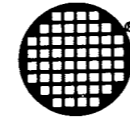


# IMS B007 evaluation board

HAI

inmos®



INMOS Limited  
1000 Aztec West  
Almondsbury  
Bristol BS12 4SQ  
UK  
Telephone (0454) 616616  
Telex 444723


INMOS Corporation  
PO Box 16000  
Colorado Springs  
CO 80935  
USA  
Telephone (303) 630 4000  
TWX 910 920 4904

INMOS GmbH  
Danziger Strasse 2  
8057 Eching  
Munich  
West Germany  
Telephone (089) 319 10 28  
Telex 522645

INMOS SARL  
Immeuble Monaco  
7 rue Le Corbusier  
SILIC 219  
94518 Rungis Cedex  
France  
Telephone (1) 46.87.22.01  
Telex 201222

INMOS International  
Room 308 Kowa No. 16 Annex  
9-20 Akasaka 1-chome  
Minato-ku  
Tokyo 107  
Japan  
Telephone 03-505-2840  
Telex J29507 TEI JPN

INMOS reserves the right to change this document and the products described herein at any time and without notice.

 inmos, IMS, and occam are trade marks of the INMOS Group of Companies.

June 1986

72 BRD 024 01

---

**1 Getting started with IMS B007**

---

- 1.1 Checking contents of the bag
- 1.2 Mounting the board for use
- 1.3 Power supply
- 1.4 Connecting up to the power supply
- 1.5 Checking the setting of the coding switches
- 1.6 Resetting the board
- 1.7 Connecting to another board

---

**2 Evaluation board architecture**

---

- 2.1 Links
- 2.2 Lower edge connector pinout
- 2.3 Lower edge connector
- 2.4 Connecting to the transputer development system

---

**3 IMS B007 graphics board functional description**

---

- 3.1 Memory
- 3.2 Display control

---

**4 IMS B007 graphics board software**

---

- 4.1 Memory test program
  - 4.2 CRT set up
  - 4.3 Running the CRT set up program
  - 4.4 Running the test program
    - 4.4.1 What you will see
  - 4.5 Running the demonstration program
  - 4.6 Software description
  - 4.7 Windows and screens
  - 4.8 Text manipulation
  - 4.9 Drawing modes
  - 4.10 IMS G170 colour look up table
  - 4.11 CRT commands
-

---

4.12 Stopping the graphics software

---

5 Using the graphics primitives

---

5.1 Graphics primitives description and access

5.2 Graphics commands

5.3 Text commands

5.4 Screen and window commands

5.5 Colour commands

5.6 General commands

5.7 CRT commands

5.8 Error codes

---

6 Graphics commands and error codes summary

---

6.1 Commands

6.2 Error codes

---

7 Character set

---

8 Memory Map

---

9 Component layout

---

10 Circuit diagram

---

Disclaimer

Every effort has been made to test this product and its operation with the transputer development system. Note, however, that the board contains a 'prequal' version of the transputer on which engineering characterisation and life tests have not been performed.

©, **inmos**, IMS and occam are trade marks of the INMOS Group of Companies.

INMOS reserves the right to make changes in specifications at any time and without notice. The information furnished by INMOS in this publication is believed to be accurate, but no responsibility is assumed for its use, nor for any infringements of patents or other rights of third parties resulting from its use. No licence is granted under any patents, trademarks or other rights of the INMOS Group of Companies.

Copyright 1986 INMOS Limited. This document may not be copied, in whole or in part, without prior written consent of INMOS Limited.

72 BRD 024 01

The IMS B007 is a high performance graphics board, suitable for use with high resolution colour graphic raster scan monitors.

The IMS B007 evaluation board enables users to evaluate and demonstrate the use of transputers. The board is a member of a family of compatible evaluation boards. It provides standard buffered INMOS link connections and external control of the transputer's Reset and Analyse functions. This allows it to control a subsystem consisting of other compatible boards, or to be a component of such a subsystem.

This manual details the product specific aspects of the IMS B007, and contains all the data necessary to power up, test and program the board.

Other information relevant to all transputer products is contained in the occam programming manual (supplied with INMOS software products and available as a separate publication), and the transputer reference manual (supplied with this board). This board is designed to be used in conjunction with a transputer development system, and reference should be made to the transputer development system user manual (supplied with the development system), for details of how to compile and load programs for a network of boards.

1.1 Checking contents of the bag

Ensure that along with this manual the contents of the bag include: a red plastic case containing the board; a set of cables plugged into a 64 way edge connector socket; a second edge connector with integral sockets for RGB coaxial cables; some other documentation and the graphics software.

1.2 Mounting the board for use

The red plastic case which holds the board also acts as a mounting rack for up to six boards. The board itself is mounted on a frame, which slides into the top of the case.

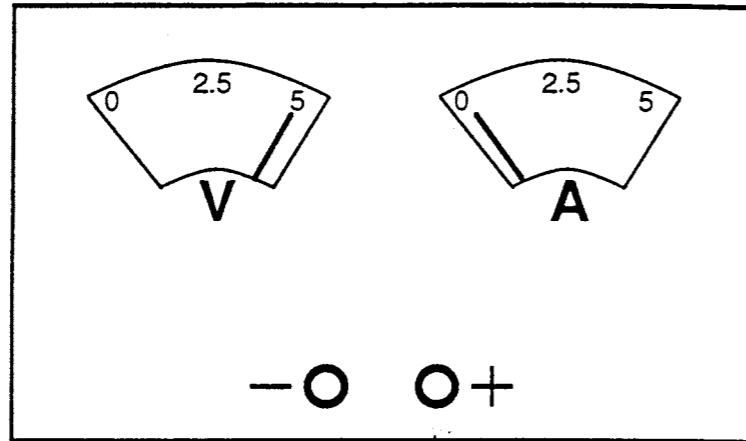
To remove the board and mount it on the top of the case:

- 1 Place the red case on a table so the white arrows are facing towards you. You should also be able to see the Anti-Static warning facing you on the frame.
- 2 With your thumbs, gently push apart the clips on either side of the frame so that the frame is released.
- 3 Tilt the case so that the frame slides out.
- 4 Without touching the circuit board, withdraw the frame from the case.
- 5 When you do touch the board, touch it first at either end of the 64 way edge connector; the pins at both ends are GND.
- 6 Turn the frame so the Anti-Static warning faces down and the component side of the board faces you.
- 7 Place the frame over the white arrows on the case. Gently slide the frame towards you into the slots either side of the case, as shown in the white painted picture on the case.
- 8 When you come to plug the cables into the board, do so carefully.

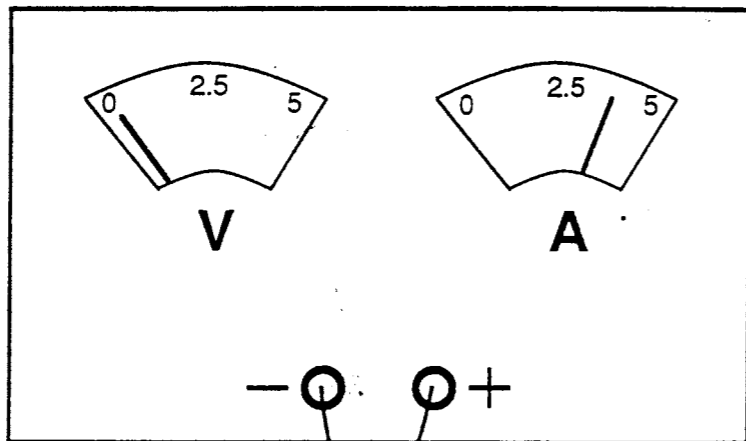
Up to six frames can be mounted onto the case.

1.3 Power supply

A power supply capable of delivering 3.5A at 5V is required for correct operation of the B007 evaluation board. Set the supply at 5V, with current limiting at 3.5 Amps.



Voltage setup



Current setup

1.4 Connecting up to the power supply

Ensure the power supply is switched off.

Leave the cables plugged into the 64 way edge connector socket and plug this socket onto the board.

Among the cables you will find a red banana plug and a black banana plug. The black banana plug connects to 0V and the red banana plug connects to +5V of the power supply. Check you have connected the power cables the correct way round and switch the power supply on.

