
4. CALL DMP (TIM,FLAG,X1,Y1,X2,Y2)

where TIM is a timeout value, FLAG is the print mode selector, and (X1,Y1) (X2,Y2) are opposite corners of the screen area to be printed.

Arguments

TIM (timeout) is specified in screen ticks (50 per second at 50Hz, 60 per second at 60Hz). This defines the amount of time the SBD will wait, after an XOFF is received from the printer, for an XON to indicate that the printer can receive more data. If zero is specified, the timeout is disabled and the SBD will wait for XON indefinitely. If this argument is omitted (call format 1), a default of 1500 (30 seconds at 50Hz) is assumed. The value of specifying a timeout is that if you wish to terminate a print request prematurely, you need only switch the printer offline (do NOT turn off the power) and after the specified interval the SBD will timeout and resume command processing. It also means that the SBD will not "hang" indefinitely if the printer's power fails.

FLAG can take the following values:

0 - the print will be in the 16-colour mode (default for call formats 1 and 2). All lines on the display will be printed on the screen. If FLAG is greater than 3 it has the same effect as if it were zero.

1 - the print will be in the 8-colour mode, with all pixel lines printed.

2 - the print will be in the 16-colour mode, but only alternate lines (1,3,5...) from the screen will be sent to the printer. Whether a "line" consists of vertical columns or horizontal rows of pixels from the screen depends on the screen-to-printer rotation (see below). You should take great care when exercising this option since significant data could be lost.

However, it doubles the print speed.

3 - the print will be in the 8-colour mode, with alternate lines being sent. The same comments apply as for FLAG = 2.

-1 - instead of the pixel page, an 8x8 array of colour squares is printed which gives the full palette provided by the printer in the 16-colour mode, as an aid to selecting colour mapping values (see the SDM command). This array is independent of the current selected mapping (see below).

Where call format 4 is used, the coordinates specified in (X1,Y1) and (X2,Y2) are used to delineate the area to be printed. If omitted, they default to (0,0) and (Xlim,Ylim), where Xlim and Ylim are the maximum X and Y dimensions respectively.

Print Modes

The inkjet printer operates using four coloured inks, which are black, cyan, magenta and yellow. It can mix any two of the colours cyan, magenta or yellow to produce dots in secondary colours:

cyan + magenta = blue
magenta + yellow = red
cyan + yellow = green

Absence of any dot produces white (as long as the paper is white!)

The 8-colour mode allows these 8 standard single-dot colours to be printed, at half the density of the 16-colour mode, but almost twice the speed. Because only half as many dots as in the 16-colour mode are printed, the colours appear paler. Since only eight print colours are available in this mode, it will not be possible to distinguish all 16 SBD colours in a print, but apart from this restriction any mapping between the 16 SBD pixel values and the 8 printed colours is possible. See the SDM command for further information.

The 16-colour mode allows 16 different distinguishable colours to be printed, which can be selected from a palette of 36 available colours (see below). The increased palette is achieved by "dithering" alternate dots which are one of the eight colours produceable by mixing inks. To maintain a "square" pixel aspect ratio, blank lines are left between the rows of dots. The density and "spreading" of the dots prevents a stripe effect. Consider the following two dots representing a pixel on the screen:

3 1

The human eye merges these dots, of yellow and red, into a single orange dot representing colour 49 (3x16 + 1) or 19 (1x16 + 3) in the colour square below.

In the 16-colour mode, each of the 16 colour values available on the SBD can be mapped to any of the 36 available printer colours. In the 8-colour mode, each of the 16 colour values available can be mapped to any of the 8 colours in the "restricted" palette. The output mode (Model A) or the contents of the colour lookup tables (Model B) have no effect on the printed colours. See the SDM command for details of how to set up the printer colour mapping.

NOTE: The dot density (16 colour mode) is 1280 dots (effectively 640 pixels) per line (a line being 7.64 inches), giving a horizontal density of 84 pixels per inch. The line density is 3/252 inches per line (84 lines per inch) so the aspect ratio of the screen is preserved when all lines are printed. If only alternate lines are printed, the SBD attempts to double the line spacing to 6/252 inches per line, to maintain the aspect ratio. Unfortunately a timing problem in the printer firmware limits the line spacing to a maximum of 5/252 inches per line at present, so that the alternate line prints are compressed in the direction of paper feed by a factor of 20%. The problem relates to the optimisation of printhead and carriage movement at the end of a print line, and a solution to this problem is under consideration by the suppliers of the printer.

Print Speed Optimisation

Typical print times are given below (whole page printed, no white space at end of lines):

Resolution	Lines printed	16-colour mode	8-colour mode
768x574	All	11 min 31 sec	6 min 25 sec
768x574	Alternate	5 min 51 sec	3 min 16 sec
512x384	All	5 min 12 sec	2 min 54 sec
512x384	Alternate	2 min 39 sec	1 min 29 sec

To speed up the printing process, the SBD will rotate the area dumped so that the minimum number of lines is printed, as long as the maximum dimension of the print area will fit on one print line (i.e. is 640 pixels or less). For example,

```
CALL DMP (1500,1,0,0,383,381)
```

would generate an unrotated print (since there is no saving in the number of lines by rotating it), but

```
CALL DMP (1500,1,0,0,381,383)
```

would cause the print to be rotated since this saves two lines. However,

```
CALL DMP (1500,1,0,0,641,381)
```

would generate a rotated print (382 dots by 642 lines) since a row of 642 pixels would not fit on a print line.

NOTE: It is not possible to send an odd number of pixels to the printer for a particular print line. Therefore, for an unrotated print the number of horizontal pixels in the window (X2-X1+1) must be even (it is rounded down if not), and for a rotated print the number of vertical pixels in the window (Y2-Y1+1) must be even. For example,

```
CALL DMP (1500,1,0,0,381,382)
```

is interpreted as

```
CALL DMP (1500,1,0,0,381,381)
```

but is still rotated even though there is no saving of lines.

To further optimise the print speed, if any line of pixels (on the printer, not the SBD) ends with one or more pixels mapped to white, this data is not sent to the printer. A line which is totally mapped to white is not sent at all - the printer carriage simply racks up one line. This results in two time savings: the time taken to send the pixel information, and the time taken for the printhead to travel to the end of the line.

Print Colour Square Array

The colour map printed by DMP (n,-1) is a square consisting of 8x8 square colour blocks. The colour squares are numbered as in Figure 3.2, i.e. increasing by 1 down a column and by 16 along a row (numbers are decimal). These numbers are used when calling the SDM routine to map one of the SBD's colour values (0-15) into an actual colour on the printer.

Note that the array reflects about a diagonal from upper left to lower right, which means that colour 1 is the same as colour 16, 22 is the same as 97 and so on. This gives a full palette of (8x8/2)+4 colours, i.e. 36.

Colours on the diagonal (0, 17, 34, 51, 68, 85, 102, 119) are unique. These are the primary and secondary colours normally used to represent black, red, green, yellow, blue, magenta, cyan and white respectively. (In the 8-colour mode, you can get only these colours, in half-density - i.e. those in the right-hand column, or bottom row.)

The algorithm for colour values becomes more obvious if you consider it as two adjacent 4-bit nybbles in a byte, each of which can contain a value from 0 to 7. This is in fact how the larger palette is achieved, as described earlier.

(black)	0	16	32	48	64	80	96	112	
	1	17	33	49	65	81	97	113	
	2	18	34	50	66	82	98	114	
	3	19	33	51	67	83	99	115	
	4	20	34	52	68	84	100	116	
	5	21	35	53	69	85	101	117	
	6	22	36	54	70	86	102	118	
	7	23	37	55	71	87	103	119	(white)

Figure 3.2 DMP colour square array mapping numbers

Examples

```
CALL DMP (1500,0)
CALL DMP
```

These two commands are functionally identical, and cause the current pixel page to be printed in the 16-colour mode, with a timeout value of 30 seconds.

```
CALL DMP (500,3,499,100,100,399)
```

This causes the rectangular part of the screen whose corners are (100,100) and (499,399) to be printed, in the 8-colour mode (alternate lines only). Note that although X2 is smaller than X1 the routine adjusts accordingly. The defined rectangle is 400 pixels wide (i.e. in the X direction) and 300 pixels high (in the Y direction). It will therefore be printed unrotated to minimise the number of lines on the printer. The timeout value specified is 10 seconds at 50 Hz, 8.3 seconds at 60 Hz.

Errors

34.(E) Peripheral not connected

The SBD did not detect a printer in the configuration on power-up. Possible causes: no printer, cable loose or faulty.

36.(E) Peripheral not ready

The specified timeout has expired without an XON from the printer. Possible causes: printer offline, powered off or faulty.

3.10.16 EAR EAR

Switches off archiving. Subsequent commands are not archived.

Call Formats

1. CALL EAR
2. CALL EAR (S)

S, if present, is set to the number of words occupied by the archive file. If archiving is not active, this is given the value zero and an error is generated.

Because the second form of this command requires the return of data, it is performed synchronously even if the current mode is asynchronous, and cannot be blocked. However, the first form can be blocked and performed asynchronously. Obviously, the command cannot be archived since it terminates archiving.

Example

```
CALL EAR (SIZE)
```

Closes the currently open archive and returns its size in SIZE.

Errors

25.(E) Archiving not active.

No BAR has been called to initiate archiving.

65.(E) Command illegal while blocking.

(Second format only).

3.10.17 EBK

Terminates the blocking of commands and optionally executes the block or returns the number of array elements occupied by the blocked commands.

Call Formats

1. CALL EBK
2. CALL EBK (S)

Arguments

S is the number of elements (words) occupied by the blocked commands. If the S parameter is not supplied the block is executed.

Only the first form of this command results in the sending of commands to the SBD. The command cannot be blocked because it terminates blocking. It can be called when archiving; the first form would result in all the blocked commands being included in the archive and the second form would have no effect on the archive.

Example

```
CALL EBK (SIZE)
```

The currently open command block array (see BBK) is closed and its size (in words) returned in SIZE. The block size (in bytes) is also written to the first word of the array. No commands are sent to the SBD.

Errors

69.(W) Not blocking.

No command block is open. The command is ignored and S is unchanged.

3.10.18 EGB EGB

Executes a block of graphics instructions. The block may be generated by the BBK and EBK commands or provided directly by the user program.

Call Format

1. CALL EGB (A,S)

Arguments

A is an INTEGER*2 array containing one or more graphics instructions coded in the standard GGS Command Block Format (see Chapter 5). The Host Library generates this format automatically while blocking. The array must not be a virtual array.

S is the size of the array in words (elements), max. 32767. This acts as a check on, and overrides, the block size stored in the block itself, which may have been altered during a previous execution if an error was detected in the block (see 5.4).

The size of a block is returned by EBK. It is the user's responsibility to make sure this remains consistent with the block. If you store a block long-term (e.g. on a magnetic disk) which has generated an error on execution, without rewriting the correct block size in bytes (S * 2) in the first element of the array, then not all the commands in the block will be executed when the block is next read and executed (unless you store the size separately).

Users creating their own command blocks independently of the library, and executing them using the library using either the EGB or OUTPUT (see 3.1) commands, should ensure the blocks are accessible for both read and write.

All the commands which are permitted within a block function as described. The Current Position is modified according to the commands executed.

If EGB is called while archiving, the effect is to include all the commands in the block as part of the archive. The EGB command itself is not archived.

Example

```
INTEGER*2 PICTR(5000)
.
.
CALL BBK (PICTR,5000)
.
. call graphics commands which are loaded into the block,
. not sent to the SBD.
.
CALL EBK (ISIZ)          ! returns nr of words used in PICTR
.
.
CALL EGB (PICTR,ISIZ)    ! sends all commands to SBD
```

Alternatively, if the command block has been read from long-term storage into array PICTR, and the length is not known:

```
ISIZ = PICTR(1) / 2      ! convert length to words
CALL EGB (PICTR,ISIZ)
```

If the array generated an error on execution before being written to disk, only the commands up to the first faulty one will be executed using this method, unless the true length (in bytes) of the block is written back to the first element.

