Copyright

This is a draft release for internal use only. Please advise in writing of any errors:

Atari Transputer Project
Atari Corp (UK) Ltd
Railway Terrace
Slough
Berks
SL2 5BZ
United Kingdom

Written by:

Richard Miller
Jack Lang
Hugo Davenport
Les Player

This manual refers to the Issue 3 ABAQ to be used with an external Mega ST

Printed in the UK

Change control

This document is liable to change without notice

Issue Date	Page Update	Date	Who

Contents

Preface

This manual is a technical reference guide to the Atari ABAQ issue 3, a Transputer based processing and video system for use with tha Atari Mega ST. It does not contain any reference material concerning the Mega ST.

The manual consists of 7 sections and 6 appendices. The appendices contain detailed hardware information which will not, in general, be neccessary for software engineers, who should find all they need to know in the main body of the text.

No detail of the operation of the Transputer and other Inmos chips is included and the user is referred to the relevant Inmos publications.

Definition of terms

In the rest of this manual the following terms are used:

Word	In our case, a word is 32 bits or 4 bytes.
Address	All addresses are byte addresses unless denoted otherwise. The address space is 32 bits and signed (in the same way as the transputer).
AD(31:0)	Buses are denoted by a range in brackets e.g. XX(msb:lsb)
BC_REG	All registers are described by two letters XX_REG
BC_REG(VEN)	A bit or range of bits within a register follows the register name and is placed in brackets. The syntax is the same for single bits and buses within registers.

1 Overview

ABAQ is the first product in a range of Transputer based machines. It is designed as a stand–alone personal workstation, but it may also be networked to form a larger system. A unique feature is the ability to plug in expansion cards containing parallel processors, to give personal systems of unprecedented power.

The issue 3 ABAQ referred to in this manual is designed to be used in conjunction with an external Mega ST.

Subsequent versions of the ABAQ will contain a 68000 based i/o processor, software compatible with an Atari ST.

An outline block diagram is given in figure 1.1 and shows the main subsystems.

The main processor, the Inmos Transputer, interfaces to the rest of the ABAQ via the main memory bus and a fast serial link to the external Mega ST. Three other fast serial links allow interfacing with add–on transputer cards or other ABAQs. The remaining I/O is handled by the external Mega ST which has its own separate memory bus.

The serial interface at the Mega ST is incorporated on an addon printed circuit board which contains the link interface, a DMA interface to the Mega ST and a SCSI interface to allow the use of an external SCSI Winchester or other SCSI peripherals. This interface card is provided with the Issue 3 ABAQ.

The system block diagram is shown in fig1.1 Expansion within the ABAQ is possible by plugging a vertical expansion backplane into the main pcb; this has 4 slots for expander cards.

The key to the ABAQ's graphics superb performance is the custom blitter chip, known as Charity. This performs all the memory control for main, expansion and video memory as well as performing 2D block operations (bitblt operations) on blocks of memory and controlling the video output. The video subsystem complements the blitter by providing three sets of Digital to Analogue Converters (DACs) to produce colour video output signals for a variety of spatial and colour resolutions and also Colour Look–Up Tables (CLUTs) for colour paletting.

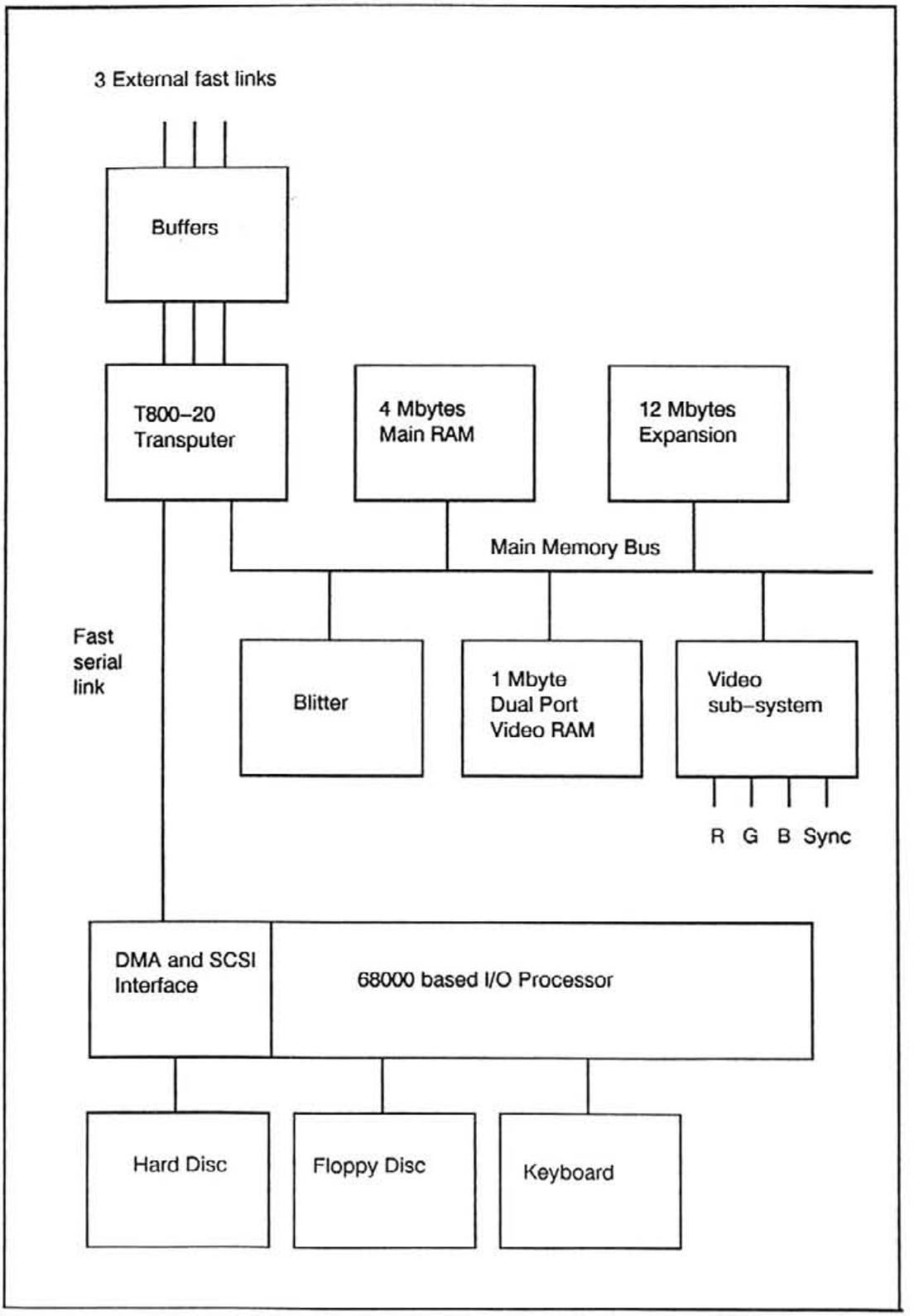


Fig 1.1 System Block Diagram

2 Physical layout and set–up

The ABAQ consists of a case in which is mounted the main printed circuit board **(the mother board)** and the mains power supply.

These are supplied in assembled form and with the power supply lead connected.

Other items are packed separately:

Video printed circuit board **(daughter board)**
SCSI interface card for Mega ST
Expansion Backplane
Video output lead
Link interface lead to Mega ST
Mains power lead
This manual
Floppy disc containing boot software

A drawing of the fully assembled unit with case cover removed is shown in figure 2.1. and a drawing of the mother board showing the connector locations is given in figure 2.2

To complete the assembly of the ABAQ the following steps should be taken:

1 Remove case cover by removing the four M3 bolts and removing the top cover (See fig 2.1 for case details)

2 Check the mains input voltage selector on the power supply to ensure that it is set for the correct voltage.

3 Remove the daughter board from its anti–static wrapping and plug it onto connectors J1 and J3 with component side uppermost. See fig 2.2

4 Connect the video lead coaxial connectors to the appropriate set of coaxial connectors on the daughter board. The choice will depend on the monitor used and the video mode to be selected. See fig 2.2 and sec 6.2

5 Connect link interface lead

6 Replace case cover

7 Install SCSI interface in Mega ST (Refer to installation leaflet supplied)

8 Connect link interface lead to Mega ST

9 Plug ABAQ mains lead into power supply

10 Plug ABAQ mains lead into mains outlet

11 Plug Mega ST mains led into supply

12 Switch on ABAQ mains outlet

13 Switch on Mega ST mains outlet

Note that it it neccessary to start the main memory refresh cycle using software. Refer to the leaflet "Getting started" or to your Software Manual for information on how to do this.

2.2 Inventory of connectors

The connectors in the ABAQ are listed below:

No.	Type	Function	See section
J1	96 way DIN 41612(f)	Daughter board conn.	Appendix4 sec 5
J2	96 way Din 41612	Not on issue 3 board	
J3	96 way DIN 41612(f)	Daughter board conn.	Appendix 4 sec 5
J4	9 way D–type	ECL link connector	Appendix 4 sec 2
J5	9 way D–type	ECL link connector	Appendix 4 sec 2
J6A	8 pin mini DIN	TTL link connector	Appendix 4 sec 2
J6B	4 pin mini DIN	TTL link connector	Appendix 4 sec 2
J6C	4 pin mini DIN	TTL link connector	Appendix 4 sec 2
J6D	4 pin mini DIN	TTL link connector	Appendix 4 sec 2
J7	Pin header	Link patch panel	Appendix 4 sec 2
J8	100 way pcb edge	Expansion backplane	Appendix 4 sec 1
J9	44 pin pcb edge	Not on issue 3 pcb	
J10	44 pin pcb edge	Not on issue 3 pcb	
J11	8 pin 0.15" header	Power input conn.	Appendix 4 sec 3
LK4	pin header	Memconfig link Not on issue 3	
LK5	pin header	Wait source jumper Not on issue 3	
LK6	pin header	CPU speed jumper	Appendix 4 sec 4
LK7	pin header	Link speed jumper	Appendix 4 sec 4
LK8	pin header	Cache card jumper Not on issue 3	
LJ9	pin header	CASA selector	Appendix 4 sec 6
LK10	pin header	CASB selector	Appendix 4 sec 6
LK11	pin header	CASC selector	Appendix 4 sec 6
LK12	pin header	CASD selector	Appendix 4 sec 6
SW1	pin header	Reset connector	

2.3 Power supply

The power supply is an internal fan–cooled 200 watt unit with the following inputs and outputs:

Input : 100/120 v.a.c | switch selected
220/240 v.a.c |

Outputs: + 5 v.d.c.15.00 A
+ 12v.d.c. 4.20 A
– 5 v.d.c. 0.30 A
– 12v.d.c. 0.25 A

2.4 Environmental and RFI

Environment: 41 to 113 F (5 to 45 C) operating or idle
–4 to 149 F (–20 to 65 C) storage
–40 to 149 F (–40 to 65 C) transport

Relative Humidity (non condensing):
20% to 80% (operating or idle)
95 % maximum (storage or transport)

Radio Frequency Interference: The ABAQ issue three has not been tested to and may not comply with national or international standards

Safety: The ABAQ iss 3 has not been tested for compliance with any national or international safety standards