WARNING

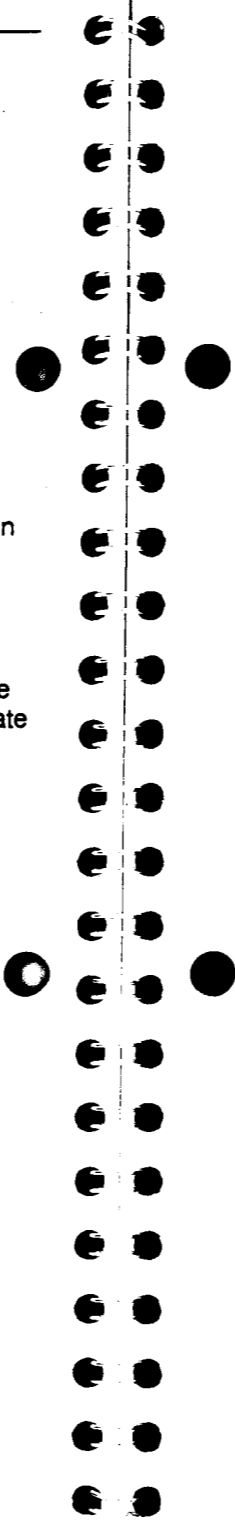
*The IMS B007 edge connector is not compatible with the edge connector of the IMS B000. The board, possibly including the transputer, will be damaged if a B007 is plugged into a B000 socket.*

1.5 Checking the setting of the coding switches

Check the coding switches on the board. There is one set of eight switches, positioned next to the transputer, as shown in the diagram on the following page. The settings are as listed below:

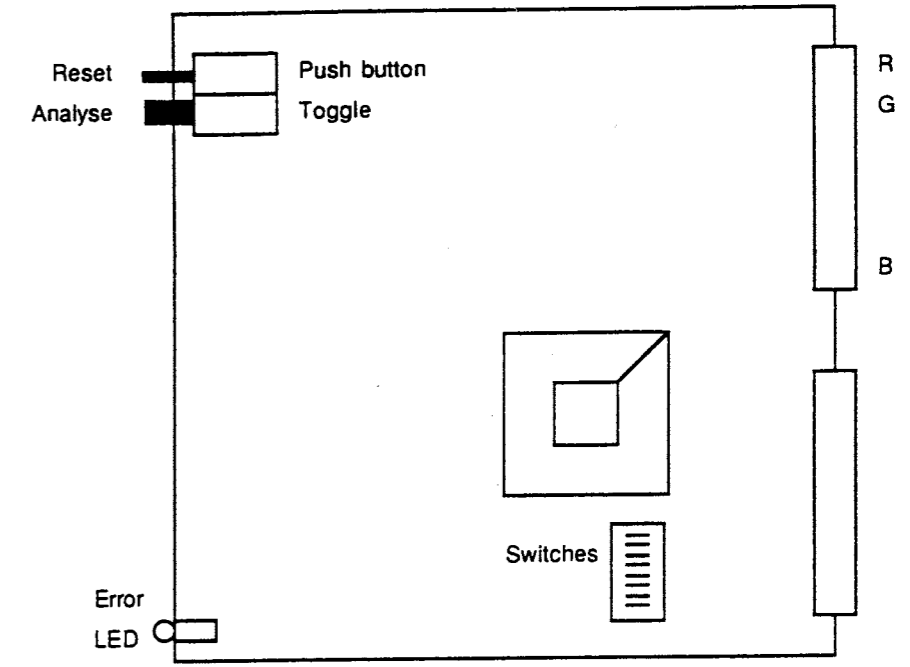
Switch

- 1 ON - do not alter
- 2 Link0Special, when set high link0 operates at non standard speed
- 3 Link123Special, when set high links 1,2 and 3 operate at non standard speed
- 4 disable internal ram, switches off fast internal ram when set high
- 5 special link speed, when high link0 or links 1,2,3 can operate at 10MHz or 20MHz, when low link0 or links 1,2,3 can operate at 5MHz or 10MHz
- 6 ON - do not alter
- 7 Not connected
- 8 Not connected



1.6 Resetting the board

Press the push button to reset the board. The reset button is situated on the opposite side of the board to the edge connectors, as shown below.





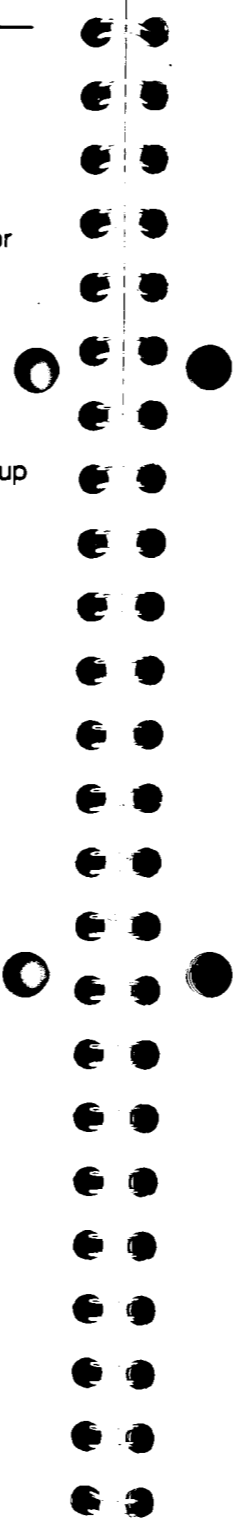
1.7 Connecting to another board

Operation of the B007 graphics board requires the use of a second evaluation board (B001, B002, B004 etc.). This second root processor board is used to provide RS232 connections to a host computer and a terminal. The IMS B001, B002, B007 and other evaluation boards share a common architecture which includes the edge connector pinout, link buffering, and system control functions. The architecture is independent of the type of transputer used, its speed, and of the number of transputers on the board. The pins of the edge connector are grouped in sets of five, suitable for the five way sockets on the cables provided. The margin diagram shows the function of each group of pins. The pinout is described in more detail later. Each five way socket is coded to make it difficult to position it incorrectly.

The banana plugs provided allow several boards to use the same power supply. Note: this power supply must be capable of providing the total current required for the system.

The Up, Down, and Subsystem sockets are concerned with system control, initialisation and error handling.

The transputer's links are brought out to the edge connector, for the user to configure the system as required. A pair of link cables is supplied to enable boards to be connected together.



2.1 Links

The memory map of each board is different, and the memory map for the B007 is given at the end of this manual.

The four link sockets on the edge connector are:

**Link0** pins c7 to c11

**Link1** pins a7 to a11

**Link2** pins c13 to c17

**Link3** pins a13 to a17

The link sockets and cables are coded to make it difficult to plug link cables into the wrong socket.

The IMS B007 buffers its links with F244s. Printed circuitry is also provided for bypassing the buffers with low value resistors (such as 22 ohm). Schottky diodes to GND and VCC in the F244s provide the transputer protection from damage by static voltages on the edge connector pins. With the buffers bypassed it is necessary to protect the transputer from static voltages in the order of 2.0 kV. The 100 ohm resistors must be removed.

