PLUTO USER MANUAL

IO RESEARCH (COMPUTER GRAPHICS)
4 EXCHANGE BUILDINGS
HIGH STREET
BARNET
HERTS EN5 5SY

Tel : 01 441 5700

Telex: 265871 - MONREF G
Quoting reference IOR001

CONDITIONS

Every effort is made to ensure that the information contained in this manual is correct. IO RESEARCH LIMITED make no representations or warranties with respect to the contents hereof and do not accept liability for any errors or omissions.

Any particular release of the product (PLUTO) may not contain all the facilities described in this manual. New (or amended) releases of the product may contain extra facilities which are not detailed in this manual. IO RESEARCH LIMITED will, and if requested, supply additional information about any extra facilities supplied with a user's particular release of the product.

PREFACE

The Pluto graphics controller is available in a number of individual packages: Pluto I (+Mini-Palette (optional), Pluto for the Sirius, Pluto for the IBM PC, Pluto II, Pluto II 24 Bit System and the Megares System. There are also Pluto I and Pluto II Interface cards for the following microcomputers: Sirius, Apricot, S100, RML, Apple, IBM P.C, Q-Bus and Nimbus. The support documentation consists of a Pluto User Manual and a Command Summary Reference Card. The Pluto User Manual describes each of the Pluto packages and the variables and commands that can be used in programming. The Command Summary Reference Card lists all the Pluto commands and cross-refers each command to it's appropriate Pluto package.

An interface library for use with high level languages is also supplied as part of the Pluto package. A summary of the interface calls can be found in Appendix F. Throughout the manual Pluto function calls are shown in two forms - for raw driving of Pluto and in a PASCAL format using the high level language interface.

TABLE OF CONTENTS

3. PLUTO RS232 PIN OUTS

4. INTERFACING PLUTO WITH NON 80 BUS SYSTEMS

5. INSTALLING IBM PLUTO

6. PLUTO II INSTALLATION NOTES

7. PLUTO II 24 BIT INSTALLATION NOTES

8. MEGARES INSTALLATION NOTES

9. PLUTO APRICOT INSTALLATION

10. PLUTO NIMBUS INSTALLATION

11. PLUTO II ENCLOSURE

SECTION ONE - INTRODUCTION TO PLUTO

Pluto is an intelligent colour graphics controller which can be interfaced with almost any computer. Software in ROM gives Pluto many inbuilt facilities which can be accessed through a high level command language.

The on-board graphics facilities provided by Pluto are extensive and include: Point set and inquire, Rectangle fill, Vector draw with definable dot-dash pattern, an in-built character font, arcs and circles with definable dot-dash patterns, definable shapes and symbols storable on board, raster operations (which include a rotate facility), image and symbol read and load, individual colour plane write protection, flood fill, boundary fill, pattern fill and shading. Also included with the Pluto II package are the SZoom command (which allows the screen picture to be magnified up to 16 times), the PanTo command (which can be used to move an image around the screen horizontally and vertically to reveal hidden parts) and an optional video frame grabber which allows the capture of an image from a video camera. The 2 versions of the Megares package also include the SZoom command (which allows the screen picture to be magnified up to 9 times) and the PanTo command.

There are several individual Pluto packages:

- Pluto I (+ Mini-Palette - optional)
- Pluto for the Sirius microcomputer
- Pluto for the IBM Personal Computer
- Pluto II
- Pluto II 24 Bit System
- Megares (Versions 1 and 2)

Each individual package has unique features and commands, and has screens of different resolution available.

The Pluto I system is contained on a 8" x 8" board which includes an 8088 processor, 256 KBytes of memory and the graphics routines in ROM. The screen resolutions available for the Pluto I system are: 2 screens of 640H x 288V (Non-Interlaced) or 1 screen of 640H x 576V (Interlaced). On a high resolution Pluto I the resolutions are 768H x 576V (Interlaced) or 2 screens of 768H x 288V (Non-Interlaced). For the Pluto I system, four bits of memory are used for each pixel, and each pixel can be independently set to one of sixteen colours. The Pluto I system is described in Section 2 of this manual.

A Mini-Palette can be added to the Pluto I system by Io Research Limited. The Mini-Palette, includes a Look-Up Table (LUT) which allows each of the sixteen basic Pluto colours to be re-defined from a selection of 4096 colours.

Pluto for the Sirius microcomputer is contained on a 9" x 5" multilayer Sirius board which includes an 8088 processor, 256 KBytes of memory and the graphics routines stored in ROM. The screen resolution is 768H x 576V (Interlaced) and can also be software selected to two displayable screens of 768H x 288V. Each pixel can be independently set to one of sixteen colours. All the commands available for Pluto I are also available for the Pluto Sirius system. Pluto for the Sirius is described in Section 3 of this manual.

Pluto for the IBM Personal Computer is contained on a six layer IBM format board which includes a 68000 processor, either 256 KBytes or 384 KBytes of memory (depending on the version) and the graphics routines stored in ROM. There are six hardware configurable versions available: 768H x 576V (Interlaced) 16 colours, one displayable screen), 768H x 576V (Non-Interlaced, 8 colours, 2 screens) and 1024H x 768V (Interlaced, 8 colours), 768H X 576V (interlaced, 16 colours, 4 screens), 768H X 576V (Non-interlaced, 8 colours, 8 screens), 1024H X 768V (Interlaced, 8 colours, 4 screens).

All the commands available for Pluto I are also available for the IBM PC, as well as a fast smooth scroll (vertical pan) capability and the facility to load a macro or user defined code. Pluto for the IBM PC is described in Section 4 of this manual.

The Pluto II system is contained on a 12" x 8" board which includes an 8088 processor, 512 KBytes of memory and the graphics routines stored in ROM. The screen resolutions are 768H x 576V (Interlaced) or 768H x 288V (Non-Interlaced). The memory can be expanded to one Megabyte which allows two displayable screens of 768H x 576V resolution or one screen of 768H x 1152V resolution. All the Pluto I commands are available with Pluto II and there are also extra features: Pan and Zoom commands to magnify the screen image and move the image around the screen, a serial port, the facility to load a macro or user defined code, a Palette which allows a 256 colour choice from a Palette of 16.7 million and an optional real-time frame grabber. The Pluto II system is described in Section 5 of this manual.

The Pluto II 24 Bit system is a special configuration of three Pluto II boards synchronised together to give a full 24 bit per pixel system. An optional real-time colour frame grabber can also be configured to this system. The Pluto II 24 Bit system is described in Section 6 of this manual.

The Megares system is contained on an 8" x 10" board which includes an 8088 processor, 512 KBytes of memory and the graphics routines stored in ROM. There are two versions of the Megares system available: Version one has a screen resolution of two screens of 768H x 576V (Non-Interlaced), and version two has screen resolutions of 1024H x 768V (Interlaced) or two screens of 1024H x 384V. Each version uses four bits per pixel. All the Pluto I commands are supplied with the Megares system and there are also extra features: Zoom and Pan commands to magnify the screen image and move the image around the screen, and a Palette which allows a sixteen colour choice from a Palette of 4096. The Megares system is described in Section 7 of this manual.

As well as the various Pluto packages available, there are Interface cards which can be used as a communications link between a Pluto graphics card and certain microcomputers. Interface cards are available for the following microcomputers: Sirius, Apricot, S100, RML 380z, Apple, IBM PC, and Q-Bus and RML Nimbus. The Interface cards for each

machine are detailed in Section 8 of this manual.

1.1 Basic Information

Each Pluto package has a large amount of memory (the FRAME BUFFER) for storing images. The frame buffer, the size of which varies between each Pluto system, is divided into a number of rectangles (PARTITIONS). Each of these partitions has the same width and is allocated a unique identifier. One (or more) of these partitions can be pre-allocated for storing a screen image which can be viewed on a monitor screen.

The remaining portion of the frame buffer is free for use as workspace or symbol storage and is allocated in partitions, as required. These are called SYMBOL PARTITIONS. When used for symbol storage, a Symbol Partition holds a number of equally sized user-specified symbols (rectangular pixel arrays). A symbol is uniquely identified by a partition identifier together with a symbol number.

One Symbol Partition (partition 255) is pre-allocated and contains the 96 standard printable ASCII characters in an 8H x 10V pixel matrix. Symbols are normally numbered from 0 to n-1, where n is the number of symbols in the partition (maximum 128). The pre-defined character set is an exception to this rule as the symbol numbers range from 32 to 127 to correspond with the ASCII character code.

The frame buffer consists of many tiny picture elements (POINTS or PIXELS) which are arranged in a rectangular matrix (a RASTER). Each pixel may be individually assigned a number which defines it's colour. To produce a screen image, a portion of the frame buffer is scanned onto a monitor screen (a RASTER SCAN).

The number of bits of memory used for each pixel is dependent on the particular Pluto package being used. For example, the basic Pluto I system uses four bits per pixel allowing a choice of sixteen colours. In this instance the four bits provide sixteen combinations for any pixel which, when connected to the correct colour monitor with four TTL level inputs, produce sixteen different colours. On other Pluto systems, for example the Pluto II system, eight bits of memory are used per pixel allowing a choice of 256 colours.

The frame buffer can be thought of as consisting of parallel one-bit-per-pixel frame buffers fixed together. These are referred to as the colour planes. Single planes, which increase the amount of storage, are very useful for storing symbol or shape masks. Colour and monochrome images may be used together as they are stored in the same way, only the way they are interpreted is different.

When displayed on a monitor, a partition has it's origin (X=0, Y=0) at the top left corner of the screen. X co-ordinates increase from left to right and Y co-ordinates increase from top to bottom.

SECTION TWO - PLUTO I

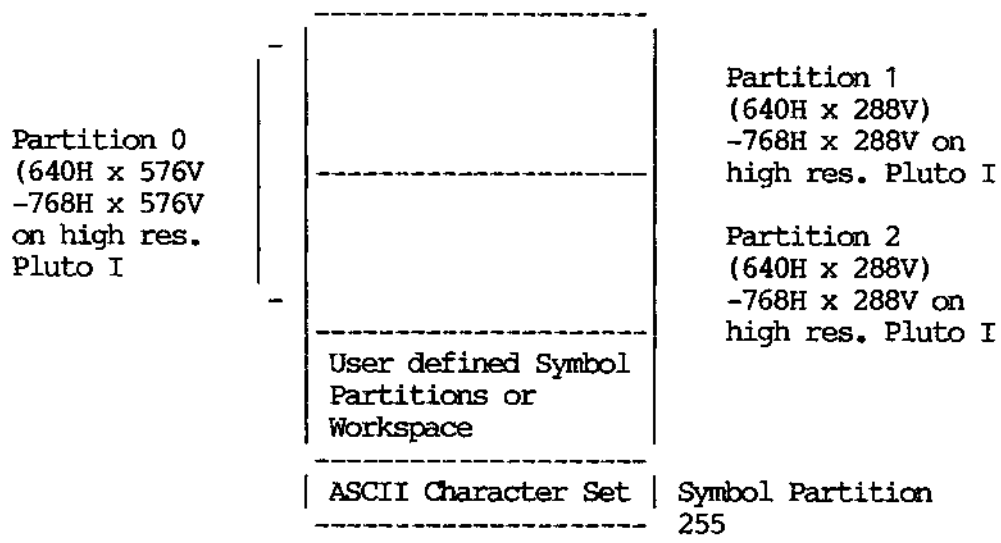
2.1 Introduction

The Pluto I graphics controller is contained on a 8" x 8" board which includes an 8088 microprocessor and 256 KBytes of memory and a serial interface. Pluto I gives screen resolutions of two screens of 640H x 288V pixels (Non-Interlaced) or one screen of 640H x 576H pixels (Interlaced). A high resolution Pluto I gives screen resolutions of one screen of 768H x 576V (Interlaced) or two screens of 768H x 288V (Non-Interlaced).

Note that this section describes the revised Pluto I as of June 1986. For previous versions of Pluto I refer to the manual supplied when the board was purchased.

As a single board Pluto I produces an eight colour display. An expansion slot on the board allows a colour palette to be added, enabling the display of a wide range of colour shades.

Pluto I has a frame buffer 640 pixels wide by over 800 high. The frame buffer is divided into partitions each of which has a width of 640 pixels and each is allocated a unique identifier. Partitions one and two (for storing screen images) each have a height of 288 pixels and can be viewed on a monitor screen. There is also the option to display the two partitions together providing double the vertical viewable image (partition 0) in a 640H x 576V display.



With the Pluto I system, four bits of memory are used to store each pixel, providing sixteen possible colours per pixel. The three bits control the green, blue, and red colour components.

2.2 Getting Started

With all connections made and Pluto I powered up, you are now ready to begin implementing the Pluto commands to create programs.

Note all the Pluto procedures described are in two forms - raw access to Pluto and in PASCAL using the high level language interface.

2.3 Pluto I State Variables

There are a number of State variables within Pluto I which define the current working mode. State variables are used to minimise the number of parameters that are needed with each command.

When Pluto I is first powered up, or after the PInit procedure is invoked (see Housekeeping commands, Section 2.4.3), the State variables are initialised with useful default values. In the following sections the default values are shown in parentheses after the variable name. All variables except the Current Position (CP) are single byte quantities.

2.3.1 Colour variables

As Pluto I uses three bits per pixel, only eight colours (0 to 7) can be stored. The values for these eight colours are as follows:

0 - Black
1 - Green
2 - Blue
3 - Cyan
4 - Red
5 - Yellow
6 - Magenta
7 - White

CCOL (7)

The Current Colour for which the default is white. The Current Colour variable can be used with most drawing and colour filling commands.

BCOL (0)

The Background Colour for which the default value is black. The Background Colour variable can be used for symbol (e.g. an ASCII character) and raster copying.

FCOL (7)

The Foreground Colour for which the default is white. The Foreground Colour variable can be used for symbol and raster copying.

TCOL (0)

The Transparent Colour for which the default is black. The Transparent Colour variable can only be used with PAINT operations (see Copy styles, Section 2.3.5.3) during raster copying.

PCOL (7)

The Perimeter (boundary) colour for which the default is white. The Perimeter Colour variable can be used to define the boundary for all boundary fill commands.

2.3.2 Partition and Workspace variables

CWP (1)

The Current Working Partition (CWP) for which the default is 1. The Current Working Partition defines the portion of the frame buffer within which commands are to be performed. Most co-ordinates are specified relative to this partition which allows a common set of Pluto routines to be used. The values available are:

0 - for working in Screen 0 - a 640H x 576V area, Interlaced (768H x 576V Interlaced on high resolution Pluto I)

1 - for working in Screen 1 - a 640H x 288V area, Non-Interlaced (768H x 288V Non-Interlaced on high resolution Pluto I)

2 - for working in Screen 2 - a 640H x 288V area, Non-Interlaced (768H x 288V Non-Interlaced) on high resolution Pluto I)

Other values may be used for working in user-allocated workspace partitions.

Note: When switching between the Interlaced display (640H x 576V or 768H x 576V) and the Non-Interlaced display (640H x 288V or 768H x 288V) with the SHires and SLores commands (see Housekeeping commands, Section 2.4.3), the Current Working partition will NOT be updated as appropriate. The Current Working Partition can be set, depending on the display, with the SCWP command (see Housekeeping commands, Section 2.4.3).

CSP (255)

The Current Symbol Partition for which the default is 255. (Current Symbol Partition 255 contains the standard ASCII character set). Most symbol operations are specified relative to this partition. Partitions can be allocated for different text fonts, macros and user-defined code.

CDP (1)

The Current Display Partition (CDP) for which the default is 1. The Current Display Partition defines what is displayed on the screen. The values available are the same as for the Current Working Partition (CWP) - 0 for a 640H x 576V (or 768H x 576V) display and 1 or 2 for one of the 640H x 288V (or 768H x 288V) displays (top and bottom of the full screen display respectively).

Note: The working partition and the display partition can be set differently. It is possible to work in partition 1 (CWP=1) while displaying partition 2 (CDP=2). The SHires and SLores commands (see Housekeeping commands, Section 2.4.3) are both extensions to the SCDP command, i.e. they only affect the display partition.

After the SLores command has been issued, the SCDP command may be used to switch into screen partition 1 or 2, but not screen partition 0. To display the full screen (0), the SHires command must be issued. Once the SHires command has been issued, the SCDP command **CANNOT** be used to switch display partitions (an ISTAT error code of 10 - Illegal partition - will be returned).

CP (0,0)

The current position (CP), for which the default is the top left corner of the screen, as X and Y co-ordinates. Most commands use the Current Position as a starting point. The Current Position can be moved and set with either the MoveTo, MoveRS command (see Current Position commands, Section 2.4.4).

2.3.3 Miscellaneous Variables

STATUS (0)

The STATUS variable records the result of the most recent command. A value of 0 signifies a successful operation and any other value signifies an error (see Pluto I Error Codes, Section 2.5).

PAT (240 = 0f0 hex)

The pattern optionally used for line and arc commands. The default is 240 in Decimal, 11110000 in Binary. This default pattern, with four pixels set and four not set, procedures the effect of a dashed line.

2.3.4 Single Colour Plane Operators

WPROT (0)

The Write Protect mask. This eight bit number uses one bit per colour plane to selectively protect colour planes from being modified. With the sixteen colour system of Pluto I, bit 0 (the least significant bit) is used to protect the green plane, bit 1 the blue plane, bit 2 the red plane. Bits 3 to 7 are not used. Setting a bit protects the corresponding colour plane. The WPROT variable can only be used with the Copy, LImagC and LSymC commands (see the Raster Copying commands, Section 2.4.8 and the Image and Symbol Read and Load commands, Section 2.4.10).

RSEL (7)

The Read Select mask. This is the opposite of WPROT and selects colour planes for use as the source of an operation. RSEL has the effect of converting a monochrome (one bit per pixel) image stored in a single colour plane into a two colour image. RSEL has its main use with text and symbol displays. Each plane can hold a different set of symbols increasing the total symbol storage space threefold.

As with WPROT an eight bit mask is used on a bit-per-plane basis to select planes from which to read an image. RSEL is normally used to select a single plane, but can be used to select multiple planes. Pixels in the selected plane(s) are converted to either the foreground (FCOL) or background (BCOL) colour before being used in an operation. The pixel value read from the frame buffer is logically ANDed with RSEL - if the result is not zero then FCOL is used otherwise BCOL is used. As an example, 1 would be the appropriate value for RSEL to read an image from the green plane.

2.3.5 Style

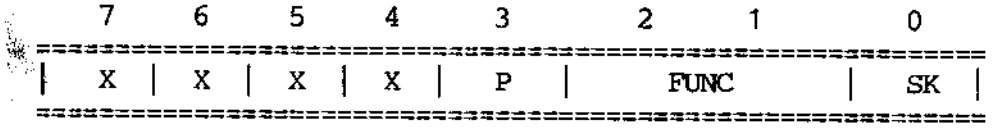
The STYLE variable for which the default is 128 in decimal (80 in hex). The STYLE variable modifies the effect of most commands. Using a single variable in this way minimises the amount of setting up required before using a command. The default STYLE produces sensible results from all functions so its value need not be set until special effects are required. STYLE is a byte variable - it can have any value from 0 to 255.

In the following three sections Line and Arc styles, Rectangle fill and Plot styles and Copy styles are detailed. The SSTYLE command (see

Housekeeping commands, Section 2.4.3) can be used to modify the STYLE variable.

2.3.5.1 Line and Arc styles

For lines and arcs only the least significant 4 bits of STYLE are used, the 4 most significant bits can be any value. STYLE can be split into the following bit fields:



FUNC	MEANING	WEIGHTING - HEX	(DEC)
RPL	Replace points on line/arc with COL	0	(0)
XOR	Logical XOR points on line/arc with COL	2	(2)
OR	Logical OR points on line/arc with COL	4	(4)
AND	Logical AND points on line/arc with COL	6	(6)

P	MEANING	WEIGHTING - HEX	(DEC)
Pattern	Apply pattern PAT while drawing line/arc	8	(8)
	Draw continuous line/arc	0	(0)

SK	MEANING	WEIGHTING - HEX	(DEC)
Skip	Don't draw first point on line/arc	1	(1)
	Draw all points on line/arc	0	(0)

To select the appropriate value of the STYLE variable, add the values in the WEIGHTING column corresponding to the facilities required. For example to draw a line using the XOR function, using a pattern and without plotting the first point on the line, a value of 11 (2+8+1) would be used.

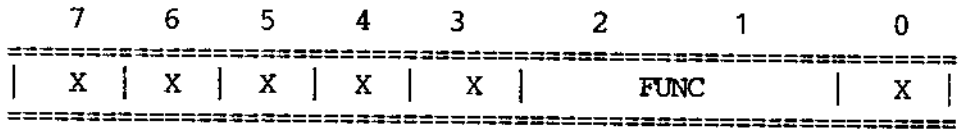
The pattern mask is used as follows: the next point on the line or arc is calculated; if the top bit of the pattern mask is set then the point is plotted using COL and FUNC otherwise no action is taken; the pattern mask is rotated left by one bit and the operation repeated for all other points on the line or arc.

The Skip bit suppresses the first point on the line or arc. This is useful when, for example, the XOR function is used while drawing connected lines since the last point of one line, being the same as the first point of the following line, would be plotted twice causing gaps to appear in the lines. It is also helpful for plotting complete circles for a similar reason.

As an example of using the STYLE variable when drawing lines or arcs, if a line were drawn using XOR with a colour of white (7), then each pixel along the line would be colour complemented. Although the WPROT flag is not effective for lines and arcs the effect of write protection can be achieved with appropriate use of colour and STYLE. For example, to draw a line only in the blue colour plane, a STYLE of OR (4) and a CCOL of blue (2) would be used. To erase a line in the blue colour plane a CCOL of yellow (5) (the complement of blue) and a STYLE of AND would be used.

2.3.5.2 Rectangle Fill and Plot styles

For Rfill (rectangle fill) and Plot commands STYLE has the following format:



FUNC	MEANING	WEIGHTING - HEX	(DEC)
RPL	Fill the rectangle with colour CCOL	0	(0)
XOR	XOR each pixel in the rectangle with CCOL	2	(2)
OR	OR each pixel in the rectangle with CCOL	4	(4)
AND	AND each pixel in the rectangle with CCOL	6	(6)

Rfill is a very fast function for operating on a rectangle using a constant colour and operator. Particularly useful operations are RPL for filling the area with a constant colour and XOR with white (CCOL=7) for colour complementing the area for highlighting. Although the write protect facility cannot be directly applied with Rfill the same effect can be achieved with appropriate combinations of FUNC and CCOL. For example, combining a FUNC of AND with a CCOL of 1 clears all but the least significant plane.

2.3.5.3 Copy styles

The definition of STYLE for the CopyS (copy symbol), Copy (rectangle) and CopyTS (copy to symbol) commands is as stated overleaf.

7	6	5	4	3	2	1	0
RS	WP	ROT			FUNC		N

FUNC	MEANING	WEIGHTING - HEX	(DEC)
RPL	Copy source raster to destination	0	(0)
XOR	Logical XOR source raster with destination	2	(2)
OR	Logical OR source raster with destination	4	(4)
AND	Logical AND source raster with destination	6	(6)
PAINT	Paint source raster onto destination	8	(8)

N	MEANING		
NOT	Logically invert source before operation	1	(1)
	Use source without inversion	0	(0)

ROT	MEANING		
0	Don't rotate source raster	0	(0)
90	Rotate source 90 degrees before operation	10	(16)
180	Rotate source 180 degrees before operation	20	(32)
270	Rotate source 270 degrees before operation	30	(48)

WP	MEANING		
Use WPROT	Write protect planes specified by WPROT	40	(64)
	Don't write protect any planes	0	(0)

RS	MEANING		
Use RSEL	Use only FCOL and BCOL in destination	80	(128)
	Perform operation normally	0	(0)

These functions use a common implementation within Pluto's firmware but offer convenient shorthand methods of specifying common operations (displaying and saving symbols). They all use a general raster (rectangle) copying operation known as RasterOp. RasterOp takes two rasters of equal size, a source and a destination, and replaces the destination raster with a function of the source and destination. The operation may be any of the FUNCs listed above and is applied between each pixel in the source raster and it's corresponding pixel in the destination. The operations are performed on all bits (planes) of the pixel simultaneously.

RPL

RPL can be used for scrolling parts of the display using a destination raster that is horizontally alligned with and above the source. A one line vertical spacing produces a smooth scroll while a 10 line spacing produces a character line scroll. RPL is also the appropriate function for defining a symbol that is already on the screen using the CopyTS command (see Raster Copying commands, Section 2.4.8).

XOR

XOR is useful for making non-destructive changes to the display. An example is superimposing a cursor on an image. The cursor is XOR'd onto the image to display it and then XOR'd again to remove it.

NOT

When NOT is combined with any function it can be used to invert (colour complement) the source raster before it is used. For example, using the same raster for both source and destination, and an RPL function, can be used to highlight an area (although the Rfill command can be used to achieve this effect more efficiently).

PAINT

PAINT produces the effect of copying non-rectangular shapes. The source and destination are defined exactly as for other RasterOps but pixels of a chosen colour are not copied to the destination. The chosen colour is defined by TCOL (transparent colour). For example, a symbol may be defined as a coloured shape on a black background and TCOL set to black. This function is so called as it is useful in painting programs using brushes defined as symbol shapes.

ROT

ROT rotates the source raster anti-clockwise through multiples of 90 degrees before performing the operation (but doesn't change the source in the frame buffer). It should be noted that while rasters are always copied non-destructively, when no rotation is specified it is impossible to always guarantee a non-destructive copy with rotation.

WP

WP invokes the WPROT (write protect) flag during a RasterOp which protects selected planes from being modified (see Single Colour Plane operators, Section 2.3.4). A situation where this is useful is when superimposing a text display on a graphics background using one plane for the text and the remaining planes for the graphics. To scroll the text, the graphics planes are write protected and an RPL RasterOp used.

RS

RS has it's main use with text and symbol display. The colours in the source raster are converted to either the foreground (FCOL) or background (BCOL) colour before the operation is performed. This means that once symbols are defined they may be displayed in any foreground and background colour combinations by simply changing FCOL and BCOL.

Pluto's RasterOps are extremely flexible enabling most required effects to be achieved. In spite of this flexibility Pluto is still able to perform simple functions such as RPL at maximum possible speed.

2.4 Pluto I Commands

2.4.1 Introduction

This section describes the commands built into Pluto's firmware. The commands are shown in two formats, the raw format (as you call them if you are sending and receiving individual codes to/from Pluto) and the high level language interface library format (if you use the demonstration disc supplied).

The format of these is as follows:-

Raw formats -

COMMAND (Parameter list)

COMMAND (Parameter list) : returned value

High level language interface library format:-

LIBRARY PROCEDURE .....

where the PROCEDURE'S are described in PASCAL.

Each command has a unique code that is sent to Pluto to invoke the command, following which parameters are sent in left to right order. All values sent to or received from Pluto are 8 bit quantities (16 bit quantities are sent as two 8 bit bytes).

Most commands use one or more of Pluto's state variables to minimise the number of parameters that need to be supplied with each call. One of the most useful state variables is the Current Position (CP) pointer which is updated after most operations to the position that would be most useful for subsequent commands. For example, after drawing a line the Current Position pointer is positioned at the end of the line, so that a sequence of line commands will draw connected lines without the need to re-define the Current Position. There will, however, be situations where the Current Position will not be appropriate and must be changed using one of the Move commands (see Current Position commands, Section 2.4.4).

Some commands use co-ordinates relative to the Current Position (a move 'by' as opposed to a move 'to') which enables position independent images to be created. Image drawing software routines using these commands can arrange to make the sum of all X movements zero and all Y movements zero, so that the final Current Position is the same as the initial Current Position.

The value of all Pluto's state variables may be set and inquired. The ability to read the current setting of these variables is important in a multi-process environment enabling a working context to be inquired and saved by an interrupting process, which temporarily takes over control of the display and then restores it to it's original state.

2.4.2 Parameter definitions

In the following sections the parameters for the Pluto I commands are shown as byte values or word values. A 'B' suffix signifies a byte value and a 'W' suffix signifies a word value sent as two bytes (low first). All co-ordinates are specified relative to the Current Position (CP). The following is a list of the parameters used with the Pluto I commands:

XW	-	X co-ordinate
YW	-	Y co-ordinate
DXW	-	X increment
DYW	-	Y increment
DXB	-	Short X increment
DYB	-	Short Y increment
DXCW	-	Arc curvature centre (X co-ordinate)
DYCW	-	Arc curvature centre (Y co-ordinate)
DXEW	-	Arc end point (X co-ordinate)
DYEW	-	Arc end point (Y co-ordinate)
WidthW	-	Width
HeightW	-	Height
nB	-	A general 8 bit unsigned value
PartitionB	-	A partition identifier
ColourB	-	colour

2.4.3 Housekeeping commands

COMMAND PInit (Hex : A6 Decimal : 166)
LIBRARY PROCEDURE PInit;

The PInit command initialises Pluto. All state variables are set to their default values, Symbol Partitions are freed and the frame buffer is cleared to all black.

COMMAND ClrCWP (Hex : AC Decimal : 172)
LIBRARY PROCEDURE ClrCWP;

The ClrCWP command clears the Current Working Partition (all pixels within it are set to zero). If the Current Working Partition is the same as the Current Display Partition then the screen will clear. The Current Position is not modified.

COMMAND AllocP (WidthW, HeightW, nB) : PartitionB
(Hex : A3 Decimal : 163).
LIBRARY PROCEDURE AllocP (Var Width, Height, N, Partition : INT);

The AllocP command allocates a Symbol or Workspace partition. The Width and Height parameters are the width and height of each symbol and n is the number of symbols required. To reserve a Workspace Partition, Width and Height would be the size of the required partition and n would be 1.