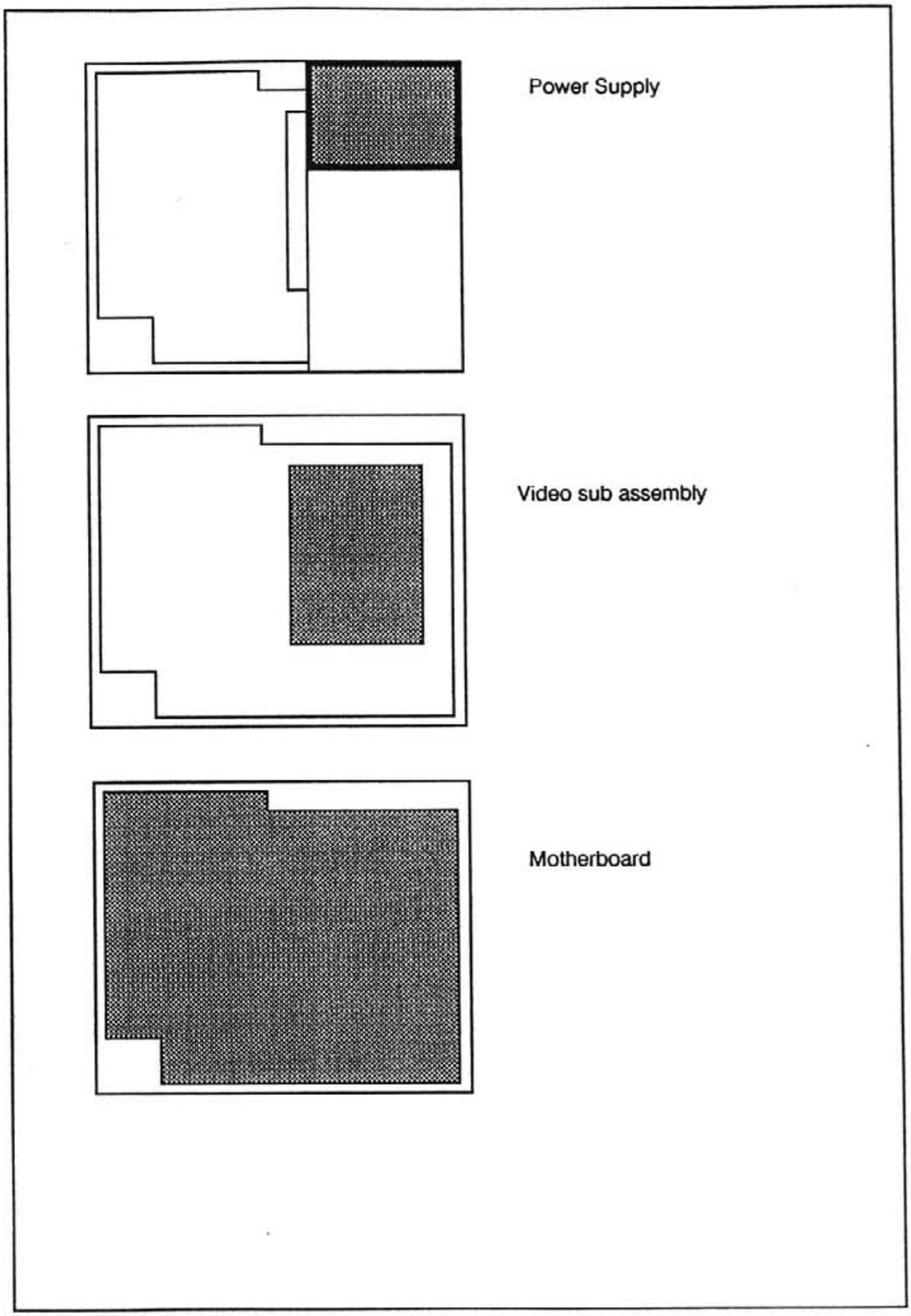


Fig 2.1 Layout of main subsystems

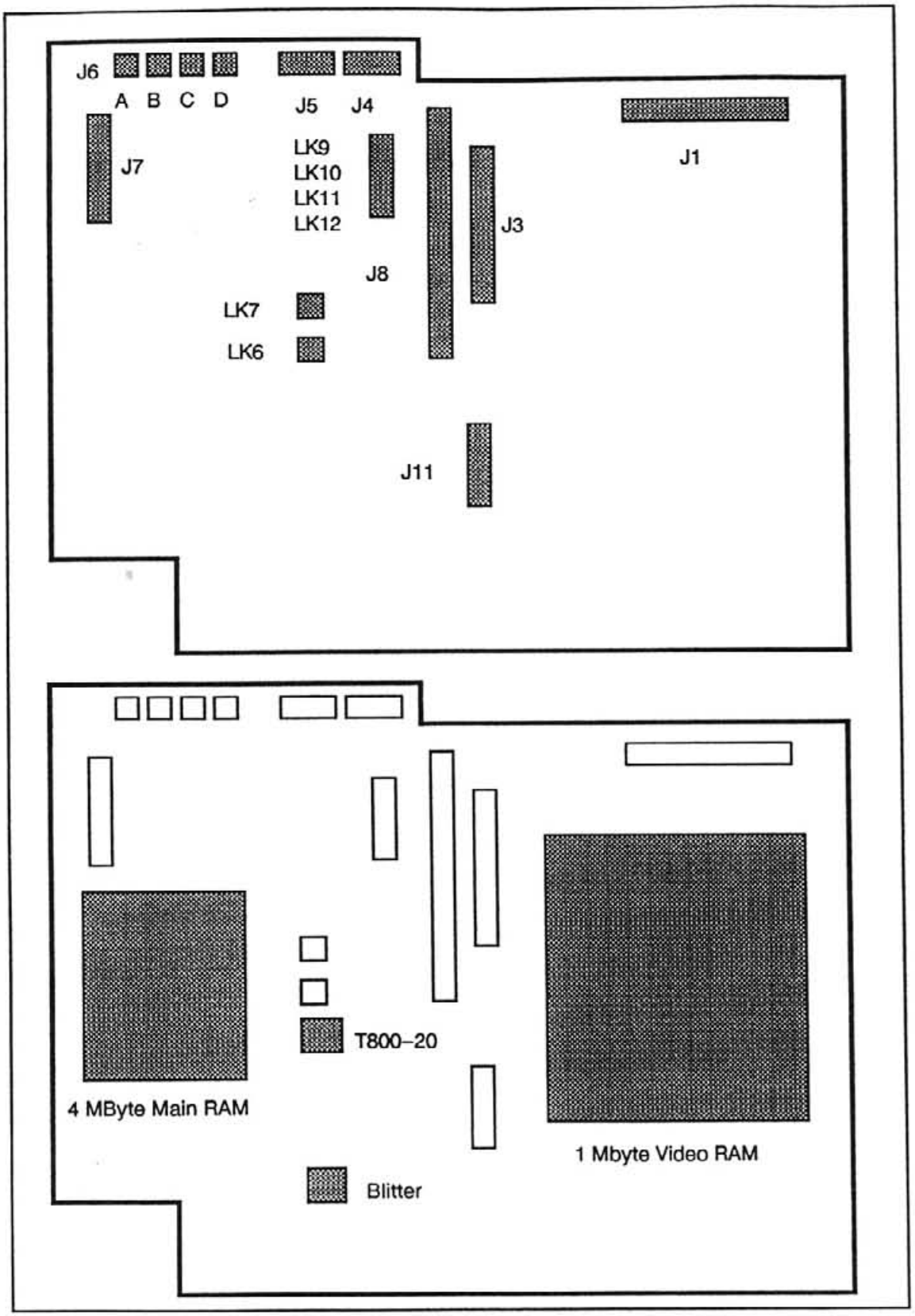


Fig 2.2 Motherboard PCB

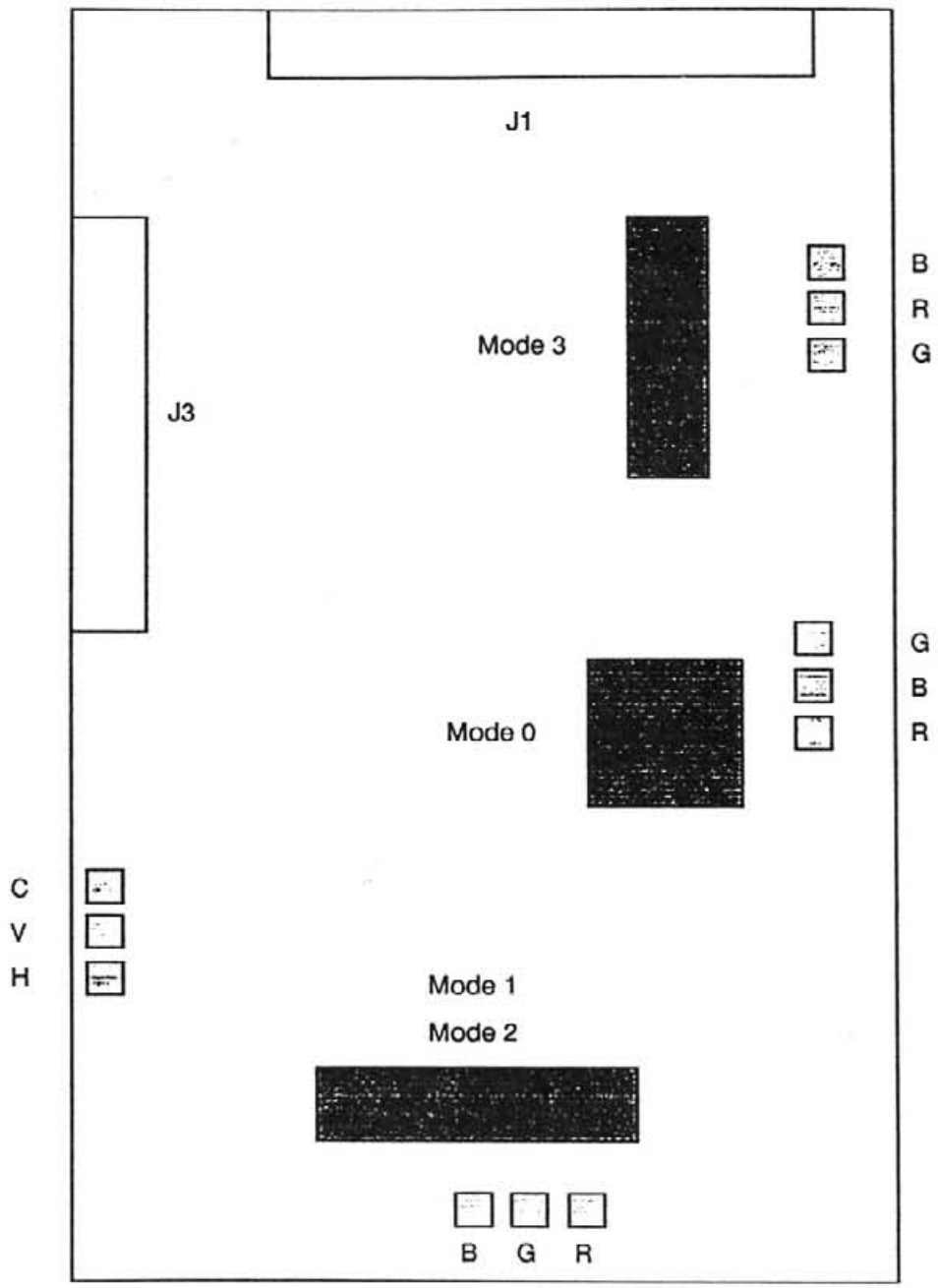


Fig 2.3 Video sub assembly

3 Architecture

3.1 Processor

The T800 processor supplied as standard has a number of unusual features from an architectural point of view; in particular it has four serial links with DMA interfaces which operate concurrently while the program is running. It is this feature which gives the Transputer its unique ability to be networked efficiently. It also incorporates 4k of fast on-board cache memory and a floating point a rithmetic unit. The T800 may be regarded as 6 separate processors working in parallel: 4 link engines, the floating paint unit and the integer CPU. As far as the processor is concerned everything it sees is memory mapped, including its own I/O registers, main memory, expansion memory, video memory and the blitter control registers.

3.2 Main memory bus and arbitration

The main bus is a multiplexed address/data bus 32 bits wide excluding control signals. Either blitter or processor can assume control of the bus for transferring data. All memory refresh is maintained by the blitter, which requests the bus when it needs to refresh main or video RAM. The processor must relinquish control within 2.5 uS of such a request. Once the blitter has control of the bus it will maintain control until the refresh is complete or until the bitblt operation is complete. In the latter case it will interrupt the bitblt operation to carry out any refresh required. Once the processor has enabled the blitter the blitter will immediately request the bus. The T800 may continue to execute program from its internal memory during blit operations.

3.3 Video memory

The video memory consists of 1Mbyte of dual-ported RAM. One port is connected to the main memory bus; the other is dedicated to video output, which is controlled by the blitter. An uninterrupted flow of digital video data is thus sent to the video subsystem while normal READ/WRITE cycles into or out of video memory are carried out on the main bus. The use of video memory means that the video refresh traffic does not use the main data-bus, and so there is no degredation of performance while supporting very high resolution displays.

3.4 Blitter

Although the Charity custom chip is known as the blitter it also contains the memory controller and the video control circuitry. Once the blitter is initialised the refresh and video control are transparent to the processor, although the video set-up can be altered at any time by the processor by writing to the appropriate video control registers. The blitter within main and video memory. It also carries out the transformation between linear and planar address spaces and copes with pixel alignment and end masking, thus freeing the processor from the arithmetic required for these.

3.5 Video subsystem

The video subsystem is connected to the remainder of the ABAQ by both the main bus and the video bus. Control over the video subsystem is exercised by writing to memory-mapped registers over the main bus but video data are output to the subsystem in a continuous stream via the video bus. The way in which this 32 bit wide video data is interpreted and routed to the three separate video output DAC systems is dependent on the video mode selected. The DACs associated with video modes 0, 1 and 2 have built-in CLUTs to allow paletting whereas in mode 3 the colour information is explicitly defined by the video data.

The outputs of each set of DACs are disabled when their video mode is not selected. The video daughter board also contains the Video Mode Register (VMODE_REG)

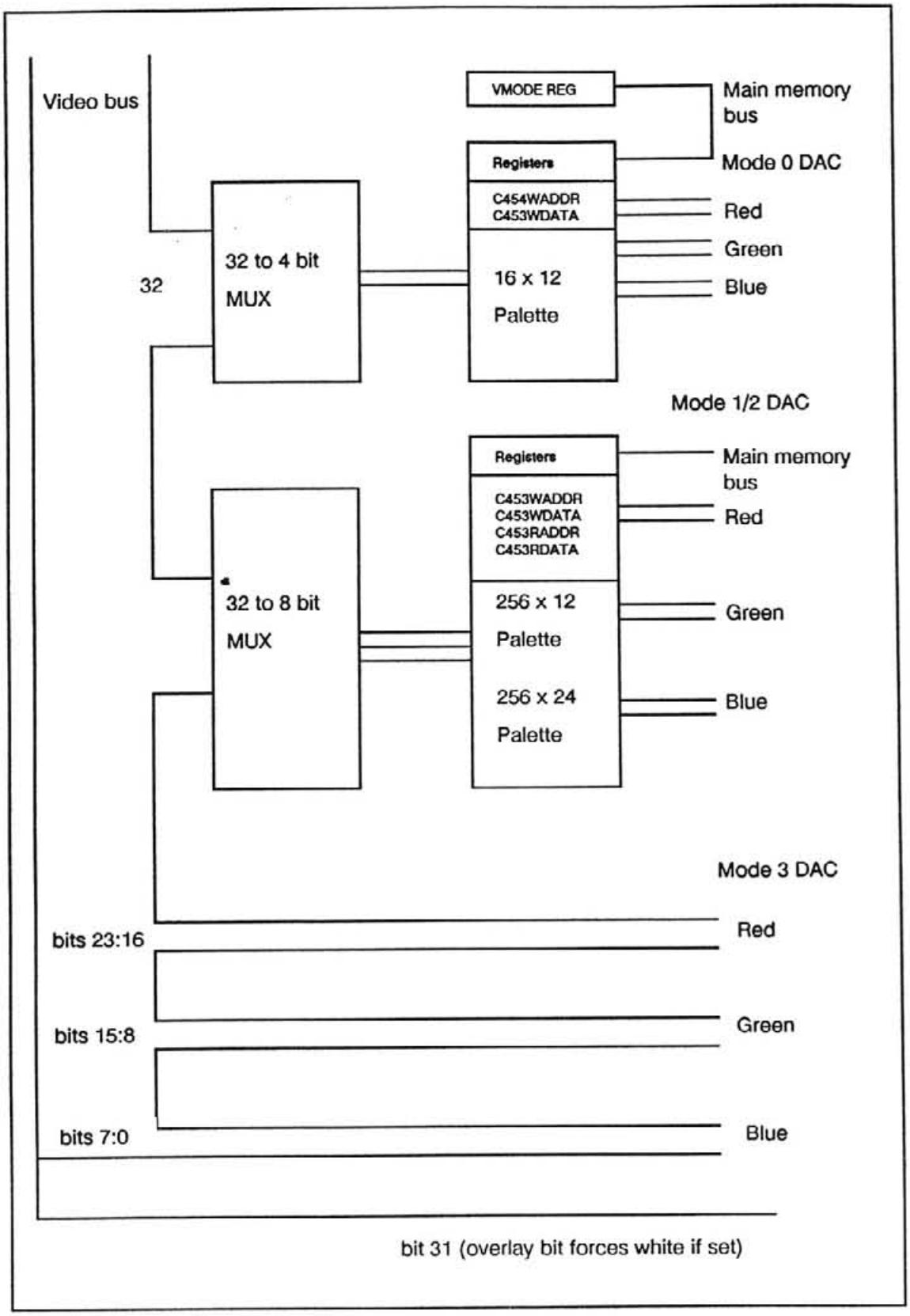


Fig 3.1 Video subsystem

4 Processor

One of the main advantages of the ABAQ is that its main processor is an advanced Reduced Instruction Set Computer (RISC) with many novel features. In particular the INMOS Transputer processor is designed to form part of a multi–processor system. This allows the ABAQ to be expanded by the addition of many extra processors by means of the serial links provided.

The main processor unit (MPU) is an INMOS T800 tranputer running at 20MHz. (17.5MHz is some versions) the T 800 is upwards pin compatible with the earlier T414 transputer which will run at the same speed. The full specification of both devices can be found in the appropriate INMOS data sheets. A summary of the T800 specification is included below:

- Full 32 bit transputer architecture
- 20 MHz processor clock, 10 MIPS
- Integral hardware 64 bit floating point unit
- ANSI–IEEE 754–1985 Floating Point Representation
- Sustained 1.5 Mega–FLOPS
- 4000K Whetstones / second (single length)
- 4 Kbytes on chip RAM for 80 Mbytes / second data rate
- 32 bit memory interface
- High performance graphics support
- Four INMOS bi–directional serial links (2350 Kbytes/second)
- Hardware scheduler
- Sub–microsecond context switch and interrupt latency
- Concurrent DMA while internal program running
- Boots from communication link
- 2 Internal timers for real–time processing

On–board link(s) enable a T414 to be plugged into the same socket and allow the software to recognise which type of processor it is running on.

Additional processor cards may be plugged into the expansion sockets.

5 Memory

5.1 Configuration and size

The Transputer can address up to 4 Gbytes of memory. The ABAQ has sockets for 32 1 Mbit x 1 DRAM parts, giving a maximum of 4 Mbytes of on-board memory. It is anticipated that these sockets will be pin-compatible with 4 Mbit parts when they become available (provided that the packaging is the same as for 1 Mbit parts). This will allow the option of upgrading to 16 Mbytes when the larger parts become available. Further memory expansion will be possible using memory expansion cards, giving a total memory size of 16/64 Mbytes depending on what size parts are used.

ABAQ supports three types of RAM parts:

4M x 1bit DRAM 11 Multiplexed address lines
1M x 1bit DRAM 10 Multiplexed address lines
256K x 1bit DRAM 9 Multiplexed address lines

Note that parts should have an access time of 120nS or better for a 20MHz processor.

User RAM is addressed with 11 address lines, some of which are redundant for the smaller RAM parts.

ABAQ has four CAS lines, called CSM0-3. Therefore, up to four banks of RAM can be addressed.

The size of each RAM bank can be programmed, and should be set by the boot software according to which type of parts are used.

The allowable widths are:

1 Mbyte	–	32 off,	256k x 1 parts
4 Mbytes	–	32 off,	1M x 1 parts
16 Mbytes	–	32 off,	4M x 1 parts

For further details of memory expansion and timing requirements see Appendix 6

5.2 Memory map

The address space of the T800 (and T414) is signed and byte addressed. Words are aligned on four-byte boundaries. Addresses in the range 80000000 to 80000FFF (4K bytes) reference on-chip memory for the T800. The T414 has only 2K of on-chip RAM. The memory map is shown in figure 5.1 below.