2.2 Lower edge connector pinout

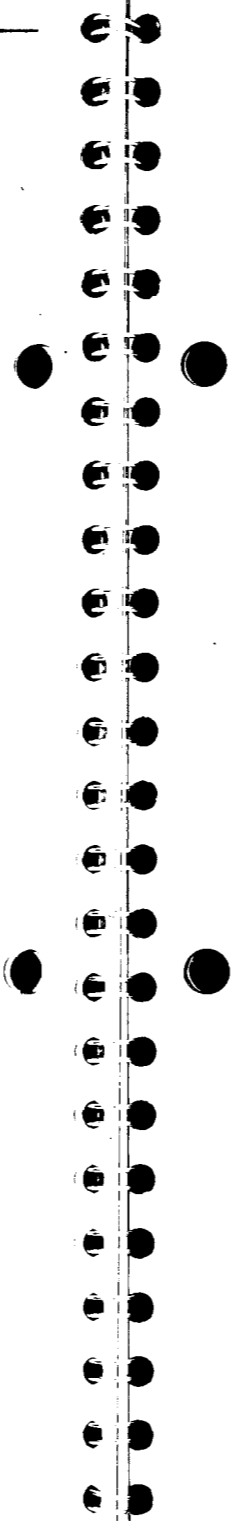
2.2 Lower edge connector pinout

Pin	c	b	a
1	GND		GND
2	VCC		VCC
3	(missing)		(missing)
4	VCC		VCC
5	GND		GND
6	(gap)		(gap)
7	GND		GND
8	(missing)		(missing)
9	LinkOut[0]		LinkOut[1]
10	LinkIn[0]		LinkIn[1]
11	GND		GND
12	(gap)		(gap)
13	GND		GND
14	(missing)		(missing)
15	LinkOut[2]		LinkOut[3]
16	LinkIn[2]		LinkIn[3]
17	GND		GND
18	(gap)		(gap)
19	PIO0		NC
20	PIO1		notBootFromROM
21	PIO2		(gap)
22	PIO3		SubsystemNotReset
23	PIO4		SubsystemNotAnalyse
24	PIO5		SubsystemNotError
25	PIO6		GND(missing)
26	PIO7		(missing)
27	(gap)		(gap)
28	UpNotReset		DownNotReset
29	UpNotAnalyse		DownNotAnalyse
30	UpNotError		DownNotError
31	GND		GND(missing)
32	GND(missing)		GND(missing)

The boards use a two part edge connector conforming to DIN 41612, the same sort of connector as is used on VME boards and Multibus II boards.

IMS B001, B002 and B007 boards use the 64 way version of this connector, with the pinout shown.

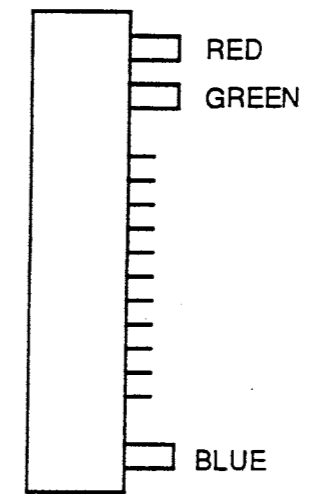
The GND pins c31 and c32 on the board are convenient for scope earth leads.



2.3 Lower edge connector

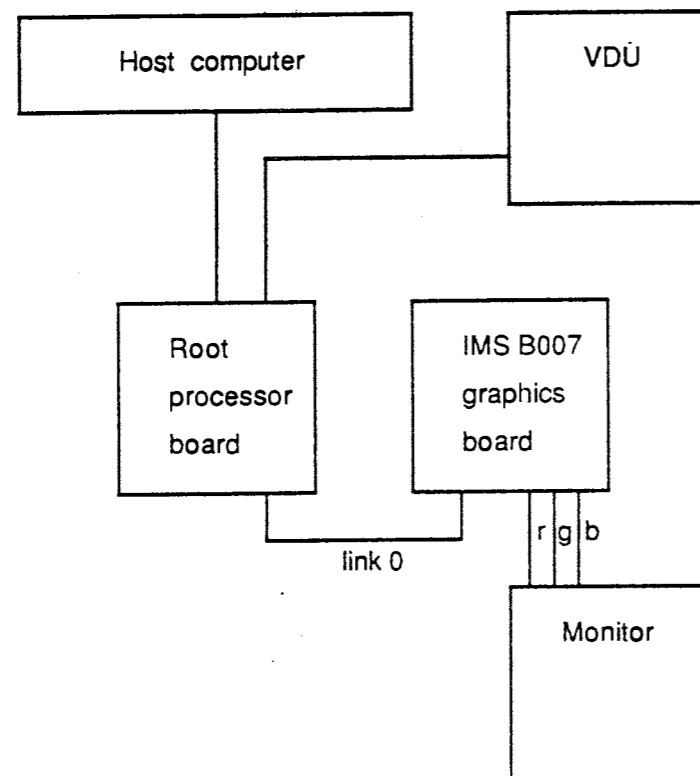
2.3 Lower edge connector

The upper edge connector has integral sockets for the three RGB 75 ohm coaxial cables provided with the board. The RGB connections are as shown below.



## 2.4 Connecting to the transputer development system

To use the board with the transputer development system (TDS) you need to have the TDS installed on your computer and to connect up the system as shown below.



The IMS B007 is a medium resolution, high performance graphics board. It is compatible with other INMOS evaluation boards in both dimension and backplane connection. The board also contains a 96 way DIN 41612 connector, which is compatible with 75 ohm SMB integral connectors for RGB output, and a single error LED.

## 3.1 Memory

The graphics system includes half a megabyte of dual ported video ram, and half a megabyte of program space.

The screen store is split into two quarter megabyte display screens, either of which is software selectable. Each screen looks to the programmer as an array of 512 by 512 bytes. Each byte represents a single pixel mapped so that [0,0] is at top left of the display.

The memory map of the IMS B007 is given at the end of this manual.

## 3.2 Display control

The display consists of 512 by 512 8 bit pixels. The pixel rate is set at 25Mhz (40ns). This can be altered by replacing the pixel clock module on the board. The display is controlled by a memory mapped CRT controller, and is suitable for any colour monitor, interlaced or non-interlaced, with line scan frequencies up to 50khz (depending on value of pixel clock module used). The displayed colours are produced using the IMS G170 colour palette, giving a range of 262,144 colours.

The display hardware can be programmed to event the processor on frame flyback to allow update of the IMS G170 without noise on the display. There is also access to a register in the CRT controller to inform a program of frame flyback.

The software provided with the board includes:

- 1 a suite of low level graphics primitives
- 2 a CRT (video monitor) set up program
- 3 test and demonstration programs

The graphics software is provided as an occam implementation. If you are still using untyped occam (i.e. variable declarations using VAR) then only use the software contained in the fold marked 'untyped occam'.

#### 4.1 Memory test program

Locate the memory test program in the software package provided, and re-compile the program with the correct terminal driver SC for your root processor board if required. Connect links 1 - 4 on the root processor board to links 1 - 4 on the graphics board. Extract the code down to the transputer system and run the terminal emulator. Run the memory test program, any problems will be reported on the terminal screen. Do not worry if the video monitor displays a rolling screen. It is very likely that the board is not set up for your monitor. The next section describes how to obtain the correct CRT values to produce a steady picture.

#### 4.2 CRT set up

The board is capable of providing control signals for a range of video monitors. The scan line frequency, frame rate and interlace are under software control. The pixel clock frequency may be altered by simply replacing the pixel clock module on the board and informing the software of the new value. The default settings are:

scan line frequency - 34,800  
 frame rate - 50  
 interlace - OFF  
 pixel clock frequency - 25,000,000

#### 4.3 Running the CRT set up program

If these are correct for your monitor, or the required values are known, the next section may be ignored.

#### 4.3 Running the CRT set up program

Locate the CRT set up program in the software package provided, and re-compile the program with the correct terminal driver SC for your root processor board if required. Extract the code down to the transputer system and run the terminal emulator program provided with the TDS.

At this point you will see one of two things on the monitor screen. If the default values of scan line frequency, frame rate, interlace and the pixel clock frequency are correct for your particular monitor, there will be two lines on the screen at right angles to each other. The red line lies along the top edge of the screen, the blue down the left edge. By using keys

u up - red line  
 d down - red line  
 l left - blue line  
 r right - blue line

on your terminal these lines may be moved around the screen. If, as is more likely, the default settings are not correct then use the following commands to alter the settings.

h horizontal line frequency  
 v vertical frame rate  
 l interlace ON/OFF  
 p new pixel clock frequency

## 4.3 Running the CRT set up program

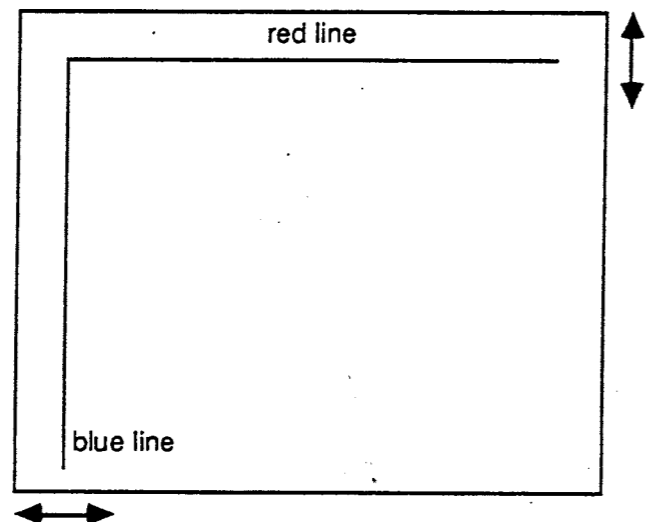
The scan line frequency and frame rate are increased/decreased by typing '+' or '-' after the command. Typing carriage return causes the value now set to be displayed on the terminal screen.