The maximum number of symbols per Symbol Partition is 128 and the maximum number of user partitions is 8. Symbols are numbered from 0 to n-1.

Width must not be greater than the horizontal screen resolution. The maximum value for the Height varies according to the board used and can be calculated as follows:

$$\frac{\text{TOTAL MEMORY IN BITS}}{\text{NO BITS PER PIXEL} * \text{HORIZONTAL SCREEN RESOLUTION}} - \text{MAXIMUM SCREEN VERTICAL RESOLUTION}$$

This produces the theoretical maximum number of lines, however, the Pluto firmware makes use of some of these, so therefore reduce the number you get by thirty to allow for this.

The value returned by the AllocP command is either 255 (Off hex) if the function failed to allocate a partition, or a partition identifier which can then be used by subsequent commands. If the function returns an error code the reason for failure can be found by using the IStat function.

COMMAND SHires (Hex : AA Decimal : 170)
LIBRARY PROCEDURE SHires;

The SHires command produces a resolution of 640H x 576V (768H x 576V on the high resolution Pluto I) using an interlaced field display. The Current Display Partition is set to 0 for consistency.

COMMAND SLores (Hex : AB Decimal : 171)
LIBRARY PROCEDURE SLores;

The SLores command produces a standard display resolution of 640H x 288V (768H x 288V on the high resolution Pluto I). The SLores command reverses the effect of the SHires command. If the Current Display Partition is currently 0 then it is set to 1 for consistency. If the Current Working Partition is changed from 0 to either 1 or 2 and the Current Position has a Y co-ordinate greater than 287, then the Y co-ordinate is set to zero.

Note: When switching screen resolutions with the SHires and SLores commands, the Current Working partition (CWP) is **NOT** amended as appropriate. The Current Working Partition can be set with the SCWP command.

```
COMMAND ScCol (ColourB)              (Hex : 89  Decimal 137)
LIBRARY PROCEDURE ScCol (Var Colour : INT);
```

The ScCol command sets the current colour to the specified Colour.

```
COMMAND SFCol (ColourB)              (Hex : 88  Decimal : 136)
LIBRARY PROCEDURE SFCol (Var Colour : INT);
```

The SFCol command sets the foreground colour to the specified Colour.

```
COMMAND SBCol (ColourB)              (Hex : 87  Decimal : 135)
LIBRARY PROCEDURE SBCol (Var Colour : INT);
```

The SBCol command sets the background colour to the specified Colour.

```
COMMAND STCol (ColourB)              (Hex : 8B  Decimal : 139)
LIBRARY PROCEDURE STCol (Var Colour : INT);
```

The STCol command sets the transparent colour used in PAINT operations (Copy and CopyTS - see Raster Copying commands, Section 2.4.8) to the specified Colour.

```
COMMAND SSTYLE (nB)                  (Hex : 8A  Decimal : 138)
LIBRARY PROCEDURE SStyle (Var n : INT);
```

The SSTYLE command sets the style for subsequent commands to n.

```
COMMAND SCDP (PartitionB)            (Hex : 83  Decimal : 131)
LIBRARY PROCEDURE SCDP (Var Partition : INT);
```

The SCDP command sets the Current Display Partition to be the specified Partition. The three possible values are 0, 1 and 2. The Current Display Partition should only be set to 1 or 2 to display one of the two 640H x 288V (768H x 288V on the high resolution Pluto I) screen images. It is automatically set to 0 by the SHires command and restored to 1 by the SLores command.

```
PROCEDURE SCWP (PartitionB)          (Hex : A5  Decimal : 165)
LIBRARY PROCEDURE SCWP (Var Partition : INT);
```

The SCWP command sets the Current Working Partition to the specified Partition. This establishes the size and position of a section of Pluto's

memory to which all specified co-ordinates are applied. The value of the Partition will normally be 0 when working in the 640H x 576V (768H x 576V on the high resolution Pluto I) display, or 1 or 2 for one of the two 640H x 288V (768H x 288V on the high resolution Pluto I) displays. User-allocated partitions may also be used for the Current Working Partition in which case the partition may be freely used for general workspace instead of storing symbols. Most commands operate within the Current Working Partition.

```
COMMAND SCSP (PartitionB)              (Hex : A4  Decimal : 164)
LIBRARY PROCEDURE SCSP (VAR Partition : INT);
```

The SCSP command sets the Current Symbol Partition to the specified Partition. The Current Symbol Partition can be set to 255 in order to use the standard ASCII character set or to any user-allocated partition number (user-allocated partitions can be specified with the AllocP command).

```
COMMAND SWprot (nB)                    (Hex : 8C  Decimal : 140)
LIBRARY PROCEDURE SWprot (Var n : INT);
```

The SWprot command sets the write protect mask to n. The WPROT variable can only be used with selected commands and then only if the WP bit in STYLE is set. WPROT uses one bit per colour plane to selectively prevent changes being made within the plane.

```
COMMAND SRsel (nB)                     (Hex : B8  Decimal : 184)
LIBRARY PROCEDURE SRsel (Var n : INT);
```

The SRsel command sets the read select mask to n. The RSEL variable can only be used for selected commands and then only if the RS bit in STYLE is set. RSEL uses one bit per colour plane to select a plane (or planes) to use in the operation.

```
COMMAND SPat (nB)                      (Hex : B6  Decimal : 182)
LIBRARY PROCEDURE SPat (Var n : INT);
```

The SPat command sets the pattern mask to n. PAT can be used for line and arc drawing if the P bit in STYLE is set.

```
COMMAND SPCol (ColourB)                (Hex : BA  Decimal : 186)
LIBRARY PROCEDURE SPCol (Var PCol : INT);
```

The SPCol command sets the perimeter colour to the specified Colour. PCOL can be used in boundary fills to define the border colour of the area to be filled.

```
COMMAND ICCol : ColourB                (Hex : 8D  Decimal : 141)
LIBRARY PROCEDURE ICCol (Var Colour : INT);
```

The ICCol command returns the current value of the Current Colour (CCOL) variable.

```
COMMAND IFCol : ColourB                (Hex : 8F  Decimal : 143)
LIBRARY PROCEDURE IFCol (Var Colour : INT);
```

The IFCol command returns the current value of the foreground colour (FCOL) variable.

```
COMMAND IBCol : ColourB                (Hex : 90  Decimal : 144)
LIBRARY PROCEDURE IBCol (Var Colour : INT);
```

The IBCol command returns the current value of the background colour (BCOL) variable.

```
COMMAND ITCol : ColourB                (Hex : 91  Decimal : 145)
LIBRARY PROCEDURE ITCol (Var Colour : INT);
```

The ITCol command returns the current value of the transparent colour (TCOL) variable.

```
COMMAND IStyle : nB                    (Hex : 8E  Decimal : 142)
LIBRARY PROCEDURE IStyle (Var n : INT);
```

The IStyle command returns the current value of the STYLE variable.

```
COMMAND ICDP : PartitionB              (Hex : 94  Decimal : 148)
LIBRARY PROCEDURE ICDP (Var Partition : INT);
```

The ICDP command returns the current value of the Current Display Partition (CDP).

```
COMMAND ICWP : PartitionB              (Hex : 93  Decimal : 147)
LIBRARY PROCEDURE ICWP (Var Partition : INT);
```

The ICWP command returns the current value of the Current Working Partition (CWP).

```
COMMAND ICSP : nB                      (Hex : 92  Decimal : 146)
LIBRARY PROCEDURE ICSP (Var n : INT);
```

The ICSP command returns the current value of the Current Symbol Partition (CSP).

```
COMMAND IWprot : nB                    (Hex : 95  Decimal : 149)
LIBRARY PROCEDURE IWprot (Var n : INT);
```

The IWprot command returns the current value of the WPROT variable.

```
COMMAND IRSel : nB                     (Hex : B9  Decimal : 185)
LIBRARY PROCEDURE IRSel (Var n : INT);
```

The IRSel command returns the current value of the RSEL variable.

```
COMMAND IPat : nB                      (Hex : B7  Decimal : 183)
LIBRARY PROCEDURE IPat (Var n : INT);
```

The IPat command returns the current value of the PAT variable.

```
COMMAND IPCol : ColourB                (Hex : BB  Decimal : 187)
LIBRARY PROCEDURE IPCol (Var Colour : INT);
```

The IPCol command returns the current value of the perimeter colour (PCOL).

```
COMMAND IStat : nB                     (Hex : 86  Decimal : 134)
LIBRARY PROCEDURE IStat (Var n : INT);
```

The IStat command returns the current value of STATUS. STATUS records the result of the last command. A value of zero signifies a successful operation and any other value signifies an error (see Pluto I Error Codes, Section 2.5).

2.4.4 Current Position commands

```
COMMAND MoveTo (XW, YW)                (Hex : 97  Decimal : 151)
LIBRARY PROCEDURE MoveTo (Var X, Y : INT);
```

The MoveTo command moves the Current Position to the specified X and Y co-ordinates within the Current Working Partition (CWP). If the specified co-ordinates are outside the partition the Current Position is not amended.

```
COMMAND MoveR (DXW, DYW)               (Hex : 98  Decimal : 152)
LIBRARY PROCEDURE MoveRL (Var DX, DY : INT);
```

The MoveR command moves the Current Position by the specified DX and DY co-ordinates (16 bit values) within the Current Working Partition, i.e. an incremental or relative movement. If the co-ordinates are outside the partition the Current Position is not amended.

COMMAND MoveRS (DXB, DYB) (Hex : 99 Decimal : 153)
NO LIBRARY PROCEDURE

The MoveRS command moves the Current Position by the specified DX and DY co-ordinates (as with the MoveR command, a relative movement) within the Current Working Partition. This command is useful for short movements (+127/-128) as only two single byte parameters are required. If the co-ordinates are outside the partition the Current Position is not amended.

COMMAND ICP : XW, YW (Hex : 96 Decimal : 150)
LIBRARY PROCEDURE ICP (Var X, Y : INT);

The ICP command returns the Current Position as X and Y co-ordinates.

2.4.5 Line Drawing commands

COMMAND LineTo (XW, YW) (Hex : 80 Decimal : 128)
LIBRARY PROCEDURE LineTo (Var X, Y : INT);

The LineTo command draws a line from the Current Position to the specified X and Y co-ordinates. The appearance of the line is controlled by the current STYLE. The Current Position is updated to point to the end of the line (the last point that was plotted). If the specified X and Y co-ordinates are outside the Current Working Partition then none of the line is drawn and the Current Position is not modified.

COMMAND LineR (DXW, DYW) (Hex : 9D Decimal : 157)
LIBRARY PROCEDURE LineRL (Var DX, DY : INT);

The LineR command draws a line from the Current Position to a point with co-ordinates of DX and DY (16 bit values), relative to the Current Position, i.e. an incremental movement. The conditions that apply to the LineTo command also apply to the LineR command.

COMMAND LineRS (DXB, DYB) (Hex : 9E Decimal : 158)
NO LIBRARY PROCEDURE

The LineRS command draws a line from the Current Position to a point with co-ordinates of DX and DY, relative to the Current Position (as with the LineR command). This command is useful for drawing short lines (end point +127/-128 from the Current Position) as only two single byte parameters are required. The conditions that apply to the LineTo command also apply to the LineRS command.

2.4.6 Arc and Circle commands

COMMAND Arc (DXCW, DYCW, DXEW, DYEW) (Hex : C1 Decimal : 193)
LIBRARY PROCEDURE Arc (Var DXC, DYC, DXE, DYE : INT);

The Arc command draws an arc from the Current Position in an anti-clockwise direction to a point with co-ordinates of DXE and DYE. The arc is drawn relative to the Current Position using a very fast incremental algorithm, and has a centre of curvature at a point with co-ordinates of DXC and DYC, relative to the Current Position.

The centre of curvature does not have to be inside the Current Working Partition, which allows an arc with a large curvature to be drawn. By selecting negative or positive values for DXC and DYC an arc can be drawn with any direction of curvature. If the specified end point is not on the arc, then the arc will stop at the nearest point on the path of the arc. If the end point is grossly in error, then the arc will stop on a quadrant boundary.

To cope with the two possible screen resolutions (640H x 288V and 640H x 576V) which have different aspect ratios, the arc is plotted asymmetrically by scaling the Y co-ordinates to make the arc appear as a segment of a circle. The aspect is chosen according to the Current Working Partition. When working in partition 0, a 640H x 576V resolution is assumed and one screen unit in the Y direction is equivalent to 0.87 of a unit in the X direction. Otherwise a 640H x 288V resolution is assumed where one screen unit in Y is equivalent to 1.66 units in X. In both cases the space between adjacent pixels in the X direction is taken as the unit of measurement. This should be taken into account when calculating XC, YC (the centre of curvature) and XE, YE (the arc end point) which are screen co-ordinates. The range of values of XC (and YC before scaling) is +/-1024.

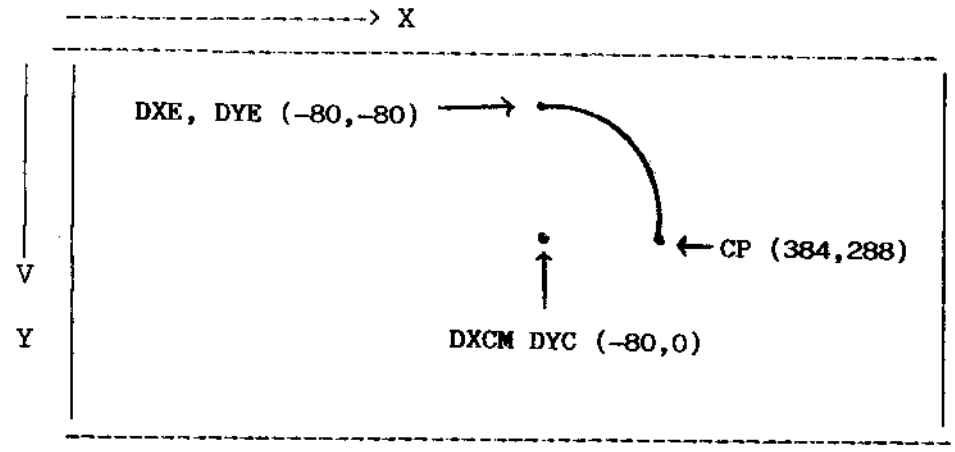
The Arc command differs from all other commands in that it may be only partially executed if it extends outside of the Current Working Partition. In this case the command is terminated when the arc reaches the partition boundary. The appearance of the arc is determined by the STYLE setting.

The Arc command can also be used to draw a circle. When drawing a circle the start and end points are both at the Current Position so the DXE and DYE co-ordinates do not need to be specified. Hence the only parameters required when drawing a circle are the centre of curvature parameters, DXC and DYC, i.e. the screen co-ordinates for the centre of the circle.

Example 1 : To draw an arc with the Current Position at (384,288), with an arc end point of (300,208) and a centre of curvature point of (300,288).

The parameters for the Arc command (using decimal notation) would then be:

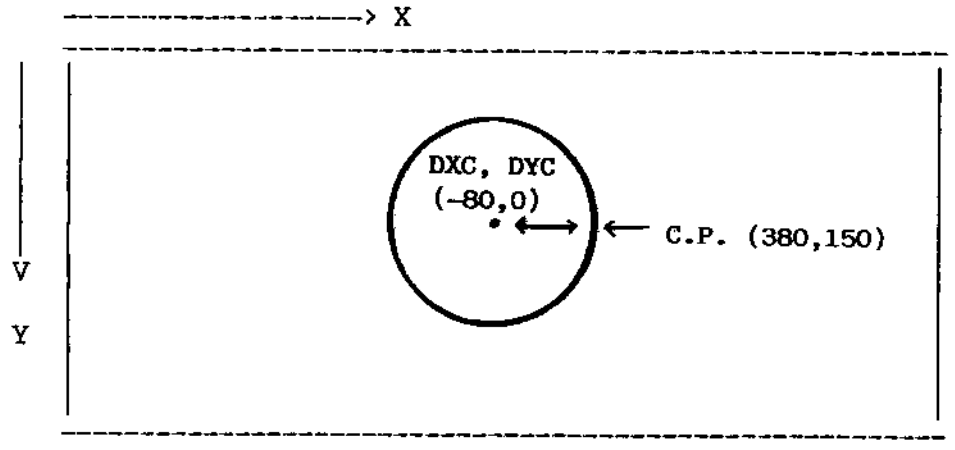
```
DXC = -80
DYC =   0
DXE = -80
DYE = -80
```



Example 2 : To draw a circle at centre (300, 150) and radius 80 units:

Move the Current Position to (300 + radius, 150). The parameters for the Arc command (using decimal notation) would then be:

DXC = -80
DYC = 0
DXE = 0
DYE = 0



Example 3 : To draw an arc from the Current Position in working partition 1 with radius 1000 and ends subtending angles of 30 degrees and 35 degrees to the horizontal.

If Dx and Dy are the distances respectively in the X and Y directions from the centre of curvature of the arc to the Current Position, then :

Dx1 = Radius*cos(30)

Dy1 = a*Radius*sin(30) - where a is 1.66 for Partition 1

For an arc in the first quadrant DXC=-Dx1, YXC=Dy1 (remembering that Y co-ordinates increase from the top of the screen to the bottom).

Similarly for the distance from the centre to the end point:

Dx2 = Radius*cos(35)

Dy2 = a*Radius*sin(35)

AND

DXE = Dx2 - DXC

DYE = Dy2 + DYC

which gives :

DXC = -1000 * 0.866 = -866

DYC = 1/1.66 * 1000 * 0.5 = 301

DXE = 1000 * 0.819 - 866 = -47

DYE = -1/1.66 * 1000 * 0.574 + 301 = -45

(all values rounded to the nearest integer)

The parameters for the Arc command (using Hex notation) would then be:

DXC = 0FC9E

DYC = 12D

DXE = 0FFD1

DYE = 0FFD3

2.4.7 Area Fill commands

Pluto I includes an extensive set of fast, powerful routines for filling any general area with solid colour or pattern. The 'pattern area fills' ease the production of information displays. Painting programs can produce composite colours and textured patterns by using subtle combinations of colours in the pattern. Graphs and bar charts may be constructed and shaded with meaningful patterns. These features all highlight Pluto's versatility and speed.

```
COMMAND Rfill (WidthW, HeightW)     (Hex : 81  Decimal : 129)
LIBRARY PROCEDURE Rfill (Var Width, Height: INT);
```

The Rfill command fills a rectangle of the specified Width and Height with the top left corner of the rectangle at the Current Position. The rectangle will be filled with the Current Colour (CCOL). The Current Position will be incremented by the Width value in the X direction. If the new X co-ordinate overflows the partition width it will be zeroed, and the Current Position incremented by the Height value in the Y direction. If the new Y co-ordinate overflows the partition height it will be zeroed. If the specified rectangle lies partially outside the Current Working Partition then no operation will be performed.

```
COMMAND Ffill                       (Hex : 82  Decimal : 130)
LIBRARY PROCEDURE Ffill;
```

The Ffill command flood fills any arbitrary closed area with the Current Colour. The colour of the pixel at the Current Position defines the colour of the area to be filled. The area is made up of this pixel and all connected pixels of the same colour. If the area is not enclosed completely then the filling colour will leak out and only stop at the partition boundary. Pixels outside the Current Working Partition are not modified and the Current Position is not moved.

```
COMMAND Bfill                       (Hex : B0  Decimal : 176)
LIBRARY PROCEDURE Bfill;
```

The Bfill command boundary fills any arbitrary closed area with the Current Colour. This differs from the Ffill command in that the filled area is defined by the boundary colour (which is defined by the perimeter colour (PCOL). The filled area starts at the Current Position and continues in all directions until the boundary colour is met. For this command to operate correctly, pixels in the area to be filled must not be coloured with the Current Colour (unless the Current Colour is the same as the perimeter colour).

This command can be used instead of the Ffill command to wipe an area containing more than one colour.

increases the speed of pattern filling on simple areas.

```
COMMAND BfilSP                          (Hex : B3  Decimal : 179)
LIBRARY PROCEDURE BFilSP;
```

The BfilSP command boundary fills a simple area with a pattern. The filled area is defined using the method described for the BfilS command and the operation performed is as described for the FfilP command.

```
COMMAND SFpatS (PartitionB,nB)      (Hex : B5  Decimal : 181)
LIBRARY PROCEDURE SFpatS (Var Partition, n:INT);
```

The SFpatS command sets the fill pattern for all pattern fills to be a symbol. The parameters are the Symbol partition identifier (Partition) and the symbol number (n). The symbol may be a user-defined symbol or one of the characters from the standard ASCII character set which is stored in Symbol partition 255.

```
COMMAND SfPatR (WidthW, HeightW, PartitionB, XW, YW)

                                    (Hex : B4  Decimal : 180)

LIBRARY PROCEDURE SFpatR (Var Width, Height, Partition, X,Y):
```

The SfPatR command sets the fill pattern for all pattern fills to be an arbitrary rectangle. The parameters are the rectangle Width and Height, the Partition in which the rectangle is stored and the co-ordinates of the top left corner of the rectangle (X and Y). The rectangle can be in any partition and of any size up to the maximum size of the partition.

This command only sets the position at which the pattern is to be found, the image within the rectangle can be amended to produce different patterns. A wide range of effects can be achieved with this command including using a text string for a pattern, using a single angled line to produce a shading pattern or using a pair of crossed lines to produce a cross-hatch shading.

2.4.8 Raster Copying commands

Pluto's three raster copy commands use a common set of routines but take different parameters. When an image is created using the line, arc and area fill commands, the Raster Copying commands can then be used to provide an efficient and flexible method of manipulating the image. These commands are further described in the Copy Styles in (Section 2.3.5.3) of this manual.

COMMAND Copy (WidthW, HeightW, Partition1B, X1W, Y1W,
Partition2B, X2W, Y2W)

(Hex : 85 Decimal : 133)

LIBRARY PROCEDURE Copy (Var Width, Height, Partition1, X1, Y1, Partition 2, X2, Y2);

The Copy command can be used as a general all-purpose copying facility. The source and destination rasters are each of specified Width and Height (if a rotation of 90 or 270 degrees is specified within the STYLE variable, then the destination raster has a width of the Height value and a height of the Width value). The source raster is in the partition specified first with it's top left corner at X1, Y1. The destination of the raster is in the partition specified second with the rasters top left corner at X2, Y2. The Copy command does not use the Current Working Partition or the Current Position and does not modify either.

The effect of the Copy command depends on the current STYLE. If any of the parameters are out of range then no operation is performed.

COMMAND CopyS (nB) (Hex : 20-7F Decimal : 32-127)
LIBRARY PROCEDURE PChar (Var n:INT);

The Copy S command copies a symbol. The source raster is symbol number n within the Current Symbol Partition (the width and height of which is already known to Pluto). The destination raster is at the Current Position within the Current Working Partition. After the operation the Current Position is updated using the method described for the Rfill command (see Area Fill commands, Section 2.4.7). The CopyS command is useful for writing text using either the default character set or a user-defined font.

This command differs from all others in that only the parameter is sent to Pluto, i.e. there is no CopyS command code. All symbol numbers are between 0 and 127 (7f hex) and command codes are between 128 and 255 (80 hex and 0ff hex). When a value less than 128 is sent to Pluto, it is interpreted as a symbol number and the CopyS command is automatically invoked with this as the parameter. This allows text to be used efficiently needing only one code to be sent per character.

COMMAND CopyTS (nB) (Hex : 84 Decimal : 132)
LIBRARY PROCEDURE CopyTS (Var n:INT);

The Copy TS command copies a raster to a symbol. It has the opposite effect to the CopyS command. The source raster is at the Current Position within the Current Working Partition, and the destination is the place in the Current Symbol Partition where symbol n is stored. The size of the raster is the size of the symbol.

This command is one way of defining a symbol. To produce an exact copy of the source, appropriate for multi-coloured symbols, the STYLE variable would be set to 0. Where symbols are drawn using only two colours (foreground and background), and symbol storage space is at a premium, the symbol can be stored in a single colour plane. To do this the WPROT

mask is set to protect all but the desired destination plane, and RSEL to select the plane on which the source symbol is defined. The foreground colour (FCOL) should be set to 7, the background colour (BCOL) to 0 and the WP and RS bits in STYLE enabled. The WPROT and RSEL variables are further described in the Single Colour Plane operators (Section 2.3.4) Section of this manual.

2.4.9 Single Pixel commands

These commands allow an image to be flexibly operated on, in situations where none of the higher level commands perform the desired effect. One such operation would be to trace out irregular outlines.

Note: Wherever possible the higher level commands should be used to minimise the number of command calls and thereby increase efficiency.

The effect of these commands can be modified by amending the STYLE variable.

```
COMMAND Plot (XW, YW)                (Hex : 9A  Decimal : 154)
LIBRARY PROCEDURE Plot (Var X,Y:INT);
```

The Plot command plots a point at the specified X and Y co-ordinates using the Current Colour (CCOL). The Current Position (CP) is moved to the X and Y co-ordinates.

```
COMMAND PlotR (DXW, DYW)             (Hex : 9B  Decimal : 155)
LIBRARY PROCEDURE  PlotR (Var DX, DY : INT);
```

The PlotR command plots a point at the DX and DY co-ordinates (16 bit values), relative to the Current Position (an incremental movement). The Current Position is moved by the DX and DY co-ordinates.

```
COMMAND PlotRS (DXB, DYB)            (Hex : 9C  Decimal : 156)
NO LIBRARY PROCEDURE;
```

The PlotRS command plots a point at the DX and DY co-ordinates (8 bit values), relative to the Current Position (as with the PlotR command).

```
COMMAND RPix (XW, YW) : ColourB      (Hex : A7  Decimal : 167)
LIBRARY PROCEDURE RPix (Var X,Y : INT);
```

The RPix command returns the Colour of the pixel at the specified X and Y co-ordinates. The Current Position is moved to the X and Y co-ordinates.

```
COMMAND RPixR (DXW, DYW) : ColourB (HEX : A8  Decimal : 168)
LIBRARY PROCEDURE RPixR (Var DX, DY, Colour :INT);
```

The RPixR command returns the Colour of the pixel at the specified DX and DY co-ordinates (16 bit values), relative to the Current Position. The Current Position is moved by the DX and DY co-ordinates.

COMMAND RPixRS (DXB, DYB) : ColourB

(Hex : A9 Decimal : 169)

NO LIBRARY PROCEDURE;

The RPixRS command returns the Colour of the pixel at the specified DX and DY co-ordinates (8 bit values), relative to the Current Position. This command is useful for short movements (+127/-128 from the Current Position). The Current Position is moved by the DX and DY co-ordinates.

2.4.10 Image and Symbol Read and Load commands

These commands allow the reading and loading of a partition (or part of a partition), and are useful for saving images on backing storage and for defining symbol shapes. A choice of data formats is provided to cater for full colour one-byte-per-pixel or compressed two-colour images. The compressed forms increase the amount of image and symbol storage by using the frame buffer as four single bit planes. They also provide a more compact form for data transfer to and from the host, improving the transfer rate and decreasing the external storage space required.

COMMAND LImage (WidthW, HeightW, <Width x Height pixels>)

(Hex : 9F Decimal : 159)

LIBRARY PROCEDURE LImage (Var Width, Height, Pixels : INT);

The LImage command transfers a sequence of pixels into a specified raster. The raster has it's top left corner at the Current Position within the Current Working Partition, and is of specified Width and Height. Each byte of data defines the colour of one pixel. The correct number of pixels must be supplied. The Current Position is moved by the Width value in the X direction (as with the Rfill command in the Area Fill commands, Section 2.4.7).

COMMAND LSym (nB, <Width x Height pixels>)

(Hex : A0 Decimal : 160)

LIBRARY PROCEDURE LSym (Var n, Pixels : INT);

The LSym command transfers a sequence of pixels into symbol n of the Current Symbol Partition, in order to define a symbol. As Pluto knows the size of the symbol it does not have to be specified in the parameter list. The LSym command works in the same way as the LImage command.

COMMAND RImage (WidthW, HeightW) : <Width x Height pixels>

(Hex : A1 Decimal : 161)

LIBRARY PROCEDURE RImage (Var Width, Height, Pixels : INT);

The RImage command is the converse of the LImage command. After sending the command, the host must read Width x Height bytes, one byte per pixel in the raster. The raster is defined in the same way as for the LImage command.

COMMAND RSym (nB) : <Width x Height pixels>

(Hex : A2 Decimal : 162)
LIBRARY PROCEDURE RSym (Var n, Pixels :INT);

The RSym command can be used to retrieve symbol definitions from the frame buffer. This command is the converse of the LSym command.

COMMAND LImagC (WidthW, HeightW, <data bytes>)

(Hex : BC Decimal : 188)
LIBRARY PROCEDURE LImagC (Var Width, Height, Pixels : INT);

The LImagC command loads an image from compressed data into selected colour planes. Each data byte defines the colour of 8 consecutive pixels in the raster being loaded, moving from left to right starting with the most significant byte. If a bit in the data byte is set (1) the foreground colour (FCOL) is written to the corresponding pixel, otherwise the background colour (BCOL) is written. The WPROT flag is used to load selected planes.

As an example, to load the image into the green plane, only the foreground colour is set to 1, the background colour is set to 0 and the WPROT flag to 6. The command parameters define the Width and Height of the raster as with the LImage command. If the Width is not an exact multiple of 8 then it is rounded up to the nearest multiple of 8, and then divided by 8 to determine the number of data bytes to be sent for each row.

COMMAND LSymC (nB, <data bytes>) (Hex : BD Decimal : 189)

LIBRARY PROCEDURE LSymC (Var n, Pixels : INT);

The LSymC command loads a symbol from compressed data. The LSymC command is useful for storing symbols which are defined using a bit-per-point mask on selected planes, which effectively quadruples the amount of symbol storage space. The symbol is normally converted into selected foreground and background colours when it is used. This command works in the same way as the LImagC command.