Errors

65.(E) Command illegal while blocking.

68.(E) Command block locked for I/O.

The specified block is currently being executed in asynchronous mode.

70.(W) Block size mismatch.

The specified size did not match the byte count stored in the first element of the array, which is overwritten.

3.10.19 ELF ELF

Draws a solid ellipse or solid quadrants of an ellipse, in the current line type, in the Access Page. The semi-major and semi-minor axes are aligned with the screen axes.

Call Formats

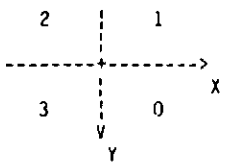
1. CALL ELF (X,Y,XR,YR)
2. CALL ELF (X,Y,XR,YR,QUAD)

Arguments

(X,Y) are the coordinates of the centre of the ellipse. The Index Registers are added to these coordinates to produce the absolute position.

XR, YR are the horizontal and vertical radii of the ellipse, in the range 0-4095. They include the perimeter and the centre. In the degenerate cases, if one radius is either zero or 1, a one-pixel-wide line is drawn. If both radii are zero or 1, a single dot is drawn.

QUAD is a 4-bit pattern (the least significant bits of the word) defining the quadrants to be drawn. Each bit represents a quadrant as follows:



If only four arguments are supplied, a complete solid ellipse is drawn. If five arguments are supplied, each specified quadrant is drawn with the current line type. The line type "stripes" are from lower left to upper right in quadrants 1 and 3, and from upper left to lower right in quadrants 0 and 2.

The Current Position is unchanged by this command.

Example

```
CALL ELF (383,286,384,287)
```

This draws a complete solid ellipse which just touches the edge of the screen at the left and top, and is 1 pixel short of it at the right and bottom. Note that the diameter is always (Radius*2) - 1 and must therefore be odd.

Errors

13.(I) Coordinate out of range.

The specified position, radii and quadrants meant that some or all of the ellipse had to be clipped to the screen edges.

32.(E) Bad radius.

One or both radii was not in the range 0-4095.

38.(E) Coordinate invalid.

The specified coordinates, when added to the Index Registers, generated a value outside the range -32768..32767.

3.10.20 ELI ELI

Draws an ellipse or quadrants of an ellipse in the current Access' Page using the current line type.

Call Formats

1. CALL ELI (X,Y,XR,YR)
2. CALL ELI (X,Y,XR,YR,QUAD)

Arguments

(X,Y) is the centre of the ellipse. The Index Registers are added to these values to determine the absolute position.

XR, YR are the radii of the horizontal and vertical axes respectively. See ELF for details.

QUAD is a 4-bit binary number specifying which of the four quadrants is to be drawn. Each bit represents a quadrant as for ELF. Only the perimeter of each quadrant is drawn, not the radius. When only four arguments are present, a complete ellipse is drawn.

The Current Position is unchanged by this command.

Example

See ELF.

Errors

13.(I) Coordinate out of range.

The ellipse was clipped.

32.(E) Bad radius.

One or both of the radii was not in the range 0-4095.

38.(E) Coordinate invalid.

The specified coordinates, when added to the Index Registers, generated a value outside the range -32768..32767.

3.10.21 FIN .FIN

Causes the library to dissociate from the Supervisor unit previously associated by INI.

Call Format

1. CALL FIN

Arguments

None.

FIN waits until all outstanding I/O with the SBD has completed. If the asynchronous mode is used and the program exits (as opposed to being aborted) some operating systems may generate an error and the last requested command may not be executed, if FIN is not called. If the synchronous mode is used, it is only necessary to call FIN if INI is to be called again to change the library's operating parameters. However, it is good programming practice always to call FIN immediately before program exit.

If you are using the RSX-11M library in its write-to-file mode (see 3.6.1), you MUST call FIN before program exit in order to close the file.

If the library is currently blocking, the FIN is ignored and you cannot call INI again.

Errors

65.(E) Command illegal while blocking.

A command block is currently open. The FIN command is ignored.

67.(E) No device active.

FIN has already been called, or INI has not been called.

3.10.22 FLO FLO

Fills a polygon in the Access Page, of random shape bounded by the background colour or the edge of the page, with the foreground colour.

Call Format

1. CALL FLO (X,Y)

Arguments

X and Y are the co-ordinates of the "seed point" within the polygon in the current Access Page. The Index Registers are added to X and Y to determine the actual seed point position.

The Current Position is unaffected.

In additive mode (see SEL), the fill takes place to any boundary which is not colour 0 (black). Also, the fill can take place to "not colour N" where N is any of the 16 available colours. In other words, any colour except N forms the fill boundary, and only colour N is overwritten. This method can be used to fill "left-over" areas whose boundaries are not necessarily all of one colour. See 3.7.4.3 for a description of this technique.

The fill works by identifying pixels of the background colour, N (or "not N") and drawing horizontal vectors in the foreground colour from left to right until they meet another pixel of the background colour (or foreground colour). To cope with complex shapes, a "stack" of seed points is built up so that all "shadowed" areas are dealt with. Because the amount of stack space (which is held in static RAM) is limited, a shape which is particularly complex may cause the stack to overflow. FLO recovers from this but the shape may not necessarily be completely filled. Also, if areas of foreground colour are present in the polygon, the fill may not be complete.

The horizontal vectors are drawn in the current line type. This means that it is possible to fill with the false colour mixtures generated by "dithering" (see 3.7). The fill boundary must, however, be solid. Coarser line types would lead to irregular (not 45-degree) stripes, generally following the line of the left-hand border.

If you try to fill with complement colour, the results are not guaranteed to be what you expect and the fill may not be complete. The fill algorithm drops seed points above and below the current fill line, at the extreme left of the fill line just inside the border. For a line to fill in complement colour, such seed points must be colour zero in the current channel, and no point immediately above or below such seed points may complement to colour zero in the channel.

As the edge of the pixel page is always considered a fill boundary, care must be taken when using the "virtual window" technique described in 3.7. If the indexed seed point lies outside the page boundary, the fill will not take place, since FLO can only examine those pixels actually present in the pixel memory. Don't forget that FLO has no knowledge of the previous commands which created the fill border.

Example

```
CALL SEL (0,0,15)      ! select all planes
CALL RUB               ! erase Display Page to purple
CALL SEL (4,0,1)       ! select plane 2 only
CALL VEC (300,100,500,400)
CALL VEC (100,400)
CALL VEC (300,100)     ! triangle in light blue (colour 11)
CALL SEL (7,1,2)       ! select planes 2 & 3
CALL FLO (300,300)     ! fill triangle with colour 7 (white)
```

This is rather a contrived example, designed to show the interaction of FLO with the current channel and the pixel values in the channel. Here FLO operates only on planes 2 and 3. Within these planes it is looking for 2 as a border colour (i.e. no bit set in plane 2, bit set in plane 3). The only pixels meeting these criteria are those forming the triangle (all other pixels have value 3 in this channel). The fill colour is 1, so this has the effect of clearing plane 3 and setting plane 2 all over the filled area, which therefore turns to colour 7 (white).

Errors

13.(I) Coordinate out of range.

The seed point position was outside the page boundary. The fill is not performed.

31.(I) Too complicated.

The internal seed point stack overflowed. As much of the fill as could be performed, was.

33.(E) Seed pixel is background colour.

Attempt to fill a degenerate area.

38.(E) Coordinate invalid.

The seed point coordinates, when added to the Index Registers, generated a value outside the range -32768..32767.

3.10.23 FSH

(Model B only.) Sets the rate of flashing between colour mapping tables 0 and 1 (see MAT).

Call Format

```
1  CALL FSH(T0,T1)
```

Arguments

T0 is the time to display mapping table 0 and T1 is the time to display table 1. Both times are unsigned 16 bit integers (range 0 to 65535) and give the time in screen frame times or "ticks" (either 1/50 or 1/60 second). This allows any value between 0.02 sec and about 22 min (at 50 Hz).

The mark/space ratio (MSR) is thus T0/T1 and the duty cycle (seconds per cycle) is (T0+T1)/F, where F is the field frequency (50 Hz or 60 Hz).

A value of zero for either T0 or T1 implies that the corresponding mapping table is never used, and the other one is permanently in use irrespective of its time. This disables flashing. If both values are zero, table 0 is permanently in operation.

To maintain compatibility between Model A and Model B, and for ease of programming, the command SOM(1) on Model B implicitly executes a FSH(22,11), as well as loading the two mapping tables appropriately (Table 0 = RGB, Table 1 = black).

Examples

```
CALL FSH (0,0)
```

Disables flashing. Mapping table 0 is permanently operative.

```
CALL FSH (0,1)
```

Disables flashing. Mapping table 1 is permanently operative.

```
CALL FSH (45,5)
```

Causes flashing between mapping tables 0 and 1 with an MSR of 9:1 and duty cycle of 1 second (at 50 HZ).

Errors

None specific.

3.10.24 IDI

Initialises the archive directory and erases all archived files.

Call Format

1. CALL IDI

Arguments

None.

Initialisation of the archive directory is automatic on power-up or reset of the SBD, and when the SET command is called, so is not normally necessary except for housekeeping, as described under DAR.

This command terminates any open archive and deletes it.

IDI implicitly calls SET and hence screen disturbance may be seen as the screen format is reconstructed. All SAS context records are also lost.

Errors

41.(I) Existing record deleted.

One or more stored context records (see SAS) has been lost. Subsequent calls to RPC, if any, will generate an error.

3.10.25 INI INI

Initialises the host library, connects to a physical interface to the SBD and sends it a MES command to set up error handling.

Call Format

1. CALL INI ([LUN],[PUN],[EFN],[MODE])

Arguments

All arguments are optional.

LUN : Logical unit number.

PUN : Physical unit name / number.

EFN : Event flag number.

MODE: Mode of operation.

Because the INI command is operating-system-dependent, you should refer to section 3.6 for details on how to use it in your particular operating environment. This section summarises the command only.

Errors

71.(E) Device already active.

Two INI calls without an intervening FIN.

75.(E) Invalid or reserved event flag

The EFN argument was not in the permitted range.

76.(E) Invalid or reserved LUN

The LUN argument was not in the permitted range.

77.(E) Invalid physical unit number

The PUN argument was not in the permitted range.

78.(E) DMA interface not supported

An attempt to use the DMA interface via a library not configured to support it.

79.(E) Serial ASCII interface not supported

An attempt to use the serial ASCII interface via a library not configured to support it.

80.(E) Serial binary interface not supported

An attempt to use the serial binary interface via a library not configured to support it.

82.(E) No serial interface supported

An attempt to use a serial interface via a library not configured to support it.

91.(E) File access not supported

An attempt to write to a file via a library not configured to support this.

92.(E) Error in filename

The filename (PUN argument) was not legal.

3.10.26 LIX LIX

Loads the Index Registers, XI and YI.

Call Format

1. CALL LIX (XI,YI)

Arguments

XI and YI are the values to be loaded into the Index Registers (range: -32768 to 32767).

Note: when an output primitive or part of a primitive is outside the pixel page area (after indexing), the informational message 13 ("Coordinate out of range") is generated. If part of a primitive has coordinates (after indexing) outside the range (-32768 to 32767) in X or Y or both, the error 38 ("Coordinate invalid") is produced. If you confine yourself to LIX values and page coordinates in the range (-16384 to 16383) the latter error can never occur.

Among the uses of LIX is the ability to relocate archived picture segments and image blocks on the screen. A picture larger than the screen dimensions can be stored as an archive and portions of it viewed by indexing and redrawing, giving a "windowing" facility. You are advised to suppress informational messages if you wish to do this (see 3.6 and the MES command).