Interlace is toggled ON/OFF simply by typing 'I' return.

If the image on the screen is too narrow/wide then the pixel clock module on the board must be changed. This device is left socketed to facilitate this procedure. If the pixel clock has been changed type the new value after the command 'p'.

Type '?' for help. Any other key will display the currently set values.

When the two lines appear in their correct positions record the values for scan line frequency, frame rate, interlace and pixel clock frequency. These new values will have to be sent to the graphics software to allow the CRT controller to be correctly initialised. How to do this is described in the following section.



## 4.4 Running the test program

## 4.4 Running the test program

If the default CRT monitor settings have to be altered, simply change the default settings at the outermost level of the program. These values will then be sent to the graphics board when the program is run. If you are using typed occam the new values can be altered in the graphics software itself, how to do this is described in the section on how to use the graphics primitives. Re-compile the test program and extract to the transputer network, using the TDS, and check that the display on the monitor corresponds to the description below.

## 4.4.1 What you will see

The test program will cycle through the following displays:

- 1 black screen
- 2 white screen
- 3 full colour tables
- 4 grey scale
- 5 red scale
- 6 green scale
- 7 blue scale
- 8 primary colour circles
- 9 white box enclosing whole screen, with white crosshairs dividing screen into four
- 10 white crosshatch on black background
- 11 white squares on black background

## 4.5 Running the demonstration program

With the terminal emulator program running, the display may be stopped at any point by typing any key on the keyboard. To continue simply press another key.

Use this program to check colour intensities, brightness, vertical and horizontal width etc.

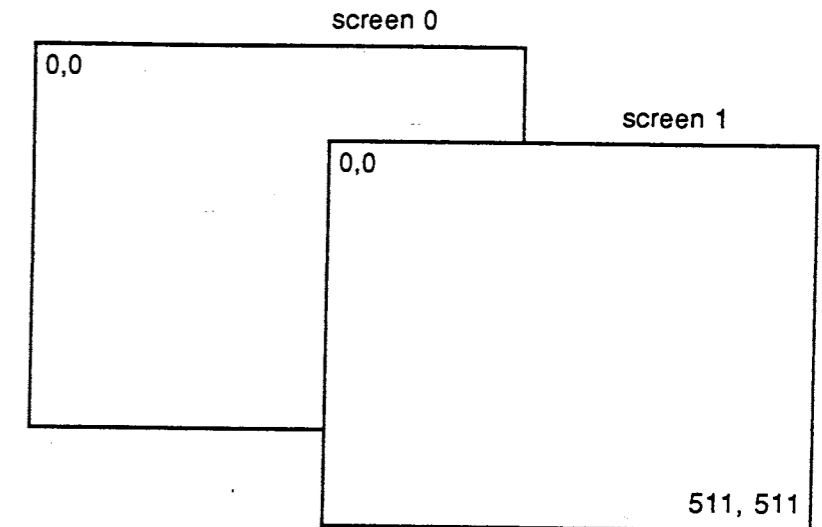
## 4.5 Running the demonstration program

To run the demonstration program set up the CRT values as for the test program. Compile and extract this program down to the boards.

## 4.6 Software description

## 4.6 Software description

The hardware allows two software selectable screens. Each is addressed as a two dimensional 512 x 512 x,y plane with [0,0] at the top left hand corner.



One screen may be displayed whilst the other screen is being used for drawing. The software also provides a mechanism for handling up to 32 windows.

The primitives in the software package include graphics routines for simple line drawing, circles and arcs, polygons and polygon fill. A character set and basic positioning routines are provided to allow text manipulation. Cursor movement affects the text positioning only. Scrolling will scroll everything displayed on the screen, graphics as well as text. With windows cursor positioning is always relative to the top left of a window.

The graphics software utilises the IMS G170 on the graphics board to provide colour. The software uses the concept of a currently selected foreground and background colour. All graphics commands use the current foreground colour as the drawing colour. Text uses both the

foreground and background colours in the default drawing mode. The various text drawing modes and their function are described in a following section.

The width of lines may be altered by changing the **x width** and **y height** values, which effectively changes the default pixel size.

## 4.7 Windows and screens

The IMS B007 hardware provides two accessible display screens, numbered 0 and 1. In addition to these two screens the graphics software provides a windowing mechanism. When a window is defined, memory from a window store is allocated for the permanent storage of any graphics or text written in the window.

The system allows a maximum of 32 windows. An error code is returned when either the maximum number of windows has been exceeded, or the requested window requires more memory than is available in the window store.

An allocated window may be selected for drawing and/or display. If a window is selected for drawing its content is not displayed on the screen until a display command is given. If a window is selected for drawing and display then the screen display is automatically updated at the end of every drawing command.

As screen displays are not saved in program memory, part of a screen display will be overwritten by the contents of a window when that window is displayed.

With a window selected all graphics and text functions apply to the window coordinates, i.e. lines will be clipped to the window, and the window will scroll independently of the screen. Windows may be moved around the screen simply by displaying the window at a new position on the screen. The previous display will not be cleared.

## 4.8 Text manipulation

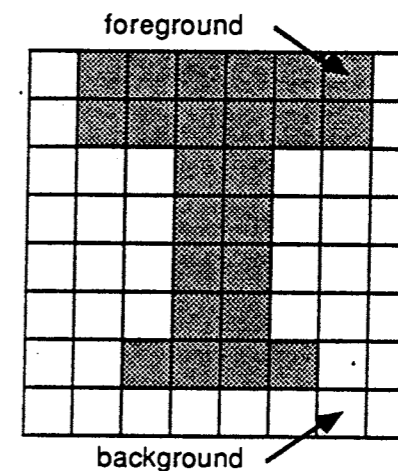
In the default mode set up by the graphics package text is drawn starting at [0,0] on the screen. The cursor position is automatically incremented by the width of the character in the x direction after the character is drawn. Carriage return - line feed is automatic at the screen/window edge as is scrolling when the bottom of the screen or window is reached. The cursor may be positioned explicitly by issuing either a **move** or a **move relative** command. After a **reflect x** or a **reflect y** command all text drawn is reflected in either the x or y plane or both. Characters may also be rotated in quarter turns in a clockwise direction.

Characters may be enlarged by increasing the **x width** or the **y height** value.

Note: these last two values also affect the width of the lines drawn by the graphics commands.

## 4.9 Drawing modes

Eight drawing modes are provided for the display of text characters. Each character in the character set has an implied foreground and background colour as shown in the diagram.



## 4.9 Drawing modes

With the first four drawing modes the whole character, both foreground and background, is drawn. The second four modes draw the foreground only.

With the REPLACE mode screen pixels are simply replaced by those of the character in the currently selected foreground or background colours.

In AND mode the bits corresponding to the colour of the pixels on the screen are anded with the current foreground or background colour. The result of this operation is the colour value written to the screen. Similarly in OR and XOR mode the character bits are ored and exclusive ored with the screen pixels.

Mode	Effect
0	REPLACE foreground and background.
1	AND foreground and background.
2	OR foreground and background.
3	XOR foreground and background.
4	REPLACE foreground only.
5	AND foreground only.
6	OR foreground only.
7	XOR foreground only.