Note that the first 18 words of the address space are used for system purposes.

5.3 User RAM addressing

The diagram Fig 5.2 shows how 1Mbit parts are addressed. The map refers to a T800 transputer, which has 4K of on chip RAM. This internal RAM overlays the first bank of user RAM. The bottom 4K of Bank 0 user RAM can however be accessed at the top of the map in each case, due to address aliasing. If the blitter is pointed to addresses 80000000 – 80001000, it will address user RAM, since it cannot access the T800's internal memory.

5.4 Video RAM

In addition to user RAM, 1 Mbyte of dual port video RAM is used in order to achieve the high video bandwidth.

Video RAM is arranged as 32 64K x 4 chips. The chips have common address and RAS lines, but individual CAS lines. Byte wide write cycles are possible in the same way as for the User RAM.

See section 6 below, and appendix 6, for more information on video RAM.

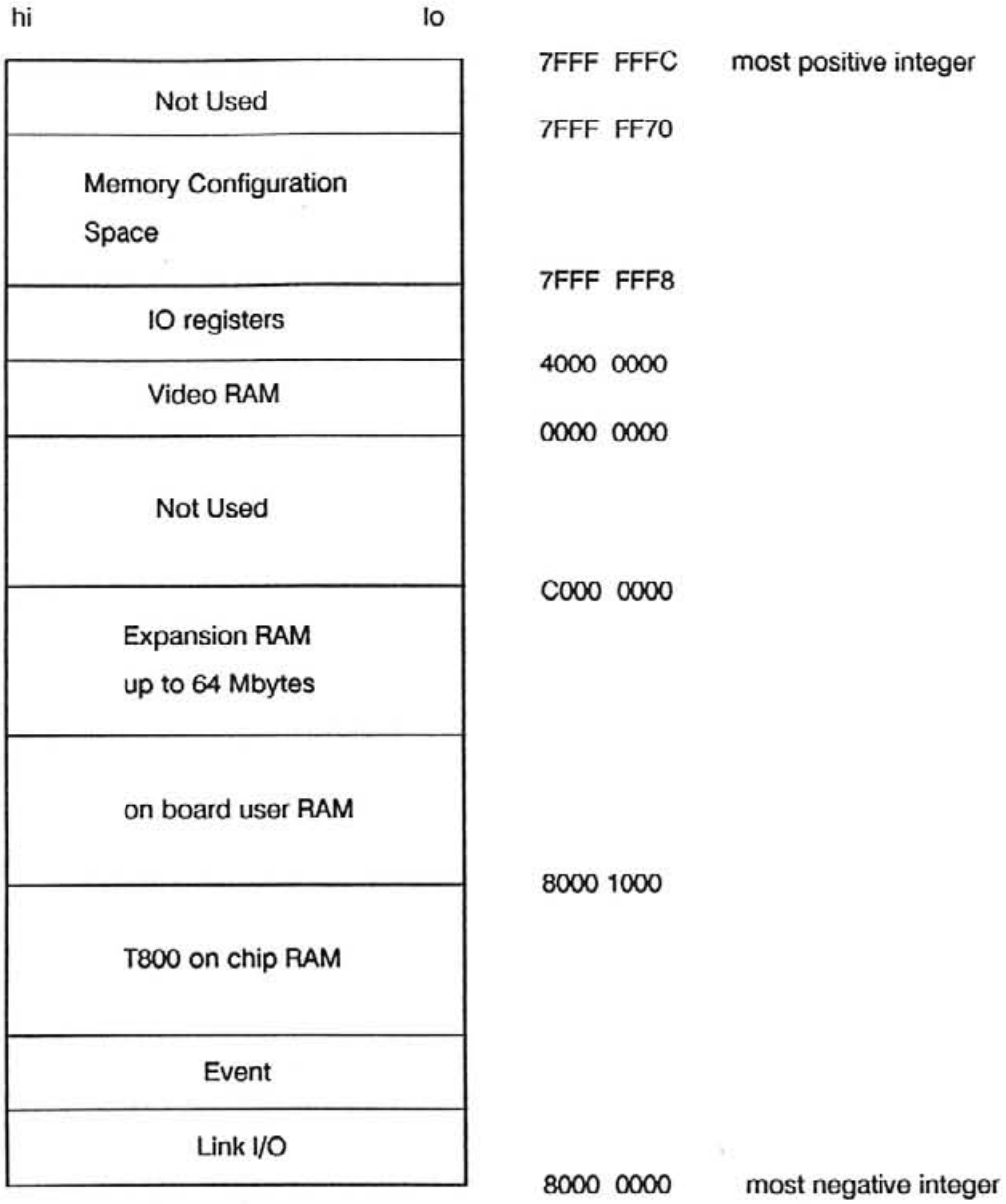
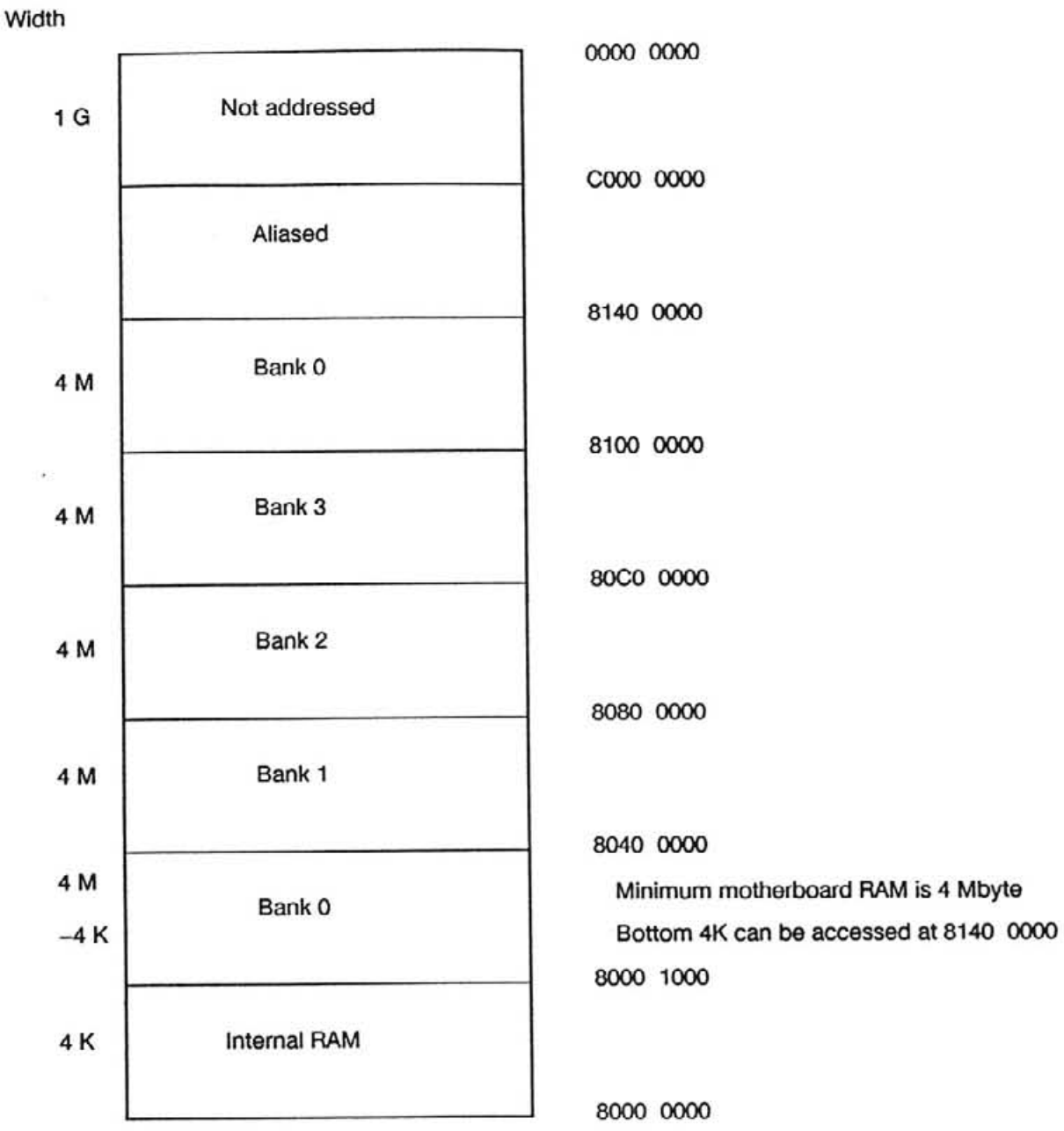


Fig 5.1 Overall address map



User RAM map for 1M x 1 parts

Fig 5.2 User RAM map

5.5. External memory interface

Charity provides all the control signals necessary to interface with up to 64 Megabytes of DRAM and 1 Megabyte of video RAM. It also provides two I/O control signals which can be programmed with different timings according to the precise requirements of the peripheral concerned.

5.5.1 Transputer cycles

Although Charity I has been designed specifically for use with the Inmos Transputer, the simplicity of the memory interface allows it to be used with other processors as well. This has been demonstrated on a VME video card.

The 32 bit multiplexed address/data bus is sampled for an address on the rising edge of MS0. The following rising edge of MCLK with MS0 low will indicate the start of a memory cycle, by RAS going low. Each memory cycle is assumed to be 250 ns long.

5.5.2 Refresh

Charity takes care of all refresh by providing 1024 row addresses every 16 milliseconds. This 10 line refresh should support the new 4 Mbit DRAMs when they become available.

Pending refresh cycles will be stacked up, or remembered by Charity until either the count reaches 15, or a bus grab has taken place (by either video or the blitter). If the count reaches 15, then Charity will grab the bus and perform all 15 pending cycles in one burst of 3.75 us. If the bus has been grabbed for any other purpose and any refresh cycles are due, then they will be slotted in at a low priority.

This system helps reduce the overhead from bus arbitration. For more discussion, see section 5.6.

5.5.3 Video DTOE cycles

At the end of each video raster line, a data transfer cycle must take place to refresh the second port of the dual-ported video RAMs. This cycle, known as a DTOE cycle, looks exactly like ordinary read cycles except that the DTOE pin is held low during the falling edge of RAS. The row address defines which row is down-loaded. The column address defines the start address of the following serial bit stream (pixels). Both of these can be programmed as part of the video control section, thus allowing horizontal and vertical scrolling.

5.5.4 I/O

Three external signals are provided for IO addressing. IOST and IOSL are active low timing signals. The position of the falling edge of IOST can be programmed by using different addresses. Appendix B shows their relationship.

The signal WE is an active low write enable and is valid almost throughout the IO cycle.

5.6 Bus arbitration

Bus arbitration on Charity I uses the two T800 signals Memreq and Memgranted. Video data transfer cycles demand that the processor releases the bus within approximately 2.5 us from a bus request.

The circuits which can control the bus, in order of priority are as follows:

Video (data transfer cycles)
Blitter
Refresh
Processor

5.6.1 Video

A data transfer (DTOE) cycle needs to take place during horizontal blanking on every line. This is the most critical aspect of the bus arbitration. If horizontal blanking is only 3us (such as video mode 0), then the processor must release the bus within about 2.5us.

5.6.2 Blitter

The blitter will always hog the bus once it has control. However, if a refresh cycle or DTOE cycle becomes due during a blit, the blitter will pause to allow it to take place. There will be no gap between blitter cycles and either video or refresh cycles; they will simply slot in.

5.6.3 Refresh

Charity I contains a four bit up / down counter, which can actually be read as one of the test registers. Each time a refresh cycle becomes due, the counter increments by one. Each time a refresh cycle actually takes place, the counter decrements by one. If the count reaches fifteen, then Charity will request the bus from the processor and then do fifteen refresh cycles. If the bus has been grabbed for any other purpose, then any pending refresh cycles will simply be slotted in; with higher priority than the blitter, but lower priority than video.

The reason behind this rather elaborate scheme is to reduce the overhead incurred by bus transfers.

5.7 Test

Charity I contains three test registers. One of these is write only (TS_REG). The other two are read only. These registers are incorporated for purposes of testing the chip. Few of the functions available are of interest to the user with the exception of TS_REG(REN) and T1_REG(ERIN)

Name	offset	Bits	Description
TS	0008	16	Test
T0	00C1	32	Test data (read only)
T1	00C2	32	Test data (read only)

TS Test control

Offset	Width	Name	Description
15:14	2	TSP(1:0)	Spare test bits
13	1	REN	Refresh enable
12	1	RTST	Forces 1 pending refresh cycle
11		QTS4	Increment bits 12–15 of Dest total
10		QTS3	4–7
9		QTS2	0–3
8		QTS	1 Increment bits 8–11 of Dest Width
7		QTS0	4–7
6		VTS6	Increment bits 0–3 of V–Total
5		VTS5	4–7 of Vsync width
4		VTS4	8–11 of V–Total
3		VTS3	4–7
2		VTS2	4–7 of Hsync width
1		VTS1	8–11 of H–Total
0		VTS0	4–7

REN must be set to 1 to enable refresh. This should obviously be done immediately after the ABAQ is switched on before loading data into dynamic memory. This would normally be done as part of the boot process.

T0 **Test register 0**

Offset	Width	Name	Description
23:12	12	VC(11:0)	Vertical count (line number)
11:0	12	HC(11:0)	Horizontal count (word number)

T1 **Test register 1**

Offset	Width	Name	Description
28	1	ERIN	Error input, (pin on chip).
27:20	8	RN(7:0)	Random number (CPU ticks)
19:16	3	RP(3:0)	Refresh pending counter
15:8	8	VW(7:0)	Vertical sync width counter
7:0	8	HW(7:0)	Horizontal sync width counter

The random number RN(7:0) is actually an eight bit counter, clocked by the processor clock MCLK.

The pin ERIN is normally connected to the error signal from a 'Farm' of transputers. The bit ERIN reflects the voltage on this pin.

6 Video

6.1 Video modes

ABAQ supports four different video modes. These are:

Mode	Resolution	Bits/pixel	Interface	Palette
0	1280 x 960	4	Analogue RGB	Yes – 4096
1	1024 x 768	8	Analogue RGB	Yes – 16 million
2	640 x 480	8	Analogue RGB	Yes – 16 million
3	512 x 480	32	Analogue RGB	No

Mode 0 provides for 16 simultaneous colours or shades of grey from a palette of 4096. (4 bits per colour using 4–bit DACs)

Modes 1 and 2 provide 256 simultaneous colours from a palette of 16,777,216 colours available. 8 bit DACs are used.

In mode 3 every pixel can have a different colour chosen from the 16,777,216 colours available.

Mode 2 is the only mode which allows two independant (switchable) screens. However, by careful use of the palette map, modes 0 and 1 can also allow several screens at the cost of colours, co–resident within video memory.

A programmable line interrupt can be used to alter the palette on line by line basis providing extra screen colours.

6.2 Video outputs

There are three sets of analogue video outputs from the daughter board, together with a set of sync outputs. These are:

Function	Signal	No. of bits	Output impedance
Mode 0	Red	4	75 Ohms
	Green	4	75 Ohms
	Blue	4	75 Ohms
Modes 1&2	Red	8	75 Ohms
	Green	8	75 Ohms
	Blue	8	75 Ohms
Mode 3	Red	8	75 Ohms
	Green	8	75 Ohms
	Blue	8	75 Ohms
Sync	Horiz.	1)	75 Ohms
	Vertical	1) +ve going	75 Ohms
	Composite	1)	75 Ohms

These outputs are not overload protected

6.3 Monitors

The following pixel and sync rates have been proposed and will be roughly adhered to. However, changes may be made to account for the available monitors, DAC speeds, RFI considerations and so on.