COMMAND RImagC (WidthW, HeightW) : <data bytes>

(Hex : BE Decimal : 190)

LIBRARY PROCEDURE RImagC (Var Width, Height, Pixels : INT);

The RImagC command reads a raster and compresses the data into 8 pixels per data byte. This command, which is the converse of the LImagC command, can be used to read an image stored on a single plane. The plane is selected using RSEL.

COMMAND RSymC (nB) : <data bytes> (Hex : BF Decimal : 191)
LIBRARY PROCEDURE RSymC (Var n, Pixels : INT);

The RSymC command reads a symbol and compresses the data. This command is the converse of the LSymC.

2.5 Pluto I Error Codes

If incorrect parameters are sent to Pluto, or for some other reason command cannot be executed, the state variable STATUS will be assigned a value to indicate the reason for failure. This value can be read using the IStat command. The possible error codes are as follows:

Error Code	Meaning
1	The co-ordinates used for a command were outside the partition boundary. This could arise from width or height values being too large, or absolute or relative offsets producing out of bounds co-ordinate values.
2	Command code not implemented.
3	Partition identifier used with the command has not been allocated.
4	No more free partitions available (in response to an AllocP call).
5	Width or height parameters for an AllocP call are zero or too large.
9	A symbol number supplied to the command was greater than the number of symbols in the partition.
10	Illegal partition number for a call to SCDP.
11	No more space for symbol partitions.
12	Rotation was specified for a CopyTS command (not allowed).
13	FfilP was given an impossible area to pattern fill and 'timed out'.
14	An arc that crossed the partition boundary was clipped.
15	Parameters to an arc command were too big.

2.6 General Programming Notes

Pluto uses two I/O ports for communication. One of these is a bi-directional data port (the COMMAND port) over which all commands and parameters are sent and results received. The second port is used as a STATUS port when read by the host, and as a reset port when written. The port addresses are selectable. In some computer applications Pluto may be memory-mapped in which case two memory locations are used.

Commands are sent to Pluto's command port as a series of bytes - the command code followed by the command parameters, in left to right order. Pluto cannot accept a command if it is processing a previous command, so a check must be made to establish whether Pluto is ready to accept data. This is achieved by reading the STATUS port. For all Pluto packages the top bit of the status byte is set (i.e. a value of >=128) if Pluto is ready to accept a data byte, and zero otherwise. For the Pluto II 24 bit system the particular bit to be set can be selected.

The STATUS port may be read at any time without affecting Pluto's operation. To avoid the necessity of checking the STATUS port before sending every byte of data, Pluto guarantees certain behaviour characteristics (see rule 4, below). To gain the optimum performance from Pluto, the following rules should be followed when sending a command:

1. Read and test the value returned from the STATUS port. If the value is greater than 127 (top bit set) then move on to step 2, otherwise repeat step 1.

2. Send the first byte of data (the command code) to the COMMAND port.

3. Wait for Pluto to decode the command by testing the STATUS port, as described in step 1.

4. Send the remaining data bytes (the parameters) without checking the STATUS port if the maximum transfer rate is 2.5 microseconds per byte.

For commands that return one or more data bytes, Pluto must be in a ready state before the first result byte is read from the COMMAND port. Thereafter result bytes may be read at the maximum data rate without the need to check the STATUS port.

There are a few commands that cannot transfer data at the maximum rate mentioned above, namely the read and load data commands (LSym, LImage, RSym and RImage) and the compressed data read and load commands (LSymC, LImagC, RSymC and RImagC). With the normal read and load commands, pixel data may be transferred at the maximum stated rate while each horizontal line in the raster is being transferred, but the STATUS port must be checked between lines. For the compressed read and load commands it is advisable to check the STATUS port for each byte transferred.

It is important to send the correct number of parameters with a command and to read the correct number of result bytes. If this is not done then Pluto may misinterpret subsequent commands. A command may be terminated during parameter or result transfer by sending 4 zero value bytes to the STATUS port (the STATUS port becomes a RESET port when writing). The STATUS port should be read first to make sure that Pluto is ready to accept the reset command. STATUS need only be checked before sending the first zero byte. The RESET will cause Pluto to ready itself for the beginning of a new command which may be sent when the STATUS port returns to the ready state.

2.7 Mini-Palette

The Mini-Palette, an option which may be fitted by Io Research onto the main Pluto controller board, enables each of the 16 basic Pluto colours to be re-defined from a selection of 4096 colours. The Mini-Palette can be added to the Pluto I board to increase the colour selection.

The Mini-Palette consists of a look-up table (LUT) that maps each of the pixel values to a colour on the screen and three high speed digital-to-analogue converters to provide variable intensity drives to the colour monitor RGB inputs. The colour is defined in terms of its red, green and blue primary components. Each of these components can be one of 16 different values allowing a total of 16 x 16 x 16 (4096) different colour combinations. The software also allows the redefinition of these values.

2.7.1 Mini-Palette Commands

The Mini-Palette option provides extra commands for manipulating the colour look-up table. Two look-up tables are provided (referred to as pages) so that two completely independent sets of colour maps can be loaded onto the Mini-palette. The particular page to be used for colour mapping can be selected with a single command - this page is referred to as the LUT display page.

The look-up table can be modified to re-define the colour mapping - the look-up table that is modified is defined as the LUT working page (the working page can be the same as the display page). As there are only 16 LUT entries, the whole LUT can be re-defined within one frame period. This facility can be used to change large areas of image quickly instead of changing the actual pixel values in the image.

All visible LUT changes are performed during a vertical blanking period so that no display tearing or interference is seen.

The Mini-Palette look-up table (LUT) page can be made to alternate between page 1 and 2 to produce a blinking facility. Each pixel blinks between the two definitions in the two look-up tables. Two commands are available with the Mini-Palette to control and monitor the blink rate: the SBlink command and the IBlank command (see Section 2.8.1,Mini-Palette Commands).

The r, g, b (red, green and blue) values in the following Palette commands specify the intensity of each colour component and have a value of 0 (zero intensity) to 15 (peak intensity).

The following is a list of the parameters used with the Mini-Palette commands:

iB	-	look-up table entry
nB	-	number of look-up table entries
gB	-	green value
bB	-	blue value
rB	-	red value
PageB	-	look-up table page
RateB	-	blink rate

COMMAND LUTInitG (Hex : C3 Decimal : 195)
LIBRARY PROCEDURE LUTInitG;

The LUTInitG command initialises both pages of the look-up table to be a linear greyscale. The LUT working page and the LUT display page are both set to be 1.

COMMAND LUTInitC (Hex : C4 Decimal : 196)
LIBRARY PROCEDURE LUTInitC;

The LUTInitC command initialises both pages of the look-up table to be a selection of colours. Colours 0 to 7 are the saturated colours available with the stand-alone Pluto (black, green, blue, cyan, red, yellow, magenta and white). The LUT working page and the LUT display page are both set to be 1.

COMMAND WLut (iB, gB, bB, rB) (Hex : C5 Decimal : 197)
LIBRARY PROCEDURE WLut (Var i, Gbrarray : INT);

The WLut command defines the colour component values of entry i in the working LUT page.

For example:-

> WLut (1,15,15,15) would define LUT entry 1 to be the brightest white, so that every pixel with value 1 would appear on the screen as white. If the working LUT page is different from the display LUT page then the operation is performed immediately and is invisible until the display page is changed. If the working page is the same as the display page then the LUT is not updated until the next vertical blanking period to prevent screen disturbances.

COMMAND WLutM (nB, iB, gB, bB, rB,.....gB, bB, rB)
(Hex : C6 Decimal : 198)
LIBRARY PROCEDURE WLutM (Var n, i, Gbrarray : INT);

The WLutM command writes multiple look-up table entries. The g, b, r values are written into the n+1 LUT entries in the working LUT page starting at entry number i (n=0 to write one entry, n=15 to write 16 entries, etc.) For example, WLutM (1,1,15,,0,0,7,7,7) would define pixel value 1 to be the brightest green, and value 2 to be half white (grey). This is equivalent to 2 WLut commands but is more convenient and efficient.

If the working page is the same as the display page then the LUT is not updated until the next blanking period to prevent screen disturbances. It is much quicker to use the WLutM command rather than several WLut commands, as all of the entries are written during one blanking period rather than only one at a time.

COMMAND RLut (iB) : gB, bB, rB (Hex : C7 Decimal : 199)
LIBRARY PROCEDURE RLut (Var i, Gbrarray : INT);

The RLut command returns the g, b, r values of entry i in the working LUT page. The values, which are returned in the order of green, blue and then red, are single byte quantities. This command is a complementary function to the WLut command.

COMMAND RLutM (nB, iB) : gB, bB, rB,.....gB, bB, rB
(Hex : C8 Decimal : 200)
LIBRARY PROCEDURE RLutM (Var n, i, Gbrarray : INT);

The RLutM command returns the g, b, r values of n+1 entries starting at entry i in the working LUT page. The values, which are returned in the order of green, blue and then red, are single byte quantities. This command is a complementary function to the WLutM command.

COMMAND SLutWP (PageB) (Hex : C9 Decimal : 201)
LIBRARY PROCEDURE SLutWP (Var Page : INT);

The SLutWP command sets the working LUT page to the specified Page (1 or 2).

COMMAND SLutDP (PageB) (Hex : CA Decimal : 202)
LIBRARY PROCEDURE SLutDP (Var Page : INT);

The SLutDP command sets the display LUT page to the specified Page (1 or 2).

COMMAND ILutWP : PageB (Hex : CB Decimal : 203)
LIBRARY PROCEDURE ILutWP (Var Page : INT);

The ILutWP command returns the value of the working LUT page. The Page parameter is a single byte.

COMMAND ILutDP : PageB (Hex : CC Decimal : 204)
LIBRARY PROCEDURE ILutDP (Var Page : INT);

The ILutDP command returns the value of the display LUT page. The Page parameter is a single byte.

COMMAND SBlink (RateB) (Hex : CD Decimal : 205)
LIBRARY PROCEDURE SBlink (Var Page : INT);

The SBlink command sets the blink rate. If Rate=0 then blinking is disabled and the LUT display page is used as the active display page. Setting Rate to a value from 1 to 5 causes the LUT display page to alternate between 1 and 2 at the following rates:

Rate	BLINK RATE
0	off (display LUT display page)
1	25 times / second
2	3.125 times / second
3	1.5625 times / second
4	once every 1.28 seconds
5	once every 2.56 seconds

COMMAND IBlink : RateB (Hex : CE Decimal : 206)
LIBRARY PROCEDURE IBlink (Var Rate : INT);

The IBlink command returns the current blink rate. The Rateparameter is a single byte.

2.7.2 Mini-Palette Error codes

Error Code	Meaning
16	LUT index value/range too big
17	Page selected was not allowed

2.8 RS232

The RS232 on all Pluto boards is driven in the same way. The baud rate defaults to 9600 baud and whilst this can be upgraded to 19.2K baud, this is not recommended. When sending to PLUTO the following sequence should be carried out :-

1. Wait for CTS to go high
2. Send byte

When receiving from PLUTO you must make sure DTR is kept high and PLUTO will send data to you. You can cause PLUTO to abandon sending by lowering DTR and then raising it again.

SECTION THREE - PLUTO FOR THE SIRIUS

The Pluto graphics card for the Sirius microcomputer is contained on a 9" x 5" multilayer Sirius board, and uses a 16 bit 8088 processor which has direct access to 256 KBytes of memory. The screen resolution is a standard 768H x 576V (Interlaced) which can be software selected to be 2 independently displayable screens of 768H x 288V (Non-Interlaced).

Each pixel (made up of 4 bits of memory) can be set to 1 of 16 fixed colours.

All the commands and facilities supplied with the Pluto I system are also supplied for Pluto on the Sirius microcomputer. These commands and facilities are detailed in sections 2.3 to 2.6 inclusive of this manual.

Note: As a general introduction to the Pluto graphics controller, Section 1, Introduction to Pluto, can be read as an overview which applies to all Pluto packages. Section 1 includes a brief description of the terms and references used in this manual, and also describes how the frame buffer is divided into partitions, and how pixels are defined in order to display different colour combinations. Also supplied with the Pluto Sirius package is a demonstration disk containing a comprehensive graphics interface library. This allows Pluto to be used directly from programs written in any high level language (e.g. 'C', BASIC, Pascal, Fortran) running under the MS-DOS operating system. The interface routines are written in assembler. The disk also contains various utility programs and a Help file on the disk (README.TXT) explains the software interface. The Demonstration disk (and the programs stored on the Demonstration disk) is further explained in Section 9 of this manual.

SECTION FOUR - PLUTO FOR THE IBM PC

4.1 Introduction

The Pluto graphics card for the IBM PC range (XT, AT and compatibles) uses a 68000 processor which has direct access to either 256 or 384, 1024 or 1536 KBytes of memory (dependent on the version). The board is a 6 layer IBM PC format board and there are 6 hardware configurable versions available:

1. 256 KBytes of memory. 768H x 576V (Interlaced) or 768H x 288V (2 screens of Non-Interlaced). This version uses 4 bits per pixel with a choice of 16 colours.

2. 384 KBytes of memory. 768H x 576V (Non-Interlaced, 2 screens). This version uses 3 bits per pixel with a choice of 8 colours.

3. 384 KBytes of memory. 1024H x 768V (Interlaced). This version also uses 3 bits per pixel with a choice of 8 colours.

4. 1024 KBytes of memory 768H x 576V (4 screens Interlaced) or 768h x 288v (8 screens Non-Interlaced). This version uses 4 bits per pixel with a choice of 16 colours.

5. 1536 KBytes of memory. 768h x 576v (Non-Interlaced, 8 screens). This version uses 3 bits per pixel with a choice of 8 colours.

6. 1536 KBytes of memory. (4 Interlaced screens). This version also uses 3 bits per pixel with a choice of 8 colours.

The Pluto IBM system is fully compatible with the Pluto I system and includes all the Pluto I commands. There are also extra facilities and commands supplied with the Pluto IBM system: a Pan command which allows vertical scrolling of the screen image, and the facility to load a macro or user-defined code.

Also supplied with the Pluto IBM package is a Demonstration disk containing a comprehensive graphics interface library. This allows Pluto to be used directly from programs written in any high level language (e.g. 'C', BASIC, Pascal, Fortran) running under the MS-DOS operating system. The interface routines are written in assembler. The Demonstration disk also contains various utility programs and a Help file on the disk (README.TXT) explains the software interface. The Demonstration disk (and the programs stored on the Demonstration disk) is further explained in Section 9 of this manual.

All the commands supplied with the Pluto I system are supplied for the Pluto IBM system. These commands are detailed in sections 2.3 to 2.6 inclusive) of this manual.

Note: As a general introduction to the Pluto graphics controller, Section 1, Introduction to Pluto, can be read as an overview which applies to all Pluto packages. Section 1 includes a brief description of the terms and references used in this manual, and also describes how the frame buffer is divided

into partitions, and how pixels are defined in order to display different colour combinations.

The procedures for installing the 6 hardware versions of Pluto for the IBM PC are detailed in Appendix F of this Manual.

4.2 Pluto IBM PC Commands

The format of the parameters for the Pluto IBM PC commands is the same as described for Pluto I (see Parameter definitions, Section 2.4.2), a 'B' suffix signifies a byte value and a 'W' suffix signifies a word value sent as two bytes (low first). The following is a list of the parameters used with the Pluto IBM commands:

XW	-	co-ordinate
YW	-	Y co-ordinate
ColourB	-	colour
PartitionB	-	a partition identifier
WidthW	-	width
HeightW	-	Height (number of partition lines)
NlinesW	-	number of lines (for symbol space)
CoixW	-	centre of interest (X co-ordinate)
CoiyW	-	centre of interest (Y co-ordinate)
ScrollModeB	-	scrollmode
StartYW	-	starting point of a partition (Y co-ordinate)
SymbolWidthW	-	symbol width
SymbolHeightW	-	symbol height
NSymbolsB	-	number of symbols
PTypeB	-	partition type
HardwareTypeB	-	graphics board identifier
HardwareSubTypeB	-	configuration details
SoftwareVersionB	-	software version
NBytesW	-	number of bytes
OffsetW	-	offset
CountB	-	count

The following are the extra commands supplied with the Pluto IBM PC system:

```
COMMAND COPYS (ASCII VALUE)
LIBRARY PROCEDURE PChar (Var Sym : INT);
```

The following extensions (control codes) are interpreted while in the default symbol partition, i.e. Current Symbol Partition 255:

CR (Hex : 0D Decimal : 13)

Moves the Current Position (CP) to the beginning of the next line.

BS (Hex : 8 Decimal : 8)

Backspaces the Current Position by one character position.

TAB (Hex : 9 Decimal : 9)

Tabs the Current Position horizontally to an 8 character boundary.

LF (Hex : 0A Decimal : 10)

Line feeds the Current Position down by 12 lines, with scroll if necessary.

COMMAND SwapEnv (Hex : D6 Decimal : 214)
LIBRARY PROCEDURE SwapEnv;

The SwapEnv command sets up an alternate working environment. The variables changed are: the perimeter colour (PCOL), the current colour (CCOL), the foreground colour (FCOL), the background colour (BCOL), the pattern (PAT), the transparent colour (TCOL), the Current Working Partition (CWP) and the Current Symbol Partition (CSP). There are two working environments which are alternated each time the SwapEnv command is invoked.

COMMAND PInitW (NlinesW) (Hex : D8 Decimal : 216)
LIBRARY PROCEDURE PInitW (Var NLines : INT);

The PInitW command is the same as the PInit command but does not clear the displayable area. Nlines must be set to zero to give the default symbol space.

COMMAND ICoi : CoixW, CoiyW, ScrollModeB

(Hex : DF Decimal : 223)
LIBRARY PROCEDURE ICoi (Var Coix, Coiy, ScrollMode : INT);

The ICoi command returns the value of the current centre of interest (see the PanTo command).

COMMAND DefDP3 (StartYW, HeightW) (Hex : E2 Decimal : 226)
LIBRARY PROCEDURE DefDP3 (Var Start Y, Height : INT);

The DefDP3 command defines the start and height of display partition 3. Display partition 3 is a user-defined partition. The StartY parameter is the start point of the partition (Y co-ordinate) and the Height parameter is the number of lines in the partition.

```
COMMAND DefWP3 (StartYW, HeightW)       (Hex : E3 Decimal : 227)
LIBRARY PROCEDURE DefWP3 (Var Start Y, Height : INT);
```

The DefWP3 command defines the start and height of working partition 3. Working partition 3 is a user-defined partition. The StartY parameter is the start point of the partition (Y co-ordinate) and the Height parameter is the number of lines in the partition.

```
COMMAND PanTo (XW, YW, ScrollModeB)     (Hex : E4 Decimal : 228)
LIBRARY PROCEDURE PanTo (Var X, Y, ScrollMode : INT);
```

The PanTo command moves the centre of interest (Coix, Coiy) of the viewing window and sets the scrolling mode. A scrollmode of zero tries to centre the Coi at the centre of the screen. The display is panned to bring the Coi as close as possible to the screen centre within the constraints of the boundary of the current display partition. As it is not possible to 'pan' from side to side, the X parameter will be 0 if the scrollmode is 1, or screen width divided by 2, if the scrollmode is 0.

A scrollmode of 1 makes the top of the screen the Coi. The visible window is allowed to wraparound to the start of the display partition if necessary. This is the mode that would be used for scrolling text, for example. The Coi and scrollmode (and the interlace mode) are kept independently for each display partition so that, for example, changes to the Coi while in partition 1 don't affect the Coi for partition 2.

```
COMMAND IWP : YW,HeightW,SymbolWidthW,SymbolHeightW,NSymbolsB,PTypeB
                                        (Hex : E5  Decimal : 229)
LIBRARY PROCEDURE IWP (Var Y, Height, Symbol Width, Symbol Height,
                                              NSymbols, PType : INT);
```

The IWP command gives information on the current working partition. The Type variable indicates a symbol/working partition (type 0), a code partition (type 1) or a macro partition (type 2).

```
COMMAND IDP : YW,HeightW                (Hex : E6  Decimal : 230)
LIBRARY PROCEDURE IDP (Var Y, Height : INT);
```

The IDP command gives information on the current display partition. The Y parameter is the start point of the partition (Y co-ordinate) and the Height is the number of lines in the partition.

```
COMMAND IVersion : HardwareTypeB,HardwareSubTypeB,SoftwareVersionB
                                        (Hex : E7 Decimal : 231)
LIBRARY PROCEDURE IVersion (Var HardwareType,HardwareSubType,Soft
```

The IVersion command identifies the board. The IBM board is a board of type 2. The HardwareSubType gives configuration information. The 6 hardware sub-types are as stated overleaf.

```
1:   768 x 576 Interlaced
2:   768 x 576 Non-Interlaced
3:   1024 x 768 Interlaced
129: 768 x 576 4 Interlaced screens
130: 768 x 576 8 Non-Interlaced screens
131: 1024 x 768 4 Interlaced screens
```

```
COMMAND AllocCM (PTypeB, NBytesW) : PartitionB
                                        (Hex : E9  Decimal : 233)
LIBRARY PROCEDURE AllocCM (Var PType, NBytes, Partition : INT);
```

The AllocCM command can be used to allocate a partition to store NBytes of executable code (type 1) or macros (type 2). The number of bytes of 68000 object code should always be even.

```
COMMAND LoadCM (PartitionB, OffsetW, NBytesW)
                                        (Hex : EA  Decimal : 234)
LIBRARY PROCEDURE LoadCM (Var Partition, Offset, NBytes, Buf : INT);
```

The LoadCM command can be used to load NBytes into the specified partition at the specified offset. The number of bytes of 68000 object code should always be even.

```
COMMAND ReadCM (PartitionB, OffsetW, NBytesW) : 1B, 2B,....nB
                                        (Hex : EB  Decimal : 235)
LIBRARY PROCEDURE ReadCM (Var Partition, Offset, NBytes, Buf : INT);
```

The ReadCM command returns the NBytes bytes stored in the specified partition at the specified offset. The number of bytes of 68000 object code should always be even.

```
COMMAND MRet                            (Hex : D5  Decimal : 213)
LIBRARY PROCEDURE MRet;
```

The MRet command can be used to terminate a macro instruction, handling control back to the IBM firmware. If the MRet command is executed from the host, error 30 will be returned by the IStat command. This can be useful in identifying which Pluto board you are addressing - Pluto I and Sirius gives a status of 02 (not a command) and Pluto II, megares and IBM - 30.

```
COMMAND ExecCM (PartitionB)             (Hex : EC  Decimal : 236)
LIBRARY PROCEDURE ExecCM (Var Partition : INT);
```

The ExecCM command executes the specified code/macro partition. The macro execution will stop at a non-zero IStat result, excessive macro nesting or an MRet command.

```
COMMAND ReleaseP (PartitionB)        (Hex : ED  Decimal : 237)
PROCEDURE ReleaseP (Var Partition : INT);
```

The ReleaseP command frees the specified partition and partition number for re-use. The symbol space is compressed to prevent memory fragmentation.

```
COMMAND IDWindow : XW, YW, WidthW, HeightW
                                     (Hex : EE  Decimal : 238)
LIBRARY PROCEDURE IDWindow (Var X, Y, Width, Height : INT);
```

The IDWindow command returns the size and position of the currently viewed portion of the display partition. This command is useful in showing what is being displayed after using the Pan command.

```
COMMAND RLLImage (WidthW,HeightW,Count1B,Colour1B,....
                                        ...CountnB,ColournB)
                                     (Hex : F0  Decimal : 240)
LIBRARY PROCEDURE RLLImage (Var Flag, Width, Height, Buf, Count : INT);
```

The RLLImage command (Run Length Load Image) loads a run-length encoded image of specified width and height. The pixel data is sent as pairs of count and colour.

Note that the 'Flag' parameter in the Library routine should be set to zero when the routine is first called. The routine will set 'Flag' to zero when a terminator (0,0 integer pair) is loaded. If the terminating (0,0) pair is not found 'Flag' will be set to '-1'.

```
COMMAND RLRImage (WidthW,HeightW) : Count1B,Colour1B...
                                     ...CountnB,ColournB,0,0
                                     (Hex : F1  Decimal : 241)
LIBRARY PROCEDURE (Var Flag, Width, Height, Buf, Count : INT);
```

The RLRImage command (Run Length Read Image) reads back a run-length encoded image of specified width and height. The pixel data is sent as pairs of count and colour and is terminated by a 0,0 pair.

Note that the 'Flag' parameter should be set to zero when the routine is first called.

4.3 Macros and User Loaded Code

Extra commands can be loaded onto the IBM card either as macros or user defined code. A macro is a sequence of bytes that are interpreted as commands and User code is executable 68000 object code.

To use this facility, a partition for storing the code/macro must be allocated using the AllocCM command (to allocate a code/macro partition). This reserves the specified amount of space in the

workspace area and also specifies whether a macro or user code is to be stored there.

4.3.1 Macros

The LoadCM command can be used to define the sequence of instructions that make up a macro. As an example, suppose the macro is to draw a rectangle of size 32 x 32 in the current colour. The sequence of bytes to be loaded with the LoadCM command would be: 129, 32, 0, 32, 0 (129 is the Rfill command code). Any number of commands can be strung together in a macro. The IBM card works through the sequence of bytes, taking the first byte as the command number and reading subsequent bytes if the command uses parameters. After executing the command, the IBM card continues from where it left off in the sequence of bytes to get the next command.

Note: The last command in a macro must be the MRet command. This tells the IBM card to stop executing the macro and wait for a new command.

The LoadCM command allows macros to be modified or extended by loading at an offset position within the macro. The macro will not be activated until an ExecCM command is specified.

The ExecCM command can be used within a macro to execute another macro. This nesting is limited by the size of the available stack space. The stack space is equal to the size of the user work space on start up. If a macro has been executed from another macro, then the MRet command will return control to the previous macro.

4.3.2 User loaded code

User code can be loaded into a code partition with the LoadCM command. The code must be 68000 in binary format. On entry A0 points to the base of the allocated partition. On exit the following return procedures should be followed.

1. The stack pointer should be restored to the value on entry.

2. The low 16 bits of register D1 should contain an IStat return code that will be passed to the host if the IStat command is used.

3. The low 16 bits of register D0 should contain the number of bytes to be sent back to the host.

4. (If register D0 is not equal to 0). A pointer to an array of 16 bit words, each of which must contain an 8 bit value to be returned to the host, should be left in register A0.

5. A "RTS" instruction should be used to return to the IBM Pluto firmware.

Reserved Registers:

The address registers A3, A4, A5 and A6 are reserved by the IBM Pluto firmware and should not be altered at any time. The high 16 bits of data register D7 are set to 0 and again should be unaltered. All other registers are free for the user to manipulate.

To use the IBM card functions from user code the following conventions are used:

1. Parameters are passed to/from the firmware in a buffer pointed at by the A0 address register. Buffers have the same format as previously described, i.e. only the low 8 bits of each 16 bit word are valid.

2. Interface calls are performed via software Traps 0 through to 4 which are detailed below:

Trap 0 "User call". The standard firmware commands may be invoked by pointing the address register A0 at a buffer containing standard command codes plus parameters (if any). On return the D1 register contains the status code returned by this command. D0 contains the number of bytes returned by this command and if D0 is not equal to zero, register A0 points to the start of a returned buffer. A1 points at the next word following the request buffer.

Trap 1 "Read from host". The low 16 bits of register D0 should contain the number of bytes to be read from the host, i.e. the IBM. The A0 address register should contain a pointer to a standard buffer large enough to contain all the bytes to be read.

Note: Each new address in the buffer contains a 16 bit word value, of which only the low 8 bits are valid. This routine will ignore a count of zero. No registers are affected.

Trap 2 "Transfer to host". The low 16 bits of register D0 should contain the number of bytes to be sent to the host, i.e. the IBM. The A0 address register should be loaded with a pointer to a standard buffer containing all the bytes to be transferred.

Note: Each new address in the buffer contains a 16 bit word value, of which only the low 8 bits are valid. This routine will ignore a count of zero. No registers are affected.

Trap 3 "Pluto Initialise". Identical to the PInit standard Pluto command. Can be used to re-initialise the hardware after an unrecoverable error condition.

Trap 4 "Trap VDU control characters". The A0 register should contain a pointer to a routine capable of handling characters in the range 0 to 1FH. These characters from the default character set (CSP = 255) can be routed to a user routine. The control character handler should return status in the normal way, i.e. Status in register D1, count and pointer in register D0/A0 respectively. Control may be

returned to the firmware handler by loading register A0 with the value 0, and issuing a "Trap 4" instruction.

Note: The firmware does not validate the address of the routine being called. If a partition is released, all partitions allocated after this partition will be moved to avoid memory fragmentation. It is therefore advisable to allocate handlers first.

4.4 Pluto IBM PC Error codes

All the Pluto I error codes (returned by the IStat command) listed in Section 2.5 of this manual also apply to the Pluto IBM PC graphics card. There are also errors unique to the Pluto IBM PC graphics card:

Error Code	Meaning
28	Code/macros nested too deep
29	Partition is not executable
30	MRet encountered while not in a macro
32	User code too big
33	Can't release CWP or CSP
34	CDP too small for high res.
35	Can't release fill source partition
37	CDP and CWP not the same for scroll
38	CDP too small to scroll

SECTION FIVE - PLUTO II

5.1 Introduction

The Pluto II system is contained on a 12" x 8" board with an 8088 processor and includes 512 KBytes of memory. The memory can be increased by 512 KBytes, to give 1MByte of memory, if required. The possible screen resolutions are 768H x 576V (Interlaced) or 768H x 288V (Non-Interlaced). With the extra 512 KBytes of memory the virtual resolution can be increased to 768H x 1152V, which allows the storage of two images of 768H x 576V and increases the spare memory available for workspace.