## 4.10 IMS G170 colour look up table

## 4.10 IMS G170 colour look up table

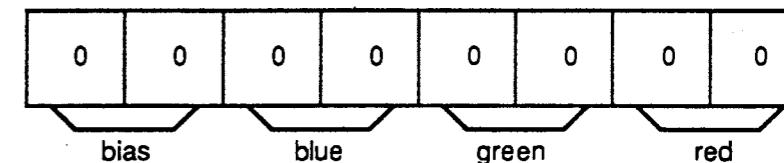
Included on the B007 is an IMS G170 colour look-up table. The 18 bit data word locations in the table allow 256 colours to be displayed from a possible 256k colour combinations. A new colour value is specified by writing the location address (0-255) to the address register, followed by three byte writes to the colour value register. The three bytes correspond to the red, green, blue intensities required. The least significant 6 bits are taken from each byte giving 64 intensities of each of the primary colours.

Two colour tables are set up by the graphics software package.

Table 0

This is the default table. It has the colours partitioned as shown in the diagram. The bottom two bits of the colour address in the table indicates the green intensity level, the next two bits the red and the 5th and 6th bits the blue intensity. The top two bits are used to add an intensity bias to each of the three colours.

colour address



The three pairs of colour bits have the following meaning:

00	no colour
01	low intensity colour
10	medium intensity colour
11	high intensity colour



## 4.10 IMS G170 colour look up table

The two top intensity bits indicate:

00 no bias  
 01 red bias  
 10 green bias  
 11 blue bias

Table 1

The second table provides primary colour scales and is set up as follows:

0 - 15 grey scale  
 16 - 31 red scale  
 32 - 47 green scale  
 48 - 63 blue scale  
 64 - 79 yellow scale  
 80 - 95 cyan scale  
 96 - 111 magenta scale

112 - 127 red to green with low intensity blue  
 128 - 143 green to blue with low intensity red  
 144 - 159 blue to red with low intensity green

160 - 175 red to green with medium intensity blue  
 176 - 191 green to blue with medium intensity red  
 192 - 207 blue to red with medium intensity green

208 - 223 red to green with full intensity blue  
 224 - 239 green to blue with full intensity red  
 240 - 255 blue to red with full intensity green

Re-selection of a colour table will set all the colour values back to their defaults and any previously set up values will be lost.

## Pixel mask register

The pixel mask register in the IMS G170 is bitwise ANDed with the address register to give a masked address. This masking can be used to alter displayed colours without altering the the video memory or the look-up table contents.

## 4.11 CRT commands

## 4.11 CRT commands

Initialising the board for a particular video monitor is done using the CRT commands provided. These allow the scan line frequency, the frame rate, the pixel clock value and the interlace to be altered from the software.

## 4.12 Stopping the graphics software

To stop the program running on the graphics board simply issue the terminate command.

Below is a list of the graphics primitives provided in software with the B007 board, with the required occam to access each.

The primitives are supplied as a pre-compiled SC. For those using untyped occam the source is given for information only, you will not be able to alter the graphics primitives as they have been written in type occam. The default CRT settings may be altered in the outermost level of the **PROGRAM** graphics.

The primitives have been provided within a test harness environment, with a few example calls to the graphics package. This should be altered for your own use.

### 5.1 Graphics primitives description and access

General format of the communication with the graphics package is:

```
to.graphics ! command; <parameter list>
from.graphics ? reply
```

This applies to all the commands listed below.

### 5.2 Graphics commands

#### draw line

The graphics commands take the currently selected foreground colour as the drawing colour. All lines are clipped to the screen/window.

A straight line is drawn specified by the two end coordinates given.

```
c.draw.line; x1; y1; x2; y2
```

e.g to draw line from top left to bottom right of monitor screen

SEQ

```
to.graphics ! c.draw.line; 0; 0; 511; 511
from.graphics ? reply
```

#### plot point

Plots a single point at the x,y position specified.

```
c.plot.point; x; y
```

### 5.2 Graphics commands

#### draw circle

Draws a circle determined by the centre coordinate and radius dimension.

```
c.draw.circle; x.centre; y.centre; radius
```

#### draw arc

Draws an arc of a circle between the three points specified.

```
c.draw.arc; x1; y1; x2; y2; x3; y3
```

#### draw rectangle

Draws a rectangle. The four parameters are: the x,y coordinate of the top left of the rectangle; the x dimension and the y dimension.

```
c.draw.rectangle; x, y, x.length, y.length
```

#### draw polygon

Draws a polygon with a specified number of sides.

```
c.draw.polygon; number.sides; x.i; y.i ... x.n; y.n
```

e.g in typed occam to draw a 5 sided polygon

```
VAL polygon IS [[100, 100] [0, 200],
                [50, 300], [250, 300], [150, 200]:
```

SEQ

```
to.graphics ! c.draw.polygon; SIZE polygon
SEQ i = 0 FOR SIZE polygon
to.graphics ! polygon[i][0]; polygon[i][1]
from.graphics ? reply
```

#### fill polygon

Any arbitrary polygon (bounded region) is filled with the currently selected foreground colour. A quick fill is provided, which may be used to fill simple convex shapes. The polygon is selected by specifying a point lying inside it.

```
c.fill.polygon; x; y
```

```
c.quick.fill; x; y
```

## 5.3 Text commands

## 5.3 Text commands

A character font of 256 text and graphic characters is provided, using a 8x8 matrix. See character set definition at the end of this manual.

## draw char

Draws the character required at the current cursor position. The cursor position is then incremented by the width of the character.

```
c.draw.char; char.number
```

## write string

Writes out a text string from the current cursor position. The string is wrapped around the edge of the screen/window. Two versions of write string are provided compatible with both typed and untyped occam.

```
c.write.string; SIZE string; "string"
```

```
c.untyped.string; string[BYTE 0]; string[BYTE 1]
                string[BYTE 2], .. string[BYTE n]
```

e.g in typed occam

```
VAL text IS "abcdefg" :
SEQ
  to.graphics ! c.write.string;
                SIZE text; text
  from.graphics ? reply
```

and in untyped occam

```
DEF text = "abcdefg" :
SEQ
  to.graphics ! c.untyped.string;
                text[BYTE 0]
  SEQ i = [1 FOR text[BYTE 0]]
  to.graphics ! text[BYTE i]
  from.graphics ? reply
```

## 5.3 Text commands

## write number

Displays a given number as a text string.

```
c.write.number; number
```

e.g to display the number 42 on the monitor screen

```
SEQ
  to.graphics ! c.write.number; 42
  from.graphics ? reply
```

## define char

Allows new characters to entered in the font table. A character is defined top to bottom in 8 byte values.

```
c.define.char; char.number; <8 byte values>
```

e.g to define the letter 'T' in typed occam

```
VAL letter.T IS [#7E, #7E, #18, #18,
                #18, #18, #3C, #00] :
```

```
SEQ
  to.graphics ! c.define.char
  SEQ i = 0 FOR SIZE letter.T
  to.graphics ! letter.T[i]
  from.graphics ? reply
```

The above `letter.t` definition is related to the drawn character as shown below.

#7E		■	■	■	■	■	
#7E		■	■	■	■	■	
#18			■	■			
#18			■	■			
#18			■	■			
#18			■	■			
#3C		■	■	■	■		
#00							

**text drawing mode** The default text drawing mode provides a simple foreground and background replacement on the screen. Any other mode may be selected with the following command. See section on drawing modes.

**c.set.draw.mode; mode**

**rotate** Rotates all subsequent characters written through 0, 1, 2 or 3 quarter turns in a clockwise direction.

**c.rotate; turn**

e.g to rotate a characters through 180 degrees

```
SEQ
to.graphics ! c.rotate; 2
from.graphics ? reply
to.graphics ! c.draw.char; INT 'a'
from.graphics ? reply
```

Rotation is cancelled by **c.rotate; 0**. Rotations are always taken from the default orientation. Hence with, **c.rotate; 2** **c.rotate; 2** the second rotation has no effect.

**reflection** Textual output may be reflected in either the x or y plane or both. A reflection can be cancelled by issuing a second reflect command.

**c.reflect.x**

**c.reflect.y**

**move** Sets the cursor to absolute pixel position given. Text will now be written from this position.

**c.move; x; y**

**relative move** Moves the cursor pixel position relative to its present position.

**c.move.rel; delta.x; delta.y**

**carriage return** This returns the cursor to the left edge of the currently selected screen or window.

**c.carriage.return**

**line feed** This increases the cursor y pixel position by the height of the characters being drawn. A screen or window is automatically jump scrolled if a line feed command is issued at the bottom of the screen/window.