Mode	Resolution	Pixel rate	Line rate	Frame rate	Horiz Blanking	Vertical Blanking
0	1280 x 960	107.500 MHz	65 KHz	60 Hz	3.5 uS	800 uS
1	1024 x 760	66.000 MHz	50 KHz	60 Hz	3.5 uS	800 uS
2	640 x 480	26.875 MHz	31.5 KHz	60 Hz	8.0 uS	800 uS
3	512 x 480	25.000 MHz	31.5 KHz	60 Hz	8.0 uS	800 uS

The monitors recommended are:

NEC Multisync XL	Modes 1,2,3
NEC Multisync Plus	Modes 1,2,3
Hitachi 4615–D–BB–3	Mode 0

6.4 Video control registers

These registers are physically situated on the video daughter board; i.e they are not in Charity 1.

VMODE_REG **Mode control** **Address: 42400000**

This register sets up the video output mode, enabling the appropriate CLUT. The bit assignment is non–explicit; the relevant magic bytes are:

Mode 0:	02
Mode 1:	3C
Mode 2:	36
Mode 3:	11

The DACs corresponding to non–selected video modes are disabled (blanked).

C453WADDR_REG **Mode 1,2 write addesses** **Address: 42D00000**

The DAC set for these modes is a Brooktree 453; hence the name of the register. Writing to this register sets a pixel value which will produce the colour value specified in the next three bytes written to C453WDATA_REG. This register is auto–incrementing (see below)

C453WDATA_REG **Mode 1,2 write colour data** **Address: 42D02000**

Colour data are written sequentially to this register in the order R,G,B and these colour values will be assigned to the pixel value held in C453WADDR_REG. After the 3rd (blue) byte is written the address (pixel value) is incremented by 1. This enables a block of colour data to be written to the CLUT without writing to C453WADDR_REG for every pixel value.

C453RADDR_REG **Mode 1,2 read addresses** **Address: 42500000**

This sets the pixel value or base for subsequent READ operations. As with C453WADDR_REG it is auto–incrementing.

C453RDATA_REG **Mode 1,2 read colour data** **Address: 42502000**

Sequential READ operations will allow the colour values associated with the pixel value held in C453RADDR_REG to be read in the order R,G,B. Further READ operations will read subsequent pixel colour vales as above.

C454WADDR_REG **Mode 0 write addresses** **Address: 42580000**

As for C453WADDR_REG but for the Brooktree 454 DAC set used for Mode 0.

C454WDATA_REG **Mode 0 write colour data** **Address: 42582000**

As for C453WDATA_REG but for the Brooktree 454 DAC set used for Mode 0.

Note that Mode 0 CLUT data remains valid even if other modes are selected (as does Mode 1 or 2 data when Mode 0 is selected).

6.4 Video Memory address maps

The ABAQ supports one megabyte of dual port video RAM. This VRAM is mapped in four ways to the address space, in 2M address spaces, starting at address 00000000. Each map is optimised for a different video mode and allows pixels to be addressed by their x–y co–ordinates rather than their address in memory, thus saving shifting and multiplication operations.

Mode 0 :	MAP0 provides one nibble per byte address
Mode 1 and 2:	MAP1 and MAP2 provides one byte per byte address
Mode 3 :	MAP3 provides one word per word address

The different modes of video RAM addressing are selected by adding multiples of 2Mbyte offsets to the video RAM base address.

In every map, the lowest address is at the top left, and the highest address is at the bottom right. The least significant byte is always to the left of the most significant byte of a word.

Although the video RAM is a 1 Mbyte linear array it can also be thought of as a 2D array VRAM[1024,1024] where byte VRAM[x.y] is at linear address offset x + y*1024.

To the end of this chapter assume the following types:

MODEn[x,y]	:	Elements of type PIXEL
VRAM[x,y]	:	Elements of type BYTE
MAPn[x,y]	:	Elements of type BYTE

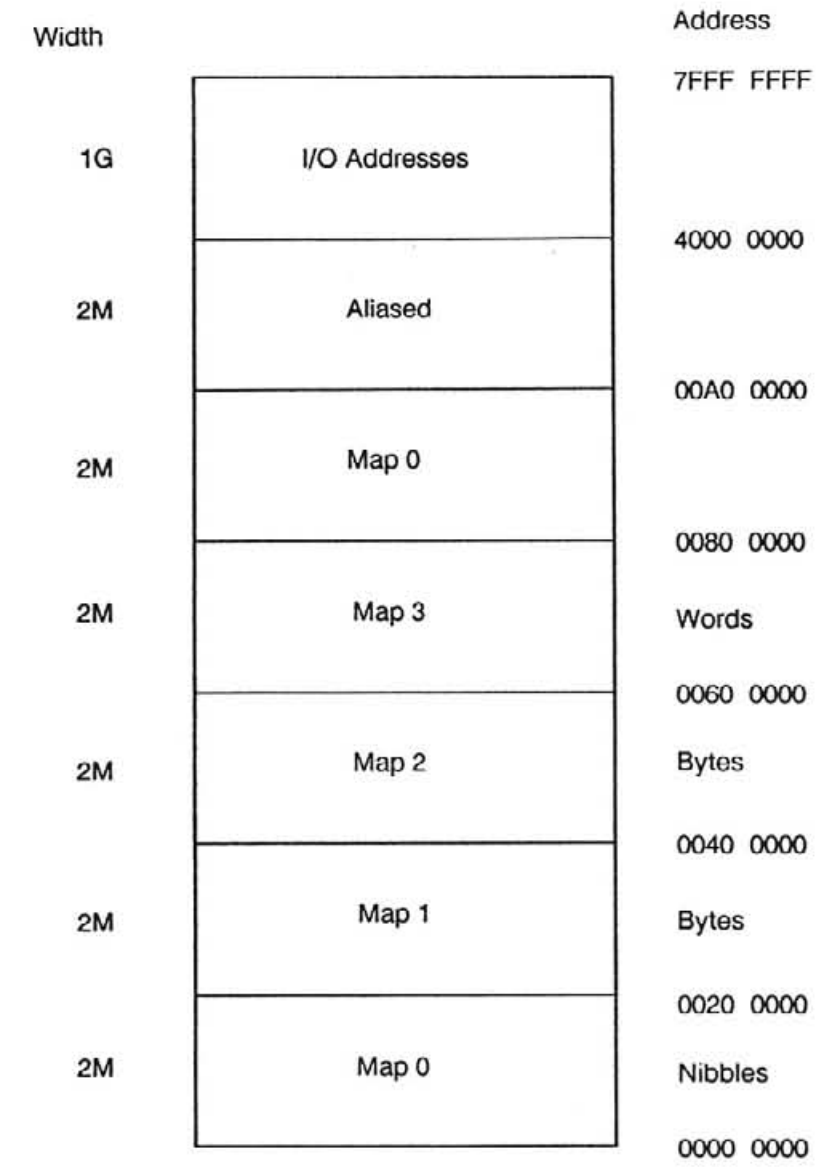


Fig 6.1 Video Memory Map

MAP0

Only the bottom nibble of each byte is used, the upper 4 bits will read as ones.

MAP0 is organised as 1024 lines of 2048 NIBBLES (4 bits) each byte address of pixel (x,y) is 2048*y + x

In mode 0, the display shows an area 1280 by 960 pixels from this map. See Fig 6.2

Mode 0
The screen file is an array MODE0[2048,1024].
Array elements (pixels) are 4 bits deep.
Pixel at x,y is at address VRAM[x/2,y], starting at bit x&1 < 2.
There are 8 pixels per word, 2 pixels per byte.

MAP1 and MAP2

MAP1 and MAP 2 are identical, and are organised as 1024 lines of 2048 BYTES each. The second 1024 bytes of each line are identical to, and a shadow copy of the first 1024 bytes. See Fig 6.3

Byte address of pixel (x,y) is 2048*y + x

In mode 1, the display shows an area 1024 by 768 pixels from these maps.

Mode1
The screen file is an array MODE1[1024,1024]
Array elements (pixels) are 8 bits deep.
Pixel at x,y is at address VRAM[x,y].
There are 4 pixels per word.

In mode 2, the display shows an area 640 by 480 pixels from these maps.

Mode2
The screen file is an array MODE2[1024,1024]
Array elements (pixels) are 8 bits deep.
Pixel at x,y is at address VRAM[x,y].
There are 4 pixels per word.

MAP3

MAP3 is organised as 512 lines of 1024 WORDS (32 bits)each. The second 512 words of each line are identical to, and a shadow copy of the first 512 words. See Fig 6.4.

Word address of pixel (x,y) is 1024*y + x
In mode 3, the display shows an area 512 by 480 pixels from these maps.

Mode3
The screen file is an array MODE3[512,512]
Array elements (pixels) are 32 bits deep.
Pixel at x,y is at address VRAM[x<2,y<1].
There is 1 pixel per word.

Fig 6.2 MAP0

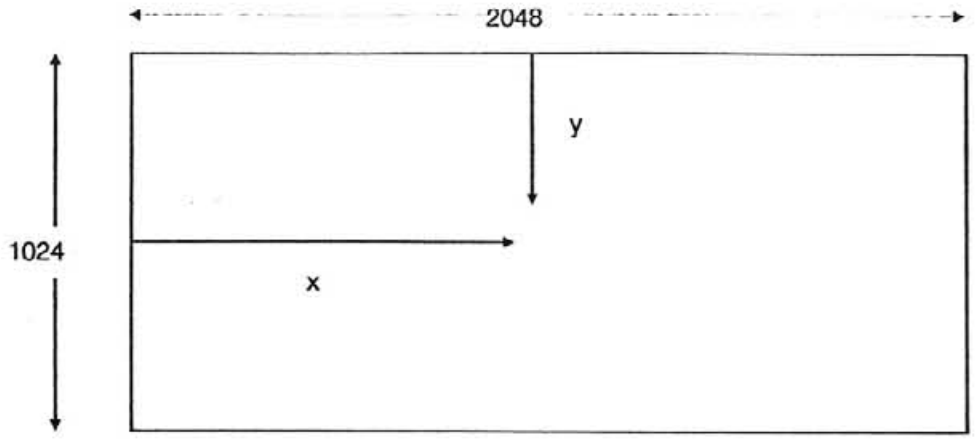


Fig 6.3 MAP0 / MAP1

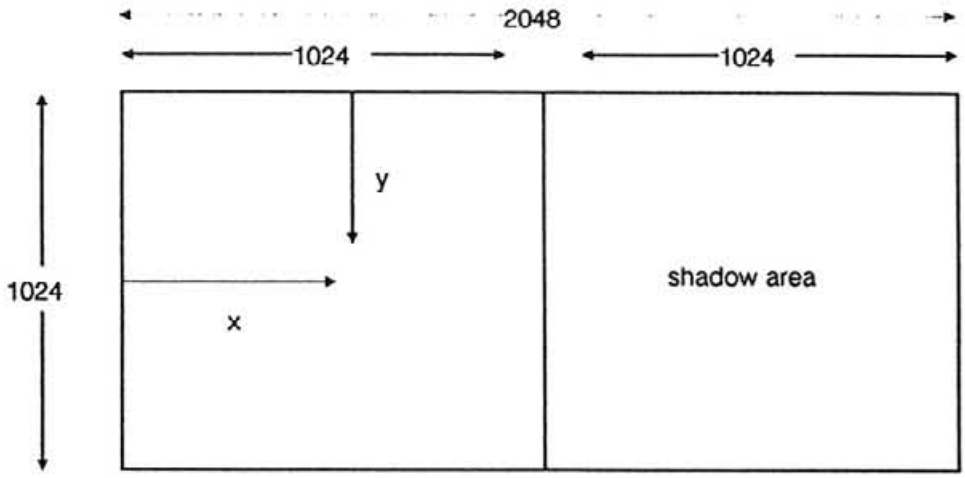
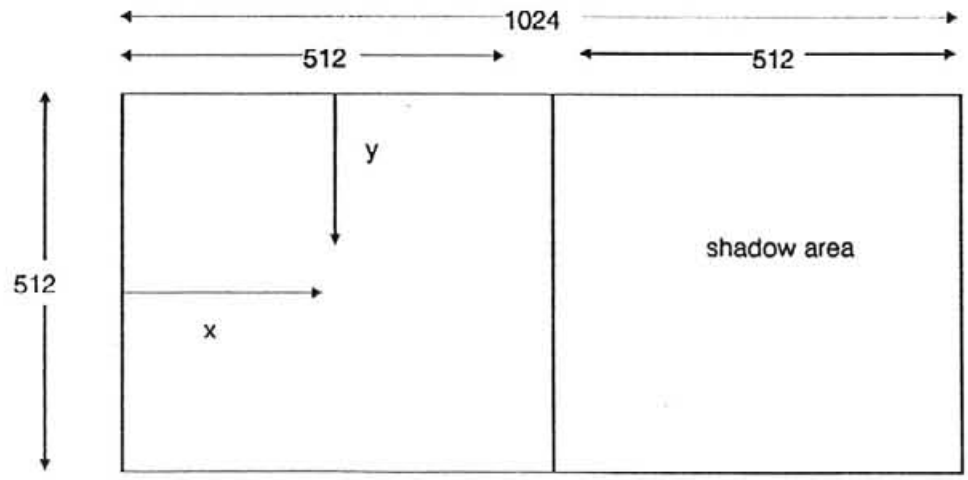


Fig 6.4 MAP3



The mapping from 32 bits to displayed colour is as follows:

bit	3130	24 23	16 15	8 7	0
MSB	| |	| RED	| GREEN	| BLUE	| LSB

bit 31 not used by the display

Overlay bit (forces white if set, regardless of other bits in word)

Bits	Function	Notes
31	Overlay bit	Forces white if set
24–30	not used	May be used by the programmer.
16–23	Red output	(0 => off, 255 => full on)
8–14	Green output	(0 => off, 255 => full on)
0–7	Blue output	(0 => off, 255 => full on)

To recap, the video modes are as follows:

Mode	Resolution	Bits/pixel
0	1280 x 960	4
1	1024 x 768	8
2	640 x 480	8
3	512 x 480	32

There are four video memory maps called MAP0–3. They represent sensible pixel mapping for the video modes 0 to 3 respectively. Which map is actually used to update screen memory is entirely at the discretion of the software writer.Each video memory map is 2 Megabytes wide. The appropriate map is accessed by adding an offset to the base address of video memory.

An additional complication is that screen start addresses can be altered from zero. The way this works is that an offset can be added to each coordinate of the 2D VRAM space. This results is a vertical scroll granularity of one line (half a line in mode 3), and a horizontal scroll granularity of one word (either 1, 4, 8 pixels, depending on the curent mode).

The resultant address VRAM[x',y'] is given by:

VRAM[x',y'] := VRAM[(x+row) MOD 1024, (y+col) MOD 1024]

where x, y are found by the above equations for each video mode.

6.5 Video memory organisation

Charity I supports a very flexible video subsystem, based around 1 Mbyte of dual ported video RAM. The architecture requires the use of 32 64K x 4 video RAM chips, organised into four parallel rows of 32 bits each:

CPU Data	31–28	27–24	23–20	19–16	15–12	11–8	7–4	3–0
ROW A								
ROW B								
ROW C								
ROW D								
Video Data	31–28	27–24	23–20	19–16	15–12	11–8	7–4	3–0

Each chip has a separate CAS line, which allows a single Write cycle to write to all 32 chips. Since the layout of the chips corresponds to a block of 8 x 4 pixels (in mode 0) or 4 x 4 pixels (in modes 1 and 2), the CPU can write to a block of pixels extremely efficiently (via the blitter, in PBS mode – see below).

The different video RAM memory maps map addresses to cas lines in different ways. For example, in MAP 1, memory in Row B is found at addresses 2048 bytes offset from memory in Row A.

6.6 Video control registers

There are nine video registers, plus one video enable bit in the master control register. The video control register (VC_REG) should be written to once before enabling video.