Note: When using the interlaced mode for Pluto II (which can be specified with the SHires command, see Section 2.4.3 Housekeeping commands), 576 lines are displayed vertically with a total number of lines (including the blanking period) of 624. This may cause compatibility problems with video or T.V. equipment where 625 lines are normally used. To overcome this problem, a Pluto II command (the SetDispOdd command, see Section 5.3) can be used to change the display to 575 lines with a total of 625 lines.

The installation of Pluto II is detailed in Appendix F of this manual.

5.2 Additional Features of Pluto II

Pluto II has all the Pluto I commands as standard, but also includes extra features: a Pan command, a Zoom command, a facility to load a macro or user-defined code, a Palette which provides increased colour selection (a 256 colour choice from a palette of 16.7 million) or increased virtual screen resolution, and an optional real-time frame grabber.

The Pan facility can be used to move an image around the screen horizontally or vertically to reveal hidden parts. This also allows smooth vertical scrolling of images or text in single line increments.

The Zoom facility allows the screen picture to be magnified (by pixel replication) up to 16 times the original size without modifying the original image. This allows small detail to be easily examined. The point at which to zoom into is easily specified using Pluto II's Zoom command.

Pluto II also offers the facility of loading extra commands either as macros or user code. A macro is a sequence of bytes interpreted as commands and user code is 8088 code in binary format.

The serial port means that Pluto II can receive commands over a serial port (RS232 interface) or a parallel port, both of which are fitted as standard.

The Pluto II Framestore has 8 bits per pixel as standard. The Palette, which is 24 bits wide, uses 8 bits for the red, blue and

green colours, allowing a choice of 256 simultaneous displayable colours from a palette of 16.7 million.

The optional real-time frame grabber allows the capture of an image from a video camera in 128 grey levels at 768H x 576V resolution in 1/25th of a second. When the image has been captured it can be manipulated using Pluto II's other commands to add colouring, enhance contrast, analyse the image or add graphics to it.

All the commands and facilities supplied with the Pluto I package are available with the Pluto II package. These commands and facilities are detailed in sections 2.3 to 2.6 inclusive of this manual.

Note: As a general introduction to the Pluto graphics controller, Section 1, Introduction to Pluto, can be read as an overview which applies to all Pluto packages. Section 1 includes a brief description of the terms and references used in this manual, and also describes how the frame buffer is divided into partitions, and how pixels are defined in order to display different colour combinations.

5.3 Pluto II Commands

The format of the parameters for the Pluto II commands is the same as described for Pluto I (see Parameter definitions, Section 2.4.2). The parameters are shown as byte values or word values. A 'B' suffix signifies a byte value and a 'W' suffix signifies a word value sent as two bytes (low first).

The following is a list of the parameters used with the Pluto II commands:

XW	-	X co-ordinate
YW	-	Y co-ordinate
ColourB	-	colour
PartitionB	-	a partition identifier
WidthW	-	width
HeightW	-	height (number of partition lines)
NlinesW	-	number of lines (for symbol space)
NVerticesB	-	number of vertices
ShadeW	-	shade
On/OffB	-	camera preview toggle
ZoomfactorB	-	zoomfactor
CoixW	-	centre of interest (X co-ordinate)
CoiyW	-	centre of interest (Y co-ordinate)
ScrollModeB	-	scrollmode
StartYW	-	starting point of a partition (Y-co-ordinate).
SymbolWidthW	-	symbol width
SymbolHeightW	-	symbol height
NSymbolsB	-	number of symbols
PTypeB	-	partition type
HardwareTypeB	-	graphics board identifier
HardwareSubTypeB	-	configuration details
SoftwareVersionB	-	software version
True/FalseB	-	displayed lines/total lines toggle

NBytesW	-	number of bytes (0 = False, 1 = True)
OffsetW	-	offset
CountB	-	count
iB	-	look-up table entry
nB	-	number of look-up table entries
gB	-	green value
bB	-	blue value
rB	-	red value
PageB	-	look-up table page

The following are the extra commands supplied with the Pluto II graphics controller:

COMMAND COPYS (ASCII value)
LIBRARY PROCEDURE PChar (Var Sym : INT);

The following extensions (control codes) are interpreted while in the default symbol partition:

CR (Hex : 0D Decimal : 13)

Moves the Current Position (CP) to the beginning of the next line.

BS (Hex : 8 Decimal : 8)

Backspaces the Current Position by one character position.

TAB (Hex : 9 Decimal : 9)

Tabs the Current Position horizontally to an 8 character boundary.

LF (Hex : 0A Decimal : 10)

Line feeds the Current Position down by 12 lines, with scroll if necessary.

COMMAND MRet (Hex : D5 Decimal : 213)
LIBRARY PROCEDURE MRet;

The MRet command is used at the end of a macro to indicate to the firmware that the macro is finished. Control is then passed back to the firmware. If the MRet command is executed from the host, error 30 will be returned by the Istat command. This can be useful in identifying which Pluto board you are addressing. Pluto I and Sirius give a status of 02 (not a command) and Pluto II, megares and IBM - 30.

```
COMMAND SwapEnv                          (Hex : D6  Decimal : 214)
LIBRARY PROCEDURE SwapEnv;
```

The SwapEnv command sets up an alternate working environment. The variables changes are: the Current Colour (CCOL), the foreground colour (FCOL), the background colour (BCOL), the pattern (PAT), the transparent colour (TCOL), the Current Working Position (CWP) and the Current Symbol Partition (CSP). There are 2 working environments which are alternated each time the SwapEnv command is invoked.

```
COMMAND PInitW (NlinesW)                 (Hex : D8  Decimal : 216)
LIBRARY PROCEDURE PInitW (Var NLines : INT);
```

The PInitW command is the same as the PInit command, but it does not clear the displayable area. The Nlines variable can be used to reserve extra lines for symbol space (0 gives the default symbol space).

```
COMMAND FGrab                            (Hex : D9  Decimal : 217)
LIBRARY PROCEDURE FGrab;
```

The FGrab command grabs a complete screenfull of information from the camera input in a single frame time. This command grabs in 128 grey levels (the top bit (bit 7) of the frame store is always zero).

```
COMMAND CPoly (ColourB, NVerticesB, X1W,Y1W,X2W,Y2W,....
                                                ....XnW,YnW)
                                       (Hex : DA  Decimal : 218)
LIBRARY PROCEDURE CPoly (Var Colour NVertices, Buf : INT);
```

The CPoly command draws a solid polygon with NVertices vertices (up to 255) in the specified colour. The co-ordinates are supplied multiplied by 16 to offer greater accuracy.

```
COMMAND GPoly (NVerticesB, X1W,Y1W,Shade1W,..
                                            ...XnW,YnW,ShadenW)
                                       (Hex : DB  Decimal : 219)
LIBRARY PROCEDURE GPoly (Var NVertices, Buf : INT);
```

The GPoly command draws a smoothly shaded polygon with NVertices vertices (up to 255). Each vertex is supplied with its own shade - Gouraud shading is used to interpolate shading between polygon edges. The co-ordinates and shading values are supplied multiplied by 16 to offer greater accuracy.

COMMAND Preview (On/OffB) (Hex : DC Decimal : 220)
LIBRARY PROCEDURE Preview (Var Onflag : INT);

The Preview command turns the camera preview on or off. A parameter of 1 turns it on and a 0 turns it off. This command provides to the frame store a non-destructive real-time view of what the camera is seeing.

COMMAND SZoom (ZoomfactorB) (Hex : DD Decimal : 221)
LIBRARY PROCEDURE SZoom (Var Zoomfactor : INT);

The SZoom command zooms the display by pixel replication. The zoomfactor can be from 1 to 16.

COMMAND IZoom : ZoomfactorB (Hex : DE Decimal : 222)
LIBRARY PROCEDURE IZoom (Var Zoomfactor : INT);

The IZoom command returns the value of the current zoom factor.

COMMAND ICoi : CoixW, CoiyW, ScrollModeB
(Hex : DF Decimal : 223)
LIBRARY PROCEDURE Icoi (Var Coix, Coiy, ScrollMode : INT);

The ICoi command returns the value of the current centre of interest (see the PanTo command).

COMMAND LutInitCam (Hex : E1 Decimal : 225)
LIBRARY PROCEDURE LutInitCam;

The LutInitCam command initialises the first 128 entries of the look-up table according to the rule: entry i:=2*i. This is a grey scale suitable for frame grabbed images.

COMMAND DefDP3 (StartYW, HeightW)
(Hex : E2 Decimal : 226)
LIBRARY PROCEDURE DefDP3 (Var StartY, Height : INT);

The DefDP3 command defines the start and height of display partition 3. Display partition 3 is a user-defined partition. The StartY parameter is the start point of the partition (Y co-ordinate) and the Height parameter is the number of lines in the partition.

```
COMMAND DefWP3 (StartYW, HeightW)   (Hex : E3 Decimal : 227)
LIBRARY PROCEDURE DefWP3 (Var StartY, Height : INT);
```

The DefWP3 command defines the start and height of working partition 3. Working partition 3 is a user-defined partition. The StartY parameter is the start point of the partition (Y co-ordinate) and the Height parameter is the number of lines in the partition.

```
COMMAND PanTo (XW, YW, ScrollmodeB) (Hex : E4  Decimal : 228)
LIBRARY PROCEDURE PanTo (Var X,Y, ScrollMode : INT);
```

The PanTo command moves the centre of interest (Coix, Coiy) of the viewing window and sets the scrolling mode. A scrollmode of zero tries to centre the Coi at the centre of the screen. The display is panned to bring the Coi as close as possible to the screen centre within the constraints of the current zoom factor and the boundary of the current display partition.

A scrollmode of 1 makes the top of the screen the Coi. The visible window is allowed to wraparound to the start of the display partition if necessary. This is the mode that would be used for scrolling text, for example. The Coi and scrollmode (and the interlace mode) are kept independently for each display partition so that, for example, changes to the COI while in partition 1 do not affect the COI for partition 2.

```
COMMAND IWP : YW,HeightW,SymbolWidthW,SymbolHeightW, NSymbolsB,PTypeB
                                        (Hex : E5  Decimal : 229)
LIBRARY PROCEDURE IWP (Var Y, Height, SymbolWidth, SymbolHeight,
                                              NSymbols, PType : INT);
```

The IWP command gives information on the current working partition. The Type variable indicates a symbol/working partition (type 0), a code partition (type 1) or a macro partition (type 2).

```
COMMAND IDP : YW,HeightW              (Hex : E6  Decimal : 230)
LIBRARY PROCEDURE IDP (Var Y, Height : INT);
```

The IDP command gives information on the current display partition. The Y parameter is the start point of the partition (Y co-ordinate) and the Height is the number of lines in the partition.

```
COMMAND IVersion : HardwareTypeB,HardwareSubTypeB,SoftwareVersionB
                                        (Hex : E7  Decimal : 231)
LIBRARY PROCEDURE IVersion (Var HardwareType, HardwareSubType,
                                           SoftwareVersion : INT);
```

The IVersion command identifies the board. Pluto II is a board of type 1. The HardwareSubType gives configuration information, as stated overleaf.

Bit 7 set = Extended Memory
Bits 6-4 = Bank number (4 = master or green bank)
Bits 0-3 = baud rate (14 = 9600 baud)

```
COMMAND SetDispOdd (True/FalseB)     (Hex : E8  Decimal : 232)
LIBRARY PROCEDURE SetDispOdd (Var Onflag : INT);
```

The SetDispOdd command can be used to change the number of displayed lines and the total number of lines. If True/False=1 when the high resolution mode is used, the number of displayed lines is 575 and the total number of lines is 625. If True/False=0 when the high resolution mode is used, the number of displayed lines is 576 and the total number of lines is 624.

```
COMMAND AllocCM (PTypeB, NBytesW) : PartitionB
                                     (Hex : E9  Decimal : 233)
LIBRARY PROCEDURE AllocCM (Var PType, NBytes, Partition : INT);
```

The AllocCM command can be used to allocate a partition to store NBytes of executable code (type 1) or macros (type 2). If the returned value is partition 255 (0ff hex), then an error is being indicated (the function has failed to allocate a partition as there are no more free partitions available).

```
COMMAND LoadCM (PartitionB, OffsetW, NBytesW)
                                     (Hex : EA  Decimal : 234)
LIBRARY PROCEDURE LoadCM (Var Partition, Offset, NBytes, Buf : INT);
```

The LoadCM command can be used to load NBytes into the specified partition at the specified offset.

```
COMMAND ReadCM (PartitionB, OffsetW, NBytesW) : 1B, 2B,....nB
                                     (Hex : EB  Decimal : 235)
LIBRARY PROCEDURE ReadCM (Var Partition, Offset, NBytes, Buf : INT);
```

The ReadCM command returns the NBytes bytes stored in the specified partition at the specified offset.

```
COMMAND ExecCM (PartitionB)          (Hex : EC  Decimal : 236)
LIBRARY PROCEDURE ExecCM (Var Partition : INT);
```

The ExecCM command executes the specified code/macro partition.

```
COMMAND ReleaseP (PartitionB)        (Hex : ED  Decimal : 237)
LIBRARY PROCEDURE ReleaseP (Var Partition : INT);
```

The ReleaseP command frees the specified partition and partition number for re-use. The symbol space is compressed to prevent memory fragmentation.

```
COMMAND IDWindow : XW, YW, WidthW, HeightW
                                        (Hex : EE  Decimal : 238)
LIBRARY PROCEDURE IDWindow (Var X, Y, Width, Height : INT);
```

The IDWindow command returns the size and position of the currently viewed portion of the display partition. This command is useful in showing what is being displayed after using the Zoom and Pan commands.

```
COMMAND RLLImage (WidthW,HeightW,Count1B,Colour1B,...CountnB,ColourB)
                                        (Hex : F0  Decimal : 240)
LIBRARY PROCEDURE RLLImage (Var Flag, Width, Height, Buf, Count:INT);
```

The RLLImage command (Run Length Load Image) loads a run-length encoded image of specified width and height. The pixel data is sent as pairs of count and colour.

Note that in the high level language interface library version of the command the parameter 'Flag' should be set to zero when the procedure is first called. 'Flag' will only be returned as zero if a 0.0 (terminator) pair is loaded. This is to allow the image to be loaded in several slices. The pairs of count and colour should be in an integer array pointed to by Buf. The parameter 'count' should contain the number of pairs to be loaded. 'Count' will be returned set to the number of pairs actually loaded.

```
COMMAND RLRImage (WidthW,HeightW) : Count1B,Colour1B...
                                       ...CountnB,ColournB,0,0
                                    (Hex : F1  Decimal : 241)
LIBRARY PROCEDURE (Var Flag, Width, Height, Count-Actual);
```

The RLRImage command (Run Length Read Image) reads back a run-length encoded image of specified width and height. The pixel data is sent as pairs of count and colour and is terminated by a 0,0 pair.

5.4 Macros and User Loaded Code

Extra commands can be loaded onto Pluto II either as macros or user code. A macro is a sequence of bytes that are interpreted as commands and User code is executable 8088 object code.

To use this facility, a partition for storing the code/macro must be allocated using the AllocCM command (to allocate a code/macro partition). This reserves the specified amount of space in the workspace area and also specifies whether a macro or user code is to be stored there.

Note: When allocating a macro partition some extra space is automatically allocated for an internal working area.

5.4.1 Macros

The LoadCM command can be used to define the sequence of instructions that make up a macro. As an example, suppose the macro is to draw a rectangle of size 32 x 32 in the current colour. The sequence of bytes to be loaded with the LoadCM command would be: 129, 32, 0, 32, 0 (129 is the Rfill command code). Any number of commands can be strung together in a macro. Pluto II works through the sequence of bytes, taking the first byte as the command number and reading subsequent bytes if the command uses parameters. After executing the command, Pluto II continues from where it left off in the sequence of bytes to get the next command.

The LoadCM command allows macros to be modified or extended by loading at an offset position within the macro. The macro will not be activated until an ExecCM command is specified.

Note: The last command in a macro must be the MRet command. This tells Pluto II to stop executing the macro and wait for a new command.

The ExecCM command can be used within a macro to execute another macro (up to 32 levels of nesting are possible). If a macro has been executed from another macro, then the MRet command will return control to the previous macro.

5.4.2 User loaded code

User code can be loaded into a code partition with the LoadCM command. The code must be 8088 code in binary format. When an ExecCM command is invoked for a code partition, the data and stack segments (ds, ss) are set to be the same as the code segment (cs), and the stack pointer (sp) is pointed to the top of the allocated space. Segments and stacks can be set up, as required.

To use Pluto II's functions for user code, the following conventions must be used:

1. Parameters are passed to Pluto II in a buffer pointed at by ds:di.

   Note: The contents of di is destroyed after return from the system call.

2. The function code is passed in al.

3. If there are any values expected back from the function, ds:si must point to a buffer big enough to receive them. Note: the contents of si is destroyed after return from the system call.

4. With the above set up, an INT 6 call is made to get the required function to execute.

On return the only registers that are guaranteed to be intact are ds,ss,es.

There are 2 extra facilities available for user code: to send data to the host and to receive data from the host. These are available through another interrupt vector as follows:

1. SendHost: Set up ds:si to point to a buffer containing the information to send and cx to hold the number of bytes to be sent. Pass the command number for a SendHost operation (0) in al. Execute a software interrupt (INT 7).

2. GetHost: Set up ds:di to point to a buffer to receive the information and cx to hold the number of bytes to be fetched. Pass the command number for a GetHost operation (1) in al. Execute a software interrupt (INT 7).

To return from an ExecCM command (including a nested ExecCM) a FAR RET must be executed.

3. MemoryPointer: Put a Y offset from the working partition in register si. Pass the command number for a memory pointer operation (2) in al and execute a INT 7. The value returned in si is a segment pointer into the display memory for that Y location (to get the absolute memory address, the value returned should be multiplied by 16).

To abort a command an INT 5 can be executed. Register al should be loaded with an error number that may be interrogated by the host (using the IStat command). ABORT always passes control back to the command level even if executed from a nested macro.

Note: A user code partition may ExecCM a macro partition and vice versa. If a Pluto II function call fails to execute correctly (through illegal parameters for example), Pluto II will abort all code/macro execution back to the command level.

5.5 Palette Commands

The Pluto II palette provides commands for manipulating a colour look-up table. Two look-up tables are provided (referred to as pages) so that two completely independent sets of colour maps can be loaded onto the palette. The particular page to be used for colour mapping can be selected with a single command - this page is referred to as the LUT display page.

The look-up table can be modified to re-define the colour mapping - the look-up table that is modified is defined as the LUT working page (the working page can be the same as the display page). As there are only 256 LUT entries, the whole LUT can be re-defined within one frame period. This facility can be used to change large areas of image quickly instead of changing the actual pixel values in the image.

All visible LUT changes are performed during a vertical blanking period so that no display tearing or interference is seen.

The r, g, b (red, green and blue) values in the following Palette commands, which specify the intensity of each colour component, have a value of 0 (zero intensity) to 255 (peak intensity) and can be incremented in single units.

COMMAND LUTInitG (Hex : C3 Decimal : 195)
LIBRARY PROCEDURE LUTInitG;

The LUTInitG command initialises both pages of the look-up table to be a linear greyscale. The LUT working page and the LUT display page are both set to be 1.

COMMAND LUTInitC (Hex : C4 Decimal : 196)
LIBRARY PROCEDURE LUTInitC;

The LUTInitC command initialises both pages of the look-up table to be a selection of colours. Colours 0 to 7 are the saturated colours available with the stand-alone Pluto (black, green, blue, cyan, red, yellow, magenta and white). The LUT working page and the LUT display page are both set to be 1.

COMMAND WLut (iB, gB, bB, rB) (Hex : C5 Decimal : 197)
LIBRARY PROCEDURE WLut (Var : i, Gbrarray : INT);

The WLut command defines the colour component values of entry i in the working LUT page.

For example,

> WLut (1,255,255,255) would define LUT entry 1 to be the brightest white, so that every pixel with value 1 would appear on the screen as white. If the working LUT page is different from the display LUT page then the operation is performed immediately and is invisible until the display page is changed. If the working page is the same as the display page then the LUT is not updated until the next vertical blanking period to prevent screen disturbances.

COMMAND WLutM (nB, iB, gB, bB, rB,.....gB, bB, rB)
(Hex : C6 Decimal : 198)
LIBRARY PROCEDURE WLutM (Var n, i, Gbrarray : INT);

The WLutM command writes multiple look-up table entries. The g, b, r values are written into the n+1 LUT entries in the working LUT page starting at entry number i (n=0 to write one entry, n=255 to write 256 entries, etc). For example, WLut (1,1,255,0,0,128,128,128) would define pixel value 1 to be the brightest green, and value 2 to be half white (grey). This is equivalent to 2 WLut commands but is more convenient and efficient.

If the working page is the same as the display page then the LUT is not updated until the next blanking period to prevent screen

disturbances. It is much quicker to use the WLutM command rather than several WLut commands, as all of the entries are written during one blanking period rather than only one at a time.

```
COMMAND RLut (iB) : gB, bB, rB        (Hex : C7  Decimal : 199)
LIBRARY PROCEDURE RLut (Var i, Gbrarray : INT);
```

The RLut command returns the g, b, r values of entry i in the working LUT page. The values, which are returned in the order of green, blue and then red, are single byte quantities. This command is a complementary function to the WLut command.

```
COMMAND RLutM (nB, iB) : gB, bB, rB,.....gB, bB, rB
                                      (Hex : C8  Decimal : 200)
LIBRARY PROCEDURE RLutM (Var n, i, Gbrarray : INT);
```

The RLutM command returns the g, b, r values of n+1 entries starting at entry i in the working LUT page. The values, which are returned in the order of green, blue and then red, are single byte quantities. This command is a complementary function to the WLutM command.

```
COMMAND SLutWP (PageB)                (Hex : C9  Decimal : 201)
LIBRARY PROCEDURE SLutWP (Var Page : INT);
```

The SLutWP command sets the working LUT page to the specified Page (1 or 2). The Page parameter is a single byte.

```
COMMAND SLutDP (PageB)                (Hex : CA  Decimal : 202)
LIBRARY PROCEDURE SLutDP (Var Page : INT);
```

The SLutDP command sets the display LUT page to the specified Page (1 or 2). The Page parameter is a single byte.

```
COMMAND ILutWP : PageB                (Hex : CB  Decimal : 203)
LIBRARY PROCEDURE ILutWP (Var Page : INT);
```

The ILutWP command returns the value of the working LUT page.

```
COMMAND ILutDP : PageB                (Hex : CC  Decimal : 204)
LIBRARY PROCEDURE ILutDP (Var Page : INT);
```

The ILutDP command returns the value of the display LUT page.

5.6 Error Codes

All the Pluto I error codes (returned by the IStat command) listed in Section 2.5 of this manual also apply to Pluto II. There are also errors unique to Pluto II which are listed below.

5.6.1 Pluto II Error codes

Error Code	Meaning
20	Illegal bank number
21	Illegal zoom parameter
22	Illegal preview parameter
27	RS232 error
28	Code/macros nested too deep
29	Partition is not executable
30	MRet encountered while not in a macro
32	User code too big
33	Cannot release CWP or CSP
34	CDP too small for high resolution
35	Cannot release fill source partition
36	Cannot release alternate CWP or CSP
37	CDP and CWP are not the same for scroll
38	CDP to small too scroll
41	Zoom is not 1 for grab or preview

5.6.2 Palette Error codes

Error Code	Meaning
16	LUT index value/range too big
17	Page selected was not allowed

SECTION SIX - PLUTO II 24 BIT SYSTEM

6.1 Introduction

The Pluto II 24 Bit system is a special configuration of three Pluto II boards. One board is configured as the master and the others as slaves. The bank numbers used are 1, 2 and 4 (Bank 4 may also be referred to as bank 0 - for compatibility with the superseded Pluto I 24 bit systems).

Each board is populated with eight planes of memory. The palette on each board has only one video output (instead of three) so that all of the eight memory planes on a board are dedicated to one colour component (256 levels, independent of the other channels). The look-up table on each channel is retained, but works on each colour component individually. For example, the pixel values on the green channel can be defined to be any brightness level of green, but cannot be defined to have any blue or red components. This differs from the standard palette where each pixel value is defined to have components of red, green and blue.

In operation the 24 Bit system is treated as three eight bit banks (one each for red, green and blue). All of Pluto's commands can work on one or more banks simultaneously. The SBank command (see Palette commands, Section 6.4) can be used to enable a combination of banks.

The SBank command should be used in the following cases:

If a bank is disabled one has to send four zeros to the status port. After this, only the commands Pinit and SBank are recognised. A Pinit command leaves the green bank enabled and the red/blue bank disabled. Data may be sent to the enabled boards without problems. If data is read back from more than one enabled board, one must ensure that the data is identical in the enabled boards. If this is not the case, data corruption will occur. To avoid data clashes one should enable the individual boards and issue the read command.

The Pluto II 24 Bit system is configured with one common parallel interface. The serial port (which is not connected on a 24 Bit system) cannot be used to communicate with all three Pluto II boards simultaneously. It is possible, however, to use three separate RS232 interfaces to communicate to each board separately.

The features included in the Pluto II 24 Bit system are: a Pan command, a Zoom command, the facility to load a macro or user-defined code, and an optional real-time colour frame grabber.

The Pan command can be used to move an image around the screen horizontally or vertically to reveal hidden parts. This also allows smooth vertical scrolling of images or text in single line increments.

The Zoom command allows the screen picture to be magnified up to sixteen times the original size without modifying the original image.

This allows small detail to be easily examined. The degree of zoom required is easily specified using Pluto's Zoom command.

Extra commands can be loaded either as macros or user code. A macro is a sequence of bytes interpreted as commands and user code is 8088 code in binary format.

The optional real-time colour frame grabber allows the capture of an image from a video camera in 128 levels of red, green and blue at 768H x 576V resolution in 1/25th of a second giving a full colour frame grab. When the image has been captured it can be manipulated using Pluto's other commands to add colouring, enhance contrast, analyse the image or add graphics to it.

All the commands supplied with the Pluto I system are also supplied for the Pluto II 24 Bit system. These commands are detailed in section 2.3 to 2.6 inclusive of this manual.

Note: As a general introduction to the Pluto graphics controller, Section 1, Introduction to Pluto, can be read as an overview which applies to all Pluto packages. Section 1 includes a brief description of the terms and references used in this manual, and also describes how the frame buffer is divided into partitions, and how pixels are defined in order to display different colour combinations.

6.2 Pluto II 24 Bit System Commands

The format of the parameters for the Pluto II 24 Bit system commands is the same as described for Pluto I (see Parameter definitions, Section 2.4.2), with a 'B' suffix to signify a byte value and a 'W' suffix to signify a word value sent as two bytes (low first). The following is a list of the parameters used with the Pluto II 24 Bit system commands:

XW	-	X co-ordinate
YW	-	Y co-ordinate
ColourB	-	colour
PartitionB	-	a partition identifier
WidthW	-	width
HeightW	-	Height (number of partition lines)
NlinesW	-	number of lines (for symbol space)
NVerticesB	-	number of vertices
ShadeW	-	shade
On/OffB	-	camera preview toggle
ZoomfactorB	-	zoomfactor
CoixW	-	centre of interest (X co-ordinate)
CoiyW	-	centre of interest (Y co-ordinate)
ScrollModeB	-	scrollmode
StartYW	-	starting point of a partition (Y co-ordinate)
SymbolWidthW	-	symbol width
SymbolHeightW	-	symbol height
NSymbolsB	-	number of symbols

PTypeB	-	partition type
HardwareTypeB	-	graphics board identifier
HardwareSubTypeB	-	configuration details
SoftwareVersionB	-	software version
True/FalseB	-	displayed lines/total lines toggle
NBytesW	-	number of bytes (0 = false, 1 = true)
OffsetW	-	offset
CountB	-	count
iB	-	look-up table entry
nB	-	number of look-up table entries
vB	-	look-up table value
PageB	-	look-up table page
BankB	-	current bank

The following are the extra commands supplied with the Pluto II 24 Bit system:

```
COMMAND COPYS (ASCII value)
LIBRARY PROCEDURE PChar (Var Sym : INT);
```

The following extensions (control codes) are interpreted while in the default symbol partition:

CR (Hex : 0D Decimal : 13)

Moves the Current Position (CP) to the beginning of the next line.

BS (Hex : 8 Decimal : 8)

Backspaces the Current Position by one character position.

TAB (Hex : 9 Decimal : 9)

Tabs the Current Position horizontally to an 8 character boundary.