**c.line.feed**

**5.4 Screen and window commands** Two screens and up to 32 windows are available. Screens/windows may be selected for drawing and/or display. This allows the option of drawing to take place on a screen or window that is not currently being displayed.

**clear screen** Clears the currently selected screen to a specified colour.

**c.clear.screen; colour.value**

**select screen** Select one of the two available screens for drawing. This screen will not be displayed until the display screen command is sent.

**c.select.screen; screen.number**

**display screen** Select one of the two available screens for display.

**c.display.screen; screen.number**

**flip screen** Swops the currently displayed and non-displayed screens over. The previously displayed screen becomes hidden, but is now selected for drawing. The number of this selected screen is returned.

**c.flip.screen**

## 5.4 Screen and window commands

e.g to flip screens

SEQ

```
to.graphics ! c.flip.screen
from.graphics ? selected.screen
```

**copy screen**

Copies the contents of one screen onto the other. The screen number given specifies which screen is to be copied.

**c.copy.screen; screen.number**

**set window**

Set up a window specified by its width and length (x and y dimensions). The window is not displayed.

**c.set.window; x.dimension; y.dimension**

The graphics system returns the number of the next free window. This number is used to select the window when required. See section on windows and screens. e.g to define a window of size 200 pixels by 300 pixels

SEQ

```
to.graphics ! c.set.window; 200; 300
from.graphics ? window.number
```

**clear window**

Clears a the currently selected window to a specified colour

**c.clear.window; colour.value**

**select window**

Selects a currently set window for drawing.

**c.select.window; window.number**

**display window**

Selects a currently set window for display at specified position on the screen.

**c.display.window; window.number; x; y**

## 5.5 Colour commands

**scroll**

The currently selected window or screen is scrolled. The currently set value of y height multiplied by eight determines the number of lines scrolled. Two scrolls are provided a smooth scroll, which moves the display up a line at a time, and a jump scroll. Scrolling is automatic after a line feed command at the bottom of a screen or window.

**c.scroll**

**c.jump.scroll**

## 5.5 Colour commands

**set colour**

The IMS G170 makes 262,144 colours available, 256 of which make be set up in the colour table at any one time.

Writes a new colour value into colour look up table. A palette of colours is provided, but any of the 256 colours may be altered simply by entering a new valued into the table. See section on IMS G170 colour look up table.

**c.set.colour; colour.value; red; green; blue**

e.g to set the colour full green into the colour table at entry 26

SEQ

```
to.graphics ! c.set.colour; 26; 0; 63; 0
from.graphics ? reply
```

Selecting colour 26 will now cause all graphics and text to be drawn in green.

**select foreground colour**

Selects one colour from the colour look up table for drawing and text. e.g to select the colour green just set up

SEQ

```
to.graphics ! c.select.fg.colour; 26
from.graphics ? reply
```

**c.select.fg.colour; colour.value**

**select background colour** Select one colour for text background.  
**c.select.bg.colour; colour.value**

**get pixel colour** This command returns the colour value of a pixel on the screen.  
**c.get.pixel.colour; x; y**

**select colour table** Select one of the colour tables provided. See section on colour look up table.  
**c.select.colour.table; table.number**

**set pixel mask register** Sets the pixel mask register on the IMS G170 to the required value. See section on colour look up table.  
**c.set.mask.reg; mask**

**5.6 General commands** The first two commands in this section affect both the graphics line drawing and the text commands.

**set x width** Sets the smallest drawing pixel width (default 1; max 512).  
**c.set.xwidth; width**

**set y height** Sets the smallest drawing pixel height (default 1; max 512).  
**c.set.yheight; height**

**termination** To close down the graphics package a termination command is sent.  
**c.terminate**

**5.7 CRT commands** The following are used to set up the graphics board for a particular video monitor.

**line frequency** Set the scan line frequency for CRT.  
**c.line.frequency; line.frequency**

**frame rate** Set the frame rate for CRT.  
**c.frame.rate; frame.rate**

**Interlace** Toggle interlace ON/OFF.  
**c.interlace**

**pixelclock rate** Set pixel clock frequency.  
**c.pixel.clock; clock.frequency**

**init CRT** The CRT controller is only reset on receipt of an init CRT command.  
**c.init.crt**

5.8	<b>Error codes</b>	If the requested action was completed successfully by the graphics software then a success code is returned, otherwise, various error codes are returned. Below is a full list of codes.
	<b>e.ok</b>	successful completion
	<b>e.out.of.drawing.range</b>	coordinates out of screen/window range
	<b>e.invalid.screen</b>	screen other than 0 or 1 selected
	<b>e.invalid.window</b>	window selected has not been set
	<b>e.no.window.store</b>	insufficient storage left for window definition
	<b>e.too.many.windows</b>	the maximum number of windows has been exceeded
	<b>e.unknown.drawing.mode</b>	drawing mode selected not in the range 0 - 7
	<b>e.invalid.rotation</b>	rotations allowed only in quarter turns, i.e (0 - 3)
	<b>e.invalid.colour</b>	colour selected out of range of colour look up table (0 - 255)
	<b>e.string.length.exceeded</b>	only 255 characters allowed in strings
	<b>e.char.out.of.range</b>	character requested was not in the range of character font table (0 - 255)
	<b>e.invalid.colour.table</b>	colour table requested does not exist
	<b>e.unknown.command</b>	command sent to graphics package does not exist

## 6.1 Commands

The following list of definitions is given in typed occam syntax. The equivalent for untyped occam is

```

DEF c.command = 1 :

VAL c.plot.point      IS 1 :
VAL c.draw.line       IS 2 :
VAL c.draw.circle     IS 3 :
VAL c.draw.arc        IS 4 :
VAL c.draw.rectangle  IS 5 :
VAL c.draw.polygon    IS 6 :
VAL c.fill.polygon    IS 7 :
VAL c.move            IS 8 :
VAL c.move.rel        IS 9 :

VAL c.clear.screen    IS 10 :
VAL c.select.screen   IS 11 :
VAL c.display.screen  IS 12 :
VAL c.flip.screen     IS 13 :
VAL c.copy.screen     IS 14 :
VAL c.clear.window    IS 15 :
VAL c.select.window   IS 16 :
VAL c.display.window  IS 17 :
VAL c.set.window      IS 18 :
VAL c.set.draw.mode   IS 19 :

VAL c.draw.char       IS 20 :
VAL c.define.char     IS 21 :
VAL c.write.string    IS 22 :
VAL c.untyped.string  IS 23 :
VAL c.write.number    IS 24 :
VAL c.scroll          IS 25 :
VAL c.jump.scroll     IS 26 :
VAL c.rotate          IS 27 :
VAL c.reflect.x       IS 28 :
VAL c.reflect.y       IS 29 :

```

## 6.2 Error codes

```

VAL c.line.feed          IS 30 :
VAL c.carriage.return    IS 31 :
VAL c.set.colour         IS 32 :
VAL c.select.fg.colour   IS 33 :
VAL c.select.bg.colour   IS 34 :
VAL c.select.colour.table IS 35 :
VAL c.mask.reg          IS 36 :
VAL c.set.xwidth         IS 37 :
VAL c.set.yheight       IS 38 :
VAL c.get.pixel.colour   IS 39 :

```

```

VAL c.line.frequency     IS 40 :
VAL c.frame.rate         IS 41 :
VAL c.interlace          IS 42 :
VAL c.pixel.clock        IS 43 :
VAL c.init.crt           IS 44 :
VAL c.quick.fill        IS 45 :

```

```

VAL c.terminate         IS 255 :

```

## 6.2 Error codes

```

VAL e.ok                IS 0 :
VAL e.out.of.drawing.range IS -1 :
VAL e.invalid.screen     IS -2 :
VAL e.invalid.window     IS -3 :
VAL e.no.window.store    IS -4 :
VAL e.too.many.windows   IS -5 :
VAL e.unknown.drawing.mode IS -6 :
VAL e.invalid.rotation   IS -7 :
VAL e.invalid.colour     IS -8 :
VAL e.string.length.exceeded IS -9 :
VAL e.char.out.of.range  IS -10 :
VAL e.invalid.colour.table IS -11 :
VAL e.unknown.command    IS -12 :