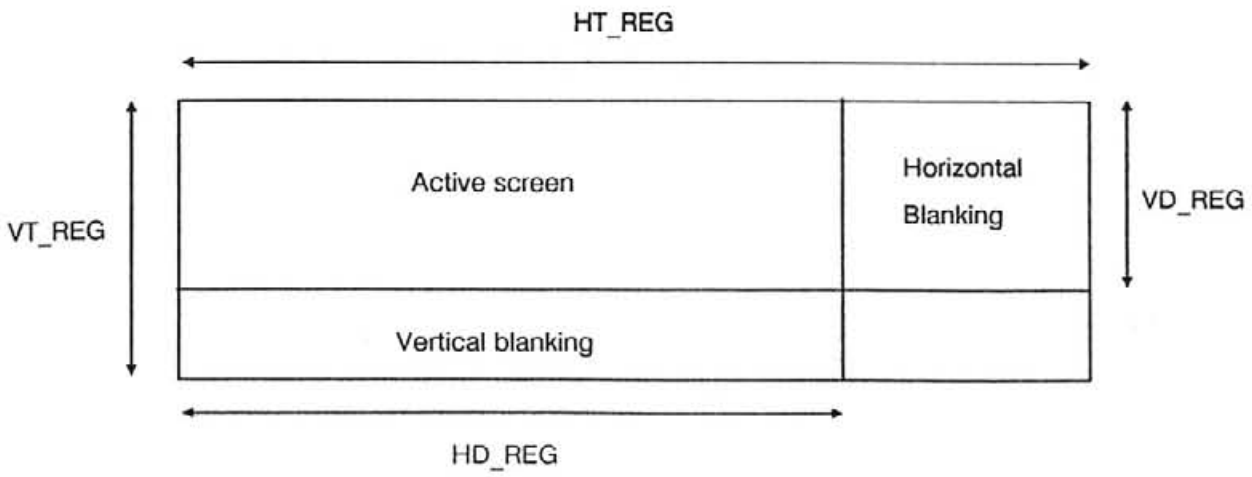
The nine registers are as follows:

Name	Address offset	Bits	Description
HT	0040	12	Horizontal Total (words)
HD	0044	12	Horizontal Displayed
HS	0048	12	Horizontal Sync
HW	004C	8	HSYNC Width
VT	0050	12	Vertical Total
VD	0054	12	Vertical Displayed
VS	0058	12	Vertical Sync
VW	005C	8	VSYNC Width
VC	0060	28	Video Control

The first eight need to be programmed with certain 'magic' numbers, dependant on which mode and monitor is being used. Apendix D gives examples for some well known monitors.

The following diagram shows the relationship between the registers:

HT_REG	Horizontal total
HD_REG	Horizontal displayed
VT_REG	Vertical total
VD_REG	Vertical displayed



The registers

HS_REG	Horizontal sync position
HW_REG	Horizontal sync width
VS_REG	Vertical sync position
VW_REG	Vertical sync width

are self explanatory. The units used for all the video registers are the period of the 'Word clock'. This is the period of the clock used to cycle the 32 bit wide video memory. Therefore in mode 0, which has 4 bit pixels and eight pixels a word, the word clock has a period eight times the pixel period (about 74 nanoseconds). In mode 1, which has four 8 bit pixel a word, the word clock has a period four times the pixel period (about 60 nanoseconds).

The video control register allows horizontal and vertical scrolling, as well as containing certain other 'magic' numbers.

VC Video control

Offset	Width	Name	Description
31:24	8	VRO(7:0)	Row start address
23:16	8	VCO(7:0)	Column word start address
15	1	VM3	Select video mode 3 (512 words / line)
14:13	2	USK(1:0)	Row 1 skew
12:11	2	LSK(1:0)	Row 0 skew
10:8	3	VPL(2:0)	Sync polarity
7:6	2	VMX(1:0)	Sync mixing
5:4	2	VSB(1:0)	Start Bank
3:2	2	VBD(1:0)	Blank skew
1:0	2	VCD(1:0)	Column equate skew

Bit	7	6	5	4	3	2	1	0	
	VM3	USK1	USK0	LSK1	LSK0	VPL2	VPL1	VPL0	**Byte 1**
	VMX1	VMX0	VSB1	VSB0	VBD1	VBD0	VCD1	VCD0	**Byte 0**

VPL(2:0)	Sync polarity	
XX0	BLANK active high	
XX1	BLANK active low	
X0X	VSYNC active high	
X1X	VSYNC active low	
0XX	HSYNC active high	see below
1XX	HSYNC active low	see below

VMX(1:0) Sync mixing

11	HSYNC = HS nand VS
10	HSYNC = HS eor VS
01	HSYNC = HS 00 | HSYNC = HS

HS = normal Hsync position
VS = VSYNC after polarity changes defined by VPL1

HSYNC can then be inverted by setting VPL(2) high

7 Raster operations

7.1 Introduction

CHARITY I is the first gate array for the ABAQ. It has about 8500 gates, roughly half of which are used to implement a blitter, which performs two–dimensional raster operations.

Raster operations for two dimensional scan graphics were first proposed by Sproull for the early PARC machines. The design used in the ABAQ is a development of the form proposed by Willis, suitable for a colour display with a colour look–up table, where the mapping from value to hue is not necessarily linear.

In essence the blitter copies, subject to various conditions and masks, from a two–dimensional Source area to a two dimensional Destination area. The blitter reads and writes one address location at a time, and has the same view of memory and display organisation as the processor. The direction of scan of the blitter may be set to allow source and destination to overlap. The blitter is implemented using 32–bit wide data paths. Depending on which display map is used, the blitter is capable of reading, manipulating and writing up to eight pixels at a time in the higher resolution modes, and a special feature allows 32 pixels to be written in one memory cycle in the video RAM, allowing super–fast area fill, line draw and most important, font plotting.

7.2 Set up

There are a few important points to note when writing software for Charity I:

1. After a hard reset, refresh is disabled by default. This makes testing easier since it is difficult to anticipate precisely when refresh cycles will occur. To enable refresh, set the bit REN (bit 13) in TS_REG (address 4000 0008). Normally this will be done by the I/O processor as part of its standard reset and boot sequence.

2. Before enabling video or the blitter, the video control register (VC_REG) and blitter control register (BC_REG) respectively should be primed. This is necessary in the test patterns since the contents of these registers are undefined after reset. In reality the contents are defined so the priming of them is not strictly speaking necessary. Nevertheless it is still considered good practice.

7.3 Blitter functions

7.3.1 Destination area

The action of the blitter is controlled by the destination area definition. The destination area is defined by the following registers.

DT_REG **Destination total 16 bits** **Address offset: 0090**

Total number of address space units to be transferred, including any partial words at the start or end of a line.

CHARITY 1 uses a sixteen bit register allowing a maximum of 65536 Words to be transferred in each blit. To transfer more than this, several blit operations should be done. Only the DT_REG register need be changed each time.

DB_REG **Destination base address 26 bits** **Address offset: 0094**

The least significant 26 bits of the address at which the first word (or partial word) is to be copied to. This is the address of one corner of the area; which corner depends on the direction of scan. Bit 31 of the address is mapped into CA_REG. The destination area can therefore be mapped onto the lowest 64 Mbytes of either main RAM or video RAM. At the conclusion of a blit operation, the destination base address will point to the next logical address, as if the operation had not terminated. In this way, blit operations can be split up into smaller blits and executed sequentially without having to reload registers.

DW_REG **Destination width (–2)** **12 bits**

Width in Words of the area, minus 2, including any partial words at start or end of line. The number of lines written (Y) will be (Total/X–width = DT/(DW+2))

DS_REG **Destination stride** **12 bits** **Address offset: 009C**

The difference, in Words, between the last word on one line and the first word on an adjacent line. DS should be set to zero if a linear destination map is required.

DS + DW + 2 = address difference (in Words) between vertically adjacent pixels (= 512 normally).

7.3.2 Source Area

The source area is defined by the following registers. The total number of address space units transferred is controlled by the Destination Total register. Therefore the source and destination areas must be the same size and width.

SB_REG **Source base address 26 bits** **Address offset: 00A0**

The least significant 26 bits of the address from which the first word (or partial word) is to be copied. This is the address of one corner of the area; which corner depends on the direction of scan. Bit 31 of the address is mapped into CA_REG. The source area can therefore be mapped onto the lowest 64 Mbytes of either main RAM or video RAM.

At the conclusion of a blit operation, the source base address will point to the next logical address, as if the operation had not terminated. In this way, blit operations can be split up into smaller blits and executed sequentially without having to reload registers.

SS_REG **Source stride 12 bits** **Address offset: 00A4**

The source stride is the difference, in address space units, between the last word on one line and the first word on an adjacent line. SS should be set to zero if a linear source map is required.

7.3.3 End masks

When the start and end of lines are not on Word boundaries, we need to mask the pixels which lie outside the active region. These masks are found in the pixel alignment register PA_REG.

The 'start' of a line is defined as the first end of the line to be operated on; therefore it will be the right hand end if the copy is backwards. The start and end masks are each 8 bits, since a word can be up to 8 pixels.

PA **Pixel Alignment Register** **Address offset: 0088**

Offset	Width	Name	Description
*31	1	MST	Select 2D Mask shifts
*30:26	5	MS(4:0)	Mask shift
25	1	SST	Select 2D Source shifts
24:20	5	SS(4:0)	Source shift
19:16	4	MM(3:0)	Middle mask (lines)
15:8	8	EM(7:0)	End Mask
7:0	8	SM(7:0)	Start Mask

PA_REG(SM) **Start mask** **8 bits**

Each bit corresponds to a 4 bit nibble of the destination word at the start of a line and must be set to one (1) for all pixels inside the active area. Bit 0 corresponds to the least significant nibble.

PA_REG(EM) **End mask** **8 bits**

Each bit corresponds to a 4 bit nibble of the destination word at the end of a line and must be set to one (1) for all pixels inside the active area. Bit 0 corresponds to the least significant nibble.

PA_REG(MM) **Mid mask** **4 bits**

This register is only active for Pixel Block Mode. It is necessary because of the two dimensional nature of this mode. Each bit then corresponds to one horizontal line and must be set to one (1) for lines inside the active area. Bit 0 corresponds to the first line to be accessed (depending on the direction of the operation).

7.3.4 Count direction

In order to allow overlapping blocks to be copied, we need to be able to control the direction of the copy. CHARITY is very flexible in this respect and allows source, destination and mask count directions to be set independently. The Count Alignment register CA_REG contains the necessary bits.

CA **Count direction Alignment** **Address offset: 008C**

Offset	Width	Name	Description
15	1	MUPL	Mask Line count UP
14	1	MUPB	Mask Bit count UP
13	1	QD31	Destination address bit 31
12	1	QS31	Source address bit 31
11:9	3	DC(5:3)	Dest' Horizontal increment
8:6	3	DC(2:0)	Dest' Vertical increment
5:3	3	SC(5:3)	Source Horizontal increment
2:0	6	SC(2:0)	Source Vertical increment

CA_REG(DC) **Destination count** **6 bits**

Two sets of three bits. One set controls the increment to the base register at the end of a line, and one set controls the increment to the base register everywhere else.

DC(2:0) **Destination count at end of line 3 bits**

DC(2:0)	DC(2)	DC(1)	DC(0)	Operation
0	0	0	0	+ Stride
1	0	0	1	+ Stride + 1
2	0	1	0	– Stride – 1
3	0	1	1	– Stride
4	1	0	0	+ 0
5	1	0	1	+ 1
6	1	1	0	– 1
7	1	1	1	+ 0

DC(5:3) Destination count everywhere else 3 bits

DC(5:3)	DC(5)	DC(4)	DC(3)	Operation
0	0	0	0	+ Stride
1	0	0	1	+ Stride + 1
2	0	1	0	– Stride – 1
3	0	1	1	– Stride
4	1	0	0	+ 0
5	1	0	1	+ 1
6	1	1	0	– 1
7	1	1	1	+ 0

CA_REG(SC) **Source count** **6 bits**

Two sets of three bits. One set controls the increment to the base register at the end of a line, and one set controls the increment to the base register everywhere else.

SC(2:0) **Source count at end of line** **3 bits**

SC(5:3) **Source count everywhere else 3 bits**

These two registers have the same structure as the Destination count registers above.

CA_REG(MUPL) **Mask line count up** **1 bit**

When set to one (1), the mask line counter counts up.

CA_REG(MUPB) **Mask bit count up** **1 bit**

When set to one (1), the mask bit counter counts up.

7.3.5 Pixel alignment

When source and destination pixels are not word aligned, the source needs to be shifted and up to one word of the residue stored. So, two source words may be included in one destination word. The pixel alignment register PA_REG controls the shifting and the Pipe control register PC_REG the pipelined source stream. See above for PA_REG structure and address offset.

PA_REG(SS) **Source shift** **5 bits**

A value in the range 0–31 indicating the number of bits by which a source word must be circularly rotated right to align with the destination word.

PC_REG **Pipe control** **8 bits** **Address offset: 00B0**

There is a pipeline in the source stream to allow a word written to the destination area to contain nibbles from adjacent words in the source area. Each bit in the pipeline control byte controls one 4 bit nibble in the source stream. Set it to a one for each nibble to be read from the previous word. Note that the value must be consistent with the value in the alignment register above for correct operation.

Source word (8 nibbles each word)	**Sn** |0 1 2 3 4 5 6 7 |	**Sn+1** |0 1 2 3 4 5 6 7 |
Shifted	|5 6 7 0 1 2 3 4 |	|5 6 7 0 1 2 3 4 |
Destination	|0 1 2 3	4 5 6 7 |
		Dn+1

In this example, the shift alignment would be $C (always the same, for forward or backward copy).

If a forward copy, the Pipe control would be set to $0F. If a backward copy, the Pipe control would be set to $F0.

PA_REG(SST) Select 2D source shifts 1 bit

When set to one (1), the barrel shifter will do a two dimensional shift. This is necessary for Pixel Block Mode since this mode is two dimensional in nature.

When SST = 1, the meaning of the shift control register bits PA_REG(SS) change. Bits 0–2 control horizonal shifting (by up to 8 bits). Bits 3–4 control vertical shifting (by up to 4 lines). Refer to the Pixel Block Mode section of this document for more details.

7.3.6 Pixel value selection tests

Destination pixels are selectively written. The blitter provides facilities to select pixels on value, allowing a multitude of visual effects (such as overlay, underlay, colour–key, or foreground/background manipulation) to be achieved. The blitter can operate on either 4 bit or 8 bit pixels. Since the data path is 32 bits wide, the blitter can therefore operate simultaneously on eight or four pixels respectively.

What is actually written to a destination pixel as a result of a blit operation, is decided by various parameters, most of which can be turned on or off;

1. Source data
2. Destination data
3. Mask data
4. Start and end masks

The equation is in fact as follows:

```
R= (        <start_mask>.start_of_line
            + <end_mask>.end_of_line
            + (not_start_of_line . not_end_of_line)
    )
  . (       <mask_data> + mask_disable
            . Test1(source;TV1,TV2,dest)
            . Test2(source;TV1,TV2,dest)
            . Test3(source,TV3,TV4;dest)
            . Test4(source,TV3,TV4;dest) )
```

. ==> Logical AND + ==> Logical OR

The result R, can be 1 (true) or 0 (false).

The masks have already been described above.

TC_REG **Test control** **24 bits** **Address offset: 0084**

The blitter will concurrently process four test conditions for each destination pixel. All four tests are ALWAYS done, and all four need to be passed in order that a pixel is written. The test control register TC_REG controls these tests.

TC Test **Control Register**

Offset	Width	Name	Description
*23	1	MKCT	
22	1	MUXB	
*21	1	BTSL	
20	1	FWR	Force write
*19	1	MEN	Mask Enable
*18	1	QFAL	Output if false
*17	1	QTRU	Output if true
16	1	TST8	8 bit tests
15:12	4	TIN(3:0)	Invert test
11:8	4	TLE(3:0)	Test for 'less than or equal'
7:4	4	TD(3:0)	Test data
3:0	4	TV(3:0)	Test value

TC_REG(TLE) **Test less than or equal** **4 bits**

TC_REG(TIN) **Invert test** **4 bits**

Each test n (n=1 to 4) can be individually programmed to be one of four types:

TINn	TLEn	Test
0	0	A > B
0	1	A < B
1	0	A <= B
1	1	A >= B

TC_REG(TD) **Test data** **4 bits**

TC_REG(TV) **Test value** **4 bits**

The data for A and B is defined by TDn and TVn:

Tests 1 and 2:

TDn	TVn	A	B
0	0	SOURCE	TV0
0	1	SOURCE	TV1
1	X	SOURCE	DEST

Tests 3 and 4:

TDn	TVn	A	B
0	0	TV2	DEST
0	1	TV3	DEST
1	X	SOURCE	DEST

The least significant bit of each of these registers corresponds with test 1. The most significant, with test 4.