LF (Hex : 0A Decimal : 10)

Line feeds the Current Position down by 12 lines, with scroll if necessary.

```
COMMAND MRet                    (Hex : D5  Decimal : 213)
LIBRARY PROCEDURE MRet;
```

The MRet command is used at the end of a macro to indicate to the firmware that the macro is finished. Control is then passed back to

the firmware. If the MRST command is executed from the Host, error 30 will be returned by the IStat command. This can be useful in identifying which Pluto board you are addressing. Pluto I and Sirius give a status of 02 (not a command) and Pluto II, Megares and IBM - 30.

```
COMMAND SwapEnv                          (Hex : D6  Decimal : 214)
LIBRARY PROCEDURE SwapEnv;
```

The SwapEnv command sets up an alternate working environment. The variables changes are: the Current Colour (CCOL), the foreground colour (FCOL), the background colour (BCOL), the pattern (PAT), the transparent colour (TCOL), the Current Working Position (CWP) and the Current Symbol Partition (CSP). There are two working environments which are alternated each time the SwapEnv command is invoked.

```
COMMAND PInitW (NLinesW)                 (Hex : D8  Decimal : 216)
LIBRARY PROCEDURE PInitW (Var NLines : INT);
```

The PInitW command is the same as the PInit command, but it does not clear the displayable area. The Nlines variable can be used to reserve extra lines for symbol space (0 gives the default symbol space).

```
COMMAND FGrab                            (Hex : D9  Decimal : 217)
LIBRARY PROCEDURE FGrab;
```

The FGrab command grabs a complete screenfull of information from the camera input in a single frame time. This command grabs in 128 grey levels (the top bit (bit 7) of the frame store is always zero).

```
COMMAND CPoly (ColourB, NVerticesB, X1W,Y1W,X2W,Y2W,...XnW,YnW)
                                         (Hex : DA  Decimal : 218)
LIBRARY PROCEDURE CPoly (Var Colour, NVertices, Buf : INT);
```

The CPoly command draws a solid polygon with NVertices vertices (up to 255) in the specified colour. The co-ordinates are supplied multiplied by 16 to offer greater accuracy.

```
COMMAND GPoly (NVerticesB, X1W,Y1W,Shade1W,...XnW,YnW, ShadenW)
                                         (Hex : DB  Decimal : 219)
LIBRARY PROCEDURE GPoly (Var NVertices, Buf : INT);
```

The GPoly command draws a smoothly shaded polygon with NVertices vertices (up to 255). Each vertex is supplied with its own shade - Gouraud shading is used to interpolate shading between polygon edges. The co-ordinates and shading values are supplied multiplied to 16 to offer greater accuracy.

```
COMMAND Preview (On/OffB)              (Hex : DC  Decimal : 220)
LIBRARY PROCEDURE Preview (Var OnFlag : INT);
```

The Preview command turns the camera preview on or off. A parameter of 1 turns it on and a 0 turns it off. This command provides a non-destructive real-time view of what the camera is seeing.

```
COMMAND SZoom (ZoomFactorB)            (Hex : DD  Decimal : 221)
LIBRARY PROCEDURE SZoom (Var ZoomFactor : INT);
```

The SZoom command zooms the display by pixel replication. The zoomfactor can be from 1 to 16.

```
COMMAND IZoom : ZoomFactorB            (Hex : DE  Decimal : 222)
LIBRARY PROCEDURE IZoom (Var ZoomFactor : INT);
```

The IZoom command returns the value of the current zoom factor.

```
COMMAND ICoi : CoixW, CoiyW, ScrollModeB
                                       (Hex : DF  Decimal : 223)
LIBRARY PROCEDURE ICoi (Var Coix, Coiy, ScrollMode : INT);
```

The ICoi command returns the value of the current centre of interest (see the PanTo command).

```
COMMAND LutInitCam                     (Hex : E1  Decimal : 225)
LIBRARY PROCEDURE LutInitCam;
```

The LutInitCam command initialises the first 128 entries of the look-up table according to the rule: entry i:=2*i. This is for a grey scale for suitable frame grabbed images.

```
COMMAND DefDP3 (StartYW, HeightW)      (Hex : E2  Decimal : 226)
LIBRARY PROCEDURE DefDP3 (Var StartY, Height : INT);
```

The DefDP3 command defines the start and height of display partition 3. Display partition 3 is a user-defined partition. The StartY parameter is the start point of the partition (Y co-ordinate) and the Height parameter is the number of lines in the partition.

```
COMMAND DefWP3 (StartYW, HeightW)      (Hex : E3  Decimal : 227)
LIBRARY PROCEDURE DefWP3 (Var StartY, Height : INT);
```

The DefWP3 command defines the start and height of working partition 3. Working partition 3 is a user-defined partition. The StartY parameter is the start point of the partition (Y co-ordinate) and the Height parameter is the number of lines in the partition.

COMMAND PanTo (XW, YW, ScrollModeB) (Hex : E4 Decimal : 228)
LIBRARY PROCEDURE PanTo (Var X, Y, ScrollMode : INT);

The PanTo command moves the centre of interest (Coix, Coiy) of the viewing window and sets the scrolling mode. A scrollmode of zero tries to centre the Coi at the centre of the screen. The display is panned to bring the Coi as close as possible to the screen centre within the constraints of the current zoom factor and the boundary of the current display partition.

A scrollmode of 1 makes the top of the screen the Coi. The visible window is allowed to wraparound to the start of the display partition if necessary. This is the mode that would be used for scrolling text, for example. The Coi and scrollmode (and the interlace mode) are kept independently for each display partition so that, for example, changes to the Coi while in partition 1 do not affect the Coi for partition 2.

COMMAND IWP : YW,HeightW,SymbolWidthW,SymbolHeightW,NSymbolsB,PTypeB
(Hex : E5 Decimal : 229)
LIBRARY PROCEDURE IWP (Var Y, Height, SymbolWidth, SymbolHeight,
NSymbols,PType : INT);

The IWP command gives information on the current working partition. The Type variable indicates a symbol/working partition (type 0), a code partition (type 1) or a macro partition (type 2).

COMMAND IDP : YW,HeightW (Hex : E6 Decimal : 230)
LIBRARY PROCEDURE IDP (Var Y, Height : INT);

The IDP command gives information on the current display partition. The Y parameter is the start point of the partition (Y co-ordinate) and the Height is the number of lines in the partition.

COMMAND IVersion : HardwareTypeB,HardwareSubTypeB,SoftwareVersionB
(Hex : E7 Decimal : 231)
LIBRARY PROCEDURE IVersion (Var HardwareType, HardwareSubType,
SoftwareVersion : INT);

The IVersion command identifies the board. Pluto II is a board of type 1. The HardwareSubType gives configuration information, as follows:-

Bit 7 set	=	Extended Memory
Bits 6-4	=	Bank Number
Bits 0-3	=	Baud Rate (14 = 9600 baud) COMMAND

SetDispOdd (True/FalseB) (Hex : E8 Decimal : 232)
LIBRARY PROCEDURE SetDispOdd (Var OnFlag : INT);

The SetDispOdd command can be used to change the number of displayed lines and the total number of lines. If True/False=1 when the high resolution mode is used, the number of displayed lines is 575 and the total number of lines is 625. If True/False=0 when the high resolution mode is used, the number of displayed lines is 576 and the total number of lines is 624.

COMMAND AllocCM (PTypeB, NBytesW) : PartitionB
(Hex : E9 Decimal : 233)
LIBRARY PROCEDURE AllocCM (Var PType, NBytes, Partition : INT);

The AllocCM command can be used to allocate a partition to store NBytes of executable code (type 1) or macros (type 2). If the returned value is partition 255 (0ff hex), then an error is being indicated (the function has failed to allocate a partition as there are no more free partitions available).

COMMAND LoadCM (PartitionB, OffsetW, NBytesW)
(Hex : EA Decimal : 234)
LIBRARY PROCEDURE LoadCM (Var Partition, offset, NBytes : INT);

The LoadCM command can be used to load NBytes into the specified partition at the specified offset.

COMMAND ReadCM (PartitionB, OffsetW, NBytesW) : 1B, 2B, ...nB
(Hex : EB Decimal : 235)
LIBRARY PROCEDURE ReadCM (Var Partition, offset, NBytes, Buf : INT);

The ReadCM command returns the NBytes bytes stored in the specified partition at the specified offset.

COMMAND ExecCM (PartitionB) (Hex : EC Decimal : 236)
LIBRARY PROCEDURE ExecCM (Var Partition : INT);

The ExecCM command executes the specified code/macro partition.

COMMAND ReleaseP (PartitionB) (Hex : ED Decimal : 237)
LIBRARY PROCEDURE ReleaseP (Var Partition : INT);

The ReleaseP command frees the specified partition and partition number for re-use. The symbol space is compressed to prevent memory fragmentation.

```
COMMAND IDWindow : XW, YW, WidthW, HeightW
                                        (Hex : EE  Decimal : 238)
LIBRARY PROCEDURE IDWindow (Var X, Y, Width, Height : INT);
```

The IDWindow command returns the size and position of the currently viewed portion of the display partition. This command is useful in showing what is being displayed after using the Zoom and Pan commands.

```
COMMAND RLLImage (WidthW,HeightW,Count1B,Colour1B,CountnB,ColournB)
                                        (Hex : F0  Decimal : 240)
LIBRARY PROCEDURE RLLImage (Var Flag,Width,Height,Buf,Count : INT);
```

The RLLImage command (Run Length Load Image) loads a run-length encoded image of specified width and height. The pixel data is sent as pairs of count and colour.

In the high level language interface library version of the procedure the 'Flag' parameter should be set to zero the first time that the procedure is called. It will be set to -1 unless a terminator (0,0 integer pair) is loaded, at which point 'Flag' will be reset to zero. This facilitates loading an image in several slices. 'Buf' points to a succession of count, colour integer pairs and 'count' is the number of pairs to be loaded. 'Count' will be returned set to the number of pairs actually loaded on return.

```
COMMAND RLRImage (WidthW,HeightW) : Count1B,Colour1B.....
                                        CountnB,ColournB,0,0
                                    (Hex : F1  Decimal : 241)
LIBRARY PROCEDURE RLRImage (Var Flag,Width,Height,Buf,Count : INT);
```

The RLRImage command (Run Length Read Image) reads back a run-length encoded image of specified width and height. The pixel data is sent as pairs of count and colour and is terminated by a 0,0 pair.

The extra parameters in the library procedure are as for RLLImage.

6.3 Macros and User Loaded Code

Extra commands can be loaded onto a Pluto II board either as macros or user code. A macro is a sequence of bytes that are interpreted as commands and User code is executable 8088 object code.

To use this facility, a partition for storing the code/macro must be allocated using the AllocCM command (to allocate a code/macro partition). This reserves the specified amount of space in the workspace area and also specifies whether a macro or user code is to be stored there.

Note: When allocating a macro partition some extra space is automatically allocated for an internal working area.

6.3.1. Macros

The LoadCM command can be used to define the sequence of instructions that make up a macro. As an example, suppose the macro is to draw a rectangle of size 32 x 32 in the current colour. The sequence of bytes to be loaded with the LoadCM command would be: 129, 32, 0, 32, 0 (129 is the Rfill command code). Any number of commands can be strung together in a macro. Pluto II works through the sequence of bytes, taking the first byte as the command number and reading subsequent bytes if the command uses parameters. After executing the command, Pluto II continues from where it left off in the sequence of bytes to get the next command.

The LoadCM command allows macros to be modified or extended by loading at an offset position within the macro. The macro will not be activated until an ExecCM command is specified.

Note: The last command in a macro must be the MRet command. This tells Pluto II to stop executing the macro and wait for a new command.

The ExecCM command can be used within a macro to execute another macro (up to 32 levels of nesting are possible). If a macro has been executed from another macro, then the MRet command will return control to the previous macro.

6.3.2 User loaded code

User code can be loaded into a code partition with the LoadCM command. The code must be 8088 code in binary format. When an ExecCM command is invoked for a code partition, the data and stack segments (ds, ss) are set to be the same as the code segment (cs), and the stack pointer (sp) is pointed to the top of the allocated space. Segments and stacks can be set up, as required.

To use Pluto II's functions for user code, the following conventions must be used:

1. Parameters are passed to Pluto II in a buffer pointed at by ds:di. Note the contents of di is destroyed after the return of the system call.

2. The function code is passed in al.

3. If there are any return values expected back from the function, ds:si must point to a buffer big enough to receive

them. Note: the contents of si is destroyed after return from the system call.

4. With the above set up, an INT 6 call is made.

On return the only registers that are guaranteed to be intact are ds,ss,es.

There are two extra facilities available for user code: to send data to the host and to receive data from the host. These are available through another interrupt vector as follows:

1. SendHost:

   Set up ds:si to point to a buffer containing the information to send and cx to hold the number of bytes to be sent. Pass the command number for SendHost (0) in al. Execute a software interrupt (INT 7).

2. GetHost:

   Set up ds:di to point to a buffer to receive the information and cx to hold the number of bytes to be fetched. Pass the command number for GetHost (1) in al. Execute a software interrupt (INT 7).

   To return from an ExecCM command (including a nested ExecCM) a FAR RET must be executed.

3. MemoryPointer:

   Put a Y offset from the working partition in register si. Pass the command number (2) in al and execute a INT 7. The value returned in si is a segment pointer into the display memory for that Y location (to get the absolute memory address, the value returned should be multiplied by 16).

To abort a command an INT 5 can be executed. Register al should be loaded with an error number that may be interrogated by the host (using the IStat command). ABORT always passes control back to the command level even if executed from a nested macro.

Note: A user code partition may ExecCM a macro partition and vice versa. If a Pluto II function call fails to execute correctly (through illegal parameters for example), Pluto II will abort all code/macro execution back to the command level.

6.4 Palette Commands

The Pluto II 24 Bit system provides commands for manipulating the colour look-up table. Two look-up tables are provided (referred to as pages) on each of the three boards so that two completely independent sets of colour maps can be used by the palette. The particular page to be used for colour mapping can be selected with a single command - this page is referred to as the LUT display page.

The look-up table can be modified to re-define the colour mapping - the look-up table that is modified, is defined as the LUT working page (the working page can be the same as the display page). As there are only 256 LUT entries, the whole LUT can be re-defined within one frame period. This facility can be used to change large areas of image quickly instead of changing the actual pixel values in the image.

All visible LUT changes are performed during a vertical blanking period so that no display tearing or interference is seen.

The red, green and blue values (indicated by 'v' in the following Palette commands) specify the intensity of the colour component and can have a value of 0 (zero intensity) to 255 (peak intensity)

```
COMMAND LUTInitG                        (Hex : C3  Decimal : 195)
LIBRARY PROCEDURE LUTInitG;
```

The LUTInitG command initialises both pages of the look-up table to be a linear greyscale. The LUT working page and the LUT display page are both set to be 1.

Note: The look-up table has to be initialised on each of the three boards to obtain a greyscale.

```
COMMAND LUTInitC                        (Hex : C4  Decimal : 196)
LIBRARY PROCEDURE LUTInitC;
```

The LUTInitC command initialises both pages of the look-up table to be a selection of colours. Colours 0 to 7 are the saturated colours available with the stand-alone Pluto (black, green, blue, cyan, red, yellow, magenta and white). The LUT working page and the LUT display page are both set to be 1.

Note: The look-up table has to be initialised on each of the three boards to obtain a selection of colours.

```
COMMAND SLutWP (PageB)                  (Hex : C9  Decimal : 201)
LIBRARY PROCEDURE SLutWP (Var Page : INT);
```

The SLutWP command sets the working LUT page on the current board to the specified Page (1 or 2).

```
COMMAND SLutDP (PageB)                (Hex : CA  Decimal : 202)
LIBRARY PROCEDURE SLutDP (Var Page : INT);
```

The SLutDP command sets the display LUT page on the current board to the specified Page (1 or 2).

```
COMMAND ILutWP : PageB                (Hex : CB  Decimal : 203)
LIBRARY PROCEDURE ILutWP (Var Page : INT);
```

The ILutWP command returns the value of the working LUT page on the currently selected board.

```
COMMAND ILutDP : PageB                (Hex : CC  Decimal : 204)
LIBRARY PROCEDURE ILutDP (Var Page : INT);
```

The ILutDP command returns the value of the display LUT page on the currently selected board.

```
COMMAND SBank (BankB)                 (Hex : CF  Decimal : 207)
LIBRARY PROCEDURE SBank (Var Bank : INT);
```

The SBank command sets the current bank, i.e. specifies the bank to which commands are sent.

Note: The SBank command must be issued before any other commands (excluding PInit) can be issued, e.g. Bank 0 = green, Bank 1 = blue, Bank 2 = red.

```
COMMAND IBank : BankB                 (Hex : D0  Decimal : 208)
LIBRARY PROCEDURE IBank (Var Bank : INT);
```

The IBank command returns the value of the current bank.

```
COMMAND BWLut (iB, vB)                (Hex : D1  Decimal : 209)
LIBRARY PROCEDURE BWLut (Var i, v : INT);
```

The BWLut command modifies the LUT entry for the current bank, i.e. changes the green level if bank = 0, changes the blue entry if bank = 1 and changes the red entry if bank = 2. The new LUT entry is v.

```
COMMAND BWLutM (nB,iB,vB,.....vB)     (Hex : D2  Decimal : 210)
LIBRARY PROCEDURE BWLutM (Var n, i, Valarray : INT);
```

The BWLutM command modifies the LUT entries for the current bank, i.e. changes the green level if bank = 0, the blue entry if bank = 1 and the red entry if bank = 2. The new LUT entries are v.....v.

```
COMMAND BRLut (iB) : vB              (Hex : D3  Decimal : 211)
LIBRARY PROCEDURE BRLut (Var i, v : INT);
```

The BRLut command returns the value of a single colour component, i.e. green, blue or red according to the currently selected bank.

```
COMMAND BRLutM (nB,iB) : vB...vB     (Hex : D4  Decimal : 212)
LIBRARY PROCEDURE BRLutM (Var n, i, valarray : INT);
```

The BRLutM command returns the value of a single colour component, i.e. green, blue or red according to the currently selected bank.

6.5 Error Codes

All the Pluto I error codes (returned by the IStat command) listed in Section 2.5 of this manual also apply to the Pluto II 24 Bit system. There are also errors, listed below, which are unique to the Pluto II 24 Bit system.

6.5.1 Pluto II Error codes

Error Code	Meaning
20	Illegal bank number
21	Illegal zoom parameter
22	Illegal preview parameter
27	RS232 error
28	Code/macros nested too deep
29	Partition is not executable
30	MRet encountered while not in a macro
32	User code too big
33	Can't release CWP or CSP
34	CDP too small for high res.
35	Can't release fill source partition
36	Can't release alternate CWP or CSP
37	CDP and CWP are not the same for scroll
38	CDP too small to scroll
41	Zoom is not 1 for grab or preview

6.5.2 Palette Error codes

Error Code	Meaning
16	LUT index value/range too big
17	Page selected was not allowed

SECTION SEVEN - MEGARES SYSTEM

7.1 Introduction

The Megares system is contained on a 8" x 10" board with a 8088 processor which has direct access to 512 KBytes of memory. There are 2 versions of Megares: Version 1 has a resolution of 2 screens of 768H x 576V (Non-Interlaced) and version 2 has resolutions of 1024H x 768V (Interlaced) or 2 screens of 1024H x 384V (Non-Interlaced). Each version uses 4 bits of memory per pixel.

The Megares system is fully compatible with the Pluto I system and includes all the Pluto I commands. The Pluto I commands are detailed in sections 2.3 to 2.6 inclusive of this manual.

Note: As a general introduction to the Pluto graphics controller, Section 1, introduction to Pluto, can be read as an overview which applies to all Pluto packages. Section 1 includes a brief description of the terms and references used in this manual, and also describes how the frame buffer is divided into partitions, and how pixels are defined in order display different colour combinations.

There are also extra features included with the Megares system: a Pan command, a Zoom command, and a Palette which allows a 16 colour choice from a Palette of 4096.

The Pan command can be used to move an image around the screen horizontally or vertically to reveal hidden parts. This also allows smooth vertical scrolling of images or text in single line movements.

The Zoom command allows the screen picture to be magnified up to nine times the original size without modifying the original image. This allows small detail to be easily examined. The point at which to zoom into is easily specified using the Megares Zoom command.

The serial port means that the Megares system can receive commands over a serial port (RS232 interface) or a parallel port, both of which are fitted as standard.

The Palette increases the number of colours available to create screen images. The 16 possible pixel values are defined in terms of their red, green and blue values allowing a choice of 4096 colours.

7.2 Megares Commands

The format of the parameters for the Megares commands is the same as described for Pluto I (see Parameter definitions, Section 2.4.2), with a 'B' suffix to signify a byte value and a 'W' suffix to signify a word value sent as two bytes (low first).

The following is a list of the parameters used with the Megares commands:

XW	-	X co-ordinate
YW	-	Y co-ordinate
ColourB	-	colour
PartitionB	-	a partition identifier
WidthW	-	width
HeightW	-	Height (number of partition lines)
NlinesW	-	number of lines (for symbol space)
ZoomfactorB	-	zoomfactor
CoixW	-	centre of interest (X co-ordinate)
CoiyW	-	centre of interest (Y co-ordinate)
ScrollModeB	-	scrollmode
StartYW	-	starting point of a partition (Y co-ordinate)
SymbolWidthW	-	symbol width
SymbolHeightW	-	symbol height
NSymbolsB	-	number of symbols
PTypeB	-	partition type
HardwareTypeB	-	graphics board identifier
HardwareSubTypeB	-	configuration details
SoftwareVersionB	-	software version
CountB	-	count
iB	-	look-up table entry
nB	-	number of look-up table entries
gB	-	green value
bB	-	blue value
rB	-	red value
PageB	-	look-up table page

The following are the extra commands supplied with the Megares II graphics controller:

```
COMMAND COPYS (ASCII value)
LIBRARY PROCEDURE PChar (Var Sym : INT);
```

The following extensions (control codes) are interpreted while in the default symbol partition:

CR (Hex : 0D Decimal : 13)

Moves the Current Position (CP) to the beginning of the next line.

BS (Hex : 8 Decimal : 8)

Backspaces the Current Position by one character position.

TAB (Hex : 9 Decimal : 9)

Tabs the Current Position horizontally to an 8 character boundary.

LF (Hex : 0A Decimal : 10)

Line feeds the Current Position down by 12 lines, with scroll if necessary.

COMMAND SwapEnv (Hex : D6 Decimal : 214)
LIBRARY PROCEDURE SwapEnv;

The SwapEnv command sets up an alternate working environment. The variables changes are: the Current Colour (CCOL), the foreground colour (FCOL), the background colour (BCOL), the pattern (PAT), the transparent colour (TCOL), the Current Working Position (CWP) and the Current Symbol Partition (CSP). There are 2 working environments which are alternated each time the SwapEnv command is invoked.

COMMAND PInitW (NlinesW) (Hex : D8 Decimal : 216)
LIBRARY PROCEDURE PInitW (Var NLines : INT);

The PInitW command is the same as the PInit command but it does not clear the displayable area. The Nlines variable can be used to reserve extra lines for symbol space (0 gives the default symbol space).

COMMAND SZoom (ZoomfactorB) (Hex : DD Decimal : 221)
LIBRARY PROCEDURE SZoom (Var Zoomfactor : INT);

The SZoom command zooms the display by pixel replication. The Zoomfactor can be from 1 to 9.

COMMAND IZoom : ZoomfactorB (Hex : DE Decimal : 222)
LIBRARY PROCEDURE IZoom (Var Zoomfactor : INT);

The IZoom command returns the value of the current zoom factor.

COMMAND ICoi : CoixW, CoiyW, ScrollModeB
(Hex : DF Decimal : 223)
LIBRARY PROCEDURE Icoi (Var Coix, Coiy, ScrollMode : INT);

The ICoi command returns the value of the current centre of interest (see the PanTo command).

```
COMMAND DefDP3 (StartYW, HeightW)   (Hex : E2  Decimal : 226)
LIBRARY PROCEDURE DefDP3 (Var StartY, Height : INT);
```

The DefDP3 command defines the start and height of display partition 3. Display partition 3 is a user-defined partition. The StartY parameter is the start point of the partition (Y co-ordinate) and the Height parameter is the number of lines in the partition.

```
COMMAND DefWP3 (StartYW, HeightW)   (Hex : E3  Decimal : 227)
LIBRARY PROCEDURE DefWP3 (Var StartY, Height : INT);
```

The DefWP3 command defines the start and height of working partition 3. Working partition 3 is a user-defined partition. The StartY parameter is the start point of the partition (Y co-ordinate) and the Height parameter is the number of lines in the partition.

```
COMMAND PanTo (XW, YW, ScrollModeB) (Hex : E4  Decimal : 228)
LIBRARY PROCEDURE PanTo (Var X, Y, ScrollMode : INT);
```

The PanTo command moves the centre of interest (Coix, Coiy) of the viewing window and sets the scrolling mode. A scrollmode of zero tries to centre the Coi at the centre of the screen. The display is panned to bring the Coi as close as possible to the screen centre within the constraints of the current zoom factor and the boundary of the current display partition.

A scrollmode of 1 makes the top of the screen the Coi. The visible window is allowed to wrap around to the start of the display partition if necessary. This is the mode that would be used for scrolling text, for example. The Coi and Scrollmode are kept independently for each display partition so that, for example, changes to the Coi while in partition 1 don't affect the Coi for partition 2.

```
COMMAND IWP : YW,HeightW,SymbolWidthW,SymbolHeightW,NSymbolsB,PTypeB
                                    (Hex : E5  Decimal : 229)
LIBRARY PROCEDURE IWP (Var Y, Height, SymbolWidth, SymbolHeight,
                                    NSymbols, PType : INT);
```

The IWP command gives information on the current working partition. The Type variable indicates a symbol/working partition (type 0), a code partition (type 1) or a macro partition (type 2).

```
COMMAND IDP : YW,HeightW            (Hex : E6  Decimal : 230)
LIBRARY PROCEDURE IDP (Var Y, Height : INT);
```

The IDP command gives information on the current display partition. The Y parameter is the start point of the partition (Y co-ordinate) and the Height is the number of lines in the partition.

```
COMMAND IVersion : HardwareTypeB,HardwareSubTypeB,SoftwareVersionB
                                        (Hex : E7  Decimal : 231)
LIBRARY PROCEDURE IVersion (Var Hardware, Subtype, Software : INT);
```

The IVersion command identifies the board. Megares is a board of type 3. The HardwareSubType gives configuration information.

```
COMMAND ReleaseP (PartitionB)       (Hex : ED  Decimal : 237)
LIBRARY PROCEDURE ReleaseP (Var Partition : INT);
```

The ReleaseP command frees the specified partition and partition number for re-use. The symbol space is compressed to prevent memory fragmentation.

```
COMMAND IDWindow : XW,YW,WidthW,HeightW
                                        (Hex : EE  Decimal : 238)
LIBRARY PROCEDURE IDWindow (Var X, Y, Width, Height : INT);
```

The IDWindow command returns the size and position of the currently viewed portion of the display partition. This command is useful in showing what is being displayed after using the Zoom and Pan commands.

```
COMMAND RLLImage (WidthW,HeightW,Count1B,Colour1B,CountnB, ColournB)
                                        (Hex : F0  Decimal : 240)
LIBRARY PROCEDURE RLLImage (Var Flag, Width,Height,Buf,Count : INT);
```

The RLLImage command (Run Length Load Image) loads a run-length encoded image of specified width and height. The pixel data is sent as pairs of count and colour.

```
COMMAND RLRImage (WidthW,HeightW) : Count1B,Colour1B...
                                     ...CountnB,ColournB,0,0
                                        (Hex : F1  Decimal : 241)
LIBRARY PROCEDURE RLRImage (Flag, Width, Height, Buf, Count : INT);
```

The RLRImage command (Run Length Read Image) reads back a run-length encoded image of specified width and height. The pixel data is sent as pairs of count and colour and is terminated by a 0,0 pair.

7.3 Palette Commands

The Megares Palette commands provide commands for manipulating the colour look-up table. Two look-up tables are provided (referred to as page) so that two completely independent sets of colour maps can be used. The particular page to be used for colour mapping can be selected with a single command - this page is referred to as the LUT display page.

The look-up table can be modified to re-define the colour mapping - the look-up table that is modified is defined as the LUT working page (the working page can be the same as the display page). As there are only 16 LUT entries, the whole LUT can be re-defined within one frame period. This facility can be used to change large areas of image quickly instead of changing the actual pixel values in the image.

All visible LUT changes are performed during a vertical blanking period so that no display tearing or interference is seen.

The r, g, b (red, green and blue) values in the following Palette commands specify the intensity of each colour component and have a value of 0 (zero intensity) to 15 (peak density), depending on the value range used. Between zero density and peak density there are 16 discrete levels which correspond to the values:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15.

COMMAND LUTInitG (Hex : C3 Decimal : 195)
LIBRARY PROCEDURE LUTInitG;

The LUTInitG command initialises both pages of the look-up table to be a linear greyscale. The LUT working page and the LUT display page are both set to be 1.

COMMAND LUTInitC (Hex : C4 Decimal : 196)
LIBRARY PROCEDURE LUTInitC;

The LUTInitC command initialises both pages of the look-up table to be a selection of colours. Colours 0 to 7 are the saturated colours available with the stand-alone Pluto (black, green, blue, cyan, red, yellow, magenta and white). The LUT working page and the LUT display page are both set to be 1.

COMMAND WLut (iB, gB, bB, rB) (Hex : C5 Decimal : 197)
LIBRARY PROCEDURE WLut (Var i, Gbrarray : INT);

The WLut command defines the colour component values of entry i in the working LUT page.

For example: WLut (1,15,15,15) would define LUT entry 1 to be the brightest white, so that every pixel with value 1 would appear on the screen as white. If the working LUT page is different from the display LUT page then the operation is performed immediately and is invisible until the display page is changed. If the working page is the same as the display page then the LUT is not updated until the next vertical blanking period to prevent screen disturbances.

COMMAND WLutM (nB, iB, gB, bB, rB,.....gB, bB, rB)
(Hex : C6 Decimal : 198)
LIBRARY PROCEDURE WLutM (Var n, i, GBRarray : INT);

The WLutM command writes multiple look-up table entries. The g, b, r values are written into the n+1 LUT entries in the working LUT page starting at entry number i (n=0 to write one entry, n=15 to write 16 entries, etc.)

For example: WLut (1,1,15,0,0,7,7,7) would define pixel value 1 to be the brightest green, and value 2 to be half white (grey). This is equivalent to 2 WLut commands but is more convenient and efficient.

If the working page is the same as the display page then the LUT is not updated until the next blanking period to prevent screen disturbances. It is much quicker to use the WLutM command rather than several WLut commands, as all of the entries are written during one blanking period rather than only one at a time.

COMMAND RLut (iB) : gB, bB, rB (Hex : C7 Decimal : 199)
LIBRARY PROCEDURE RLut (Var i, GBRarray : INT);

The RLut command returns the g, b, r values of entry i in the working LUT page. This command is a complementary function to the WLut command.