```

The character set provided with the graphics software is :

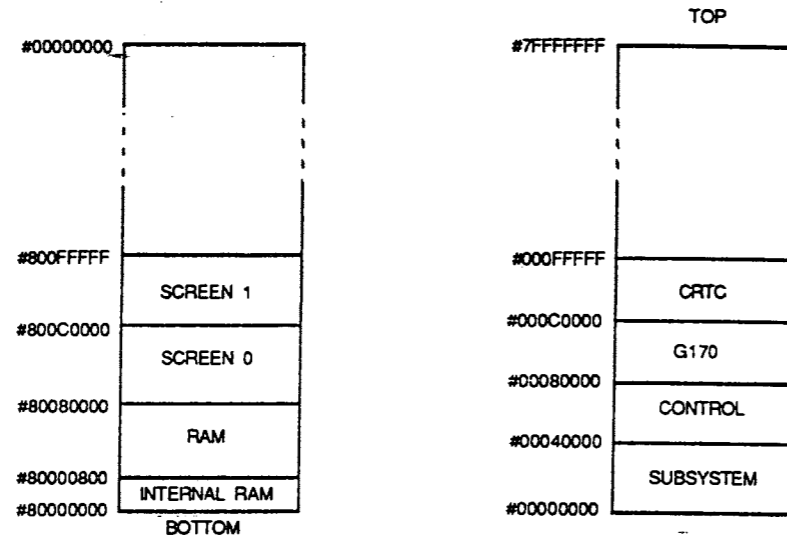
```

0 - 31 graphics
   32 space
   33 !
   34 "
   35 #
   36 $
   37 %
   38 &
   39 '
   40 (
   41 )
   42 *
   43 +
   44 ,
   45 -
   46 .
   47 /
48 - 57 0-9
58 - 90 A-Z
   91 [
   92 \
   93 ]
   94 ^
   95 _
96 - 122 a-z
   123 {
   124 |
   125 }
   126 ~
127 - 255 graphics

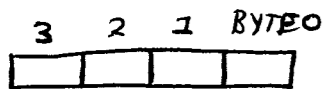
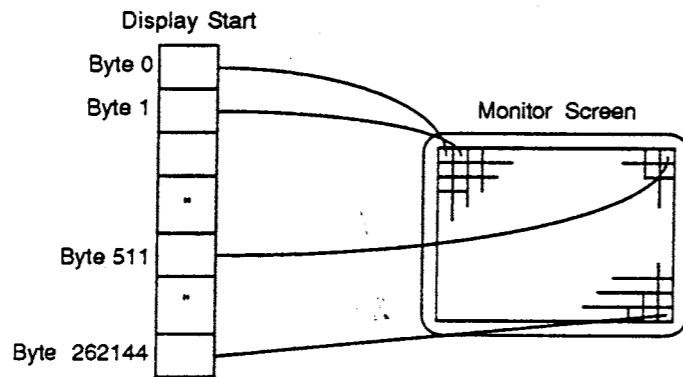
```



The memory map of the B007 is as shown. The control registers and peripherals are shown mapped as byte addresses but are word aligned.



Each display is mapped as shown:



#800C0060

Subsystem

The Subsystem control is mapped as shown. Each control bit is active high.

WRITE

BIT Position	7	6	5	4	3	2	1	0
WORD 1	—	—	—	—	—	—	—	SAna
WORD 0	—	—	—	—	—	—	—	SRes

READ

BIT Position	7	6	5	4	3	2	1	0
WORD 1	—	—	—	—	—	—	—	SErr
WORD 0	—	—	—	—	—	—	—	—

SAna Controls Subsystem Analyse.

SRes Controls Subsystem Reset.

SErr This shows status of Subsystem Error.

Video control latch is addressed as shown.

Video Control Latch

WRITE

BIT Position	7	6	5	4	3	2	1	0
WORD 2	—	—	—	—	—	—	—	VRes
WORD 1	—	—	—	—	—	—	—	—
WORD 0	—	—	—	—	—	—	—	Scr

VRes Resets the video port.

Scr Controls which screen is being displayed. (0 = screen0)

Colour Look Up Table

The CLUT is addressed as shown. Refer to data sheet for register description.

WRITE

BIT Position	7	6	5	4	3	2	1	0
WORD 2	PIXEL MASK REG.							
WORD 1	COLOUR VALUE REG.							
WORD 0	ADDRESS REG.							

CRTC

The video controller is addressed as shown.