Note: In Phil Willis's paper the four tests are set up so as to provide selection for values in some closed

source and destination ranges, for example

((v1 <= source <= v2) & (v3 <= destination <= v4))

but the blitter allows more general tests than the example above. Never (or always) can be achieved by using suitable test values.

TC_REG(TST8) **Cascade tests to 8 bits** **1 bit**

The comparison is usually four bits wide, but can be cascaded to eight bits by setting this bit to one (1). The 8 bit result will be associated with the upper (odd) nibble. The bit TC_REG(MUXB) should be cleared in order to associate this 8 bit result with the lower nibble as well. The effect is important since nibble wide write cycles are only possible to video RAM. TC_REG(MUXB) Independent nibble results 1 bit

The rules for TST8 and MUXB are summarised below:

Destination in Main Ram:

TST8	MUXB	
0	0	Write if odd nibble result true
0	1	Write if even nibble result true
1	0	Write if 8 bit result true
1	1	Write if even nibble result true

Destination in video RAM, Map 0: NB. Even nibble data is written, odd nibble data is discarded.

TST8	MUXB	
0	0	Write if odd nibble result true
0	1	Write if even nibble result true
1	0	Write if 8 bit result true
1	1	Write if even nibble result true

Destination in Video RAM, not Map 0: NB. All 8 nibbles valid candidates for independent writing.

TST8	MUXB	
0	0	Write byte if odd nibble result true
0	1	Write nibble if aligned nibble result true
1	0	Write byte if 8 bit result true
1	1	Write odd nibble if 8 bit result true
		Write even nibble if even nibble result true

Nibble number	Main RAM	VRAM not Map0	VRAM Map0
7		x	
6	x	x	x
5		x	
4	x	x	x
3		x	
2	x	x	x
1		x	
0	x	x	x

TST8, wghen set cascades the tests to produce an 8 bit result, on the upper (odd) nibble. The result on the even nibble represents the result of the even nibble test.

MUXB, when cleared, equates the results of even and odd nibbles.

7.3.6.1 Writing to destination area

It will help in understanding how pixels are selectively written to different areas of memory, by explaining the write characteristics.

Main RAM has four write lines. Therefore, trying to selectively write nibbles to main RAM will not work! See the table above to find out which test determines whether a particular byte is written.

Video RAM effectively has eight write lines. This allows nibbles to be selectively written. However, MAP0 discards odd nibbles, so it has the same problems as Main RAM.

The rule is, if operating on four bit pixels, copy from:

Video RAM MAP1 to MAP1	(fast! 8 pixels a cycle)
Video RAM MAP0 to Main RAM	(4 pixels a cycle)
Main RAM to Video RAM MAP1	(8 pixels a cycle)

If the tests are disabled (always true), then of course one can copy from MAP1 to main RAM.

TC_REG(FWR)	**Force write**	**1 bit**	
TC_REG(QTRU)	**Action if true**	**1 bit**	**** not in CHARITY1**
TC_REG(QFAL)	**Action if false**	**1 bit**	**** not in CHARITY1**

The register bits FWR (force a write), QFAL (false) and QTRU (true) determine what will happen if the result is true or false.

R	FWR	QTRU	QFAL	Action
0	0	X	X	Destination unchanged – No write *
0	1	X	0	Write colour register
0	1	X	1	Write source data *
1	X	0	X	Write colour register
1	X	1	X	Write source data

X= Don't care.

NB. The Colour register is not implemented on CHARITY1.

TV_REG **Test values** **32 bits** **Addressoffset: 00B8**

V1–V4 are four eight bit values which are used to compare against the source and destination data. The tests will always occur. The first two compare the source data with either V1, V2 or destination data. The second two compare the destination data with either V3, V4 or the source data. The results of all four tests must be true for the overall result to be true. V1 is the least significant byte of this register, V4 being the most significant byte.

If four bit comparisons are used, then the eight bit value registers (V1–V4) should have the four bit test value duplicated in the upper and lower four bits.

7.3.7 PIXEL BLOCK MODE

The blitter can take advantage of the organisation of video memory to treat each 32–bit source word as a block of 1–bit pixels 8 pixels wide by 4 pixels high, and write 32 nibbles to the destination area in one cycle. PBS mode can only be used where the destination area is in the video memory. When the PBS bit is set (one) the following occurs:

Each source word is treated as 4 rows of 8 1–bit descriptors.

Source alignment is two dimensional.

The mask data is ignored. The planar mask, and the source and destination value tests have no effect.

For each 4-bit nibble of the destination area that corresponds to a position where the source input is set (1) the word aligned nibble of the Test Value register TV_REG is copied to the destination area. Where the source input is zero, no write takes place, and the video memory is unchanged.

Destination area variables are interpreted as before, except that the stride becomes four lines long. Stride, therefore, should be set to a value to move 4 lines down, rather than one line.

In addition to the start and end mask, and additional mask MID_MASK is also provided, so that destination areas that are not multiples of 4 lines deep may be written. MID_MASK is 4 bits wide, and corresponds to the 4 rows written at one time in PBS mode. Rows corresponding to bits set to a one will be written, rows corresponding to bits set to a zero will not be changed.

PBS mode updates video memory very fast indeed. In the high resolution mode 32 pixels are written every two cycles (one to read the source data, one to write), that is 32 pixels in 500ns or a rate of 64 million pixels/sec, or better than 50 full screens per second. If latched source data can be used to write a single colour or 8 x 4 repetative pattern, 128 Million pixels per second can be written for area fill or better than 100 full screens per second.

7.3.8 Blitter control

BC_REG **Blitter Control** **5 bits**

The blitter control register controls the master state machine, and also provides the master reset and GO functions.

The register is 5 bits wide. The control bits are called SRC, DST, MSK, QS0 and PBS. Writing to this register will always cause the blitter to start. The blitter will then hog the main memory bus until the total count is reached. Since the blitter saturates the entire memory bandwidth, the CPU can determine whether a bit op is complete simply by attempting to read external memory.

BC Blitter **Control Register** **Address offset: 0080**

Offset	Width	Name	Description
4	1	REST	Reset total (active high)
3	1	SRC	Read source area
2	1	DST	Read destination area
1	1	MSK	Read mask area
0	1	QS0	Read source preamble

Bit	7	6	5	4	3	2	1	0
				REST	SRC	DST	MSK	QS0

BC_REG(SRC) **Read source area** **1 bit**

When set (1), the source area will be read into the source data register.

BC_REG(DST) **Read destination area** **1 bit**

When set (1), destination data will be read into the destination data register.

BC_REG(MSK) **Read mask area** **1 bit**

When set (1), mask data will be read into the mask data register, once per line. Not implemented in CHARITY1.

The Source, Destination and Mask data registers can each be written to directly by the CPU, rather than be

updated by the incoming data stream.

BC_REG(QS0) **Precharge source pipe** **1 bit**

When set (1) and SRC = 1, a source word will precharge the source data register at the start of each line. This is neccesary if the first destination word requires data from two source words.

BC_REG(REST) **Reset counters** **1 bit**

Writing a one (1) to this bit will reset the counter state machines. Do not set this bit if you want to continue a blit from where the last one left off.

MC_REG(PBS) **Pixel Block Mode** **1 bit**

Pixel bock mode (see below). Note: this bit is actually in the master control register: see below.

MC_REG(QEN) **Blitter enable** **1 bit**

Setting this bit in the master control register will enable the blitter. However, the Blitter master control register BC_REG should be written to first to initialise the system. This is also true of the video system incidentally.

7.4 Master control register

With the exception of MC_REG(QEN) and MC_REG(PBS) the functions of the Master control register relate to non-blit functions.

MC Master Control **8 bits** **Address offset: 0000**

Bit(s)	Name	Description	
7	QEN	QAD (Blitter) Enable	
6	PBS	Pixel block select (for Blitter to VRAM)	
5:4	MEM(1:0)	Memory type	10 4 Mbits
			01 256 Kbits
			00 1 Mbit
3	EEN	Event enable	
2	VEN	Video enable	
1	ROUT	Reset out Inverted	
0	AOUT	Analyse out True	

All bits are reset to zero (0) by master reset. Therefore, ROUT is reset to active and AOUT is reset to inactive.

The bits QEN and VEN should not be set until the video and blitter registers have been initialised at least once. The results are conmpletely undefined if these registers are not written to first (and in fact, the system may well crash).

7.5 Predicted blit rates

Predicted Charity rates (4 bits per pixel)

SRC	DST	MSK	PBS	QS0	Mpixels per second
0	X	X	1	0	128
1	X	X	1	0	64
0	0	0	0	0	32
1	0	0	0	0	16
1	0	1	0	0	<16
1	1	0	0	0	10.6

These rates should be halved for 8 bits per pixel. If mask data is read, it will only add one memory cycle per line. The same situation occurs if QS0 is set. Therefore, for short lines, the overhead becomes more significant.

8 Interfaces and connectors

8.1 Links

The T800 transputer has four, bi-directional, INMOS links to communicate with the outside world (and other transputers). Normally, one link is used for connection to an intelligent I/O processor, two links to implement a fast ring network, and the last link for connection to further transputers for processor expansion.

These links are capable of transferring data at 20MHz or 2350 Kbytes/second each, over desktop distances. Signal conditioning and acknowledgment times may reduce throughput over longer distances.

Links 1,2,3 are available on a patch panel J7, together with uncommitted ECL and TTL buffers, and on external connectors J4, J5 and J6a,b,c,d. **Note that J6a is normally used to connect to the Mega ST on issue 3 ABAQs.**

Error and Analyse

A specification for a general Debug, Analyse, Soft and Hard Error system between transputers has been developed. This is not yet available.

8.2 Mega ST Expansion Board and SCSI interface

This board is fitted into a Mega ST and forms the interface between the ST and an Abaq.

J3 carries the busses and control signals from the Mega ST onto the Expansion Board. Power is supplied to the Board via J4 **8.2.1**

Features:

SCSI interface allowing data transfer rates up to 1.5 MBytes per second.
High performance DMA controller

Inmos Link Adapter allowing data transfer at up to 20MHz

Communication between the Mega ST and the Abaq is conducted down the INMOS link. The SCSI interface is connected to a 40 MByte Winchester Disc.

The RESET and ANALYSE inputs of the transputer on the Abaq mother board can be directly controlled from the Mega ST. (Diagnostic connector)

8.2.2 Connector Summary

J1 SCSI connector 50 way DIL connects to 40 MByte Winchester

J2 Link connector 9 way D type connector (Female) connects to the Abaq mother board.

J3 Expansion connector 64 way DIN41612 connects to the Mega ST main board.

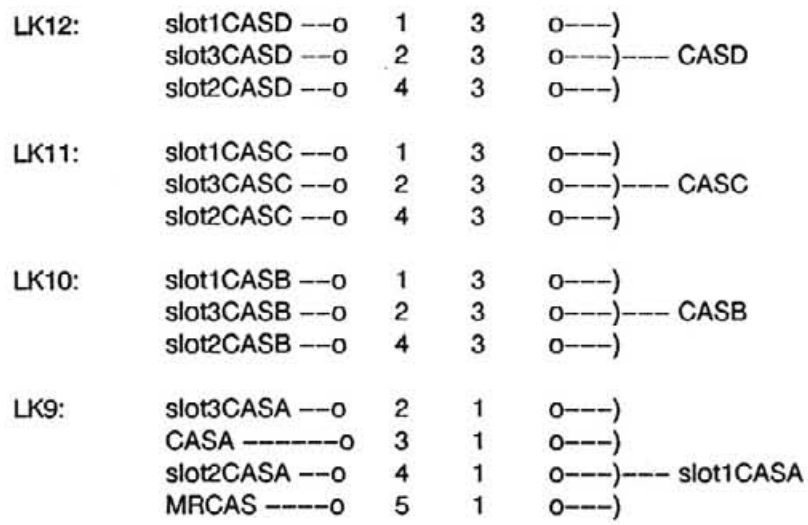
J4 Power connector 5 Way molex connects via a flying lead to the Mega ST main board.

J5 Diagnostic connector 9 way D type (Male) connects to the Abaq mother board.

8.3 Expansion Memory Addressing Jumpers

Jumper areas LK9 – LK12 are provided to allow the four decoded CAS lines to be routed either to expansion slots J101–103 or to the 4Mbytes of on board main RAM (CASA only).

The pinouts of these links are given below. Note that LK9 is incorrectly laid out on the Issue 3 board, producing a slightly nonsensical arrangement of pins.



Factory setting is for CASA to select the on–board RAM. This places the on–board RAM at #80000000 (excluding space overlayed by T800 internal RAM).

Issue 3 factory settings: jumper LK9 p3 – p1
p5 – p1
LK10–12 not jumpered.

8.4 Video connectors (on video daughter board)

There are three sets of analogue video outputs from the daughter board, together with a set of sync outputs. These are:

Function	Signal	Type of connector
Mode 0	Red	Subminiature co–axial
	Green	Subminiature co–axial
	Blue	Subminiature co–axial
Modes 1&2	Red	Subminiature co–axial
	Green	Subminiature co–axial
	Blue	Subminiature co–axial
Mode 3	Red	Subminiature co–axial
	Green	Subminiature co–axial
	Blue	Subminiature co–axial
Sync	Horiz.	Subminiature co–axial
	Vertical	Subminiature co–axial
	Composite	Subminiature co–axial

These outputs are not overload protected

Appendix 1 Outline specification

The Atari Transputer Workstation

The Atari Transputer Workstation Development System is now being distributed to software developers. The workstation in its development system form, requires an Atari Mega ST for use as a front end processor with seperate monitor. The final design will be a single box, single monitor, possibly tower style case, with the 68000 front end processor built in. The 68000 will have at least 500K memory thus allowing standard Atari ST software products to run.

Designed and manufactured in the United Kingdom.

Outline Specification:

Processor: Inmos T800 –20

Memory:

4 Mbyte 120ns DRAM expandable to 16Mbyte DRAM (1 Mbyte parts)
16 Mbytes expandable to 64 Mbytes DRAM (4Mbyte parts)

Video: 1 Mbyte dual–port Video RAM:
1 port to main memory bus,the other to video output

All screen modes 60Hz frame rate landscape format

Mode 0	1280 x 960	16 out of 4096 colours
Mode 1	1024 x 768	256 out of 16 million colours
Mode 2	640 x 480	256 out of 16 million colours
Mode 3	512 x 480	32 bits/pixel true colour

	Pixel Rate	**Line Rate**
Mode 0	107.5 Mhz	65 KHz
Mode 1	66 Mhz	50 KHz
Mode 2	26.875 Mhz	31.5 Khz
Mode 3	25.0 Mhz	31.5KHz

Suitable monitors:

NEC Multisync plus/XL	Mode 1,2,3
Hitachi 4615–D–BB–3	Mode 0

Hardware pixelblt engine (Charity I):

Predicted bit rates:

Square area fill:	128 Mpixels/sec
Character draw:	64 Mpixels/sec
Full raster operation:	16 Mpixels/sec

4 x 75 ohm BNC (RGB, synch on green or composite synch)

Inter–processor links:

Three 10/20 Mhz bi–directional interprocessor links, optionally ECL or TTL buffered, allowing 3 x 2.35 Mbyte/sec transfers simultaneously

Expansion:

Ram expanded internally up to 16 Mbytes (64 Mbytes with 4mx1 parts).
Currently upto 17 transputers can be installed internally.
Other expansion cards planned:

- Ethernet card
- 16 channel asynchronous comms card
- X25 comms card
- Backplane with C004 cross bar for soft control of link configuration

I/O subsystem: Atari Mega ST
DMA SCSI port

All Atari ST peripherals available to transputer system including:

- Atari Laser Printer
- Atari Hard Disc
- Atari CD Rom

Power supply: Internal, fan–cooled
Input: AC 100/120V
AC 220/240V switch selected
50/60 Hz 200 W
Output: + 5v 15A
+12v 4.2A
– 5v 0.3A
–12v 0.25A

Environment: 41 to 113 F (5 to 45 C) operating or idle
–4 to 149 F (–20 to 65 C) storage
–40 to 149 F (–40 to 65 C) transport
Relative Humidity (non–condensing) 20 to 80% operating or idle
up to 95 % storage or transport

Availability: Development systems available now.
Production units available December 1988.