COMMAND RLutM (nB, iB) : gB, bB, rB,.....gB, bB, rB
(Hex : C8 Decimal : 200)
LIBRARY PROCEDURE RLutM (Var n, i, GBRarray : INT);

The RLutM command returns the g, b, r values of n+1 entries starting at entry i in the working LUT page. This command is a complementary function to the WLutM command.

COMMAND SLutWP (PageB) (Hex : C9 Decimal : 201)
LIBRARY PROCEDURE SLutWP (Var Page : INT);

The SLutWP command sets the working LUT page to the specified Page (1 or 2).

COMMAND SLutDP (PageB) (Hex : CA Decimal : 202)
LIBRARY PROCEDURE SLutDP (Var Page : INT);

The SLutDP command sets the display LUT page to the specified Page (1 or 2).

```
COMMAND ILutWP : PageB                (Hex : CB  Decimal : 203)
LIBRARY PROCEDURE ILutWP (Var Page : INT);
```

The ILutWP command returns the value of the working LUT page.

```
COMMAND ILutDP : PageB                (Hex : CC  Decimal : 204)
LIBRARY PROCEDURE ILutDP (Var Page : INT);
```

The ILutDP command returns the value of the display LUT page.

7.4 Error Codes

All the Pluto I error codes (returned by the IStat command) listed in Section 2.5 of this manual also apply to Megares. There are also errors unique to the Megares system, and these are listed below.

7.4.1 Megares Error codes

Error Code	Meaning
21	Illegal zoom parameter
22	Illegal preview parameter
27	RS232 error
33	Can't release CWP or CSP
35	Can't release fill source partition
36	Can't release alternate CWP or CSP
37	CDP and CWP are not the same for scroll
38	CDP to small to scroll

7.4.2 Palette Error codes

Error Code	Meaning
16	LUT index value/range too big
17	Page selected was not allowed

SECTION EIGHT - INTERFACE CARDS FOR DIFFERENT MACHINES

8.1 Introduction

There are Pluto I, Pluto II and Megares Interface Cards available for certain microcomputers. The Interface Cards act as a communications link between a host microcomputer and a boxed Pluto system. The Interface Cards are 8 bit parallel port Interface Cards, and are available for the following microcomputers: Sirius, Apricot, S100, RML, Apple, IBM P.C., RML Nimbus and Q-bus. A boxed Pluto system can be configured to a BBC microcomputer without the need for an Interface Card (the Pluto interface connector can be wired directly to the printer and/or user ports on the BBC microcomputer. In addition to the Io Research Interface Cards mentioned here, certain third party Interface Cards are available such as a BBC card and a VME card. Please ask for further details.

8.2 Sirius

The Pluto Interface Card for the Sirius plugs straight into any of the internal expansion bus slots inside the Sirius microcomputer.

8.3 Apricot

The Pluto Interface Card for the Apricot is a printed circuit board which plugs straight into either of the internal expansion slots inside the Apricot microcomputer.

8.4 S100

The Pluto Interface Card for the S100 is a printed circuit board which plugs straight into the S100 bus inside the S100 microcomputer.

8.5 RML 380z

The Pluto Interface Card for the RML 380z is a printed circuit board which plugs straight into the expansion bus inside the RML microcomputer.

8.6 Apple II

The Pluto Interface Card for the Apple II is a printed circuit board which plugs into any of the seven bus expansion slots inside the Apple microcomputer.

8.7 IBM PC

The Pluto Interface Card for the IBM PC range plugs straight into a short expansion bus slot inside the IBM PC range of microcomputers.

8.8 RML Nimbus

The Pluto Interface Card for the RML Nimbus can be positioned into any one of the free slots inside the Nimbus microcomputer.

8.9 Q-Bus

The Q-Bus Interface Card can be used with any DEC Q-Bus Modline (LSI II or MicroVAX) and occupies one dual-height slot.

8.10 Connections between Pluto and a BBC microcomputer

Pluto can be interfaced to the BBC microcomputer by wiring directly between the 15 way D-type parallel interface connector on Pluto and the printer port and/or user port on the BBC.

Using only the printer port or the user port provides a uni-directional interface, i.e. data and commands can be written to Pluto but data cannot be read back. This caters for most situations, but if data is also to be read back from Pluto, both the printer port and the user port must be used together.

SECTION NINE - DEMONSTRATION DISK

The Demonstration disk (V5.0) is not supplied as standard with all Pluto packages. A Demonstration disk is supplied for each of the Pluto graphics cards which is configured to a host microcomputer running under the MS-DOS operating system. The contents of the Demonstration disk is as follows:

ReadMe.Txt	:	Help file
Pluto.Asm	:	Pluto interface library (assembler source)
SPluto.Obj	:	Object : 16 bit data/ 16 bit code pointers (Suitable for prospero pascal)
MPluto.Obj	:	Object : 16 bit data/ 32 bit code pointers (Suitable for microsoft pascal and basic)
LPluto.Obj	:	Object : 32 bit data/ 32 bit code pointers (suitable for Microsoft Fortran)
CPluto.Obj	:	Object : for Microsoft C
Pluto.Hlp	:	Description of Pluto interface library
Pls.Exe	:	Pluto load/save picture (Executable code)
Demo.Txt	:	Picture list for PLS (demo option)
PTest.Exe	:	Pluto function test (Executable code)
Demo1.Exe	:	Pluto demonstration (Executable code)
Demo1.Bas	:	Basic source for above
Demo2.Bas	:	Pluto demonstration (Basic source)
Demo3.Pas	:	Pluto demonstration (Pascal source)
?????.Pic	:	Selection of pictures
PInstall.Exe	:	Loads code/macros onto Pluto II board
Usercode.asm	:	Example of Pluto usercode (8088-code)
Tutor.Exe	:	Demonstrates Pluto commands
Tutor?.Pas	:	Source code for Pluto Tutor

The 'ReadMe' program explains in more detail each of the programs stored on the Demonstration disk.

The source for most of the programs on the Demonstration disk can be obtained (at a small extra charge) from IO Research Ltd.

APPENDIX A - SAMPLE PASCAL PROGRAM

```
PROGRAM PASCAL EXAMPLE (input, output);

(*-------------------------------------------------------------*)
(*                                                             *)
(*   EXAMPLE OF DRIVING PLUTO THROUGH PLUTO = ASM USING PASCAL *)
(*                                                             *)
(*-------------------------------------------------------------*)


CONST

     NoCols  =  640;   (*--- Pluto 1 ---*)
     NoLines =  576;   (*--- Pluto 1 interlaced ---*)

TYPE

     INT  =  32768..32767;


(*----- FROM PLUTO.ASM -----*)
PROCEDURE LineTo (Var dx,dy: INT); EXTERNAL;
PROCEDURE MoveTo (Var X,Y: INT); EXTERNAL;
PROCEDURE Arc (Var Xc,Yc,Xe,Ye: INT); EXTERNAL;
PROCEDURE PInit; EXTERNAL;
PROCEDURE SHires; EXTERNAL;
PROCEDURE SCCol (Var Cwp: INT); EXTERNAL;
PROCEDURE CIRCwp; EXTERNAL;
PROCEDURE SCWP (Var Cwp: INT); EXTERNAL;
PROCEDURE FFill; EXTERNAL;


(*-- RANDOM NUMBER GENERATOR - GENERATES REALS IN THE RANGE 0.0-1.0 ---*)
FUNCTION Rand: real; EXTERNAL;



PROCEDURE Drawline;
VAR
     X,Y,NewX,NewY: INT;
begin

     X : = round (rand * (NoCols - 1));
     Y : = round (rand * (NoLines - 1));
     MoveTO (X, Y);        (*----- set CP to start point -----*)


     NewX : = round (rand * (NoCols - 1));
     NewY : = round (rand * (NoLines - 1));
     LineTo (NewX, NewY);  (*----- draw line to end point -----*)
end;
```

```
PROCEDURE DrawBox;
VAR
       X,Y,NewX,NewY: INT;
begin

      X : = round (rand * (NoCols - 1));
      Y : = round (rand * (NoLines - 1));
      NewX :  = round (rand * (NoCols - 1));
      NewY :  = round (rand * (NoLines - 1));

      MoveTo  (X, Y );              (*----- set CP to top corner -----*)

      LineTo (newX, Y);             (*----- Draw Box -----*)

      if (Y<>newY) then LineTo ( newX, newY );
      if (X<>newX) then LineTo ( X, newY );
      if (Y<>newY) then LineTo ( X, Y );
end;



PROCEDURE DrawCircle;
VAR
     DeltaX, DeltaY, XEdge, XCentre, YCentre, Zero: INT;
begin
     Zero : = 0;


     XEdge : = round (rand * (NoCols - 1));
     YEdge : = round (rand * (NoLines - 1));

     XCentre : = round (rand * (NoCols - 1));
     YCentre : = round (rand * (NoLines - 1));

     DeltaX : = (XEdge - XCentre);
     DeltaY : = (YEdge - YCentre);

     MoveTo  (XEdge, YEdge);              (*----- set CP to edge -----*)

     Arc (DeltaX, DeltaY, Zero, Zero);    (*----- Draw Full circle -----*)
end;



PROCEDURE FloodFill:
VAR
     X,Y: INT;
begin
    X : = round (rand * (NoCols - 1));
    Y ; = round (rand * (NoLines - 1));
    MoveTo (X,Y);                   (*----- set CP -----*)

    FFill;                          (*----- flood fill -----*)
end;
```

```
PROCEDURE SetCurrentColour;
VAR
     CCol : INT;
begin
     CCol : = round (rand * 7);
     SCCol  (CCol);
end;
```

```
PROCEDURE Initialise;
VAR
     Zero: INT;
begin
     PInit;                         (*----- initialise pluto -----*)

     SHires;                        (*--- set screen into high res ---*)

     Zero : = 0;
     SCWP (Zero);                   (*--- set current working partition ---*)
end;
```

```
PROCEDURE Main;
VAR
     i, j : INT;
begin
     Initialise:

     (*---- Draw miscellaneous selection of boxes lines and circles ----*)

     i  :  =  0;
     while TRUE do
     begin

          SetCurrentColour;

          DrawLine;
          DrawBox;
          DrawCircle;

          i : = i + 1;
          if i = 10 then
          begin
```

```
            FloodFill;
            for j : = 0 to 30000 do (*--- delay ---*)
            i : = i DIV 11;

            CirCWP;


    end;
    end;
end;

begin;

    Main;

end;
```

APPENDIX B - SAMPLE BASIC PROGRAM

```
10 REM
20 REM  -------EXAMPLE OF DRIVING PLUTO THROUGH PLUTO.ASM USING BASIC
30 REM
40 REM
50 REM  -------INITIALISE
60 REM
70 CALL PINIT
80 CALL SHIRES
90 ZERO% = 0
100 CALL SCWP (ZERO%)
110 I = 0
114 REM
115 REM  -----SET CURRENT COLOUR
120 CCOL% = RND * 639
130 CALL SCCOL (CCOL%)
134 REM
135 REM  -----DRAW LINE
140 X% = RND * 639
150 Y% = RND * 575
160 CALL MOVETO (X%, Y%)
170 NEWX% = RND * 639
180 NEWY% = RND * 575
190 CALL LINETO (NEWX%, NEWY%)
200 REM
210 REM  -----DRAW BOX
220 X% = RND * 639
230 Y% = RND * 575
240 NEWX% = RND * 639
250 NEWY% = RND * 575
260 CALL MOVETO (X%, Y%)
270 CALL LINETO (NEWXY, Y%)
280 IF Y% <> NEWY% THEN CALL LINETO (NEWX%, NEWY%)
290 IF X% <> NEWX% THEN CALL LINETO (X%, NEWY%)
300 IF Y% <> NEWY% THEN CALL LINETO (X%, Y%)
310 REM
320 REM -----DRAW CIRCLE
325 ZERO% = 0
330 XEDGE% = RND * 639
340 YEDGE% = RND * 575
350 XCENTRE% = RND * 639
360 YCENTRE% = RND * 575
370 DELTAX% = XEDGE% - XCENTRE%
380 DELTAY% = YEDGE% - YCENTRE%
390 CALL MOVETO (XEDGE%, YEDGE%)
400 CALL ARC (DELTAX%, DELTAY%, ZERO%, ZERO%)
410 REM
420 REM
430 I = I + 1
440 IF I <> 10 THEN GOTO 120
450 X% = RND * 639
460 Y% = RND * 575
470 CALL MOVETO (X%, Y%)
480 CALL FFILL
490 FOR J = 0 TO 30000
500 I = I \ 11
510 NEXT J
520 CALL CLRCWP
530 GOTO 120
```

APPENDIX C - PLUTO GRAPHICS BOARD - TEST PROGRAM

The following program can be used to test that any of the Pluto graphics boards are operating correctly. The program, which is written in 'C' for a Gemini Galaxy PC amended to run on any other computer.

Note: Within the code of the test program, /* indicates the start of a comment (i.e. text that the compiler will ignore), and */ indicates the end of the same comment.

The test program uses the STYLE parameter, which can be set to any value from 0 to 255. The value of the STYLE parameter may also be inquired. By setting the STYLE parameter to every value from 0 to 255, inquiring each new value and comparing the two, the Pluto board's 8 bit parallel communications interface can be tested.

If the STYLE parameter can be set to all 256 values successfully, a dot (".") will display on the VDU of the host computer. If an error occurs, the expected value and the actual value inquired will display in hex and decimal.

Note: Ensure that the STATUS port and the COMMAND port addresses are set correctly before running the test program. The STATUS and COMMAND port addresses are set with the global definitions STATPORT and DATAPORT.

Two primitive routines are required to interface between the host computer and the Pluto graphics board. In the following test program these are called OUTPORT and INPORT. These routines can be used with a Pluto board that is addressed in an isolated I/O (as in the following code) or memory map.

(1) OUTPORT(address, value);

In this function the address is taken as a 16 bit unsigned integer which is the address of the STATUS port or the COMMAND port. The value is taken as an unsigned integer and is the 8 bit value to send to the required port.

(2) INPORT(address);

In this function a value is read (and then returned to the calling function) from the port specified by the address which is taken as a 16 bit unsigned integer. The value returned is an unsigned integer, with values from 0 to 255.

There is one further routine, called "KEYHIT", which can be used to stop the test once a key has been hit. A BDOS call of "Get Console Status" is used which returns either TRUE (if a key has been hit) or false.

Note: Further test software may be available on disk from Io Research. Please ask our Technical Support Development for further details.

```
/*****************************************************/
/* Global definitions (can be amended as required) */
/*****************************************************/

# define STATPORT       0xA0    /*Pluto STATUS port address*/
# define DATAPORT       0xA1    /*Pluto COMMAND port address*/
# define FALSE          0       /*Boolean type equates*/
# define TRUE           1



/*****************************/
/* Pluto Command Definitions */
/*****************************/

# define cmd_istyle     0x8E    /*Pluto inquire style command
                                                    code */
# define cmd_sstyle     0x8A    /*Pluto set style command
                                                    code */
# define cmd_pinit      0xA6    /*Pluto initialise command  */


/*****************************/
/*  Start of main routine    */
/*****************************/

main()
{
int count,error,v;
    unsigned int style,getstyle();


    signon();                   /*Display hello message*/
    pinit();                    /*Initialise Pluto */
    while( keyhit() == FALSE ) {     /*While no key is hit*/
        error = FALSE;                /*Clear error flag*/
        for(style=0; style<=255; ++style)  {
                                /*Loop for 1 full cycle*/
            setstyle( style );  /*Set the style to "style"*/
                                 /*Read the Style back and
                                   compare the two*/
            if ((v=getstyle())!=style)  {
                                 /*If an error occurs....*/
                 printf("\nError: ");
                                 /*Show Txed and Rxed byte*/
                 printf("SENT %dDec (%xHex)",style,style);
                 printf(" RECEIVED %dDec (%xHex)",v,v );
                 error = TRUE;      /*Flag the error*/
            }
        }                            /*Finish the cycle*/
        if( error == TRUE )         /*If a bad run then don't*/
                printf("\n");       /*display a "." */
        else                        /*Else (a good run) so */
                printf(".");        /*indicate 1 good cycle */
    }                               /*Continue until a key
                                      hit has been detected */
    printf("\nTest Sequence Aborted");      /*Else quit*/
```

```
/************************************************************/
/* Support Routine: KEYHIT(): Returns TRUE if a key has     */
/* been hit, else FALSE. In this example the "C" compiler   */
/* is being used on a CP/M 2.2 system. The standard BDOS    */
/* "Get Console Status" function is used to determine if    */
/* a key has been hit. This routine can be amended,as       */
/* required, or patched out by unconditionally returning    */
/* FALSE.                                                   */
/************************************************************/

keyhit()
{
     return( bdos( 0x0B,0 ) );     /*Returns TRUE if a key */
                                   /*has been hit, else    */
                                   /*returns FALSE         */
}


/************************************************************/
/* Sign-on message. Displayed on the VDU of the host        */
/* computer                                                 */
/************************************************************/
signon()
{
    char getchar();

    printf("STYLE:\tData Communications Test: Vers 1.00\n");
    printf("tCopyright (c) IO Research Ltd: 17/11/1985\n");
    printf("tEvery \".\" Represents 256 checked Read/Write
                                              Cycles\n\n");
    printf("Connect your PLUTO and hit RETURN to start ");

    while( getchar() != '\n' );  /*Loop till RETURN is hit*/

    printf("\nPress any key to stop\n\n");
}


/************************************************************/
/* Pluto Initialise Routine: PINIT(); This routine          */
/* attempts to set the Pluto graphics card to the same      */
/* as when first turned on.                                 */
/************************************************************/
pinit()
{

     chk_pluto_ready();          /*Check Pluto is not busy  */
     outport( STATPORT,0 );      /*Do a hardware reset, i.e.*/
     outport( STATPORT,0 );      /*Write 4 zero bytes to the*/
     outport( STATPORT,0 );      /*status port              */
     outport( STATPORT,0 );
     chk_pluto_ready();
     outport( DATAPORT,cmd_pinit );  /*Issue the PINIT    */
                                                /*command*/
}
```

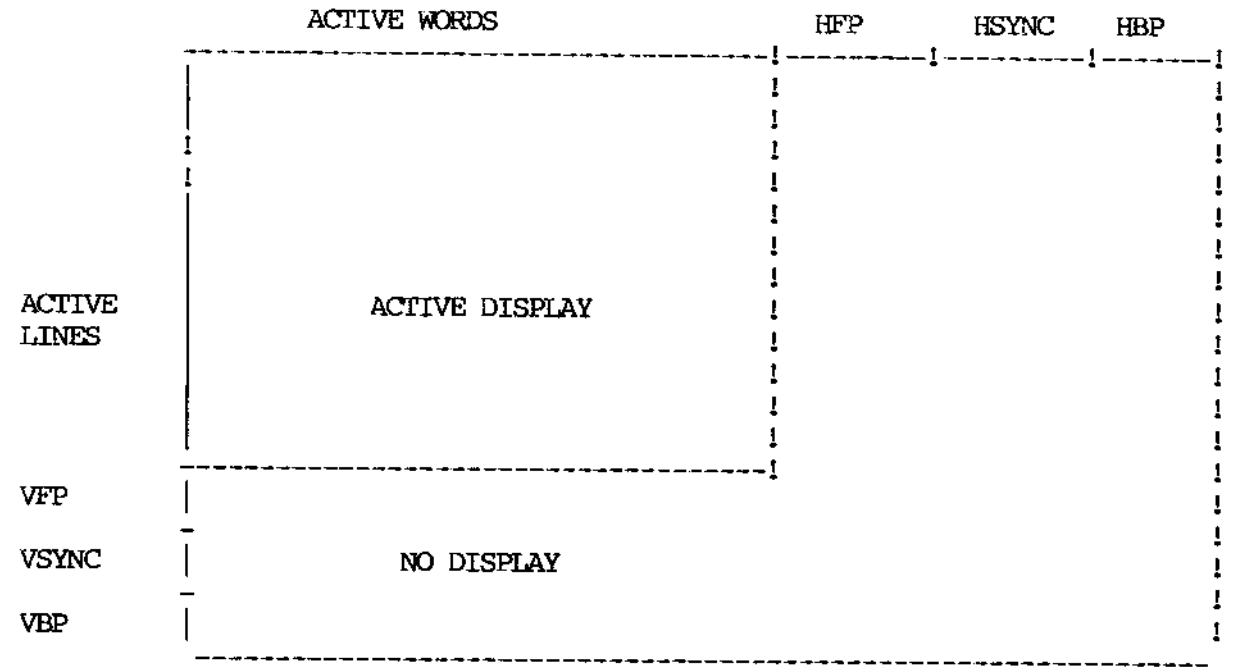
```
/*********************************************************/
/* Pluto Set Style Routine: SETSTYLE( V ); This routine  */
/* sets the STYLE parameter on the Pluto graphics card to */
/* the value V. It returns nothing.                      */
/*********************************************************/

setstyle( value )

unsigned int value;

{
     chk_pluto_ready();          /*Check Pluto is not busy*/
     outport( DATAPORT,cmd_sstyle );    /*Issue the set   */
                                        /*style command  */
     chk_pluto_ready();          /*Check Pluto is not busy*/
     outport( DATAPORT,value );  /*Issue the style value  */
}



/*********************************************************/
/* Pluto Inquire Style Routine: GETSTYLE(); This routine  */
/* inquires the style value that the Pluto card is       */
/* currently set to. It returns this as an 8 bit unsigned */
/* integer to the calling routine.                       */
/*********************************************************/

unsigned int getstyle()

{

     unsigned int inport();
     chk_pluto_ready();          /*Check Pluto is not busy*/
     outport( DATAPORT,cmd_istyle );   /*Issue the inquire*/
                                       /*style command    */
     chk_pluto_ready();          /*Check Pluto is not busy*/
return( inport(DATAPORT) );      /*Return the STYLE value */
                                 /*that Pluto is currently*/
                                 /*set to.                */
}



/*********************************************************/
/* Routine to check the status of the Pluto card: i.e. it */
/* will make sure that Pluto is free to accept the next   */
/* command. A free board is defined as one where bit 7 of */
/* the status port is high. Conversely, a busy board will */
/* have bit 7 low.                                       */
/* Some form of "timeout" can be added to this section    */
/* to prevent a faulty board causing a "lock-up".        */
/*********************************************************/

chk_pluto_ready()
{

     unsigned int inport();
     while( inport(STATPORT) <= 127 );    /*Loop till free*/
}
```

```
/************************************************************/
/* Primitive routine to issue an 8 bit quantity to an       */
/* isolated I/O port.                                       */
/************************************************************/


outport( addr,val )
unsigned int addr,val;
{
     out8( addr,val )                /*Patch for your system*/
}




/************************************************************/
/* Primitive routine to receive an 8 bit quantity from an */
/* isolated I/O port.                                       */
/************************************************************/


unsigned inport( addr )
{
     return(inp( addr ) );           /*Patch for your system*/
}
```

APPENDIX D

CRTC timings for: - PLUTO I - (768)
==========

RESOLUTION	:	768 X 576	PIXELS (INTERLACED)
CRYSTAL FREQUENCY	:	30	Mhz
PIXEL CLOCK	:	15	PIXELS/uS
WORD SIZE	:	8	PIXELS/WORD



WORD TIME (WORDSIZE / PIXELCLOCK)	:	.5333			uS / WORD
ACTIVE WORDS	:	96	WORDS	(51.20	us)
HORIZONTAL FRONT PORCH (HFP)	:	7	WORDS	(3.73	us)
HORIZONTAL SYNC (HS)	:	8	WORDS	(4.27	us)
HORIZONTAL BACK PORCH	:	9	WORDS	(4.80	us)
HORIZONTAL TOTAL	:	120	WORDS	(64.00	us)

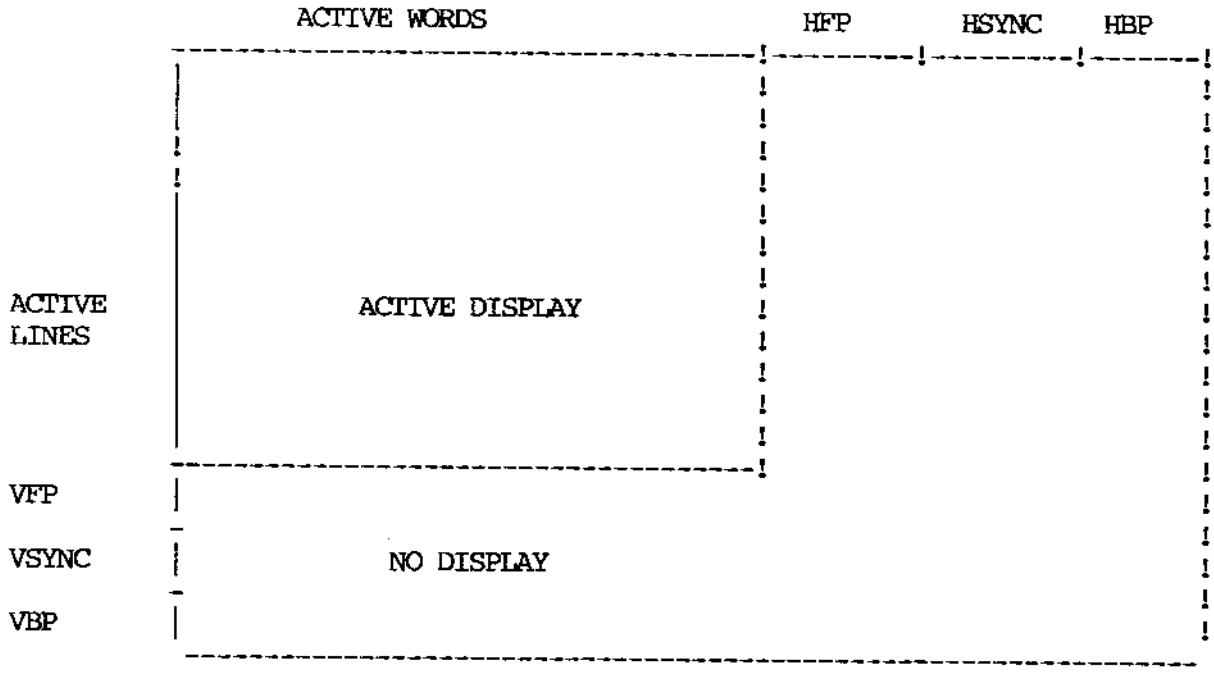
LINE RATE = 1 / HORIZONTAL TOTAL = 15.625 Khz

ACTIVE LINES	:	288	LINES / FRAME
VERTICAL FRONT PORCH (VFP)	:	8	LINES / FRAME
VERTICAL SYNC (VS)	:	8	LINES / FRAME
VERTICAL BACK PORCH (VBP)	:	8	LINES / FRAME
VERTICAL TOTAL	:	312	LINES / FRAME (+ .5 line if interlaced)

$$\text{FRAME TIME} = \frac{\text{HORIZONTAL TOTAL} \times \text{VERTICAL TOTAL} \times \text{WORDSIZE}}{\text{PIXEL CLOCK}} = 20.000 \text{ mS}$$

CRTC timings for: - PLUTO I - (640)
==========

RESOLUTION	:	640 X 576	PIXELS (INTERLACED / NON INTERLACED)
CRYSTAL FREQUENCY	:	25	Mhz
PIXEL CLOCK	:	12.5	PIXELS/uS
WORD SIZE	:	8	PIXELS/WORD



WORD TIME (WORDSIZE / PIXELCLOCK)	:	.64			uS / WORD	
ACTIVE WORDS	:	80	WORDS	(	51.20	us)
HORIZONTAL FRONT PORCH (HFP)	:	5	WORDS	(	3.20	us)
HORIZONTAL SYNC (HS)	:	8	WORDS	(	5.12	us)
HORIZONTAL BACK PORCH	:	7	WORDS	(	4.48	us)
HORIZONTAL TOTAL	:	100	WORDS	(	64.00	us)

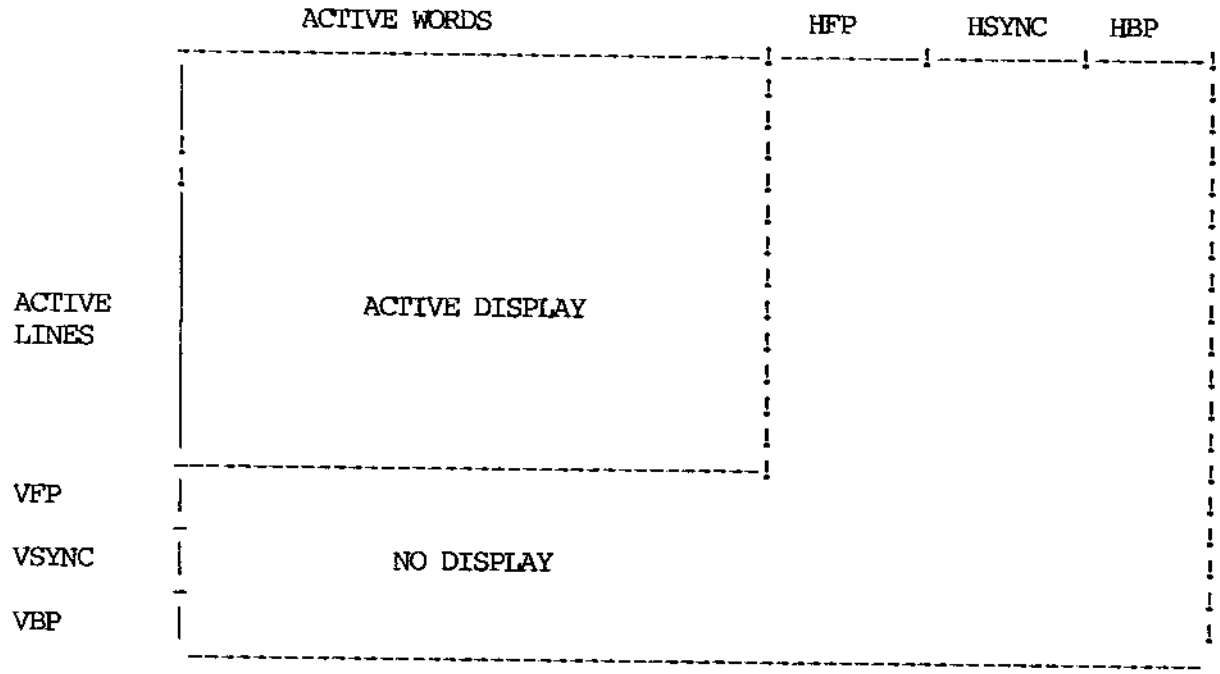
LINE RATE = 1 / HORIZONTAL TOTAL = 15.625 Khz