WRITE

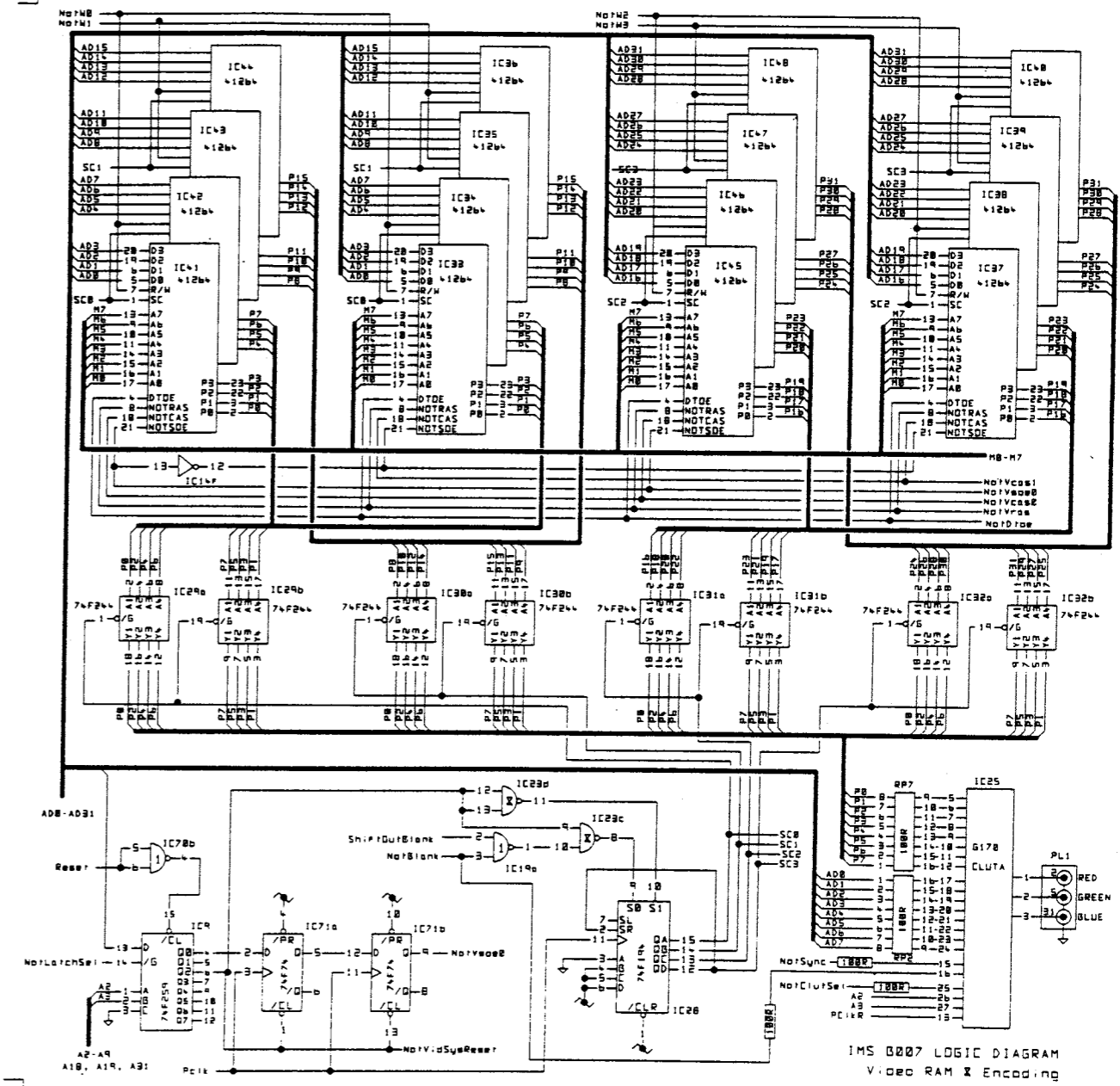
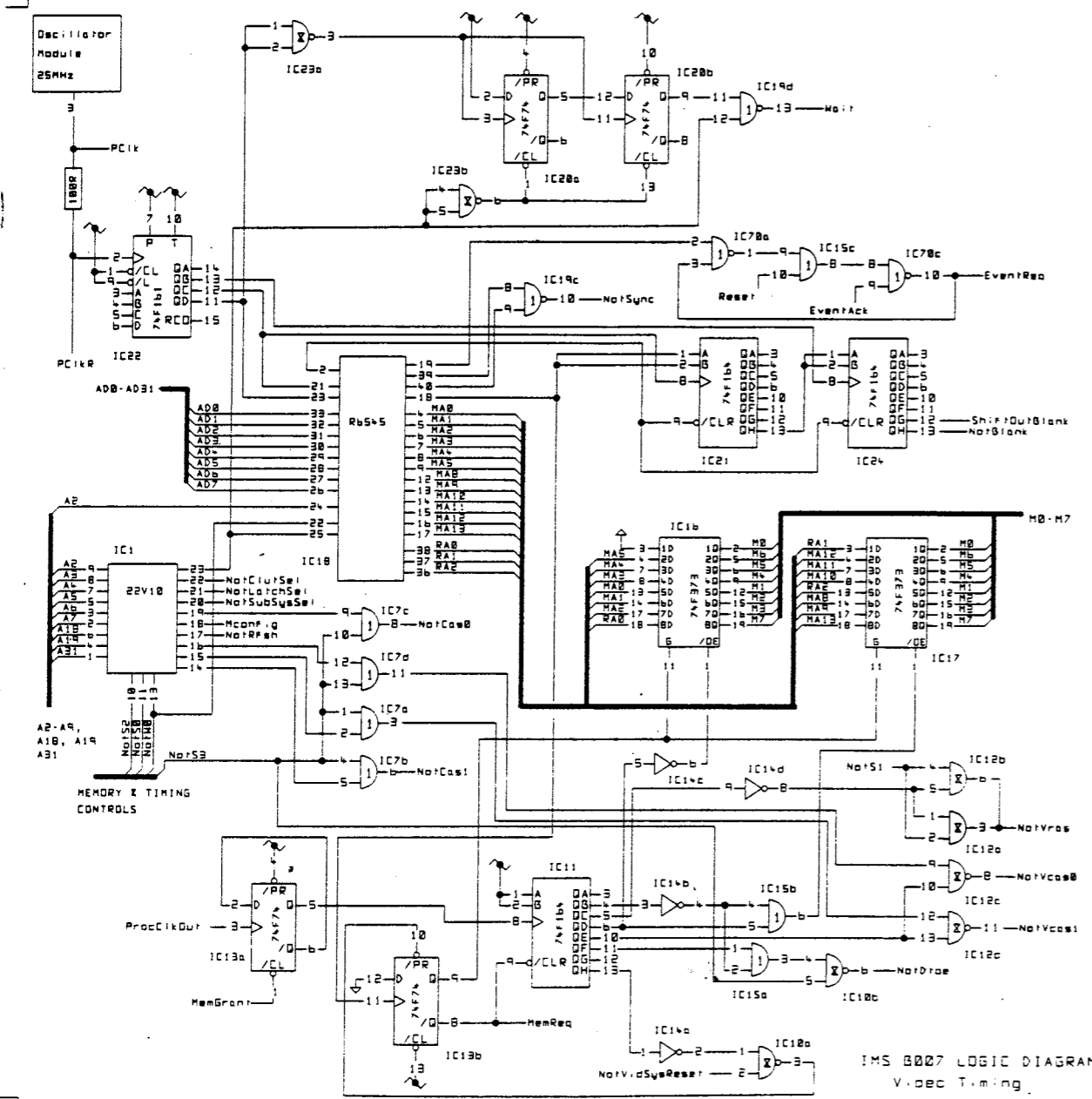
BIT Position	7	6	5	4	3	2	1	0
WORD 1	DATA Reg.							
WORD 0	ADDRESS Reg.							

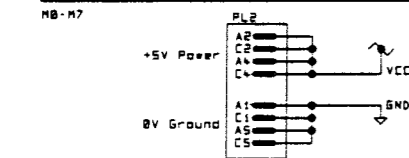
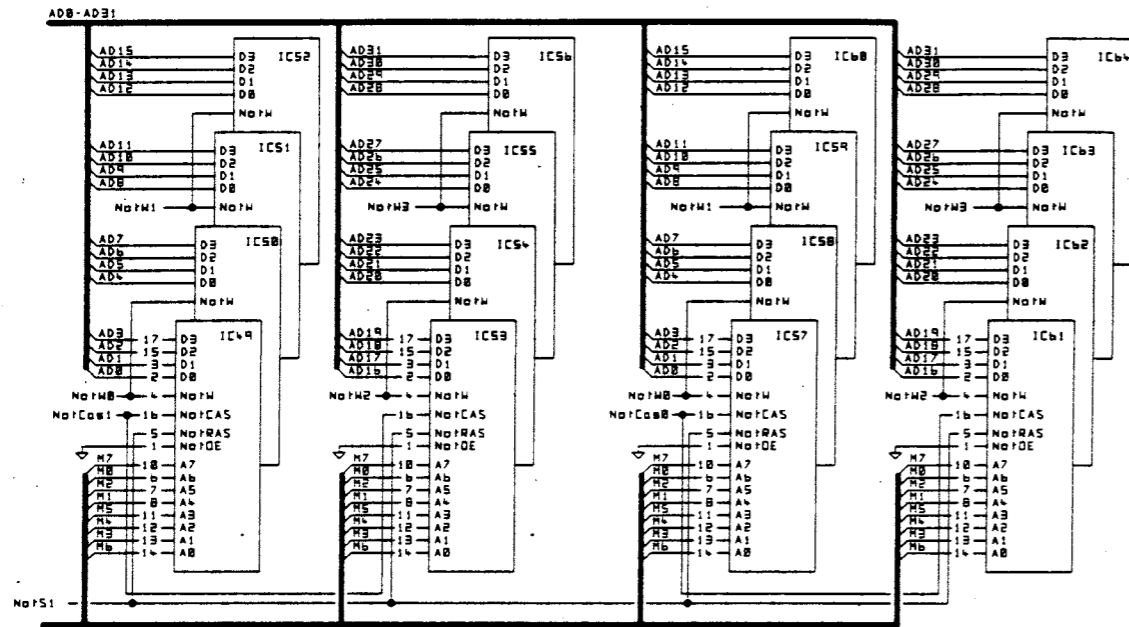
READ

BIT Position	7	6	5	4	3	2	1	0
WORD 0	—	—	VRT	—	—	—	—	—

It is not recommended to program the CRTC as a detailed knowledge of the hardware is necessary. A seperately compiled procedure is supplied to do this for the programmer. Synchronising with frame flyback can be acheived by polling VRT (vertical retrace bit). This bit is active high.







Type	IC Ref	VCC	GND	Type	IC Ref	VCC	GND	Type	IC Ref	VCC	GND
74F08	10 23	14	7	74F132	27	14	7	74264	33 10 48	12	24
74F02	19 69 78	14	7	74F164	11 21 22 24	14	7	74264	49 10 64	9	18
74F04	14	14	7	74F194	28	14	8	Rb545	18	28	1
74F08	12	14	7	74F244	4 29 30 31 32 37	28	18	G178	25	28	14
74F32	7 15	14	7	74F259	9	14	8	74148	8	C2 C5 E18 H4	A9 C1 G2 G9 J7
74F74	13 28 26 71	14	7	74F373	2 3 5 16 17	28	18				
74F86	6	14	7	PAL22V18	1 68	24	12				

IMS 0007 LOGIC DIAGRAM  
RAM Memory Plane