NOTE: when LIX is called outside any archive, it sets the Index Registers ABSOLUTELY; in other words, it overwrites them. No errors are possible from an absolute LIX. However, when LIX is called by an executing archive, it sets the Index Registers RELATIVELY; in other words, when an archive is recalled, the current Index Registers are saved on a "stack" and then LIX ADDS its specified values and the stacked "base offset" to generate the value written to the Index Registers. If the resulting coordinates are invalid (see above) an error is generated. When the archive completes, the Index Registers are restored from the "stack". Two LIX commands within the same level of archive use the same "base offset", and do not interact with each other. Outside any archive the "base offset" is (0,0).

This feature allows relative positioning of nested archives; see the example face-drawing program in Appendix B.

Example

```
CALL LIX (-100,100)
```

This effectively shifts subsequent output primitives to the left and down - in other words, "pans" 100 pixels to the right and 100 pixels up in the virtual coordinate space.

Errors

38.(E) Coordinate invalid.

The specified coordinates, when added to the current base offset, generated a value outside the range -32768..32767.

3.10.27 MAG

Draws magnified ASCII strings from the specified position, in the Access Page, with the current character attributes (see SCA).

Call Formats

1. CALL MAG (MF,STR)
2. CALL MAG (MF,NC,HOL)
3. CALL MAG (X,Y,MF,STR)
4. CALL MAG (X,Y,MF,NC,HOL)

Arguments

X and Y are the co-ordinates of the mimimum corner (closest to the origin) of the rectangular cell containing the string, as for ALP. The cell is erased to the background colour unless in Additive Mode.

MF is the magnification factor which may be from 1 to 16.

STR is an ASCII string terminated in a null byte (PDP) or a string descriptor (VAX/VMS). The characters are drawn in foreground colour and are the standard fonts, magnified by pixel replication. Apart from <SPACE>, non-printing characters in the string are ignored.

NC and HOL form a Hollerith string. NC is the number of characters in the byte array HOL.

The Current Position is set as described under ALP.

Example

```
CALL SCA (7,10,1,2)     ! select 5x7 font double ht. upwards
CALL MAG (100,100,3,'Magnified')
```

Each character cell resulting from this is (3*7) = 21 pixels by (10*2*3) = 60 pixels. The whole string occupies the area from (100,100) to (159,288).

Errors

13.(I) Coordinate out of range.

One or more of the characters was not wholly inside the page boundary.

37.(E) Magnification factor too large.

A magnification factor greater than 16 was specified.

38.(E) Coordinate invalid.

The specified coordinates, when added to the Index Registers, generated a value outside the range -32768..32767.

3.10.28 MAT MAT

(Model B only.) Loads the pixel colour mapping tables and VDU colour.

Call Formats

1. CALL MAT(A)
2. CALL MAT(2,VC)

Arguments

A is a sixteen element array. The values of the array elements are:

T,V1,V2,V3,V4,V5,V6,V7,V8,V9,V10,V11,V12,V13,V14,V15

where T is the table number (0 or 1), and Vn is the value for colour n (see below for value specification). Note that, since the value for colour 0 must be 0 (black) there is neither necessity nor means to enter it.

In the second format, which is used to set the VDU colour, the first argument represents the mapping table for the VDU (Table 2), which contains two entries: entry 0 = zero (black), entry 1 = VDU colour. In reverse video the mapping is swapped so that the background appears in the VDU colour and the text in black. VC is the VDU colour which is specified in the same way as the pixel mapping colours (see below).

Elements 2 to 16 of array A (and VC) contain an RGB colour value. This is defined as:

256*B + 16*G + R

where R, G and B are the relative intensities of red, green and blue respectively, in the range zero to 15. Only the least significant 12 bits of the word are examined.

It is possible to dynamically switch between pixel tables 0 and 1 at a fixed frequency, allowing area flash (see the FSH command).

The SOM command (q.v.) automatically sets up the mapping tables and flash attributes to correspond to the fixed values provided by the Output Modes in Model A. See also 3.7.3.

The single-argument form of MAT can be used to set the VDU colour, if A(1) = 2 and A(2) = VDU colour. The array A must still be 16 words long, so this is a wasteful method.

Examples

```
    INTEGER*2 PIXMP0(16),PIXMP1(16)
    DATA PIXMP0/ 0, 1, 2, 3, 4, 5, 6, 7,
   +             8, 9,10,11,12,13,14,15/
    DATA PIXMP1/ 1, 0, 0, 0, 0, 0, 0, 0,
   +             0, 0, 0, 0, 0, 0, 0, 0/
     .
     .
     .
    CALL MAT (PIXMP0)
    CALL MAT (PIXMP1)
    CALL MAT (2,15)
    CALL FSH (0,0)
```

These operations set mapping table zero to 16 levels of red, and mapping table 1 to black. Table 0 is permanently operative so that the colours are solid. The VDU colour is set to full intensity red. This might be the mode of operation, say, in a military installation under red lighting. To cause the whole screen to flash it is then only necessary to call:

```
    CALL FSH (33,17)
```

which causes the whole screen to flash to black at 1 Hz with a 2:1 MSR.

Errors

42.(E) Invalid argument.

No such mapping table (not 0, 1 or 2).

3.10.29 MES

Controls the handling of errors by the SBD.

Call Formats

1. CALL MES
2. CALL MES(E)

Arguments

E is the error handling mode, made up as follows:-

Bit 0 determines whether errors detected by the SBD are to be communicated to the host. If zero, errors are not returned. If 1, errors are returned. Note that status values are always returned by the DMA interface, but this bit determines whether the library reports them.

Bit 1 determines whether errors detected by the SBD cause messages to be displayed in the scrolling text store (on the bottom line) with automatic MIX if necessary. If zero, errors are displayed. If 1, errors are not displayed.

Bits 2:5 determine the handling of errors of varying degrees of severity - see 3.6. Bit 2 set suppresses I (informational) severity, bit 3 is W, bit 4 is E and bit 5 is F.

Bits 6:15 are ignored.

Note: These bits correspond to bits 0, 1 and [10:13] respectively of the Mode argument of the INI command (see 3.6). INI in fact transparently generates a MES command of the second format.

If argument E is not present (call format 1), the effect is to 'toggle' bit 0, i.e. if errors were being returned, they are subsequently not returned, and vice versa. This form of the command is supplied for compatibility with earlier Supervisor products. On power-up, status values are not returned until a MES command is received.

Example

CALL MES (15)

Errors are returned but not displayed in the VDU area. Errors of severity I and W are not generated.

CALL INI (,,,3091)

Generates a MES (15) via the DMA interface.

Errors

None specific.

3.10.30 MIX MIX

Switches into view the Current Viewing Page in the upper part of the screen and scrolling VDU text in the lower part of the screen.

Call Formats

1. CALL MIX
2. CALL MIX (R)

Arguments

R specifies the number of rows of scrolling text in the range 0 to 24. The default value of R at power-up is configurable (see section 2.3): this value is used if no argument is specified (call format 1), or if the MIX key on the keyboard is pressed. However, MIX with one argument (call format 2) changes this default value until the SBD is reset or re-powered. The value zero for R is not advisable as a power-up default, since the bottom row of the VDU should always be visible for the confidence message. If the zero value is configured, then the value 3 is used for the MIX key and for MIX with no arguments.

The MIX mode of display is entered automatically if EITHER a graphics command is received while in VDU mode OR VDU text is received while in PIX mode (including error messages generated by the SBD). This is called "auto-MIX". Note that the VDU or pixel information just written is not guaranteed to be visible. The number of VDU lines visible is the same as the argument to the last MIX command (or the power-up default), including zero.

The values zero and 24 for R are not quite the same as calling PIX and VDU (q.v.) respectively, because auto-MIX is then disabled. For instance, if you do not want any VDU lines to appear even if something is written to the VDU area, use MIX(0). To restore auto-MIX, call PIX or VDU.

After MIX(0) is called, pressing the MIX key on the keyboard (or calling MIX with no arguments, which is functionally equivalent) will cause a reversion to the last selected non-zero MIX value. This prevents disabling of the MIX key and locking the SBD into a situation where the VDU cannot be seen. (Calling PIX or VDU, or pressing the PIX/VDU key, has the same effect, but this would not be noticed until next time the MIX mode was entered.)

Example

Call MIX (1)

Makes one VDU line visible at the bottom of the screen.

Errors

12.(E) Too many lines.

The argument was greater than 24 or negative.

3.10.31 PAG PAG

Defines a pixel page to be the Current Viewing Page and switches it into view.

Call Format

1. CALL PAG (P)

Arguments

P is the pixel page number. Numbering starts at zero. The maximum page number depends on the current display resolution and the board type, according to Table 3.7. The SET command can specify a number of pages less than the maximum permissible, to increase available archiving space.

SET code	Resolution	Max page number	
30	768x586	0	)
30	768x574	1	)
31	512x384	3	) High-resolution SBD
32	768x574	1	)
33	512x384	3	)
2	768x586	0	)
2	768x574	1	)
3	384x293	7	) Low-resolution SBD
4	512x512	2	)
5	256x256	12	)
40	768x574	1	)
41	704x526	1	) North American SBD
42	640x480	1	)
43	480x360	4	)

Table 3.7 Maximum page numbers at different resolutions

The Viewing Page following power-up or reset, or after a SET command, is always Page 0, as is the Access Page (see SAP).

For an example of the use of PAG and SAP to control hidden update, see 3.7.6.

Example

CALL PAG (0)

Future output primitives affect page 0.

Errors

5.(E) Too many pages.

Requested page number larger than the maximum permitted for the current resolution (see SET).

3.10.32 PIX PIX

Switches into view the Current Viewing Page and no scrolling VDU text.

Call Format

1. CALL PIX

Arguments

None. Functionally equivalent to pressing the PIX/VDU key on the keyboard twice (once if already in VDU mode). Subsequent calls to MIX with no arguments will revert to the most recently specified non-zero value.

Errors

None specific.

3.10.33 RAR RAR

Recalls and executes an archived picture segment.

Call Format

1. CALL RAR (N)

Argument

N is the number of the archive to be displayed, in the range 0 to 65534.

RAR calls can be nested, i.e. an archive can itself contain RAR commands. The nested archive (sub-segment) need not exist at the time the RAR for it is stored in the calling segment. (See the LIX command for further information relating to nested archives). However, if RAR is called outside any archive, then an error is reported if the recalled archive does not exist.

Example

Examples of the use of archives are given in 3.7 and Appendix B.

Errors

20.(E) Invalid archive code.

An archive number of -1 (65535) was specified.

22.(E) Archive not found.

The specified archive did not exist. This error is only generated if archiving is not active.

24.(E) Archive nesting level too deep.

More than 10 levels of nested archive requested.

28.(F) Cannot recall open archive.

Attempt to call an archive from itself. This is a fatal message since these cannot be suppressed when archiving, and this command must not be allowed to appear in an archive.

3.10.34 RIB RIB

Reads a two-dimensional array of pixels from the current Access Page to a host array.

Call Format

1. CALL RIB (X1,Y1,X2,Y2,P,A)

Arguments

(X1,Y1) and (X2,Y2) are the diagonal corners of the screen array. P is the data packing mode specifying the number of pixel values packed into each word of the host array, A.

P is provided for compatibility with other Supervisor display systems. Its value must be 4, indicating that four pixel values are packed into each word. The leftmost/uppermost pixel on the screen occupies the least significant four bits of the word. All four planes are read regardless of the current display channel.

See under WIB for details of the format of array A. Note that A can subsequently be used in a WIB command without modification (the corner values must bound the same size block). As the Index Registers are added to the corner coordinates to arrive at the output position, this provides a flexible method of moving blocks of information around between different displays. For block moves on the same display, it is more efficient to use ZOO (q.v.).

Array A can also be used after RIB (without change) by the EGB command to produce a WIB, for example:

```
ISIZ = IFIX(A(1)/2)
CALL EGB (A,ISIZ)
```

The effect is identical to calling WIB(.....,A). The user should not change the first 8 elements of the array, but the pixel data which occurs in the elements after this can be processed to produce various raster effects (see 3.7.8).