For more information please contact:

Atari Transputer Project Manager
Atari House
Railway Terrace
Slough
Berkshire
SL2 5BZ
United Kingdom

Tel: (44) (0753) 33344
Fax: (44) (0753) 822914

Composed on **Atari Mega 2 ST** and printed on an **Atari Laser Printer**

Appendix 2 Charity 1

1 Register summary

The internal registers occupy a 4 Megabyte address space, starting at address $4000 0000. Registers and bits marked with an asterisk are not implemented on Charity I.

Name	Offset Address	Bits	Description
MC	0000	8	Master control
EA	0004	12	Event line Address
TS	0008	16	Test
HT	0040	12	Horizontal Total (words)
HD	0044	12	Horizontal Displayed
HS	0048	12	Horizontal Sync
HW	004C	8	HSYNC Width
VT	0050	12	Vertical Total
VD	0054	12	Vertical Displayed
VS	0058	12	Vertical Sync
VW	005C	8	VSYNC Width
VC	0060	28	Video Control
BC	0080	5	Blit master control
TC	0084	24	Test control
PA	0088	32	Pixel Alignment
CA	008C	16	Count direction Alignment
DT	0090	16	Destination Total
DB	0094	24	Destination Base address
DW	0098	12	Destination Width
DS	009C	12	Destination Stride
SB	00A0	24	Source Base address
SS	00A4	12	Source Stride
PC	00B0	8	Pipe Control
TV	00B8	32	Test Values (4x8 bits) / Colour data
SD	00BC	32	Source data
DD	0024	32	Destination data
SP	00C0	32	Source pipe (read only)
T0	00C1	32	Test data (read only)
T1	00C2	32	Test data (read only)

Early I/O: $4200 0000 Bits 31 to 23 inclusive are decoded
Late I/O: $4280 0000 Bits 31 to 23 inclusive are decoded

MC Master Control

Bit(s)	Name	Description	
7	QEN	QAD (Blitter) Enable	
6	PBS	Pixel block select (for Blitter to VRAM)	
5:4	MEM(1:0)	Memory type	10 4 Mbits
			01 256 Kbits
			00 1 Mbit
3	EEN	Event enable	
2	VEN	Video enable	
1	ROUT	Reset out	Inverted
0	AOUT	Analyse out	True

All bits are reset to zero (0) by master reset. Therefore, ROUT is reset to active and AOUT is reset to inactive.

The bits QEN and VEN should not be set until the video and blitter registers have been initialised at least once. The results are conmpletely undefined if these registers are not written to first (and in fact, the system may well crash).

EA Event line address

Bit(s)	Name	Description
11:0	EA(11:0)	Video line number on which an event is to occur.

An event will occur each time the raster reaches the end of the line number EA(11:0), providing EEN is active (1). The event will occur at the start of horizontal blanking on that line. The line number can include non displayed lines (during vertical blanking), but must be less than the number in VT_REG (total number of video lines including vertical flyback).

TS Test control

Offset	Width	Name	Description
15:14	2	TSP(1:0)	Spare test bits
13	1	REN	Refresh enable
12	1	RTST	Forces 1 pending refresh cycle
11		QTS4	Increment bits 12-15 of Dest total
10		QTS3	4-7
9		QTS2	0-3
8		QTS1	Increment bits 8-11 of Dest Width
7		QTS0	4-7
6		VTS6	Increment bits 0-3 of V-Total
5		VTS5	4-7 of Vsync width
4		VTS4	8-11 of V-Total
3		VTS3	4-7
2		VTS2	4-7 of Hsync width
1		VTS1	8-11 of H-Total
0		VTS0	4-7

This register is of no interest to mere softies. However, if the bit REN is not set (high), refresh cycles will not take place! Never set any other bits unless you want some wierd and wonderful (and undefined) effects. Games writers please note.....

VC Video control

Offset	Width	Name	Description
31:24	8	VR0(7:0)	Row word start address
23:16	8	VC0(7:0)	Colomn word start address
15	1	VM3	Select video mode 3 (512 words / line)
14:13	2	USK(1:0)	Row 1 skew
12:11	2	LSK(1:0)	Row 0 skew
10:8	3	VPL(2:0)	Sync polarity
7:6	2	VMX(1:0)	Sync mixing
5:4	2	VSB(1:0)	Start Bank
3:2	2	VBD(1:0)	Blank skew
1:0	2	VCD(1:0)	Column equate skew

Bit	7	6	5	4	3	2	1	0	
	VM3	USK1	USK0	LSK1	LSK0	VPL2	VPL1	VPL0	Byte 1
	VMX1	VMX0	VSB1	VSB0	VBD1	VBD0	VCD1	VCD0	Byte 0

VPL(2:0)	Sync polarity	
XX0	BLANK active low	
XX1	BLANK active high	
X0X	VSYNC active high	
X1X	VSYNC active low	
0XX	HSYNC active high	} see below
1XX	HSYNC active low	}

VMX(1:0)	Sync mixing
11	HSYNC = HS nand VS
10	HSYNC = HS eor VS
01	HSYNC = HS
00	HSYNC = HS

HS = normal Hsync position
VS = VSYNC after polarity changes defined by VPL1

HSYNC can then be inverted by setting VPL2 high

BC Blitter Control Register

Offset	Width	Name	Description
4	1	REST	Reset total (active high)
3	1	SRC	Read source area
2	1	DST	Read destination area
1	1	MSK	Read mask area
0	1	QSO	Read source preamble

Bit	7	6	5	4	3	2	1	0
	---	---	---	REST	SRC	DST	MSK	QSO

TC Test Control Register

Offset	Width	Name	Description
22	1	MUXB	
20	1	FWR	Force write
16	1	TST8	8 bit tests
15:12	4	TIN(3:0)	Invert test
11:8	4	TLE(3:0)	Test for 'less than or equal'
7:4	4	TD(3:0)	Test data
3:0	4	TV(3:0)	Test value

The blitter will concurrently process four test conditions for each destination pixel. All four tests are ALWAYS done, and all four need to be passed in order that a pixel is written.

Each test n (n=0 to 3) can be individually programmed to be one of four types:

TINn	TLEn	Test
0	0	A > B
0	1	A < B
1	0	A <= B
1	1	A >= B

The data for A and B is defined by TDn and TVn:

Tests 0 and 1:

TDn	TVn	A	B
0	0	SOURCE	TV0
0	1	SOURCE	TV1
1	X	SOURCE	DEST

Tests 2 and 3:

TDn	TVn	A	B
0	0	TV2	DEST
0	1	TV3	DEST
1	X	SOURCE	DEST

Destination in Main Ram:

TST8	MUXB	
0	0	Write if odd nibble result true
0	1	Write if even nibble result true
1	0	Write if 8 bit result true
1	1	Write if even nibble result true

Destination in video RAM, Map 0:
NB. Even nibble data is written, odd nibble data is discarded.

TST8	MUXB	
0	0	Write if odd nibble result true
0	1	Write if even nibble result true
1	0	Write if 8 bit result true
1	1	Write if even nibble result true

Destination in Video RAM, not Map 0:
NB. All 8 nibbles valid candidates for independent writing.

TST8	MUXB	
0	0	Write byte if odd nibble result true
0	1	Write nibble if aligned nibble result true
1	0	Write byte if 8 bit result true
1	1	Write odd nibble if 8 bit result true
		Write even nibble if even nibble result true

Nibble number	Main RAM	VRAM not Map0	VRAM Map 0	
7		X		
6	X	X	X	
5		X		
4	X	X	X	
3		X		
2	X	X	X	
1		X		
0	X	X	X	

TST8, when set, cascades the tests to produce an 8 bit result, on the upper odd nibble. The result on the even nibble represents the result of the even nibble test.

MUXB, when cleared, equates the results of even and odd nibbles.

PA Pixel Alignment Register

Offset	Width	Name	Description
25	1	SST	Select 2D Source shifts
24:20	5	SS(4:0)	Source shift
19:16	4	MM(3:0)	Middle mask (lines)
15:8	8	EM(7:0)	End Mask
7:0	8	SM(7:0)	Start Mask

CA Count direction Alignment

Offset	Width	Name	Description
15	1	MUPL	Mask Line count UP
14	1	MUPB	Mask Bit count UP
13	1	QD31	Destination address bit 31
12	1	QS31	Source address bit 31
11:9	3	DC(5:3)	Dest' Horizontal increment
8:6	3	DC(2:0)	Dest' Vertical increment
5:3	3	SC(5:3)	Source Horizontal increment
2:0	6	SC(2:0)	Source Vertical increment

BIT 2	BIT 1	BIT 0	Increment
0	0	0	+S
0	0	1	+(S+1)
0	1	0	-(S+1)
0	1	1	-S
1	0	0	+0
1	0	1	+1
1	1	0	-1
1	1	1	+0

S = STRIDE

CA Count direction Alignment

Offset	Width	Name	Description
15	1	MUPL	Mask Line count UP
14	1	MUPB	Mask Bit count UP
13	1	QD31	Destination address bit 31
12	1	QS31	Source address bit 31
11:9	3	DC(5:3)	Dest' Horizontal increment
8:6	3	DC(2:0)	Dest' Vertical increment
5:3	3	SC(5:3)	Source Horizontal increment

2:0	6	SC(2:0)	Source Vertical increment

BIT 2	BIT 1	BIT 0	Increment
0	0	0	+S
0	0	1	+(S+1)
0	1	0	-(S+1)
0	1	1	-S
1	0	0	+0
1	0	1	+1
1	1	0	-1
1	1	1	+0

S = STRIDE

T0 Test register 0

Offset	Width	Name	Description
23:12	12	VC(11:0)	Vertical count (line number)
11:0	12	HC(11:0)	Horizontal count (word number)

T1 Test register 1

Offset	Width	Name	Description
27:20	8	RN(7:0)	Random number (CPU ticks)
19:16	3	RP(3:0)	Refresh pending counter
15:8	8	VW(7:0)	Vertical sync width counter
7:0	8	HW(7:0)	Horizontal sync width counter

The random number RN(7:0) is actually an eight bit counter, clocked by the processor clock MCLK.

2. Pads and pins

PAD#	PAD NAME	PIN#	PIN NAME
1	CAB1	1	CSA1 (VCASA0)
3	CAB2	2	CSA2
5	CAB3	3	CSA3
7	CMB1	4	CSM1
9	CAB4	5	CSA4
11	CAB5	6	CSA5
13	CAB6	7	CSA6
14	CAB7	8	CSA7
15	CMB2	9	CSM2
16	CBB0	10	CSB0 (VCASB0)
17	CBB1	11	CSB1
18	CBB2	12	CSB2
19	CBB3	13	CSB3
20	CMB3	14	CSM3
21	CBB4	15	CSB4
22	VSSE	16	
23	VDDE	17	
24	VDD	18	
25	VSS	19	
26	CBB5	20	CSB5
27	CBB6	21	CSB6
28	CBB7	22	CSB7
29	MCLK	23	ProcClkOut
30	CCB0	24	CSC0 (VCASC0)
31	CCB1	25	CSC1
32	CCB2	26	CSC2
33	CCB3	27	CSC3
34	MSOB	28	notMemS0
35	CCB4	29	CSC4
37	CCB5	30	CSC5
39	CCB6	31	CSC6
41	CCB7	32	CSC7
43	VDD	33	
45	VSS	34	
46	WIB3	35	notMemWrB3
48	CDB0	36	CSD0 (VCASD0)
49	VSS	49	
50	CDB1	37	CSD1
50	VDD	50	
51	VDDE	51	
52	VSSE	52	
52	WIB2	38	notMemWrB2
53	TMC	53	Do not connect
54	CDB2	39	CSD2
56	CDB3	40	CSD3
57	CDB4	41	CSD4
58	CDB5	42	CSD5
59	WIB1	43	notMemWrB1

PAD#	PAD NAME	PIN#	PIN NAME
60	CDB6	44	CSD6
61	CDB7	45	CSD7
62	WIBO	46	notMemWrBO
63	ISL	47	IOSL
64	IST	48	IOST
70	ERNB	54	ERIN
71	VRNB	55	VRIN
72	ERQ	56	EREQ
73	WEB	57	IOWE
74	MGT	58	MGNT
75	AOTB	59	AOUT
76	VOR	60	VOER
77	VOL	61	VOEL
79	MRQ	62	MREQ
81	ROTB	63	ROUT
83	VOS	64	VOES
84	EA8	65	EACK
86	RINB	66	RINB
88	WVB	67	VWBO
90	VKB	68	VCLK
92	DTB	69	DTOE
94	OAD2	70	D2
97	OADO	71	DO
98	OAD1	72	D1
99	BNK	73	BLNK
100	HNC	74	HSNC
101	OAD3	75	D3
102	VNC	76	VSNC
103	OAD4	77	D4
104	VW1	78	VRW1
105	OAD5	79	D5
106	VWO	80	VRWO
107	OAD6	81	D6
108	VSSE	82	
109	VDDE	83	
110	VDD	84	
111	VSS	85	
112	OAD7	86	D7
113	OAD8	87	D8
114	OAD9	88	D9
115	OAD10	89	D10
116	OAD11	90	D11
117	OAD12	91	D12
118	RASB	92	RASB
119	OAD13	93	D13
120	OAD14	94	D14
122	WOBO	95	WBOO
124	OAD16	96	D16
125	WOB1	97	WBO1
127	OAD17	98	D17
129	OAD18	99	D18

PAD#	PAD NAME	PIN#	PIN NAME
131	WOB2	100	WBO2
133	OAD19	101	D19
134	OAD20	102	D20
136	WOB3	103	WBO3
138	OAD21	104	D21
140	OAD22	105	D22
142	AO10	106	A10
143	OAD23	107	D23
144	AO9	108	A9
145	AO8	109	A8
146	OAD24	110	D24
147	AO7	111	A7
148	AO6	112	A6
149	OAD25	113	D25
150	AO5	114	A5
151	VSS	115	
152	VDD	116	
153	VDDE	117	
154	VSSE	118	
155	AO4	119	A4
156	OAD26	120	D26
157	AO3	121	A3
158	AO2	122	A2
159	OAD27	123	D27
160	AO1	124	A1
161	AOO	125	AO
162	OAD28	126	D28
163	OAD29	127	D29
164	OAD30	128	D30
166	OAD31	129	D31
167	OAD15	130	D15
169	CMBO	131	CSMO
171	CABO	132	CSAO

3 Pins used for external memory control

The pins used by EMC are as follows:

Pin(s)	I/O	Description
AD(31:0)	I/O	CPU multiplexed address / data
MSO	I	Address strobe (T800 = NotMemSO)
MCLK	I	Master clock
RAS	O	RAS
CAS(3:0)	O	Main RAM CAS lines
WBO(3:0)	O	Main RAM byte write lines
VCAS(31:0)	O	Video RAM CAS lines (one per chip)
WBOV	O	Video RAM write line
VOEL	O	Video RAM write left side

VOER	0	Video RAM write right side
VOES	0	Video RAM write through
DTOE	0	Video RAM Data transfer / output enable
IOST	0	I/O Strobe
IOSL	0	I/O Select
WE	0	I/O Write enable

Appendix 3

Video setup

Calling the routine Setup_video(mode,monitor) will setup the video registers for the modes and monitors described in the array crt[8][9]. Some of these 'magic' numbers have not been fully validated (March 1988). However, all the ones for the NEC multisync plus (NEC_PL) should work.