ACTIVE LINES	:	288	LINES / FRAME
VERTICAL FRONT PORCH (VFP)	:	8	LINES / FRAME
VERTICAL SYNC (VS)	:	8	LINES / FRAME
VERTICAL BACK PORCH (VBP)	:	8	LINES / FRAME
VERTICAL TOTAL	:	312	LINES / FRAME (+ .5 line if interlaced)

$$\text{FRAME TIME} = \frac{\text{HORIZONTAL TOTAL} \times \text{VERTICAL TOTAL} \times \text{WORDSIZE}}{\text{PIXEL CLOCK}} = 20.000 \text{ mS}$$

CRTC timings for: - SIRIUS (SIR001)
==========

RESOLUTION	:	768 X 576	PIXELS (INTERLACED)
CRYSTAL FREQUENCY	:	30	Mhz
PIXEL CLOCK	:	15	PIXELS/uS
WORD SIZE	:	8	PIXELS/WORD



WORD TIME (WORDSIZE / PIXELCLOCK)	:	.5333			uS / WORD
ACTIVE WORDS	:	96	WORDS	(51.20	us)
HORIZONTAL FRONT PORCH (HFP)	:	7	WORDS	(3.73	us)
HORIZONTAL SYNC (HS)	:	8	WORDS	(4.27	us)
HORIZONTAL BACK PORCH	:	9	WORDS	(4.80	us)
HORIZONTAL TOTAL	:	120	WORDS	(64.00	us)

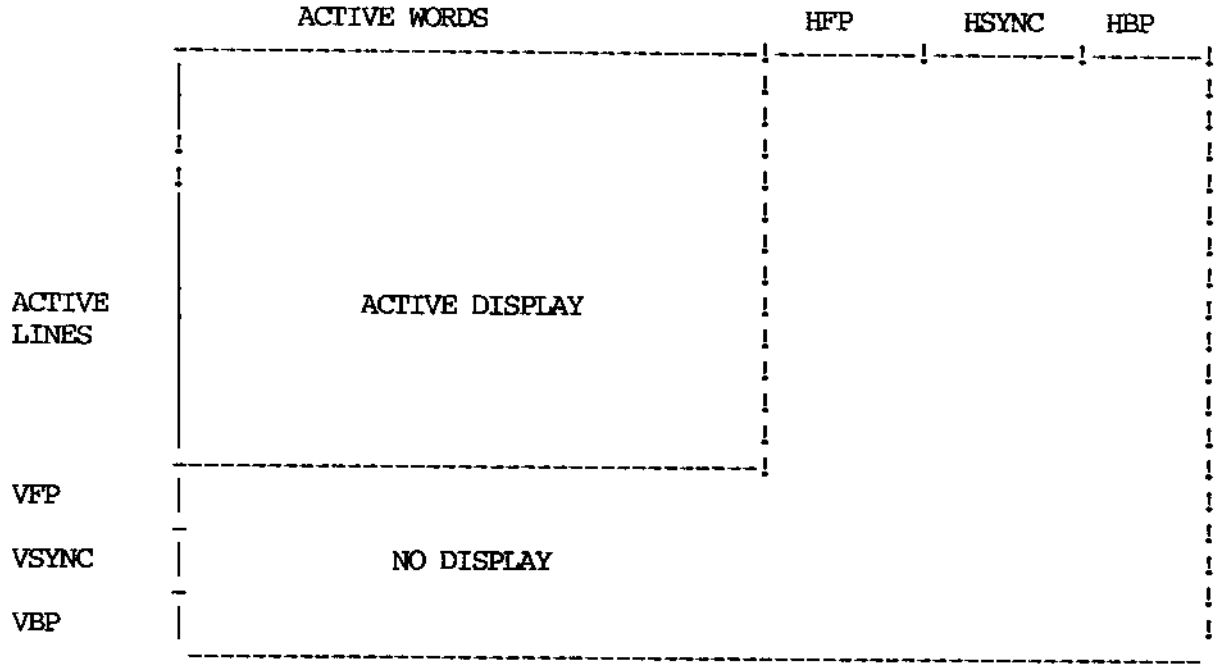
LINE RATE = 1 / HORIZONTAL TOTAL = 15.625 Khz

ACTIVE LINES	:	288	LINES / FRAME
VERTICAL FRONT PORCH (VFP)	:	4	LINES / FRAME
VERTICAL SYNC (VS)	:	8	LINES / FRAME
VERTICAL BACK PORCH (VBP)	:	12	LINES / FRAME
VERTICAL TOTAL	:	312	LINES / FRAME (+ .5 line if interlaced)

$$\text{FRAME TIME} = \frac{\text{HORIZONTAL TOTAL} \times \text{VERTICAL TOTAL} \times \text{WORDSIZE}}{\text{PIXEL CLOCK}} = 20.000 \text{ mS}$$

CRTC timings for: - PLUTO II / 24 BIT
==========

RESOLUTION	:	768 X 576	PIXELS (INTERLACED)
CRYSTAL FREQUENCY	:	30	Mhz
PIXEL CLOCK	:	15	PIXELS/uS
WORD SIZE	:	8	PIXELS/WORD



WORD TIME (WORDSIZE / PIXELCLOCK)	:	.5333			uS / WORD
ACTIVE WORDS	:	96	WORDS	(51.20	us)
HORIZONTAL FRONT PORCH (HFP)	:	4	WORDS	(2.13	us)
HORIZONTAL SYNC (HS)	:	9	WORDS	(4.80	us)
HORIZONTAL BACK PORCH	:	11	WORDS	(5.87	us)
HORIZONTAL TOTAL	:	120	WORDS	(64.00	us)

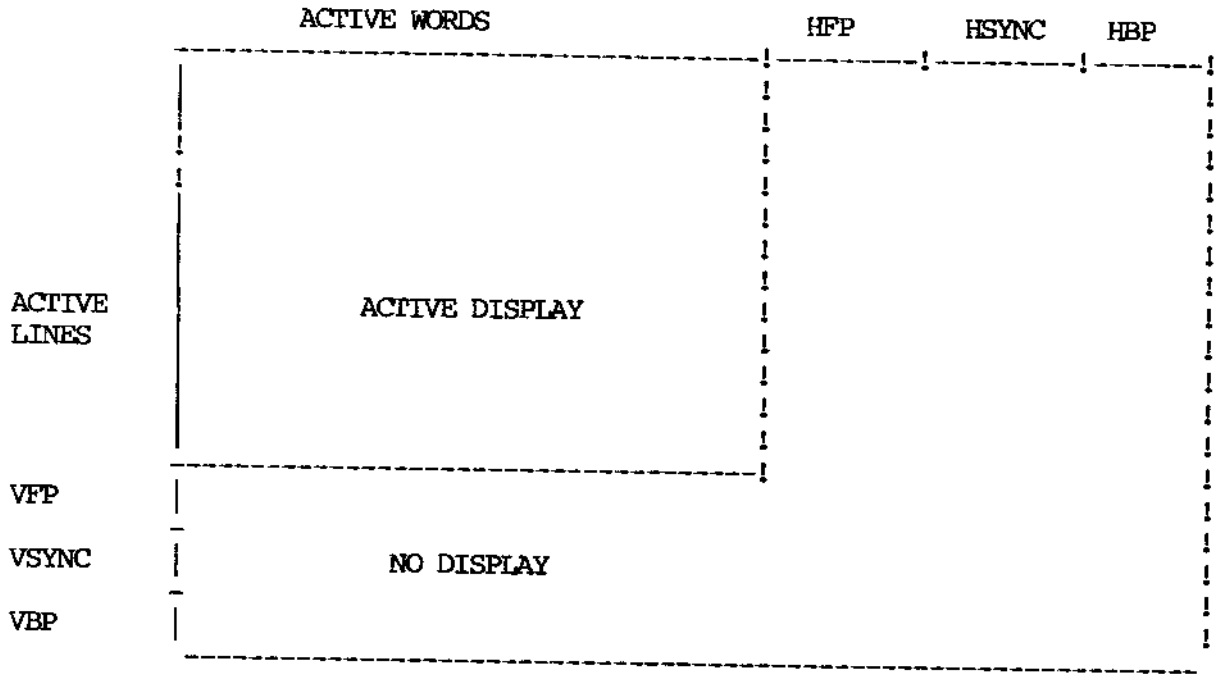
LINE RATE = 1 / HORIZONTAL TOTAL = 15.625 Khz

ACTIVE LINES	:	288	LINES / FRAME
VERTICAL FRONT PORCH (VFP)	:	2	LINES / FRAME
VERTICAL SYNC (VS)	:	8	LINES / FRAME
VERTICAL BACK PORCH (VBP)	:	14	LINES / FRAME
VERTICAL TOTAL	:	312	LINES / FRAME (+ .5 line if interlaced)

$$\text{FRAME TIME} = \frac{\text{HORIZONTAL TOTAL} \times \text{VERTICAL TOTAL} \times \text{WORDSIZE}}{\text{PIXEL CLOCK}} = 20.000 \text{ mS}$$

CRTC timings for: - MEG 001
==========

RESOLUTION	:	768 X 576	PIXELS (NON INTERLACED)
CRYSTAL FREQUENCY	:	30	Mhz
PIXEL CLOCK	:	30	PIXELS/uS
WORD SIZE	:	16	PIXELS/WORD



WORD TIME (WORDSIZE / PIXELCLOCK)	:	.5333			uS / WORD
ACTIVE WORDS	:	48	WORDS	(25.60	us)
HORIZONTAL FRONT PORCH (HFP)	:	2	WORDS	(1.07	us)
HORIZONTAL SYNC (HS)	:	8	WORDS	(4.27	us)
HORIZONTAL BACK PORCH	:	4	WORDS	(2.13	us)
HORIZONTAL TOTAL	:	62	WORDS	(33.07	us)

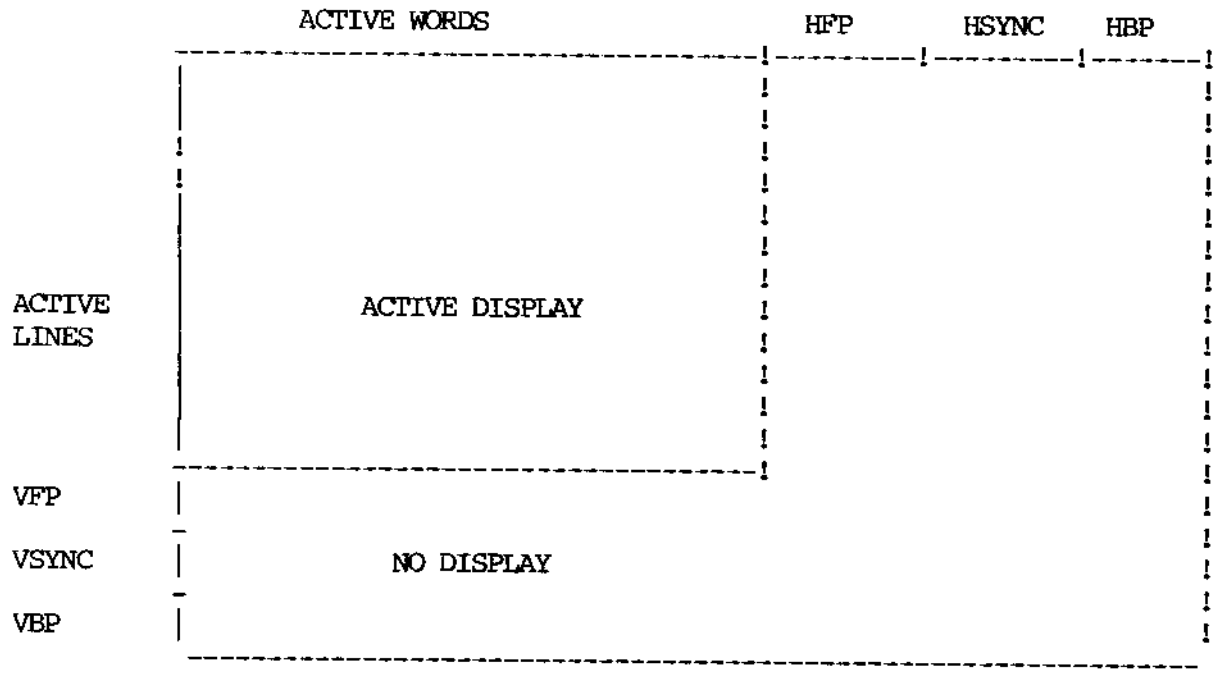
LINE RATE = 1 / HORIZONTAL TOTAL = 30.242 Khz

ACTIVE LINES	:	576	LINES / FRAME
VERTICAL FRONT PORCH (VFP)	:	1	LINES / FRAME
VERTICAL SYNC (VS)	:	3	LINES / FRAME
VERTICAL BACK PORCH (VBP)	:	31	LINES / FRAME
VERTICAL TOTAL	:	611	LINES / FRAME (+ .5 line if interlaced)

$$\text{FRAME TIME} = \frac{\text{HORIZONTAL TOTAL} \times \text{VERTICAL TOTAL} \times \text{WORDSIZE}}{\text{PIXEL CLOCK}} = 20.203 \text{ mS}$$

CRTC timings for: - MEG 002
==========

RESOLUTION	:	1024 X 768	PIXELS (INTERLACED)
CRYSTAL FREQUENCY	:	30	Mhz
PIXEL CLOCK	:	30	PIXELS/uS
WORD SIZE	:	16	PIXELS/WORD



WORD TIME (WORDSIZE / PIXELCLOCK)	:	.5333			uS / WORD
ACTIVE WORDS	:	64	WORDS	(34.13	us)
HORIZONTAL FRONT PORCH (HFP)	:	3	WORDS	(1.60	us)
HORIZONTAL SYNC (HS)	:	7	WORDS	(3.73	us)
HORIZONTAL BACK PORCH	:	5	WORDS	(2.67	us)
HORIZONTAL TOTAL	:	79	WORDS	(42.13	us)

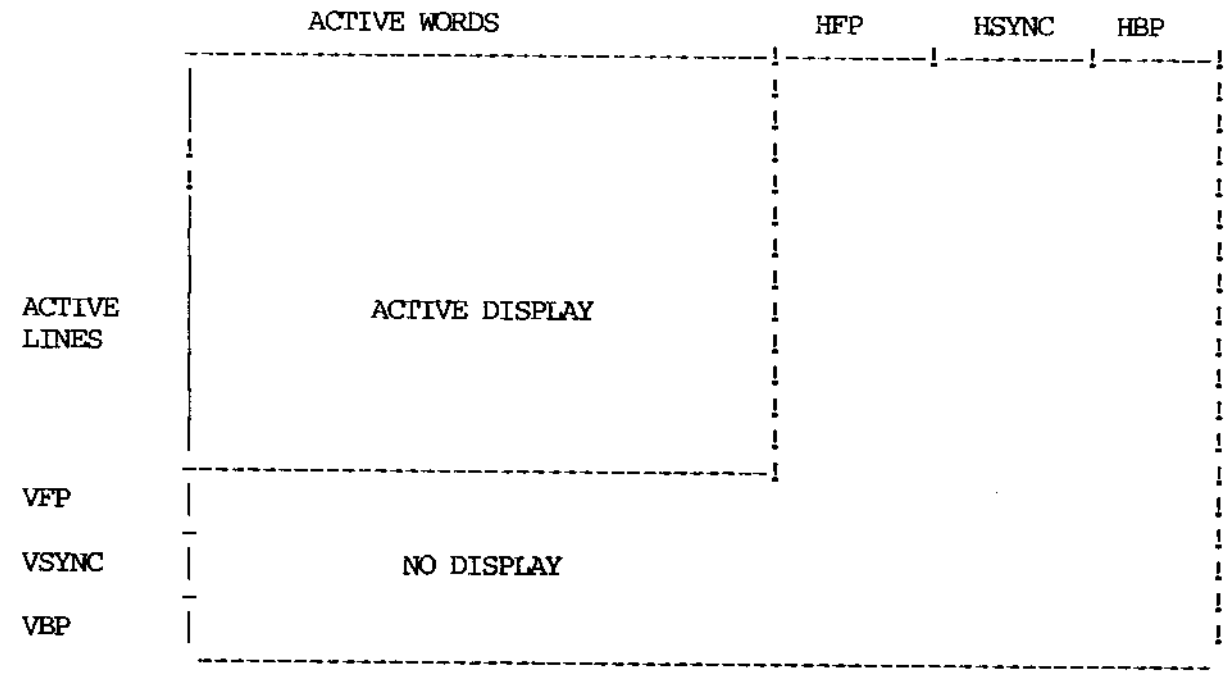
LINE RATE = 1 / HORIZONTAL TOTAL = 23.734 Khz

ACTIVE LINES	:	384	LINES / FRAME
VERTICAL FRONT PORCH (VFP)	:	15	LINES / FRAME
VERTICAL SYNC (VS)	:	15	LINES / FRAME
VERTICAL BACK PORCH (VBP)	:	61	LINES / FRAME
VERTICAL TOTAL	:	475	LINES / FRAME (+ .5 line if interlaced)

$$\text{FRAME TIME} = \frac{\text{HORIZONTAL TOTAL} \times \text{VERTICAL TOTAL} \times \text{WORDSIZE}}{\text{PIXEL CLOCK}} = 20.034 \text{ mS}$$

CRTC timings for: - IBM 001
==========

RESOLUTION	:	768 X 576	PIXELS (INTERLACED)
CRYSTAL FREQUENCY	:	30	Mhz
PIXEL CLOCK	:	15	PIXELS/uS
WORD SIZE	:	8	PIXELS/WORD



WORD TIME (WORDSIZE / PIXELCLOCK)	:	.5333			uS / WORD
ACTIVE WORDS	:	96	WORDS	(51.20	us)
HORIZONTAL FRONT PORCH (HFP)	:	4	WORDS	(2.13	us)
HORIZONTAL SYNC (HS)	:	8	WORDS	(4.27	us)
HORIZONTAL BACK PORCH	:	12	WORDS	(6.40	us)
HORIZONTAL TOTAL	:	120	WORDS	(64.00	us)

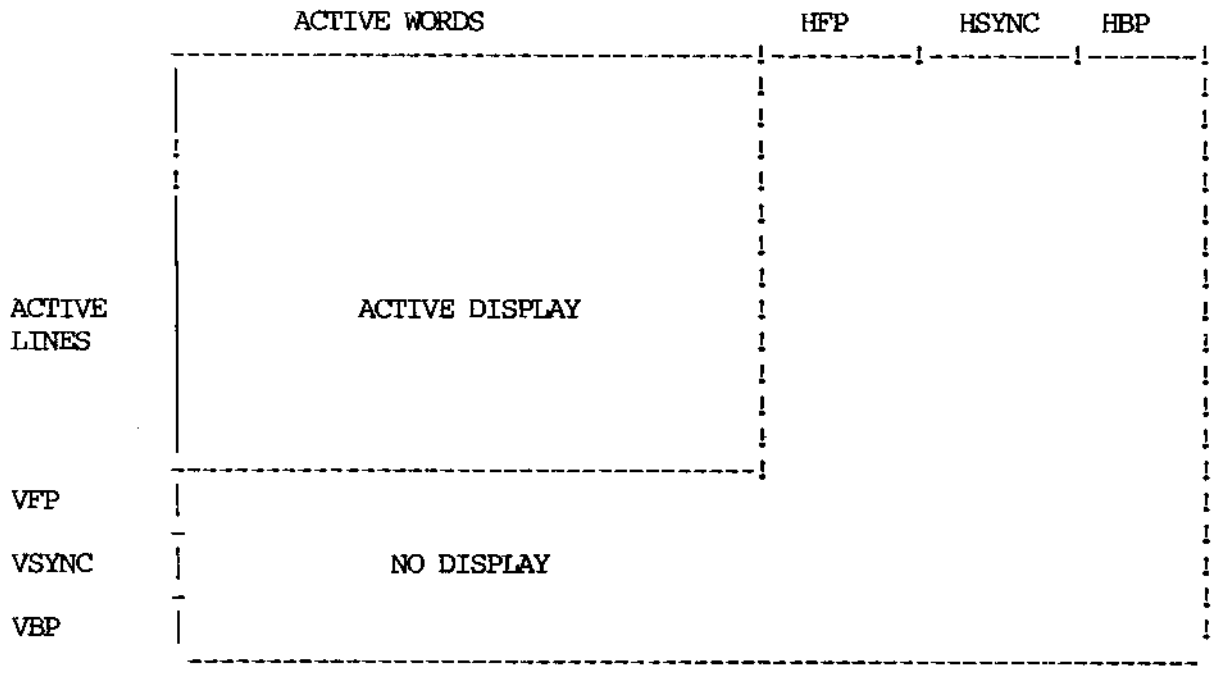
LINE RATE = 1 / HORIZONTAL TOTAL = Khz

ACTIVE LINES	:	288	LINES / FRAME
VERTICAL FRONT PORCH (VFP)	:	2	LINES / FRAME
VERTICAL SYNC (VS)	:	8	LINES / FRAME
VERTICAL BACK PORCH (VBP)	:	14	LINES / FRAME
VERTICAL TOTAL	:	312	LINES / FRAME (+ .5 line if interlaced)

$$\text{FRAME TIME} = \frac{\text{HORIZONTAL TOTAL} \times \text{VERTICAL TOTAL} \times \text{WORDSIZE}}{\text{PIXEL CLOCK}} = 20.000 \text{ mS}$$

CRTC timings for: - IBM 002
==========

RESOLUTION	:	768 X 576	PIXELS (NON INTERLACED)
CRYSTAL FREQUENCY	:	30	Mhz
PIXEL CLOCK	:	30	PIXELS/uS
WORD SIZE	:	16	PIXELS/WORD



WORD TIME (WORDSIZE / PIXELCLOCK)	:	.5333			uS / WORD
ACTIVE WORDS	:	48	WORDS	(25.60	us)
HORIZONTAL FRONT PORCH (HFP)	:	2	WORDS	(1.07	us)
HORIZONTAL SYNC (HS)	:	8	WORDS	(4.27	us)
HORIZONTAL BACK PORCH	:	4	WORDS	(2.13	us)
HORIZONTAL TOTAL	:	62	WORDS	(33.07	us)

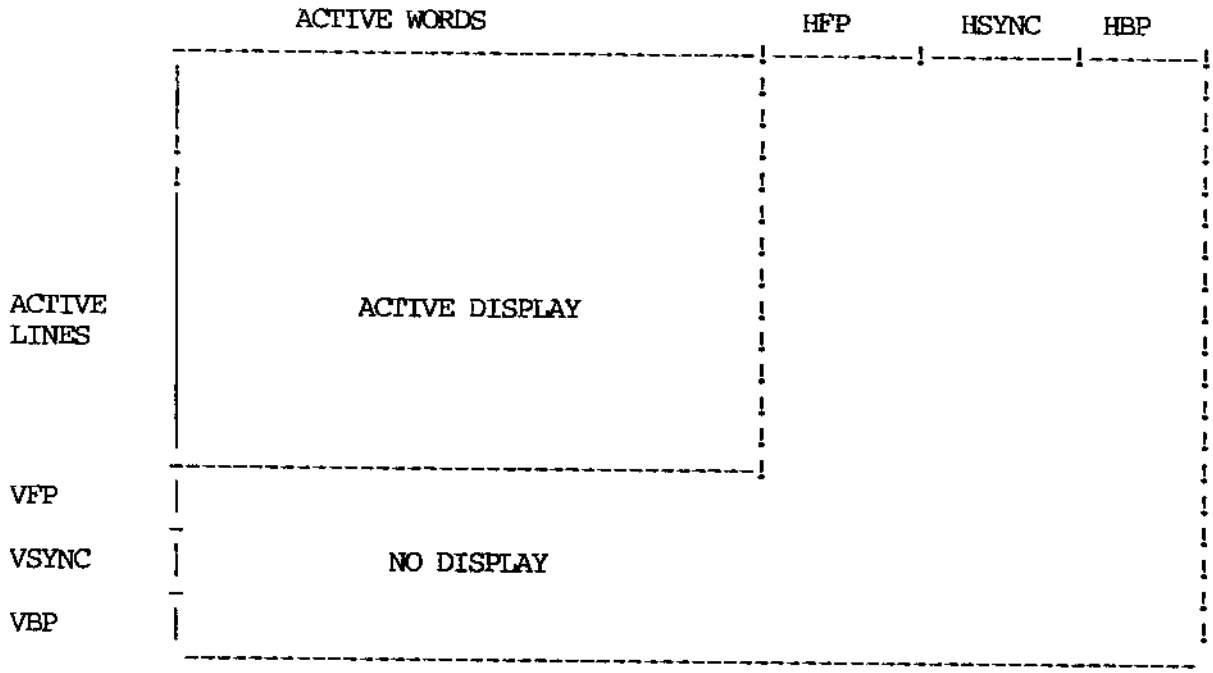
LINE RATE = 1 / HORIZONTAL TOTAL = 30.242 Khz

ACTIVE LINES	:	576	LINES / FRAME
VERTICAL FRONT PORCH (VFP)	:	1	LINES / FRAME
VERTICAL SYNC (VS)	:	24	LINES / FRAME
VERTICAL BACK PORCH (VBP)	:	4	LINES / FRAME
VERTICAL TOTAL	:	605	LINES / FRAME (+ .5 line if interlaced)

$$\text{FRAME TIME} = \frac{\text{HORIZONTAL TOTAL} \times \text{VERTICAL TOTAL} \times \text{WORDSIZE}}{\text{PIXEL CLOCK}} = 20.005 \text{ mS}$$

CRTC timings for: - IBM 003

RESOLUTION	:	1024 X 768	PIXELS (INTERLACED)
CRYSTAL FREQUENCY	:	30	Mhz
PIXEL CLOCK	:	30	PIXELS/uS
WORD SIZE	:	16	PIXELS/WORD



WORD TIME (WORDSIZE / PIXELCLOCK)	:	.5333			uS / WORD
ACTIVE WORDS	:	64	WORDS	(34.13	us)
HORIZONTAL FRONT PORCH (HFP)	:	3	WORDS	(1.60	us)
HORIZONTAL SYNC (HS)	:	7	WORDS	(3.73	us)
HORIZONTAL BACK PORCH	:	5	WORDS	(2.67	us)
HORIZONTAL TOTAL	:	79	WORDS	(42.13	us)

LINE RATE = 1 / HORIZONTAL TOTAL = 23.734 Knz

ACTIVE LINES	:	384	LINES / FRAME
VERTICAL FRONT PORCH (VFP)	:	15	LINES / FRAME
VERTICAL SYNC (VS)	:	15	LINES / FRAME
VERTICAL BACK PORCH (VBP)	:	61	LINES / FRAME
VERTICAL TOTAL	:	475	LINES / FRAME (+ .5 line if interlaced)

$$\text{FRAME TIME} = \frac{\text{HORIZONTAL TOTAL} \times \text{VERTICAL TOTAL} \times \text{WORDSIZE}}{\text{PIXEL CLOCK}} = 20.034 \text{ mS}$$

APPENDIX E - PIN OUTS FOR VIDEO CONNECTIONS

PLUTO IBM

PIN OUTS FOR 9 WAY D-TYPE VIDEO CONNECTOR

PIN NUMBER	SIGNAL
1	GROUND
2	GROUND
3	RED
4	GREEN
5	BLUE
6	I
7	C.SYNC
8	H.SYNC
9	V.SYNC

PLUTO BOX

PIN OUTS FOR 9 WAY D-TYPE VIDEO CONNECTOR

PIN NUMBER	SIGNAL
1	V.SYNC
2	H.SYNC
3	GROUND
4	RED
5	GREEN
6	GROUND
7	COMPOSITE (GREEN)
8	COMBINED SYNC
9	BLUE

PLUTO SIRIUS

9 WAY 'D' CONNECTOR	PL1 CONNECTOR	SIGNAL
1	5	V.SYNC
2	6	H.SYNC
3	8	GROUND
4	2	RED
5	4	GREEN
6	1	INTENSITY
7	-	-
8	9	C.SYNC
9	3	BLUE
-	7	+ 5V
-	10	NOT CONNECTED

APPENDIX F - HARDWARE TECHNICAL NOTES

1. PLUTO I I/O PORT CONFIGURATION

Only two I/O ports are required for communication with Pluto. The ports have consecutive addresses that may be selected to be on any 20 hex byte boundary. Pluto I decodes 4 addresses, two of which are not used but reserved for future use.

To select the I/O address, link the point marked COM (next to IC30) to one of the points 0 to 7, as required.

STATUS Port addresses (hex)	COMMAND Port addresses (hex)	Link COM to
00	01	4
20	21	5
40	41	6
60	61	7
80	81	0
A0	A1	1
C0	C1	2
E0	E1	3

Pluto I is pre-configured with a base address of A0

For compatibility with Nascom systems a NASIO signal is optionally provided by Pluto. Only one board in the entire system should provide this signal which is asserted when an I/O address for the Nascom main board is decoded. If this signal is to be provided by Pluto then the points marked NASIO should be linked. Pluto asserts this signal for all I/O addresses from 0 to 07f hex inclusive which means that all peripheral boards (including Pluto) should use an I/O address above 80 hex. The Nascom Internal/External I/O addressing switch must be set to enable external addressing.