The Current Position becomes (X2,Y2).

As with WIB, this command may only be used via the serial line if the Binary transmission format is used. The size of command permitted is similarly limited. For RIB, however, this limitation is imposed by the host library - the SBD can return any amount of information in response to a single serial line RIB command, if your host can cope with it.

When the serial line is used for RIB, the amount of data sent back is always the same for a particular number of pixels requested. If an error occurs, the data sent back is unreliable, but would normally be zero. (This is the case for parts of the pixel block which are outside the screen boundary). Unlike every other command which returns values via the serial line binary protocol, the pixel values are returned in a binary encoded form (see 7.1). When the DMA interface is used, those parts of the host array which cannot be written with actual pixel data are left unchanged.

The RIB command is always synchronous, and is illegal when archiving or blocking.

Example

```
INTEGER*2 PIXELS(2509)
CALL RIB (0,0,99,99,4,PIXELS)
```

Reads back a 100x100 array of pixels. The array PIXELS is just large enough to contain the 10 000 pixels read plus the required format words (9). The pixel data will be contained in PIXELS(9) to PIXELS(2508) inclusive (2 500 words).

Errors

13.(I) Coordinate out of range.

All or part of the pixel array was outside the screen boundary, after the Index Registers were applied.

26.(E) Command illegal in archive.

38.(E) Coordinate invalid.

After the Index Registers were applied, some of the pixel addresses were outside the range (-32768...32767).

65.(E) Command illegal while blocking.

84.(E) Image block too large.

The rectangle bounded by the corner coordinates contained more than 131032 pixels.

85.(E) Binary protocol not configured for imaging support.

The library was not configured to support WIB and RIB via the serial binary protocol.

86.(E) Imaging routine not translateable to ASCII.

Attempt to send WIB or RIB via the ASCII protocol. This error also appears on the host terminal when any other error is detected in WIB or RIB, since the library attempts to translate the offending command and display it in ASCII on the user's terminal.

3.10.35 RPC RPC

Restores the parameters currently at the top of the context stack.

Call Format

1. CALL RPC

Arguments

None.

The values restored are defined under the SAS command.

Errors

26.(E) Command illegal in archive.

40.(E) Nothing saved.

The context stack is empty. The command has no effect.

3.10.36 RUB RUB

Erases the currently selected background channel to the background colour/intensity.

Call Formats

1. CALL RUB
2. CALL RUB (X1,Y1,X2,Y2)

Arguments

(X1,Y1) and (X2,Y2) are the corners of the rectangular area to be erased. When these are not specified (call format 1) the area to be erased is defaulted to the whole of the current Viewing Page.

NOTE: To erase the whole of the Access Page you MUST use call format 2.

The current values of the Index Registers XI and YI are added to X1, Y1, X2, and Y2. The resulting minimum X coordinate is rounded down to a multiple of 32. The resulting maximum X coordinate is rounded down to (a multiple of 32) minus 1. (See example.) The rounding allows RUB to operate faster than BLK. You are not prevented from erasing the whole page at any of the resolutions available, since all possible screen widths (768, 704, 640, 512, 480, 384, 256) are multiples of 32.

RUB operates faster if the background pixel value is zero or equal to the maximum allowed by the number of pixel planes in the current channel (see the SEL command).

The Current Position is not affected.

Examples

```
CALL PAG (0)              ! Display Page is 0
CALL SAP (1)              ! Access Page is 1
CALL RUB                  ! Erase whole of page 0
CALL RUB (10,0,760,573) ! Erase page 1 from (0,0) to (735,573)
```

Errors

13.(I) Coordinate out of range.

The specified rectangle was not wholly within the page boundaries.

38.(E) Coordinate invalid.

The specified coordinates, when added to the Index Registers, generated a value outside the range -32768..32767.

3.10.37 SAP SAP

Selects an Access Page for reading or writing of pixel data.

Call Format

1. CALL SAP (P)

Argument

P is the pixel page number. The minimum value is zero and the maximum varies according to the number of pages specified or defaulted by SET (q.v.).

On power-up or reset the Access Page is page 0.

Example

For an example of the use of PAG and SAP, see 3.7.

Errors

5.(E) Too many pages.

The page value specified was invalid for the current resolution.

3.10.38 SAS SAS

Saves a Display Context record on the context stack. The context stack is built up from archiving memory, so this command (and RPC) is ~~part of the optional set purchased with the Arcaid package.~~

Call Format

1. CALL SAS

Arguments

None. The following parameters are saved by SAS in a context record:

Channel Number
Foreground Colour
Background Colour
Current Position (XC,YC)
Index Registers (XI,YI)
Access Page and Viewing Page
Line type pattern
MES error handling bits
Number of MIX VDU lines
ASCII serial introducer character
Tablet transformations
Character attributes
Symbol orientation
Output Mode (Model A)
Mapping tables 0 and 1, Flash rates, VDU colour (Model B)

The context stack is a last-in-first-out (LIFO) buffer held in archive memory. Use of context storage therefore reduces the amount of archive storage available. A single context record occupies 56 16-bit words (Model A), which is the equivalent of roughly 11 4-argument vector commands. (On Model B, 90 words are used). There is no limit to the depth of the context stack except the availability of archive memory.

SAS and RPC facilitate the operation of asynchronous tasks by controlling the context stack. SAS is a 'push' operation (it adds a record at the top of the stack and increases the stack pointer) and RPC is a 'pop' (removes the record currently at the top of the stack and decreases the stack pointer).

Using SAS and RPC, multiple independent tasks can access the same SBD using completely different operating characteristics. The tasks need to agree some conventions between themselves (such as not changing the ASCII introducer if the ASCII serial protocol is in use, not redefining the symbol size which might destroy another task's symbols, etc.). A window management application, for instance, could

have different characteristics depending on which window was the "active" one.

Another possible use might be to restore attributes after calling an archive which could potentially alter them.

Note that the use of SET or IDI, resetting the SBD or switching power off re-initialises the context stack. A warning message is issued in the case of SET and IDI if the stack is not empty.

Errors

23.(E) Insufficient archive memory.

The archive memory is full, or else Arcaid has not been enabled (see 2.3.3).

26.(E) Command illegal in archive.

The SAS command may not be archived, since it requires the use of archive memory.

3.10.39 SCA SCA

Sets the character attributes of horizontal pitch, vertical pitch, orientation and font. These attributes affect characters drawn by ALP and MAG, but not those drawn by SYM or TXT.

Call Format

1. CALL SCA (W,H,O,F)

Arguments

W is cell width, H is cell height, O is string orientation and F is the character font code. All arguments have configurable power-up defaults.

The cell width or horizontal pitch (W), is in the range 6 to 20. The absolute maximum for any font is 20 and the minima for the various fonts are given in Table 3.8 below.

The cell height or vertical pitch (H) is in the range 8 to 32. The absolute maximum for any font is 32 and the minima for the various fonts are given in Table 3.8 below.

These cell size parameters specify the background rectangle for each character in a string, which is written to the background colour if additive mode is not in operation. The cell also determines whether or not a character is clipped - the whole cell must be within the page boundary for the character to appear.

NOTE: cell width and height are defined relative to the character, not to the X and Y axes. They may operate in different directions depending on the string orientation.

When a double height font (see below) is selected, specify the cell height as if for the single height font - the cell is automatically doubled in height, by duplicating rows of pixels.

The orientation code (O) can take the following values:

```
0 = Character    0 deg., string    0 deg.  (Normal)
1 = Character    0 deg., string +270 deg.  (Hotel sign)
2 = Character  +90 deg., string  +90 deg.  (Up)
3 = Character +270 deg., string +270 deg.  (Down)
```

Rotation is anti-clockwise from the horizontal. When orientations 0 and 1 are used, the character and cell are written together, one row at a time. With orientations 2 and 3, the entire cell is erased for each character before the character data are written. This can cause flicker on rapidly updated strings using these orientations.

The font code (F) can take the following values:

0 = 5 x 7, single height
1 = 5 x 7, double height
2 = 7 x 9, single height
3 = 7 x 9, double height
4 = 12 x 16, single height
5 = 12 x 16, double height

Characters are centred within the cell, with the odd pixel (if any) at the right hand side and at the bottom. Allowance is made for descender characters, which means that an entirely upper-case string in a cell with a large vertical pitch may not appear vertically centred.

Lower-case characters with descenders (g, j, p, q and y) are aligned with the descenders below the base line (as in normal typescript) if the cell vertical pitch permits this, always allowing at least one blank pixel row at the bottom of the cell (see Table 3.8). If the cell is not high enough, the descender characters are moved up out of line with the other characters in order to fit in the descenders, which may tend to give an unnatural appearance to the text. If this occurs, increase the cell height to the minimum shown in Table 3.8 or else use upper case characters only.

Font	5 x 7	7 x 9	11 x 16
Minimum cell width	6	8	12
Minimum cell height	8	10	17
Descender size (pixels)	2	3	4
Minimum cell height for proper descender alignment	10	13	21

Table 3.8 Standard font cell selection parameters

Examples

```
CALL SCA (6,10,1,1)
```

This sets attributes to be:

- Font 5 x 7 double height
- Cell size 6 pixels horizontal by 10 pixels (doubled to 20 pixels) vertical
- Orientation "Hotel Sign" - characters have normal orientation but the string progresses downwards.

```
CALL MAG (100,200,2,'SAVOY')
```

This string, drawn with the attributes above, would appear as follows:

S
A
V
O
Y

The string background would be 12 pixels wide (X = 100 to 111) and 100 pixels high (Y = 200 to 299). It would appear the same irrespective of the Y axis orientation.

Errors

15.(E) Invalid font code.

The font code (F) was not in the range 0 to 5.

16.(E) Invalid orientation code.

The orientation code (O) was not in the range 0 to 4.

17.(E) Invalid or incompatible cell height.

The cell height (H) was not in the range 8 to 32, or else was too small for the specified font.

18.(E) Invalid or incompatible cell width.

The cell width (W) was not in the range 6 to 20, or else was too small for the specified font.

3.10.40 SDM SDM

Sets the current mapping of SBD colour values to printed colours on the inkjet printer.

Call Formats

1. CALL SDM (TABLE)

where TABLE is an array of 16 printer colour values.

Arguments

TABLE is a 16-element INTEGER*2 array containing printer colour values in the range 0..7, 16..23, ..., 112..119 (see the DMP command for the range and meaning of these values). The first element of the array corresponds to SBD pixel value 0, the second element to 1 and so on. This means that when DMP is called, pixels on the screen of value 0 will be printed according to the colour specified in the first element of the array. The interpretation of values in TABLE differs between the 8-colour and 16-colour print modes (see below).

The user should ensure that the array is at least 16 words long, otherwise the SBD (or host library) will attempt to read off the end of it.

Colour Mapping

The pixel-value-to-printer mapping defined by SDM is independent of the current Output Mode (Model A) and of the contents of the lookup tables (Model B).

When using the 8-colour print mode, the printer only has a palette of 8 colours (black, red, green, yellow, blue, magenta, cyan, white, corresponding to colour map values zero to 7). This is because in the 8-colour mode, only one ink dot is printed per pixel to obtain colours, rather than two as in the 16-colour mode. The methods used by the printer to obtain the different colours are described under the DMP command.

However, the SBD provides a means whereby you can map any of the pixel values (zero to 15) to any of the eight printable colours in the 8-colour mode.

When SDM is called, the SBD stores the array TABLE internally, and also sends appropriate command sequences to the printer which set up the printer's internal mapping. (If any element of TABLE contains -1 or an illegal value, the previously set internal value is used.

Power-up defaults are given below.)

When DMP is subsequently called, the action taken by the SBD depends on the print mode specified:

- For the 16-colour mode, the SBD ignores the internally-stored table (except when determining when the pixels at the end of a line are mapped to white for speed optimisation purposes) and just sends raw pixel values (zero to 15) to the printer, which then selects the appropriate colours from its palette of 36. NOTE: if the printer is switched off and on it reverts to its default mapping (see below) for subsequent 16-colour prints.

- For the 8-colour mode, the SBD uses the pixel value as an index to its internal mapping table. It divides the table value by 16 and uses the remainder (which is always in the range zero to 7) as the colour value to send to the printer. The printer uses this value to select from the restricted palette of 8 colours (as defined above) and does not use its own 36-colour palette. The printer can therefore be switched on and off again without affecting subsequent 8-colour prints, though it would not be a recommended practice to rely on this.

It may help to follow the method of colour mapping if you study the colour square described under the DMP command (preferably with an actual printed colour square array, produced by DMP(n,-1), in front of you). The colour squares down the right-hand side, corresponding to SDM table values 112 to 119, are the colours available in the 8-colour mode. When these values are divided by 16 and the remainder taken, they become zero to 7. This means that if only the numbers 112 to 119 are used in an SDM array, a subsequent print will appear identical whether the 8-colour or the 16-colour mode is used.

However, if you prefer to think in terms of values zero to 7 mapping to the standard RGB primary and secondary colours, you may use these instead. (In the 16-colour mode, these map to very dark versions of the RGB colour set.)

If your application produces both 8-colour prints (for speed) and 16-colour prints (for density), the 'diagonal' table values (0,17,34,51,68,85,102,119) would be a good choice since these map to the full-intensity colour in the 16-colour mode, and to the same colour in half-intensity in the 8-colour mode.

It is not good practice to use values other than the above for the 8-colour mode.

A comparison between colours produced in 8-colour and 16-colour modes, for a sample of TABLE values, is given in Table 3.9.

Word N of TABLE contains:	Pixel value N appears as: (8-colour mode)	Pixel value N appears as: (16-colour mode)
0	black half intensity	black full intensity
1	red " "	dark red
2	green " "	dark green
3	yellow " "	dark yellow
4	blue " "	dark blue
5	magenta " "	dark magenta
6	cyan " "	dark cyan
7	white	black half intensity
17	black half intensity	black full intensity
34	red " "	red " "
51	green " "	green " "
68	yellow " "	yellow " "
85	blue " "	blue " "
102	magenta " "	magenta " "
118	cyan " "	cyan " "
19	yellow " "	orange
33	red " "	dark brown
112	black half intensity	black half intensity
113	red " "	red " "
114	green " "	green " "
115	yellow " "	yellow " "
116	blue " "	blue " "
117	magenta " "	magenta " "
118	cyan " "	cyan " "
119	white	white

Table 3.9 Printed colours for a sample of map values

Examples

```
INTEGER*2 PRTMAP(16)
DATA PRTMAP /  0, 17, 34, 51, 68, 85,102,119,
1            112,  2, 19,118,  4,  1,119,  5/
CALL SDM (PRTMAP)
```

For the 16-colour mode, this sets up the printing colours to an approximation of the 16 colours available in Output Mode 3 (although colour 14, light grey, has to be set to white). For the 8-colour mode the printing colours would be half-intensity black to white (see above) for pixel values 0 to 7, and the mapping for colours 8 to 15 would be as in Table 3.10 below.

Pixel value	Printed colour (8-colour mode)	Printed colour (16-colour mode)
8	black half intensity	black half intensity (dark grey)
9	green " "	dark green
10	yellow " "	orange
11	cyan " "	cyan half intensity (light blue)
12	blue " "	dark blue
13	red " "	dark red (brown)
14	white	white
15	magenta " "	dark magenta (purple)

Table 3.10 Printed colours for suggested mapping (high pixel values)

Note: when printing simple line drawings rather than solid images, it is advisable to map SBD colour 0 (black) to printed colour 119 (white), and SBD colour 7 (white) to printed colour 0 (black). This has the following advantages:

- it saves ink, since the printer does not print a dot for the colour white, which normally would form the picture background and is therefore the majority of the pixels;
- it saves time, since white pixels at the end of a print line are not sent to the printer, and the printhead does not then have to travel the full width of the line;
- it produces a clearer picture.

Default values

On power-up the values in the SBD's internal mapping table are set to simulate Output Mode 0 (8 colours plus white overlay) for the 8-colour mode. This is equivalent to the following code:

```
INTEGER*2 PRTMAP(16)
DATA PRTMAP /  0, 17, 34, 51, 68, 85,102,119,
1             119,119,119,119,119,119,119,119/
CALL SDM (PRTMAP)
```

The printer can be switched off and on again without affecting the SBD's internal mapping table. However, since the printer's internal mapping for the 16-colour mode is lost when it is switched off, it is always necessary to call SDM again before printing in the 16-colour mode. If this is not done, the printer's default mapping, which is not particularly useful, is used. This is equivalent to the following code:

```
INTEGER*2 PRTMAP(16)
DATA PRTMAP /119,  4,  2,  6,  1,  5,  3,  7,
1             0, 68, 34,102, 17, 85, 51,119/
CALL SDM (PRTMAP)
```

Errors

34.(E) Peripheral not connected

No printer was in the configuration on power-up.

42.(E) Illegal argument

One or more elements of array TABLE was not in the required range. The element was ignored and the previous set value, or default, was used. This error is here treated as a warning only.

44.(E) Peripheral not ready

Printer powered off. If it is simply offline, SDM will wait for ever.

3.10.41 SEL SEL

Selects a display channel and specifies the foreground and background colours/intensities. SEL can also select separate channels for foreground and background.

Call Formats

1. CALL SEL (C,F,B)
2. CALL SEL (C,F,B,D)

Arguments

C is the channel number, in the range 0 to 15 (default channels are 0 to 7). For call format 1, this also defines the background channel number.

D is the background channel number, which has the same range as the foreground channel number.

F is the foreground colour (pixel value). Subsequent output primitives write this value into the planes defined by the foreground channel. Its maximum value is therefore (2 to the power N) - 1, where N is the number of planes in the channel. For example, in channel 0 (4 planes) the maximum is 15. The minimum value of F is zero, but negative values also have meaning in that they select complement colour (see 3.7.4.2) as foreground.

B is the background colour (pixel value). Where required by a particular primitive (e.g. RUB), this value can be written into the pixel planes defined by the background channel. The maximum value is as defined for F (foreground colour). However, negative values select additive mode (see 3.7.4.3) and the value (100 + N), where N is a pixel value in the range defined above, has a special effect on the FLO command (q.v.), in that it selects "not N" as a border colour (any pixel value in the background channel which is not N).

All values have configurable defaults on power-up. The background channel is set to the same as the foreground channel.

Examples

See the examples under the FLO command.

```
CALL SEL (6,3,7,1)
```

This command sets the foreground channel as 6 (planes 0 and 1) and the background channel as 1 (planes 0, 1 and 2) and selects the maximum pixel value for each channel as foreground and backgroun colours.

```
CALL SEL (0,1,100)
CALL SEL (0,1,-1)
```

These two commands have the same effect in that they cause the border pixel value for FLO to be set to "not zero" in channel 0. However, the second call also sets additive mode whereas the first does not.

Errors

6.(E) Undefined channel.

The selected channel had not been defined by a DEF command. (It is not necessary to define channels 0 to 7).

7.(E) Bad colour.

The foreground or background colour was not in the permitted range for the channel, defined above.

8.(E) Invalid channel number.

The channel number was not in the range 0 to 15.

3.10.42 SET SET

Sets the pixel resolution, raster standard and number of pixel pages, and performs a number of initialisations. Optionally erases all pixel memory and creates a background VDU page.

NOTE: as SET is a reserved word in PASCAL, you can also call this routine in the host library as SETP.

Call Formats

1. CALL SET (R)
2. CALL SET (R,P)

Arguments

R is the resolution code, and P is the number of pages. The pages are numbered from zero to (P-1). The resolution code has a set of 4 "base values" (which are different depending on whether the SBD is high, low or North American resolution), and different effects are achieved by adding various constants to the base values. Table 3.11 below specifies the range of base values (R).

R	Horiz. Pixels	Vert. Pixels	Field Frq Hz	Interlaced / Non	Max. Pages	
30	768	586*	50	I	2	)
31	512	384	50	N	4	) High-resolution
32	768	574	50	N	2	) version
33	512	384	60	N	4	)
2	768	586*	50	I	2	)
3	384	293	50	N	8	) Low-resolution
4	512	512	50	I	3	) version
5	256	256	50	N	13	)
40	768	574	50	N	2	)
41	704	526	60	N	2	) North American
42	640	480	60	I	2	) version
43	480	360	60	N	5	)

* Reduced to 574 with 2 pages.

Table 3.11 Resolution base code values

When call format 1 is used (number of pages omitted), the number of pages is set to the maximum possible.

Resolution Code Modifiers

The following modifying values can be added to the base resolution code. Any combination of modifiers may be added.

50 causes the value of Y coordinates to increase UP the screen, causing the screen origin (0,0) to be at lower left. The default Y co-ordinate convention is that Y increases DOWN the screen.

100 suppresses erasure of pixel memory. You cannot use this modifier as a default on power-up (see 2.3.2) since it is not a sensible operation at that time. If this modifier is not present, all pixel pages are cleared to the configured background colour.

200 allocates memory for a second VDU page from archiving memory. See the VDU command for details of using the second VDU page. The hidden VDU page takes up approximately 8K words (see 3.3.3.).

Initialisations

Apart from setting the resolution and other parameters as above, SET also performs initialisations as in Table 3.12 below.

Parameter	Initialised to
Channel definitions	Default values. Other channels deleted
Pixel Viewing Page	0
Pixel Access Page	0
Selected channel number (foreground & background)	Configured value (SETUP)
Foreground colour	Configured value (SETUP)
Background colour	Configured value (SETUP)
X and Y Index Registers	0,0
Current Position	0,0
Video output mode	Configured value (SETUP)
VDU/PIX/MIX	MIX with configured VDU rows (SETUP) (If SET format 1, then PIX)
Character attributes	Configured values (SETUP)
Symbol attributes	Configured values (SETUP)
Symbol definitions	Deleted
Archives	Deleted - directory initialised
SAS context stack	Emptied
ASCII introducer	Unchanged
TIN tablet transforms	scale=1.0, shift=0.0, sin=0.0, cos=1.0
MES error handling bits	Unchanged

Table 3.12 SET initialisations

Because the SET command initialises the archive directory, it is rejected if in archiving mode and cannot be archived.

The pixel pages are deleted optionally (see above). If the SET command changes the base resolution ~~code from its previous value~~, the VDU area is also erased and the VDU cursor placed at the bottom left. The hidden VDU page (see above) is always erased.

Examples

```
CALL SET (332,2)
```

The resolution is set to 768x574 non-interlaced 50 Hz, with 2 pixel pages. (This is a high-resolution SBD). Some archive storage is set aside for a hidden VDU page and is cleared. The pixel pages are not cleared (this assumes that the resolution is not, therefore, being changed and the purpose of the SET was to allocate the extra VDU page. If this is the case, the visible VDU page is unchanged).

Errors

4.(E) Bad resolution code.

The specified resolution code was not one of those described above (with modifiers).

5.(E) Too many pages.

A number of pages was requested which could not be achieved at the specified resolution.

26.(E) Command illegal in archiving.

SET cannot be archived. Even if you suppress errors of severity E (see MES and 3.6) you cannot archive SET. It is unique among commands in this respect. SET is not executed if archiving is active.

41.(I) Existing record deleted.

The context stack (see SAS) was not empty. It has been emptied.

3.10.43 SLT SLT

This selects one of 8 predefined line types for use with subsequent output primitives.