For compatibility with Nascom 1 systems a DBDR option is provided, by linking the points marked on the CCT board next to the edge connector.

The points marked WAIT should not be linked for any system.

2. PLUTO I PLA CONNECTOR

The connector marked PLA provides the video output signals which are suitable for connection to a standard RGB monitor.

Separate horizontal and vertical sync pulses are provided as well as a combined (horizontal and vertical) sync pulse to cater for most types of monitor. All syncs are negative TTL level pulses and the three colour signals (Red, Green and Blue) are positive TTL levels. Alternate pins on PLA are grounded for maximum noise immunity.

A composite video signal is provided on the green channel for use with a standard monochrome monitor (IV video level).

It is advisable to use a high resolution RGB colour monitor. A medium or low resolution colour monitor can be used but the picture quality will suffer. High resolution monitors have a dot spacing of .31mm or less and are capable of displaying a 640H x 288V screen resolution. A standard horizontal scan rate of 15.625 and vertical frame rate of 50Hz is used. For the double resolution display of 640H x 576V, Pluto produces an interlaced display. Most standard monitors are capable of working with such a display.

The connections to PLA are listed below. Pin 1 is marked with a small arrow and pins are numbered as for standard 20 way ribbon connectors.

PLA pin	Signal
1	GREEN
3	GROUND
5	HORIZONTAL SYNC
7	BLUE
9	GROUND
11	VERTICAL SYNC
13	COMBINED HORIZ. AND VERT. SYNCS
15	COMPOSITE VIDEO
17	GROUND
19	RED
2,4,6,8,10,12,14 16,18,20	GROUND

3. PLUTO RS232 PIN OUTS

The following are the pin numbers for the 25 way D-type socket on the back of the Pluto box:

PIN	2	:	Rx DATA
PIN	3	:	Tx DATA
PIN	4	:	CTS
PIN	5	:	RTS
PIN	6	:	(connected to Pin 8)
PIN	7	:	SIGNAL GROUND
PIN	8	:	DTR
PIN	20	:	DCD/DSR

All other pins are not connected.

4. INTERFACING PLUTO WITH NON-80 BUS SYSTEMS

This section lists Pluto's signal and timing requirements. Numbers in brackets are the pin numbers on Pluto's 78-way bus.

ADDRESS LINES A2-A4 (32-34)
These should be at logic 0 level to select Pluto.

ADDRESS LINES A5-A7 (35-37)
These address Pluto according to the setting of the address decode link (as explained in Links, Section 6 of this appendix). For example, if COM is linked to address decode 4 then A5-A7 should be at logic 0 level to select Pluto.

ADDRESS LINES A0 (30)
Selects between the COMMAND (A0 logic level 1) and STATUS (A0 logic level 0) ports.

DB0-DB7 (50-57)
Bi-directional data bus.

/IORQ (26)
This should be at logic level 0 to select Pluto (it may be kept permanently at logic 0).

/RD (29)
Read strobe (active low) to enable data from Pluto onto the data bus. Data access time from the leading edge (high-to-low) of /RD is 70nsec max. The address and /IORQ lines should be stable 30 nsecs prior to /RD leading edge and stay valid for 40 nsecs after the trailing edge.

/WR (28)
Write strobe (active low) to write data from the data bus to Pluto. Data is latched on the trailing edge (low to high transition) of /WR. Data need not be valid before the leading edge of /WR, but must be valid 30 ns before the trailing edge and remain valid for 40 ns after. The address and /IORQ lines should be stable 30 nsecs prior to the /WR leading edge, and stay valid until 40 ns after the trailing edge.

+5 v (75,76,77,78)
A single +5 volt supply is required. Typical current consumption is 1.5 Amps.

GROUND (1,2,3,4)

All other signals are defined by the 80-Bus specification and are not required by Pluto.

5. INSTALLING IBM PLUTO

The Pluto IBM PC graphics card is contained on a 6 layer IBM PC board. The procedures to install the 3 hardware versions of the Pluto IBM PC graphics card are as follows:

1. Disconnect the system unit power cord from the mains.

2. Disconnect all cables from the rear of the system unit.

3. Move the keyboard away from the work area.

4. Use a flat blade screwdriver and remove the 5 cover mounting screws from the back of the unit (if necessary refer to the IBM Technical documentation for more details).

5. Slide the system unit's cover away from the rear and set it aside.

6. Select one of the spare slots inside the unit and remove the matching expansion slot cover at the back of the unit (If using an AT microcomputer, use one of the PC lookalike slots).

7. Plug the Pluto board firmly in the expansion slot and install the new back panel with the 9-way D-type in place of the old back cover.

8. Position and screw the back panel onto the rear of the unit.

9. Replace the cover and fasten the five screws at the back of the unit.

10. Replace all cables and connect the colour monitor via the 9-way D-type to the Pluto card. The system is now ready for use.

When the machine is switched on there may be a random pattern on the colour monitor momentarily, but this will be cleared and a blank screen will display. The machine will operate normally with the board plugged in.

Port address

The board supplied is configured to I/O port address 832 for status and address 834 for data/commands (these are in the IBM's 8088 I/O space, not the memory space). If different addresses are required these must be specified when ordering the board or may be changed by a replacement chip.

Colour Monitors

The Pluto IBM PC board provides 4 video outputs to drive an IBM compatible monitor. Three of the outputs can be used to drive non-IBM monitors with normal RGB inputs. Separate vertical and horizontal sync drives are provided as well as a composite horizontal and vertical sync signal. The vertical and horizontal syncs are positive going. If negative going syncs are required then this must be specified when ordering.

It is recommended that a high resolution RGB colour monitor is used for Version 1 (768H x 576V, Interlaced, 16 colours) of the Pluto IBM PC graphics card. A medium or low resolution colour monitor can be used but the picture quality will suffer. High resolution monitors have a dot spacing of .31mm, and use a horizontal scan rate of 15.625 and a vertical frame rate of 50Hz. Most standard monitors are capable of working with an interlaced screen display.

Video connectors

The video output is obtained from the 9-way D-type connector, with the pin connections as follows:

Pin	Function	
1	Ground	
2	Ground	
3	Red Output	
4	Green Output	
5	Blue Output	
6	IBM Intensity Output	
7	Composite Sync (Negative)	Configurable
8	Horizontal Sync (Positive)	
9	Vertical Sync (Positive)	

Cotron Hi-Line Rate monitor connections

Pin	1	----	Pin	14	+ 12 volts
	2	----		15	Contrast
	3	Composite Sync		16	TTL Horizontal Sync
	4	+ 12 volts		17	Ground
	5	TTL Vertical Sync		18	Ground
	6	Brightness		19	Ground
	7	Ground		20	TTL Red LSB
	8	TTL R MSB		21	TTL Green LSB
	9	TTL G MSB		22	TTL Blue LSB
	10	TTL B MSM		23	Red Ground
	11	Analog Red		24	Green Ground
	12	Analog Green		25	Blue Ground
	13	Analog Blue			

6. PLUTO II INSTALLATION NOTES

Pluto II is contained on a 8" x 12" board and has an 80-Bus interface. A Serial port (for an RS232 Interface) is included as standard on the Pluto II system. The Auxiliary connector can be used to connect mutliple Pluto II boards.

Power supplies

The Pluto II system requires only +5 volts if it is to be operated over the parallel port. For RS232 operation (over a serial port), +/- 12 volts are also required. The power is supplied over the 80-Bus on the pin numbers as stated overleaf.

Pin	Voltage	
1	GROUND	
2	GROUND	Common on the PCB
3	GROUND	
4	GROUND	
68	-5v	Common on the PCB
69	-5v	
70	-12v	Common on the PCB
71	-12v	
(72	keyway)	
73	+12v	Common on the PCB
74	+12v	
75	+5 v	
76	+5 v	Common on the PCB
77	+5v	
78	+5v	

Links

The hardware configuration is defined using the following links:

Host Address Link

This is situated near the 80-Bus and is labelled E0 at one end and ON at the other. The ON position permanently selects the board (all the bus address lines and /IORQ are ignored). The next position up from ON selects base address 00, and subsequent positions select hex addresses of: 20H, 40H, 60H, 80H, A0H, C0H and E0H.

Status Readback Link

This is situated between IC58 and IC68 and is used to select which data line the hardware status is routed onto when the STATUS port is read. Normally this is on data line 7. This option is useful if multiple Pluto II boards are to be controlled from the same port address.

Video Sync On Green

To produce a composite sync signal on the green channel, J5 should be linked. To disable this facility leave J5 unlinked. Video Setup J4 selects the voltage difference between 'black level' and the absolute blanking level. This is known as the 'setuplevel'. For zero setup (compatible with PAL systems) the link should be in position b, for 142 mV setup it should be in position a and for 71 mV setup it should be left unlinked (compatible with NTSC systems).

THE REMAINING LINKS SHOULD BE LEFT AS SUPPLIED.

RS232 connector

The RS232 connector is a 20-way header labelled 'S'. The pin numbers are labelled as for a ribbon connector. All odd numbered pins are grounded. The even numbered pins are detailed overleaf.

Pin	Signal
2	DCD
4	DSR
6	Rx Data
8	CTS
10	Tx Data
12	Tx clock (ext. clock if J1 is linked)
14	DTR
16	RTS
18	-12 volts
20	+12 volts

Pin numbers for cable connectors

The following are the pin numbers for the 25 way D-type socket on the back of the Pluto box:

PIN 2 : Rx DATA
PIN 3 : Tx DATA
PIN 4 : CTS
PIN 5 : RTS
PIN 6 : (connected to Pin 8)
PIN 7 : SIGNAL GROUND
PIN 8 : DTR
PIN 20 : DCD/DSR

All other pins are not connected.

Pluto II will abort the current command if the DTR line is dropped for longer than 200 microseconds. Pluto II drops the RTS line when it's busy, to stop incoming data.

The RS232 port is configured for:

8 bit data

1 stop bit

parity disabled

9600 Baud (from 50 Baud to 19.2K Baud can be specified when ordering)

External Oscillator connector

This is available to enable Pluto II to be run from an external 30 MHz oscillator, allowing the addition of an external sync unit. The socket is positioned near the AUX connector and is supplied with a 30 MHz oscillator module fitted. The pin connections on the 14 pin (Dual in Line) socket are detailed overleaf.

Pin	Signal
1	GND
2	Pluto II Horizontal sync (-ve)
3	GND
4	Pluto II Vertical sync (-ve)
5	GND
6	Pluto II Reset (active low)
7	GND
8	Master 30 MHz input
9	GND
10	GND
11	GND
12	GND
13	GND
14	+5 volts

Auxiliary connector

The AUX connector is a 10-way header used to connect multiple Pluto II boards. The signals are as follows:

Pin	Signal
1	Composite H and V sync (-ve)
3	Horizontal sync (-ve)
5	Master oscillator output
7	Master reset (active low)
9	Vertical sync (-ve)

All signals are at TTL levels. All even numbered pins are grounded.

Video output

A row of 6 miniature SMB type connectors are used for all video connections. These are marked R G B S S and C. The two S connectors supply a combined horizontal and vertical sync (negative going), one is for the monitor and the other is to sync a camera (the camera must be capable of external synchronisation onto Pluto II). The C connector is an input for a monochrome camera (1 volt into 75ohms). An RGB colour camera can be used with a 24 Bit Pluto system.

Camera adjustment

RV1 is for adjusting brightness and RV2 is for adjusting contrast. The contrast is factory set and should not be adjusted. The brightness will need to be adjusted for a particular camera. Adjust this while 'previewing', i.e. issue the Preview command, with a parameter of 1 to turn it on. If RV2 is adjusted it should be set up so there is -1 volt on pin 23 of IC70 (the socketed 24 pin chip near the C connector).

Using multiple Pluto II boards

The simplest way to use multiple Pluto II boards is to operate them at different hardware addresses and control them independently. If it is desired to operate them from a common interface or all at the same address, then the software bank select mechanism must be used.

Up to 7 Pluto II boards may be controlled using the software bank select mechanism. Each board must have it's status line linked to a different data line and each board must be ordered with a unique bank number in it's ROM (this is a factory set option). The boards would have bank numbers 1,2,4,8,16,32 and 64. To select one or more boards the SBank command is used - the parameter sent with SBank is the sum of the bank numbers to be enabled. For example, if bank 4 and bank 32 are to be enabled, then the command SBank (36) would be used.

Software protocol for multiple Pluto II boards

When using multiple Pluto II boards on one bus as described above (the Pluto 24 Bit system, see Section 3.5, is an example of this type of usage), the following guidelines must be used in the software interface.

Commands are executed in the normal way by all banks which are enabled - this includes the SBank command. To reselect the banks which are enabled, the SBank command is used. Once a board is disabled it will no longer listen to commands or any data transfers until a software RESET is performed. This is done by sending four zero bytes to the STATUS port (wait for a ready status from all boards before doing this). This will get the attention of all boards in the system, whether they are bank selected or not. All boards will listen to the very next command and all boards will respond normally to the command if it is a PInit or an SBank. If any other command is sent instead, then the deselected banks will go back to the 'not responding' state.

Synchronising multiple Pluto II boards

The video outputs of multiple boards may be synchronised together using a 10-way ribbon cable running between the AUX connectors on each board. One board must be factory configured as a MASTER to generate the main synchronising signals and all other boards must be configured as SLAVES.

7. PLUTO II 24 BIT SYSTEM INSTALLATION NOTES

The Pluto II 24 Bit system is a special configuration of 3 Pluto II boards (and must be ordered as a 24 Bit system). One board is configured as a master and the others as slaves. The bank numbers used are 1,2 and 4 (bank 4 is equivalent to bank 0 for compatibility with Pluto I). Each board has only one video output (either red, green or blue) allowing free control over each colour component giving over 16.7 million simultaneously displayable colour combinations.

Note: The serial interface port cannot be used in a 24 Bit system to communicate to all boards simultaneously.

Using the Pluto II 24 Bit system with a BBC micro

With the standard Pluto/BBC interface it is not possible to issue a write to the Pluto Status Port. Instead, the following (equivalent) procedure has to be adopted:

1. Generate a WR low pulse (as for a normal write command).
2. While WR is low generate a RD low pulse.
3. Take WR and RD high together.

This simultaneous read and write operation signals a software RESET to Pluto (similar to writing to the STATUS port).

8. MEGARES INSTALLATION NOTES

Megares is contained on a 8" x 10" board and has an 80-Bus interface. A Serial port (for an RS232 Interface) is included as standard on the Megares system. The Auxiliary connector can be used to connect multiple Megares boards.

Power supplies

The Megares system requires only +5 volts if it is to be operated over the parallel port. For RS232 operation (over a serial port), +/- 12 volts are also required. The power is supplied over the 80-Bus on the following pin numbers:

Pin	Voltage
1	GROUND
2	GROUND
3	GROUND
4	GROUND
70	-12 v
71	-12 v
(72	keyway)
73	+12 v
74	+12 v
75	+5 v (also to video output stage)
76	+5 v (" " " " ")
77	+5 v (" " " " ")
78	+5 v (" " " " ")

Links

The hardware configuration is defined using the following links:

Host Address Link

This is situated near the 80-Bus and is labelled E0 at one end and ON at the other. The ON position permanently selects the board (all the bus address lines and /IORQ are ignored). The next position up from ON selects base address 00, and subsequent positions select hex addresses of: 20H, 40H, 60H, 80H, A0H, C0H and E0H.

Status Readback Link

This is labelled LK11 and is used to select which data line the hardware status is routed onto when the STATUS port is read. Normally this is on data line 7. This option is useful if multiple Megares boards are to be controlled from the same port address.

Sync On Green

To produce a composite sync signal on the green channel, link 2 of the 3 pins on LK3 so that the free pin is nearest the video connectors. To disable this facility, link the middle pin of LK3 to the opposite side (the default setting is no sync on green).

THE REMAINING LINKS SHOULD BE LEFT AS SUPPLIED.

RS232 connector

The RS232 connector is a 20-way header labelled 'S'. The pin numbers are labelled as for a ribbon connector. All odd numbered pins are grounded. The even numbered pins are as follows:

Pin	Signal
2	DCD
4	DSr
6	Rx Data
8	CTS
10	Tx Data
12	Tx clock (ext. clock if J1 is linked)
14	DTR
16	RTS
18	-12 volts
20	+12 volts

Pin numbers for cable connectors

Detailed overleaf are the pin numbers for the 25 way D-type socket on the back of the Pluto box:

PIN	2	:	Rx DATA
PIN	3	:	Tx DATA
PIN	4	:	CTS
PIN	5	:	RTS
PIN	6	:	(connected to Pin 8)
PIN	7	:	SIGNAL GROUND
PIN	8	:	DTR
PIN	20	:	DCD/DSR

All other pins are not connected.

Megares will abort the current command if the DTR line is dropped for longer than 200 microseconds. Megares drops the RTS line when it's busy, to stop incoming data.

Auxiliary connector

The AUX connector is a 10-way header used to connect multiple Megares boards. The signals are as follows:

Pin	Signal
1	Composite H and V sync (-ve)
3	Horizontal sync (-ve)
5	Master oscillator output
7	Master reset (active low)
9	Vertical sync (-ve)

All signals are at TTL levels. All even numbered pins are grounded.

Video output

A row of four miniature SMB type connectors are used for all video connections. These are marked R G B S. The S connector supplies a combined horizontal and vertical sync (negative going).

Using multiple Megares boards

The simplest way to use multiple Megares boards is to operate them at different hardware addresses and control them independently. If it is desired to operate them from a common interface or all at the same address, then the software bank select mechanism must be used.

Up to 7 Megares boards may be controlled using the software bank select mechanism. Each board must have it's status line linked to a different data line and each board must be ordered with a unique bank number in it's ROM (this is a factory set option). The boards would have bank numbers 1,2,4,8,16,32 and 64. To select one or more boards the SBank command is used - the parameter sent with SBank is the sum of the bank numbers to be enabled. For example, if bank 4 and bank 32 are to be enabled, then the command SBank (36) would be used.

Only the currently enabled bank(s) respond to commands. The exceptions to this are PInit and SBank which all banks execute.

Synchronising multiple Megares boards

The video outputs of multiple boards may be synchronised together using a 10-way ribbon cable running between the AUX connectors on each board. One board must be factory configured as a MASTER to generate the main synchronising signals and all other boards must be configured as SLAVES.

9. PLUTO APRICOT INTERFACE INSTALLATION

INSTALLATION

Plug the interface card into either of the internal expansion bus slots inside the Apricot.

Plug the 16 pin header on the inter-connecting ribbon cable into the red socket on the interface card (it may help to do this before installing the card). The side of the ribbon marked red, should be connected to the end of the red socket marked by a white blob on the interface card (ie. pin 1 of the plug goes to pin 1 of the socket). Take ribbon cable out through the slot in the back of the Apricot and plug the 15-way D connector into the back of PLUTO.

USE

Two "ports" are required to communicate with PLUTO - the DATA and STATUS ports. On the Apricot these are I/O mapped.

The two ports used -

PORT 100 hex (256 decimal) is the STATUS read/write PORT
PORT 102 hex is the DATA read/write PORT

The addresses of these ports may be altered by changing a link on the interface card.

SIGNAL	R-TO VCC	R-TO GND
CSX	R1	R2
BALE	R3	R4
BRD	R5	R6
BWR	R7	R8
BIORQ	R9	R10
BUFDIR	R11	R12

The resistor to Vcc should be 220 ohms, and to ground 330 ohms. This interface board is sent without these terminating resistors.

The 15 way D-type pin configuration is shown below :-

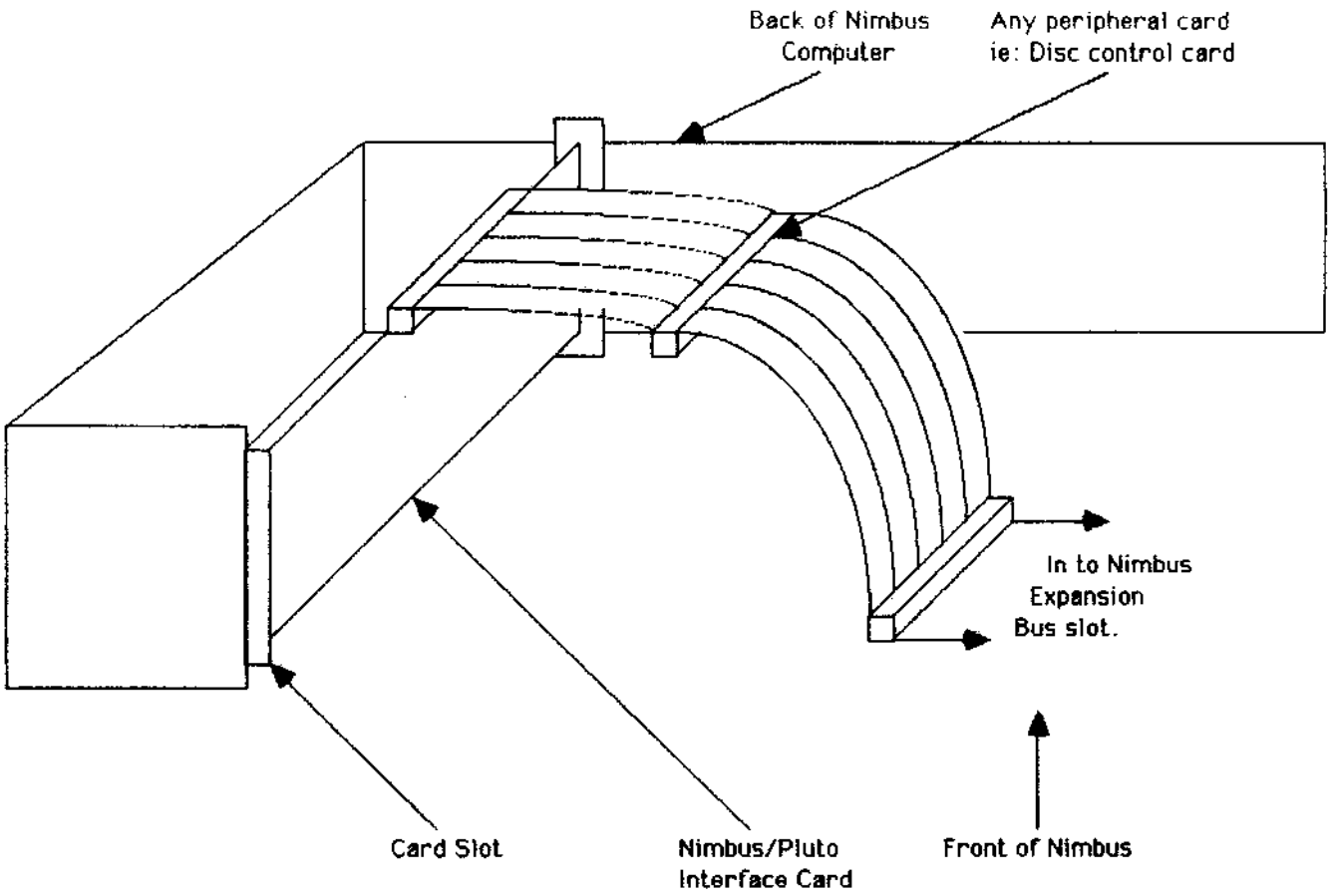
PIN	SIGNAL
1	D0
2	D2
3	D4
4	D6
5	GROUND
6	GROUND
7	GROUND
8	GROUND
9	D1
10	D3
11	D5
12	D7
13	/WR
14	/RD
15	NIMBUS A1

The two ports required to communicate with PLUTO -
the DATA and STATUS ports are as follows :-

PORT 1280 (decimal) is the STATUS read/write port.

PORT 128.2 (decimal) is the DATA read/write port.

10. Pluto Nimbus Interface Installation.

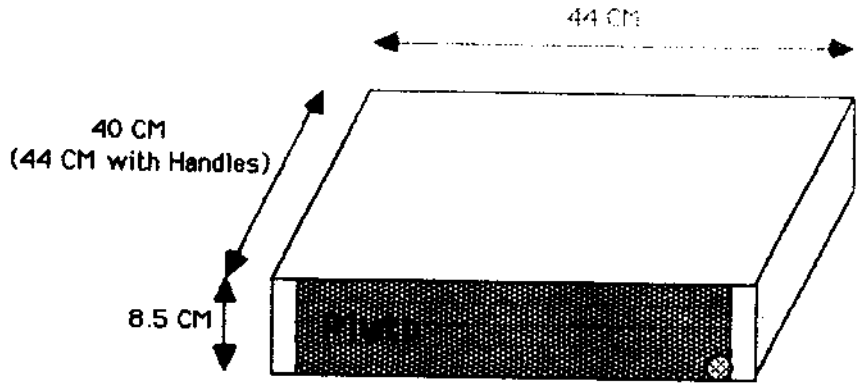


General Diagram showing installation of Nimbus/Pluto interface in the Nimbus Computer See your Hardware Handbook for exact details.

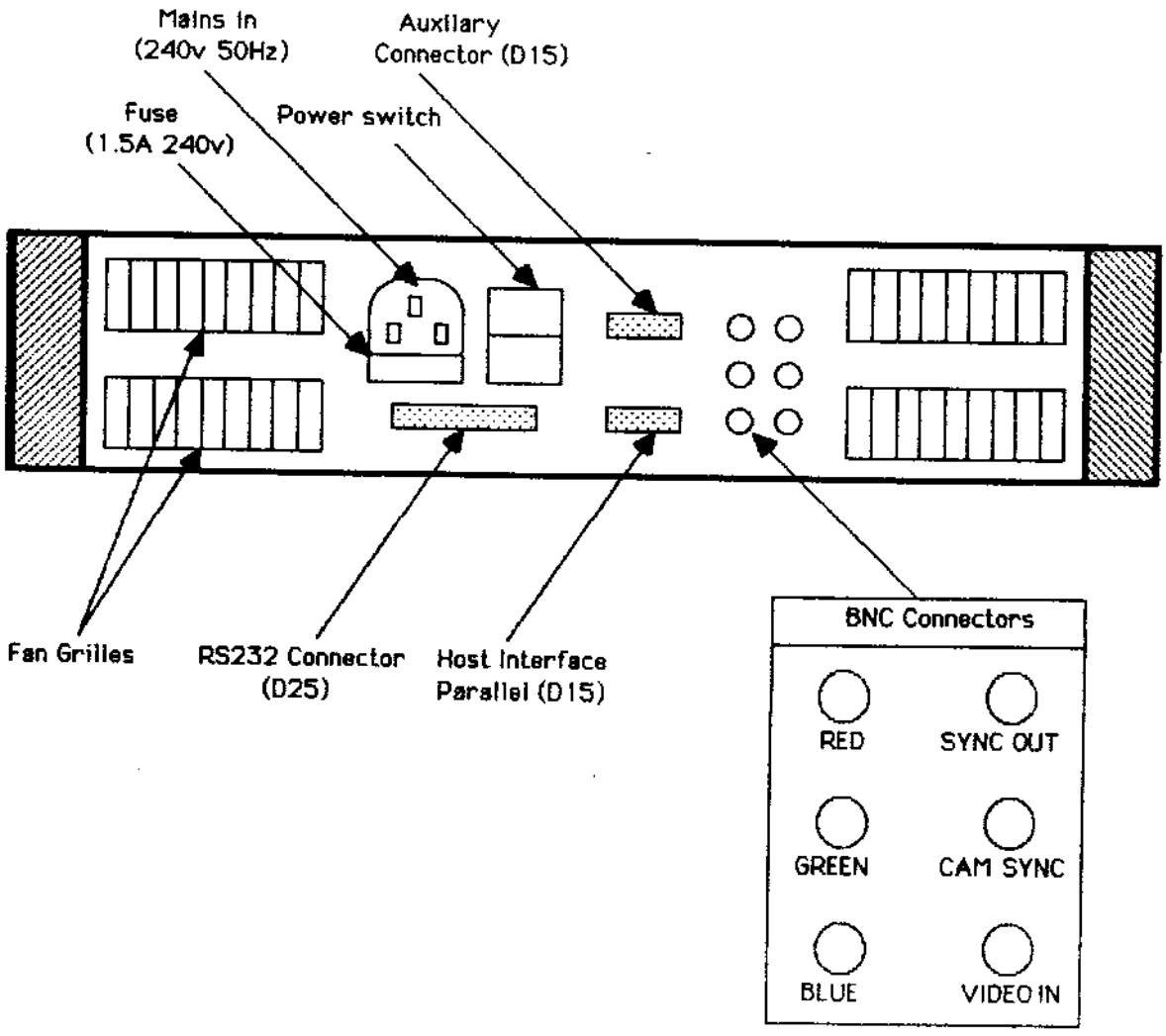
SECTION 11

The PLUTO Box

19" Rack Mount Box



Back panel layout



PLUTO USER MANUAL

We welcome your comments and suggestions. They help us provide you with better product information.

****

Please tell us how you rate this manual in the following areas by ticking the appropriate boxes:

	Good	Average	Poor
1. Completeness of information	[ ]	[ ]	[ ]
2. Clarity of information	[ ]	[ ]	[ ]
3. Accuracy of information	[ ]	[ ]	[ ]
4. Layout of the manual	[ ]	[ ]	[ ]
5. Overall rating of the manual	[ ]	[ ]	[ ]

Name ..... Date .....

Title ..... Telephone No .....

Company Name/Department .....

Address .....

.....

All comments and suggestions become the property of:

IO RESEARCH LIMITED

4 Exchange Buildings

High Street

Barnet

HERTS EN5 5SY