Call Format

1. CALL SLT (T)

Arguments

T is the line type code. The line types are based on patterns which repeat within 16 or less pixels as follows:

Code	Pattern
0	1111111111111111
1	1010101010101010
2	1111101011111010
3	1111100011111000
4	1111111111110000
5	1111111110101010
6	1111111111001100
7	1111111100000000

Note: 1 = foreground colour, 0 = background colour.

Example

```
CALL SLT (3)
```

Sets the line type to a 5-pixel dash pattern. This is equivalent to CALL DLT ("174370).

Errors

29.(E) Bad predefined line type code.

Not in range 0-7.

3.10.44 . SOM SOM

Selects one of the four Video Output Modes (6 for Model B) by which pixel data is presented on the screen.

Call Format

1. CALL SOM (M)

Argument

M is the output mode, in the range 0-3 (0-5 for Model B).

For Model A, the exact nature of each mode depends on a hardware option. Section 3.7.3 contains all the necessary detail relating to output modes.

Example

CALL SOM (3)

Selects the 16-colour mode, which is the same on both monochrome and colour versions of Model A and on Model B.

Errors

11.(E) Invalid mode.

The mode was not in the specified range.

3.10.45 SSA -SSA

Sets the symbol attributes of horizontal pitch, vertical pitch and orientation.

Call Formats

1. CALL SSA (O)
2. CALL SSA (W,H,O)

Arguments

W is the symbol cell width, in the range 1 to 16.

H is the symbol cell height, in the range 1 to 16.

O is the orientation code. Valid codes are:

0 = Symbol array rotated 0 degrees
1 = Symbol array rotated 90 degrees
2 = Symbol array rotated 180 degrees
3 = Symbol array rotated 270 degrees

Rotation is anti-clockwise from the horizontal. Both the symbols and the "string" containing them are rotated the same way, unlike the different character orientaions set by SCA.

Changing the value of either W or H (call format 2) deletes all existing symbol definitions, as does calling SET.

When defining symbols (see SYM) the value of H defines how many words in the defining array are actually used (the first words). The value of W defines how many bits are used from each word (the least significant bits).

All argument values have configurable power-up defaults.

Example

CALL SSA(10,8,0)

Defines a symbol cell 10 pixels broad by 8 high. Orientation is unrotated.

Errors

16.(E) Invalid orientation code.

The orientation code was not in the range 0 to 3.

17.(E) Invalid cell height.

The cell height was not in the range 1 to 16.

18.(E) Invalid cell width.

The cell width was not in the range 1 to 16.

3.10.46 SYM SYM

Draws a string of symbols from the Current Position or a specified position with the current symbol attributes, in the Access Page. Alternatively defines the dot patterns for a symbol.

Call formats

1. CALL SYM (STR) (Draws)
2. CALL SYM (A,STR) (Defines)
3. CALL SYM (X,Y,STR) (Draws)
4. CALL SYM (L1,L2,L3[,L4]...[,L14],STR) (Defines)

Arguments

STR is an ASCII string containing symbol codes in the range A to Z (upper case) plus asterisk (*) or <SPACE>. For defining symbols, the string must contain only one character, in the range A-Z. The string must be terminated with a null character (PDP-11) or be a string descriptor (VAX/VMS).

Within the string, when drawing symbols, the following codes have special effect:

* Start newline of symbols. This allows a two dimensional array of symbols to be drawn. The next symbol is drawn under the first symbol on the previous line.

<SPACE> Skip one symbol position.

X and Y are the co-ordinates of the top left-hand corner of the first symbol in the array. The Index Registers are added to X and Y to determine the position at which the string is drawn.

The Current Position is updated to (X,Y) if these are specified.

A is an integer array of size 16 containing the lines of dot patterns. The first element in the array is the top line of the unrotated symbol. The least significant bit in the integer is the right-most bit of the unrotated symbol. If the symbol cell is less than 16 pixels on a side, the lower elements and/or less significant bits are used.

If the origin (0,0) is at the lower left of the screen, the first element of the array corresponds to the bottom line of the symbol, and X and Y are the coordinates of the bottom left hand corner of the first symbol in the string. In other words, if the Y axis is inverted, so is the symbol. This is in contrast to ALP and MAG which do not, under any circumstances, display text upside down.

L1, L2 etc. are line patterns expressed as binary constants. Not more than 14 patterns or less than 3 can be included in call format 4. Subject to these limits it is only necessary to supply as many line patterns as are required to define the symbol. Words not supplied are assumed to be zero. When less than 3 or more than 14 lines are necessary, the array mode (call format 2) must be used to define the symbol.

All defined symbols are deleted if SET is called or the symbol cell size is changed by SSA.

Examples

```
INTEGER*2 GRSH(16)
DATA GRSH/   48,   72,32712,36808,36936,37680,38016,38142,
+          38129,38025,37641,36873,36849,32769,32769,32766/
CALL SSA (16,16,0)
CALL SYM (GRSH,'G')
```

This defines the symbol 'G' to be the logo of a well-known group of electronics companies.

```
Call SYM (100,100,'GGGG*G  G*G  G*GGGG')
```

This displays a hollow 4x4 array of the previously defined symbol.

Errors

13.(I) Coordinate out of range.

(Displaying) - part or all of the specified symbol string was outside the page boundaries.

30.(E) Invalid character.

A character in STR is not (A-Z), * or <SPACE>. When defining a symbol, * and <SPACE> are also illegal. This error can also occur when displaying a symbol string, if the symbol corresponding to a character in the string is not defined (the symbol is displayed as a space).

38.(E) Coordinate invalid.

(Displaying) - the specified coordinates, when added to the Index Registers, generated a value outside the range -32768...32767.

3.10.47 SYN SYN

Synchronise to field - causes further commands to be deferred until after the next field flyback (screen interrupt).

Call Format

1. CALL SYN
2. CALL SYN(1)

Arguments

SYN with no arguments will cause the SBD to suspend all operations until the next field flyback on the display monitor. SYN with one argument will cause the SBD to wait until the currently requested page is actually displayed on the screen. In point of fact the argument value is totally irrelevant but it is recommended that the value one is used.

SYN is designed to assist in those situations where a picture is changing rapidly and by synchronising to the field flyback it is possible to reduce flicker and in some cases ensure that some entities are at least seen before they are erased.

The one argument form needs some explanation. On the SBD the display page is changed at field flyback time. On receipt of a PAG command the SBD sets a "New page" value and begins execution of the next command immediately. Every field flyback this "New page" value is compared to the current displayed page and if they differ the current displayed page is changed. SYN with one argument pauses whilst the "New page" value is different from the current displayed page number.

NOTE: Although the actual displayed page does not change until field flyback the logical displayed page is altered by the PAG immediately. Accordingly the following code:

```
CALL PAG(0)
CALL RUB
```

will always RUB page 0 even if, owing to scan delays, page 0 is not actually displayed until after it has been RUBbed.

Examples

It is possible to display a full VDU text page and graphics simultaneously by flicking from one to another very quickly. The objectionable flickering produced makes this more suitable for an error warning than a standard procedure. An example of code which will do this is: (nb. This code does not terminate.)

```
1000  CALL VDU
      CALL SYN
      CALL PIX
      CALL SYN
      GOTO 1000
```

A standard technique is hidden update, where one page is displayed whilst another is being drawn. Because of scan delays it is possible for a page still to be displayed when it begins to be updated. If the first update command is a RUB then this will wreck any smooth animation effect. The no argument SYN command will work perfectly here, but the one argument form will avoid wasting any unnecessary time. For example:

```
      CALL SAP(0)
     draw page 0 whilst page 1 is displayed
      CALL PAG(0)          ! Display finished page
      CALL SAP(1)          ! Prepare to redraw new page
     calculate page 1 updates and parameters
      CALL SYN(1)          ! Wait only if necessary
     draw page 1 whilst page 0 is displayed
```

In this example, by calculating as much as possible before starting to display and possibly using command blocks (see BBK and EBK) it is quite possible that a field flyback will already have taken place before the host computer is ready and that the SYN(1) is redundant most of the time. On those occasions it will merely waste a few micro-seconds checking but when it is needed it will act as a safety net.

Errors

None specific.

3.10.48 TAB TAB

Activates the Digitising Tablet (if fitted), or gets the current digitiser coordinates. Optionally causes the local pixel cursor (see CUR) to be tied to the movement of the puck or stylus on the tablet.

Call Format

1. CALL TAB (IX,IY,ID,IB,IW,IC)
2. CALL TAB (IX,IY,ID,IB,IW,IC,IT)

Arguments

IX,IY are loaded with the coordinates of the digitiser, either before or after the transformations defined by TIN (q.v.) have been applied, depending on the value of ID. Their original values are ignored.

ID is a flag indicating whether the raw digitiser coordinates are returned in IX and IY (any value except -1), or whether the returned coordinates are after application of the transformations defined by TIN (if -1). On output, ID is loaded with the Table ID (0 or 1). If the tablet is used, zero is always returned in this argument.

IB is loaded with the value of the button on the puck which was pressed (1 to 15). Various button values can be achieved depending on the type of locator device in use. A stylus can only return the value 1 (when the tip is depressed). The four-button pucks can return a value between 1 and 15 since the four buttons have the values 1, 2, 4, and 8, and pressing more than one button simultaneously adds their codes together. Values of IB corresponding to the buttons on the various puck styles are given in Table 3.13.

IW is a flag specifying whether to wait for a button to be pressed. If zero, the program continues, and the coordinates of the digitiser at the time of the call are loaded into IX and IY. If IW is non-zero, the host program waits until a button on the digitiser is pressed. The cursor (if displayed) meanwhile follows the movements of the digitiser (transformed) on the screen.

IC is a flag specifying whether a cursor is displayed while the tablet is active. If zero, no cursor appears. This facility is useful when setting up the tablet, i.e. before suitable transformations have been arrived at. If IW is non-zero, the cursor is displayed for one field time (1/50 or 1/60 second) and then erased. The pixel cursor's appearance can be controlled as described under the CUR command. Only the small cross-hair cursor, not the box cursor, is available.

IB value	4-button pucks		13-button puck	Stylus
	Summagraphics	TDS		
0	none	none	none	Up
1	yellow	red	Grey	Down
2	green	yellow	-	-
3	green+yellow	yellow+red	-	-
4	blue	green	*	-
5	blue +yellow	green +red	7	-
6	blue+green	green+yellow	4	-
7	blue+green+yellow	green+yellow+red	1	-
8	red	blue	0	-
9	red +yellow	blue +red	8	-
10	red +green	blue +yellow	5	-
11	red +green+yellow	blue +yellow+red	2	-
12	red+blue	blue+green	Hash	-
13	red+blue +yellow	blue+green +red	9	-
14	red+blue+green	blue+green+yellow	6	-
15	red+blue+green+yellow (3) (2) (1) (Z)	blue+green+yellow+red	3	-

Table 3.13 Return key values corresponding to tablet puck buttons

IT is an optional parameter. If zero, TAB will retry continuously if no data is received from the tablet within 2 seconds. The cursor, if requested, will not appear until communication is established. Normally the reason for communication breakdown is either that the auxiliary power unit is switched off, or that the puck is not in proximity to the active surface. If the condition is rectified, TAB will then continue as normal. If IT is non-zero, TAB times out after two seconds and returns an error (36). If omitted, IT defaults to zero.

All the first 6 arguments of TAB are returned. In the case of IW and IC, they are returned unchanged.

Although the two types of digitiser (tablet and table) have slightly different communication formats, the TAB routine can distinguish between them dynamically and adjust itself. The two types are not, however, distinguished on power-up.

See 4.4.3 for details of setting up the digitizers for operation with the SBD.

Because this command requires the return of data, it is performed in synchronous mode always. It cannot be archived or blocked.

Example

```
WAIT = 0
CUR = 0
ID = -1
CALL TAB (X,Y,ID,BUT,WAIT,CUR)
```

This call returns immediately with transformed coordinates in X and Y. The local pixel cursor is not displayed; presumably the user will generate his own form of cursor for visual feedback as described in 3.7.5.

Errors

26.(E) Command illegal in archive.

34.(E) Peripheral not connected.

No digitiser is present in the configuration.

35.(E) Cannot identify peripheral.

Neither of the expected communications protocols was received from the digitiser. Probably the digitiser was incorrectly set up - see 4.4.3.

36.(E) Peripheral not ready.

The digitiser timed out after two seconds (call format 2 only).

65.(E) Command illegal while blocking.

3.10.49 TIN

TIN

Initialises the Digitizing Tablet (if fitted) with parameters defining transformations to be applied to the digitizer position in order to produce the current screen coordinates (i.e. the position of the cursor if visible). The three possible transformations are scaling, translation (shifting the origin) and rotation.

Call Format

1. CALL TIN (XF,YF,XS,YS,SIN,COS)

Arguments

Unlike any other primitive in the SBD's command set, the parameters to TIN are all REAL*4 (DEC floating-point format - see 3.5.3).

XF, YF are scaling factors in the X and Y directions respectively (negative values invert the directions of the axes).

XS, YS are origin shifts in the negative X and Y directions respectively (negative values shift the origin in the positive direction).

SIN, COS are the sine and cosine of the angle of rotation (anticlockwise from the positive X direction).

The rotation is performed first, then scaling, then translation.

The Current Position is unaffected by this command. If the Digitising Tablet option is not fitted, an error is generated on a subsequent call to TAB.

The transformation equations are:

$$Xc = (Xd*COS + Yd*SIN) * XF - XS$$
$$Yc = (Yd*COS - Xd*SIN) * YF - YS$$

where (Xc,Yc) are transformed cursor coordinates,
(Xd,Yd) are the raw digitizer coordinates.

All transformation values are set to "no transformation" on power-up, reset and when SET is called., i.e. XF, YF, COS = 1.0, and XS, YS, and SIN = 0.0.

Example

```
XSCAL=FLOAT(XRES/2794)
YSCAL=FLOAT(YRES/2794)
CALL TIN(XSCAL,YSCAL,1.0,1.0,0.0,1.0)
```

Scales the TDS tablet (in Bit Pad Emulation Mode) to correspond exactly to the screen coordinates. (Tablet coordinates are from 0 to 2793 in both axes). (XRES,YRES) are the X and Y resolutions respectively (e.g. (768,574)).

Appendix B contains an example program which allows you to map any arbitrary rectangle, at any angle, from the tablet to the screen.

Errors

None specific. It is the user's responsibility to provide valid floating-point arguments.

3.10.50 TRA

Causes the trackerball to become permanently active in the background and allows sampling of the current coordinates and key-value. Works with all types of trackerball and the mouse. This allows these devices to be used as low-cost alternatives to a digitizing tablet. Other graphics commands may be executed while the trackerball/mouse is active in the background.

Call Format

1. CALL TRA(X,Y,K)

Arguments

(X,Y) return the current trackerball coordinates, or initialise them if activating the trackerball/mouse. Note: although trackerballs and mice are inherently relative positioning devices, the SBD internally updates an absolute position based on the relative movements of the transducer. These are absolute screen coordinates and are independent of the Index Registers.

K is a flag value on input, and a key value on output. The K argument is examined to determine the action of TRA:

K = -1 : activate with standard cross-hair local pixel cursor visible and tied to motions of the transducer. The values of X and Y are used to initialise the SBD's internal absolute coordinates. X and Y are returned unchanged (unless out of range in which case they are set to the exceeded limit) but K is set to zero. If sample mode is already active, the SBD's internal absolute coordinates are overwritten by X and Y, and the cursor made visible or invisible according to the value of K.

K = other negative value: as for K = -1, but without any cursor visible. It is assumed that the user will provide his own method of visual feedback.

K = zero or any positive value: the SBD's internal absolute coordinates overwrite X and Y, and the value of any button pressed is returned in K (zero if no button depressed). A warning is generated if sample mode is not active, and the action is then the same as for K=-1.

While this "sample mode" is active, the trackerball(s) and/or the mouse are used to adjust the internal absolute coordinates. This proceeds entirely locally to the SBD and nothing is returned to the host except when requested by TRA. If visible, the SBD's local cursor "tracks" the internal coordinates as they change.

Note that if you enter SETUP (see 2.3) and leave it, or call SET, while the sample mode is active, then sample mode is terminated (subsequent calls to TRA with positive K will re-enable sample mode with a warning and visible cursor). This does not apply to Terminal Setup (see 9.5) unless the RESET key is pressed.

The keypad keys on the keyboard also generate key values in K in this mode, and the Direction Keys are used to move the cursor. None of these keys send data to the host during sample mode. Apart from this, the VDU is operative as normal while in sample mode.

If no trackerball or mouse is connected (via a PCU), then whenever a keypad key is pressed, that key value (see Table 3.6 under CUR) is returned in K for all subsequent calls to TRA, until another keypad key is pressed. To revert to zero in K, the zero key on the keypad must be pressed. In other words, pressing a key once on the keypad is equivalent to holding a button down on the trackerball or mouse. This allows you to use a trail mode even if no transducer is available, with the slight drawback of having to press another key (rather than simply release a button) to terminate the mode. If a trackerball or mouse is present, keypad keys generate a single K value (but only if you happen to call TRA immediately afterwards and before the next transducer value is received - these occur every 80 milliseconds). Even if you hold down a keypad key, causing it to autorepeat, only about 2 out of every seven calls to TRA, on average, will return the correct key value. K always reverts to zero if no transducer button is being held down. It is therefore not wise for your application to rely on K values generated by the keypad alone (i.e. greater than 7) if a trackerball or mouse is likely to be used.

Note: when sampling continuously, the key value in K cannot be guaranteed to change until about 200 msec after the button(s) are pressed or released. In practice, for most applications this will not be noticeable. Furthermore, when several buttons are pressed in combination it may take many milliseconds before the value of K stabilises on the desired value. Your application should therefore take several samples before taking action based on a key value, if combinations are meaningful. Otherwise, the application must be designed so that useful state transitions are achieved by single button depressions or releases, e.g. from 2 to (2+1), but not from zero to (2+1).

When the SBD's pixel cursor is made visible by TRA, it is blanked while any commands are being executed. This may cause the cursor to flicker slightly, but is necessary in order to avoid leaving 'shadows' of the cursor behind if a primitive is drawn through it. Consider also the possibilities for disaster in the PAG command. The local cursor is drawn in the Access Page, not the Display Page (contrast CUR), and so is not guaranteed to be visible.

If the standard cursor is not employed, users may draw their own cursors based on sampled values of X and Y, as described in 3.7.5. In this case, users must take appropriate steps to avoid drawing primitives through their cursors, and to ensure that the Access and Display pages are the same at the time the cursor is drawn.

Commands which use other peripherals (e.g. SDM, DMP and TAB) can still be used without disabling sample mode. If TAB is called with a visible cursor, and a pixel cursor is already visible from the last TRA call, then the pixel cursor is under the control of the tablet until the command is terminated, when the transducer regains control of it.

Trackerball sample mode is terminated by calling TRO, CUR or SET, or by resetting the SBD.

Because this command requires the return of data, it is always performed in synchronous mode. It cannot be archived or blocked.

Example

```
X = 100
Y = 200
K = -1
CALL TRA (X,Y,K)
```

Activates trackerball sample mode with a visible cursor at (100,200). Movements of the available transducers then move the cursor around the screen regardless of other activity on the board.

Errors

13.(I) Coordinate out of range.

The coordinates which were out of range are assumed to be zero.

26.(E) Command illegal in archive.

34.(E) Peripheral not connected.

No trackerball or mouse was available in the current configuration, or else the PCU did not respond on power-up.

45.(W) Peripheral not active.

Attempt to call TRA with positive value in K when sample mode is off. K is assumed to be -1 and is set to zero.

65.(E) Command illegal while blocking.

3.10.51 TRO TRO

(Model B only.) Cancels the background cursor mode activated by TRA.

Call Format

1. CALL TRO

Arguments

None. The trackerball / mouse is deactivated.

Errors

45.(W) Peripheral not active.

The trackerball sample mode was not active.

3.10.52 TXT

Writes the scrolling VDU text store with characters.

Call Format

1. CALL TXT (A)

Argument

A is defined as a byte array ending with a null, or as an ASCII string. (For VMS, a string descriptor.)

The first byte is treated in a similar way to a FORTRAN carriage control character, coded as follows:-

<SPACE>	<LF> before string, <CR> after.
0	2 <LF>s before string, <CR> after.
1	Clear VDU area, cursor to (1,1), string followed by <CR>.
+	String followed by <CR> (no <LF>)
$	<LF> before string.(no <CR>)

Subsequent bytes until the terminal null are treated exactly as if they were VDU characters received from the host serial line. Note that if the ASCII introducer appears in the TXT string it will be treated as a character not as an introducer. This means that this is a long winded but effective way of getting the ASCII introducer on the VDU screen. This scrolling text is always written to the current VDU cursor position.

Escape sequences embedded in the text have their normal effect on the VDU.

Example

```
CALL TXT('$No <CR> after this :')
```

Errors

None specific.

3.10.53 VDU VDU

Switches into view 24 rows of scrolling text and no pixel data
or
Saves and restore current VDU screen.

Call Formats

1. CALL VDU
2. CALL VDU(FLAG)

Arguments

1. CALL VDU

The no argument form causes the SBD to display 24 lines of VDU text on its display monitor and no graphics. The actual primitive sets a flag requesting 24 lines of VDU and exits immediately. The change in screen format is only actioned at the time of the next field flyback.

After a VDU command the SBD is in VDU mode and will automatically go into MIX mode (ie execute a MIX with no arguments) if a graphics primitive such as VEC or CIR is executed. There is more information on this in the chapter on MIX where the difference between VDU and MIX(24) is explained.

2. CALL VDU(FLAG)

This is an entirely different command from VDU with no arguments. VDU with one argument saves or restores the VDU page. If a SET command is given with a resolution code modifier of 200 then approximately 8K words of the archive space (see 3.3.3 for exact amounts) are set aside as a second VDU page. The details of the SET command are described in its own section.

There are three valid values of "FLAG" giving the commands:

CALL VDU(0) Save VDU page
CALL VDU(1) Restore VDU page
CALL VDU(-1) Swap back VDU page with the current VDU page

Although at first glance the VDU(n) commands resemble the PAG commands they are entirely different in concept and are, in fact, closer to the ZOO commands.

There are two VDU pages, the front page and the back page. Only the front page is ever displayed. A VDU(0) command copies the front page onto the back page overwriting any previously saved data. Similarly a VDU(1) command, in restoring the saved data, destroys the current text screen information. Only a VDU(-1) command preserves

all the information (cf. ZOO(-1)).

NOTE: allocating the second VDU page with SET does not create a readable blank hidden VDU page. It is necessary to execute a VDU(0) command to save some data before either a VDU(1) or a VDU(-1) may be performed.

When a screen is saved or restored the current cursor position, scrolling regions and character attributes are transferred with it.

Errors

5.(E) Too many pages.

An attempt has been made to save or restore a VDU page without the hidden page having been allocated by SET. The VDU command is not executed.

40.(E) Nothing saved.

An attempt has been made to restore a VDU page or swap VDU pages without first saving a page. The VDU command is not executed.

42.(E) Invalid argument

VDU(n) was called where n was neither 1, 0 nor -1

3.10.54 VEC

This command draws a vector in the current line type, in the Access Page.

Call Formats

1. CALL VEC (X2,Y2)
2. CALL VEC (X1,Y1,X2,Y2)

Arguments

(X1,Y1) and (X2,Y2) are coordinates in the current Access Page. The contents of the Index Registers are added to the coordinates to determine the location at which the vector will be drawn.

In call format 1, the vector is drawn from the Current Position to (X2,Y2). In call format 2, the vector is drawn from (X1,Y1) to (X2,Y2). Chained vectors can therefore be drawn more efficiently using call format 1.

The Current Position becomes X2,Y2.

Example

```
CALL VEC (0,0,767,0)
CALL VEC (0,573)
CALL VEC (0,0)
```

These commands draw a triangle enclosing half the screen area.

Errors

13.(I) Coordinate out of range.

Part or all of the vector was outside the screen boundary.

38.(E) Coordinate invalid.

The specified coordinates, when added to the Index Registers, generated a value outside the range -32768..32767.

3.10.55 WIB WIB

Writes a two-dimensional array of pixels to the current Access Page with data from a host array.

Call Format

1. CALL WIB (X1,Y1,X2,Y2,P,A)

Arguments

(X1,Y1) and (X2,Y2) are the diagonal corners of the screen array. P is the data packing mode specifying the number of pixel values packed into each word of the host array, A.

P is provided for compatibility with other Supervisor display systems. Its value must be 4, indicating that four pixel values are packed into each word. The leftmost/uppermost pixel on the screen occupies the least significant four bits of the word. All four planes are written to, regardless of the currently selected channel.

The WIB and RIB (q.v.) commands are different from the other commands because the library does not build a command block using space within the library. Instead it writes the command data into the first 8 locations (and the location after the last pixel word) of the user array, A. This avoids the need to laboriously copy the pixel data into a large buffer within the library thus saving space and time. The user must ensure that the first 8 elements in the array are free for command data and that the pixel data begins in element 9. Alternatively the user can write command data into the array, following the usual command block syntax, and use the EGB command instead of WIB or RIB. The number of elements of A used is given by A(8)+9, or A(1)/2.

The Index Registers are added to X1, Y1, X2, Y2 to determine the actual corner positions.

The Current Position becomes X2,Y2.

This command may only be used via the serial line if the Binary transmission format is used. The library must be configured to support these commands on the serial line. An error is generated if the ASCII format is in use.

Note that the maximum size of a command block (see Chapter 5) is 32767 words, so the largest number of pixels that can be read/written with one command is 4*(32767-9), i.e. 131032. At the maximum resolution (768x586) this implies only 170 pixel lines at the full width, so a minimum of 4 separate commands would be required to read/write the whole screen. Less commands would be required at lower resolutions. In practice, since the host library does not

support virtual arrays, the size of a command block on a 16-bit machine such as a PDP-11 is limited to about 20000 words when using FORTRAN, so more commands will be needed to read or write an entire screen.

When using WIB or RIB over a serial line, the size of the command block is limited to 127 words owing to the limitations of the binary protocol. Since 7 words are used by the command arguments, only 120 words are available for pixel data and therefore only 480 pixels can be transmitted by a single command. (See also RIB.)

Example

```
    INTEGER*2 PIXDAT(2509)
    .
    .
    DO 10 I=9,2508
    READ (1,*) PIXDAT(I)            ! unformatted read
    PIXDAT(I) = (PIXDAT(I).AND."73567)
10  CONTINUE
    CALL WIB(0,0,99,99,4,PIXDAT)
```

Reads the pixel data out of a file into the correct parts of the array PIXDAT, clearing the top bit of each pixel, then copies the data to the screen.

Errors

13.(I) Coordinate out of range.

The specified coordinates, when added to the Index Registers, defined a pixel block which was not completely on the screen.

26.(E) Command illegal in archive.

38.(E) Coordinate invalid.

The specified coordinates, when added to the Index Registers, generated a value outside the range (-32768...32767).

65.(E) Command illegal while blocking.

The WIB array is a self-contained command block and cannot be copied into another block.

84.(E) Image block too large

The rectangle bounded by the corner coordinates contained more than 131032 pixels.

85.(E) Binary protocol not configured for imaging support.

The library was not configured to support WIB and RIB via the serial binary protocol.

86.(E) Imaging routine not translateable to ASCII.

Attempt to send WIB or RIB via the ASCII protocol. This error also appears on the host terminal when any other error is detected in WIB or RIB, since the library attempts to translate the offending command and display it in ASCII on the user's terminal.

3.10.56 ZOO ZOO

Copies a block of pixels to a destination block with optional scaling.

ZOO is a particularly versatile and complex command. There are eleven forms with numbers of arguments varying from zero to 10. The basic elements of the command are a source rectangle, a destination rectangle, a source page, a destination page and scaling factors in the X and Y directions. By using different forms of the command, the various elements can be defaulted or specified.

Call Formats

1. CALL ZOO (XS1,YS1,XS2,YS2,XD1,YD1,XD2,YD2,PS,PD)
2. CALL ZOO (XS1,YS1,XS2,YS2,XD1,YD1,PS,PD,M)
3. CALL ZOO (XS1,YS1,XS2,YS2,XD1,YD1,XD2,YD2)
4. CALL ZOO (XS1,YS1,XS2,YS2,XD1,YD1,M)
5. CALL ZOO (XS1,YS1,XS2,YS2,PS,PD)
6. CALL ZOO (XS1,YS1,PS,PD,M)
7. CALL ZOO (XS1,YS1,XS2,YS2)
8. CALL ZOO (XS1,YS1,M)
9. CALL ZOO (PS,PD)
10. CALL ZOO (M)
11. CALL ZOO

(XS1,YS1) and (XS2,YS2) are the coordinates of opposite corners of the source rectangle,

(XD1,YD1) and (XD2,YD2) are the coordinates of opposite corners of the destination rectangle,

PS and PD are the source page and destination page respectively,

M is the magnification factor (integer) in both X and Y directions.

Arguments

NOTE: the contents of the Index Registers have no effect on the source and destination rectangle coordinates. The source and destination rectangles need not lie entirely within the pixel page. However, only those parts of the source rectangle which do actually lie in the source page will be used, and only those parts of the destination rectangle which lie within the destination page will be written. Areas of the destination rectangle which lie within the page boundaries, but correspond to areas of the source rectangle which lie off the source page, will not be changed. If part of either source or destination lies outside a page boundary, a warning is produced.

If M, the magnification factor, is negative, the actual value is ignored and the source and destination rectangles are exchanged. The facility for swapping blocks of pixels is particularly useful for implementing pop-up menus (see 3.7).

M cannot be fractional; for scaling down, or scaling up by fractional amounts, the source and destination rectangles should be chosen accordingly. Valid values for M are therefore -32768 to -1 and 1 to 32767. In practice, values greater than 10 or so are of little value.

Scaling up is done by pixel replication (no additional detail is produced). Scaling down is by pixel sampling, so fine detail such as text may well become obscured. However, images without fine lines tend to remain recognisable even with severe reduction.

Different forms of the command assume varying default values for the unspecified arguments. The default values assumed are summarised by Table 3.14.

	-------Source-------					-----Destination----					-Scaling-	
Narg	Xs1	Ys1	Xs2	Ys2	Ps	Xd1	Yd1	Xd2	Yd2	Pd	Mx	My
10	Xs1	Ys1	Xs2	Ys2	Ps	Xd1	Yd1	Xd2	Yd2	Pd	Calculate	
9	Xs1	Ys1	Xs2	Ys2	Ps	Xd1	Yd1	Calculate		Pd	M	M
8	Xs1	Ys1	Xs2	Ys2	Dp	Xd1	Yd1	Xd2	Yd2	Ap	Calculate	
7	Xs1	Ys1	Xs2	Ys2	Dp	Xd1	Yd1	Calculate		Ap	M	M
6	Xs1	Ys1	Xs2	Ys2	Ps	0	0	Xlim	Ylim	Pd	Calculate	
5	Xs1	Ys1	Calculate		Ps	0	0	Xlim	Ylim	Pd	M	M
4	Xs1	Ys1	Xs2	Ys2	Dp	0	0	Xlim	Ylim	Ap	Calculate	
3	Xs1	Ys1	Calculate		Dp	0	0	Xlim	Ylim	Ap	M	M
2	0	0	Xlim	Ylim	Ps	0	0	Xlim	Ylim	Pd	1	1
1	0	0	Calculate		Dp	0	0	Xlim	Ylim	Ap	M	M
0	0	0	Xlim	Ylim	Dp	0	0	Xlim	Ylim	Ap	1	1

Table 3.14 Default values for different forms of ZOO

Key:

Dp is the current Display Page
Ap is the current Access Page
Xlim, Ylim are the maximum X and Y coordinates at the current resolution (e.g. 767 and 573 if the current resolution is 768x574).

One form of the ZOO command which is frequently used is magnifying by a factor of 2. For this reason, a faster technique (using table lookup) is used for ZOO if the following conditions are met:

- magnification factor (explicit or calculated) is 2
- source rectangle X start is a multiple of 16
- source rectangle X width is a multiple of 16
- destination rectangle X start is a multiple of 16.

Under these conditions, writing speeds (per destination pixel) of approximately 1.8 microseconds (Model A) or 1.2 microseconds (Model B) can be achieved. The corresponding figures for an "unaligned" zoom are 8.9 and 5.9 microseconds respectively, so there is a five-fold increase in speed. For example, if zooming a 128x128 rectangle up to 256x256 on Model B, it is possible to do this 12 to 13 times per second (faster if the ZOO commands themselves are archived, since this reduces the DMA access time).

Examples

```
CALL ZOO (-1)
```

This exchanges the Access Page and the Display Page.

```
CALL ZOO (100,100,199,199,200,100,249,349)
```

This copies a block of pixels from a rectangle 100 pixels on a side whose upper left corner (assuming conventional Y-axis direction) is at (100,100) to a rectangle whose upper left is at (200,100), i.e. adjacent to the source rectangle, and which is 50 pixels in the X direction and 250 pixels in the Y direction. The X and Y magnification factors are calculated by ZOO: the X magnification is 0.5 and the Y scale factor is 2.5. This means that half the vertical columns of pixels (taken alternately) are moved from source to destination, and the other half are not copied. The rows of pixels are replicated by 2.5, in other words the first row is repeated 3 times, the second twice, and so on.

The source and destination pages in this example are defaulted to the Display Page and the Access Page respectively.

```
CALL ZOO (385,288,767,573)
```

This expands the bottom right-hand quarter of the Display Page (resolution 768x573) into the whole Access Page, thereby magnifying both X and Y by 2. Note that this still works even if the Display Page and the Access Page are the same (though a warning is produced).

Such an expansion, where source and destination rectangles overlap and the source is entirely contained within the destination, would probably not work correctly if the maximum Y coordinate of the source was less than the maximum Y of the destination. This is because the pixel movement is done row by row starting with the lowest Y value, and therefore the chances are that the higher Y rows of the source would have been overwritten before they could be read in.

On each pixel row of the source, the pixels with lowest X value are examined first, in normal raster fashion. Therefore, an overlapping zoom in which the source rectangle was against the right-hand edge (highest X) of the page, but not on the bottom (highest Y), would work correctly only if there was no expansion in the Y dimension, and the destination was higher up (smaller Y) or level with the source.

You can achieve some interesting effects by experimenting with overlapping areas. Consider, for example, the following:

```
CALL ZOO (0,0,767,573,384,287,767,573,0,0)
```

This copies the whole screen into the bottom right quadrant of the same page (page 0). The effect observed is an "infinite" regression of images, and can be used to determine what reduction in scale is possible before an image becomes unrecognisable.

Errors

5.(E) Too many pages.

Page number larger than the maximum permitted.

13.(I) Coordinate out of range.

One or more corners of source or destination rectangle are outside the screen boundaries.

37.(E) Invalid magnification factor.

The value of M is zero.

38.(E) Coordinate invalid.

One or more corners of the destination rectangle are outside the range (-32768 to 32767). This can only occur for calculated coordinates, e.g.

```
CALL ZOO (0,0,500,300,0,0,20000)
```

39.(I) Overlapping areas.

The source and destination rectangles overlap and are on the same page. The results of the copy may therefore differ from that expected.