The header files follow at the end of the code.

```
WORD v_mode[NVM_MODES] = {0x02,0x3C,0x36,0x11);
/* CLUT enables */

WORD crt[8][9] =  {
/* Hitachi mode 0 */ { 208,158,160,16,1100,958,1000,4,3},
/* NEC xl   mode 1 */ { 320,254,268,21,797,766,774,4,0x10000000},
/* NEC pl   mode 1 */ { 352,254,284,21,896,766,824,4,3},
/* NEC xl   mode 1 */ { 320,254,268,21,797,766,774,4,0x00000000},
/* NEC pl   mode 2 */ { 210,158,176,20,544,478,500,3,0},
/* NEC xl   mode 2 */ { 210,158,176,20,544,478,500,3,0},
/* NEC pl   mode 3 */ { 800,511,616,88,544,478,500,3,0x00008005},
/* NEC xl   mode 3 */ { 800,511,616,88,544,478,500,3,0x00008005},
                   };

setupvreg(index)
WORD index;
{
WORD i;
WORD *x =       VREG_BASE;
WORD *ptr1 =    0x80000800;
WORD *ptr2 =    0x80000804;

        for (i=0; i <= NUM_VREGS; i++) {

/*              *x++  =  crt[index][i] ; */
                if (*ptr1 == 0)
                        *ptr2  =  crt[index][i] ;
                *x++ = *ptr2++;
        }
}

void setup_video(mode,monitor)
WORD mode;      /* screen mode 0 - 3 */
WORD monitor;   /* monitor supplied 0 - 2 */
{
WORD *px;       /* pointer for register stuffing */

        if (mode < 0 || mode >= NUM_MODES) return;

        switch (mode) {
```

```
            case 0:         setupvreg(MODE0);
                            break;

            case 1: switch (monitor) {

                      case HITACHI: setupvreg(MODE1_HITACHI);
                                    break;
                      case NEC_PL:  setupvreg(MODE1_NEC_PL);
                                    break;
                      case NEC_XL:  setupvreg(MODE1_NEC_XL);
                                    break;
                      default:      /* error */
                                    break;
                    }
                    break;

            case 2: switch (monitor) {

                      case NEC_PL:  setupvreg(MODE2_NEC_PL);
                                    break;
                      case NEC_XL:  setupvreg(MODE2_NEC_XL);
                                    break;
                      default:      /* error */
                                    break;
                    }
                    break;

            case 3: switch (monitor) {

                      case NEC_PL:  setupvreg(MODE3_NEC_PL);
                                    break;
                      case NEC_XL:  setupvreg(MODE3_NEC_XL);
                                    break;
                      default:      /* error */
                                    break;
                    }
                    break;

            default: /*error */
                    break;
        }

        px = MC_REG;
        *px = 4;                    /* Enable video */
        px = VMODE;
        *px = v_mode[mode];         /* Enable the CLUT */
}
```

```
/*****************************************************************
        Video.h

        Charity version for video

        RGM 30/12/87
*****************************************************************/

#define NUM_MODES       4

#define MAP0    0x00000000
#define MAP1    0x00200000
#define MAP2    0x00400000
#define MAP3    0x00600000

/*****************************************************************
                Video board addresses and values
*****************************************************************/

#define VMODE 0x42400000        /* Video mode register */

/* BT453 is Brooktree CLUT for modes 1 and 2 (8 bits / pixel) */

#define C453WADDR 0x42D00000    /* BT453 write addresses */
#define C453WDATA 0x42D02000    /* BT453 write colour data */
#define C453RADDR 0x42500000    /* BT453 read addresses */
#define C453RDATA 0x42502000    /* BT453 read colour data */

/* BT454 is Brooktree CLUT for mode 0 (4 bits / pixel) */

#define C454WADDR 0x42580000    /* BT454 write addresses */
#define C454WDATA 0x42582000    /* BT454 write colour data */

/*****************************************************************
                Magic numbers for monitors and modes
*****************************************************************/

#define HITACHI         0
#define NEC_PL          1
#define NEC_XL          2

#define MODE0           0
#define MODE1_HITACHI   1
#define MODE1_NEC_PL    2
#define MODE1_NEC_XL    3
#define MODE2_NEC_PL    4
#define MODE2_NEC_XL    5
#define MODE3_NEC_PL    6
#define MODE3_NEC_XL    7
```

```
/*****************************************************************
        Charity.h

        Charity registers

        RGM 30/12/87
*****************************************************************/

#define NUM_VREGS 9             /* Number of video registers */
#define VREG_BASE 0x40000040    /* Base of video registers */

/* !! N/A !! means that this register is not implemented on
   Charity1 */

#define MC_REG 0x40000000       /* Master control */
#define EA_REG 0x40000004       /* Event line address */
#define TS_REG 0x40000008       /* Test (leave alone!) */

#define HT_REG 0x40000040
#define HD_REG 0x40000044
#define HS_REG 0x40000048
#define HW_REG 0x4000004C

#define VT_REG 0x40000050
#define VD_REG 0x40000054
#define VS_REG 0x40000058
#define VW_REG 0x4000005C
#define VC_REG 0x40000060

#define BC_REG 0x40000080       /* Blit control */
#define TC_REG 0x40000084       /* Test control */
#define PA_REG 0x40000088       /* Pixel alignment */
#define CA_REG 0x4000008C       /* Count direction */

#define DT_REG 0x40000090       /* Destination total */
#define DB_REG 0x40000094       /* Destination base address */
#define DW_REG 0x40000098       /* Destination width */
#define DS_REG 0x4000009C       /* Destination stride */

#define SB_REG 0x400000A0       /* Source base address */
#define SS_REG 0x400000A4       /* Source stride */
#define ML_REG 0x400000A8       /* Mask line start !! N/A !! */
#define MB_REG 0x400000AC       /* Mask bit start  !! N/A !! */

#define PC_REG 0x400000B0       /* Pipe control */
#define CD_REG 0x400000B4       /* Colour data     !! N/A !! */
#define TV_REG 0x400000B8       /* Test values */
#define SD_REG 0x400000BC       /* Source data */

#define MD_REG 0x40000020       /* Mask data       !! N/A !! */
#define DD_REG 0x40000024       /* Destination data */

/*****************************************************************/
```

Appendix 4 Issue 3 ABAQ motherboard connector information.

1. The Expansion Bus

The expansion bus connector on Issue 3 boards is a 100-way 0.1" edge connector, carrying signals as listed below. It is referred to as J8 in ABAQ diagrams. A standard-issue expansion backplane is normally fitted into J8; this board carries the four expansion slots, each of which is a 124-way 0.1" edge connector.

The expansion slots on the backplane, referred to as J101 (bottom) to J104 (top), are almost identical in pinout. The differences are that each of the bottom three slots receives a different set of four CAS lines on pins 6, 8, 69 and 71. Consult the chapter on the memory map for the addresses at which the different slot CAS lines become active.

In addition, signals MGIN and MGOUT (pins 105, 106) and EVTACKIN and EVTACKOUT (pins 112, 113) are connected in a daisy-chain, with J101 having the highest priority in both cases. Note that although the expansion daughterboard carries these signals between edge connectors, they do not connect into the motherboard circuitry on Issue 3 boards.

Fig 1 ABAQ Issue 3 motherboard 100-way edge connector J8

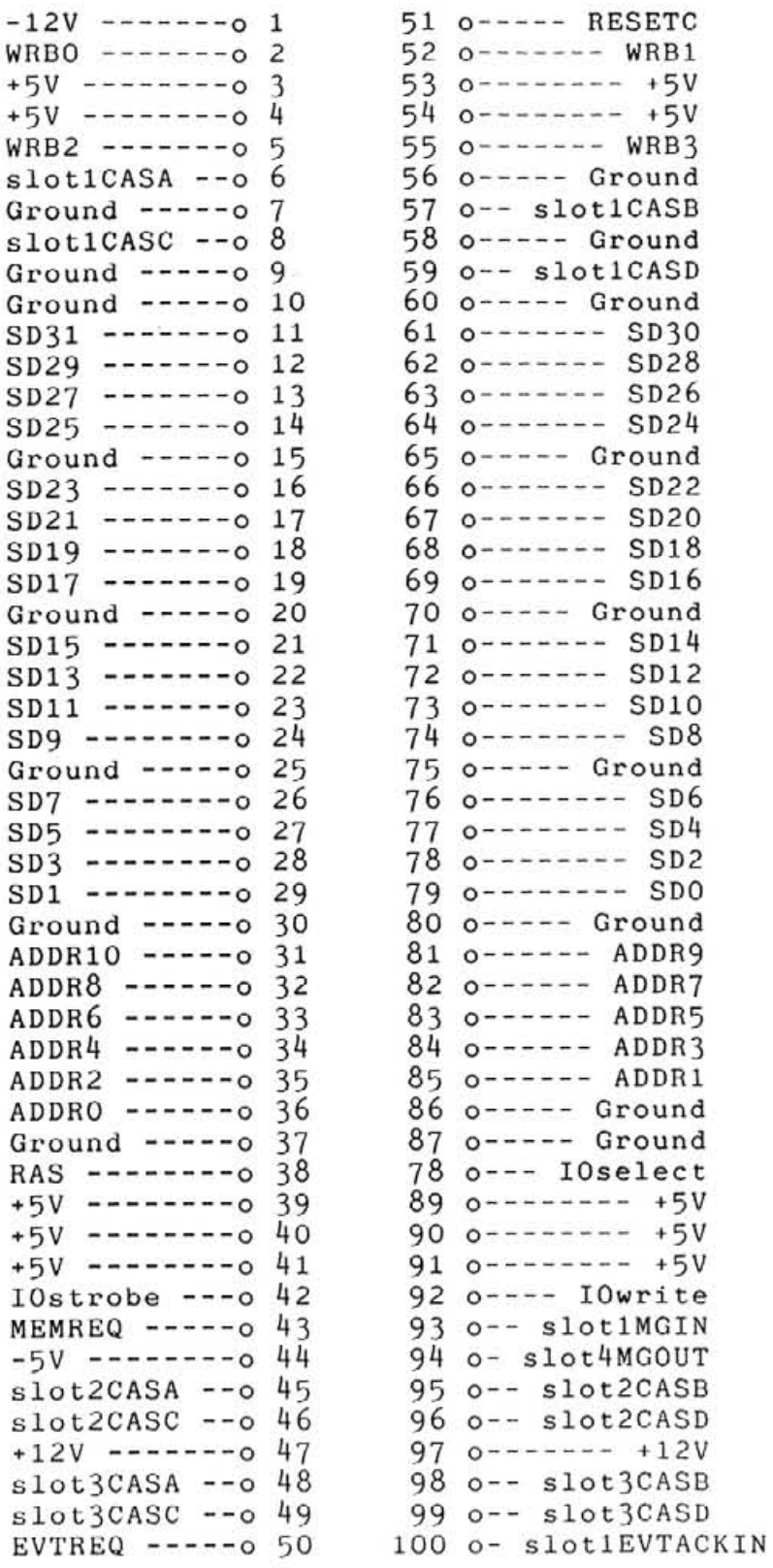


Fig 1.2 ABAQ Expansion plane 124-way connectors: J101-104

	-12V	1	63	RESETC	
	WRB0	2	64	WRB1	
	+5V	3	65	+5V	
	+5V	4	66	+5V	
	WRB2	5	67	WRB3	
*	CASA	6	68	Ground	
	Ground	7	69	CASB	*
*	CASC	8	70	Ground	
	Ground	9	71	CASD	*
	Ground	10	72	Ground	
	SD31	11	73	SD30	
	SD29	12	74	SD28	
	SD27	13	75	SD26	
	SD25	14	76	SD24	
	Ground	15	77	Ground	
	SD23	16	78	SD22	
	SD21	17	79	SD20	
	SD19	18	80	SD18	
	SD17	19	81	SD16	
	Ground	20	82	Ground	
	SD15	21	83	SD14	
	SD13	22	84	SD12	
	SD11	23	85	SD10	
	SD9	24	86	SD8	
	Ground	25	87	Ground	
	SD7	26	88	SD6	
	SD5	27	89	SD4	
	SD3	28	90	SD2	
	SD1	29	91	SD0	
	Ground	30	92	Ground	
	ADDR10	31	93	ADDR9	
	ADDR8	32	94	ADDR7	
	ADDR6	33	95	ADDR5	
	ADDR4	34	96	ADDR3	
	ADDR2	35	97	ADDR1	
	ADDR0	36	98	Ground	
	Ground	37	99	Ground	
	RAS	38	100	IOselect	
	+5V	39	101	+5V	
	+5V	40	102	+5V	
	+5V	41	103	+5V	
	IOstrobe	42	104	IOwrite	
	MEMREQ	43	105	MGIN	**
	-5V	44	106	MGOUT	**
	n.c.	45	107	n.c.	
	n.c.	46	108	n.c.	
	+12V	47	109	+12V	
	n.c.	48	110	n.c.	
	n.c.	49	111	n.c.	

Expansion plane connectors (continued)

```
EVTREQ -----o 50    112 o--- EVTACKIN  **
Ground -----o 51    113 o-- EVTACKOUT  **
MWAIT ------o 52    114 o--------- WA
Ground -----o 53    115 o--------- WB
BCLK -------o 54    116 o----- Ground
Ground -----o 55    117 o--------- WC
NMSO -------o 56    118 o--------- WD
Ground -----o 57    119 o----- spare3
spare1 -----o 58    120 o----- spare2
+5V --------o 59    121 o-------- +5V
+5V --------o 60    122 o-------- +5V
+5V --------o 61    123 o-------- +5V
+5V --------o 62    124 o-------- +5V
```

* = not connected on J104. On J101 these pins are connected to signals slot1CASA - slot1CASD, on J102 to slot2CASA - slotCASD, on J103 to slot3CASA - slot3CASD

2. Link Connectors

2.1 The Link Configuration Header

The four Inmos links of the ABAQ CPU are connected to jumper area J7 on the ABAQ motherboard. There are three rows of 24 pins concerned with the links - J71A, J71B and J72A. J72B has a collection of control signals designed for controlling extra transputer add-in cards. The pinout of all 4 rows is given below.

J71B consists of four sets of six pins, of which two are grounded and two others are directly connected to the LinkIn and LinkOut pins on the T800 CPU. It is possible to communicate with the transputer by plugging a link cable onto this header, but two other options are also offered, and these have the advantage of signal buffering.

2.2 ECL-buffered Links

If a pair of jumpers is used to connect a pair of link signals on J71B to the adjacent pins of J72A (e.g. LOUT0 to ECLin0, and LIN0 to ECLout0), then these link signals are connected to 10H124 (outputs) and 10H125 (inputs) ECL/TTL buffers. Communication with the ABAQ transputer would then be via 9-way 'D' connectors J4 and J5, where the ECL-level signals are available in true and complement polarity. The pinouts for J4 and J5 are given below.

N.B. The Issue 3 motherboard as shipped does not have the ECL buffer chips fitted. If required, insert a 10H124 in the 16-pin IC socket labelled U120 and a 10H125 in the socket marked U121. These sockets are located directly behind J4 - J6d on the Issue 3 motherboard.

2.3 TTL-buffered Links

If a pair of jumpers is used to connect a pair of link signals on J71B to the adjacent pins of J71A (e.g. LOUT0 to TTLin0, and LIN0 to TTLout0), then these link signals are connected to a 74F244 TTL buffer. Communication with the ABAQ transputer would then be via the four mini-DIN connectors J6a - J6d. J6b, J6c and J6d are four-pin mini-DINs, and each carries the TTL-buffered signals required for one link connection. J6a is an 8-way mini-DIN, and carries the two grounds and two buffered signals comprising link 0, and also the Reset and Analyse control signals from the ST.

Fig 2.1 ABAQ Issue 3 Link Configuration Header J7

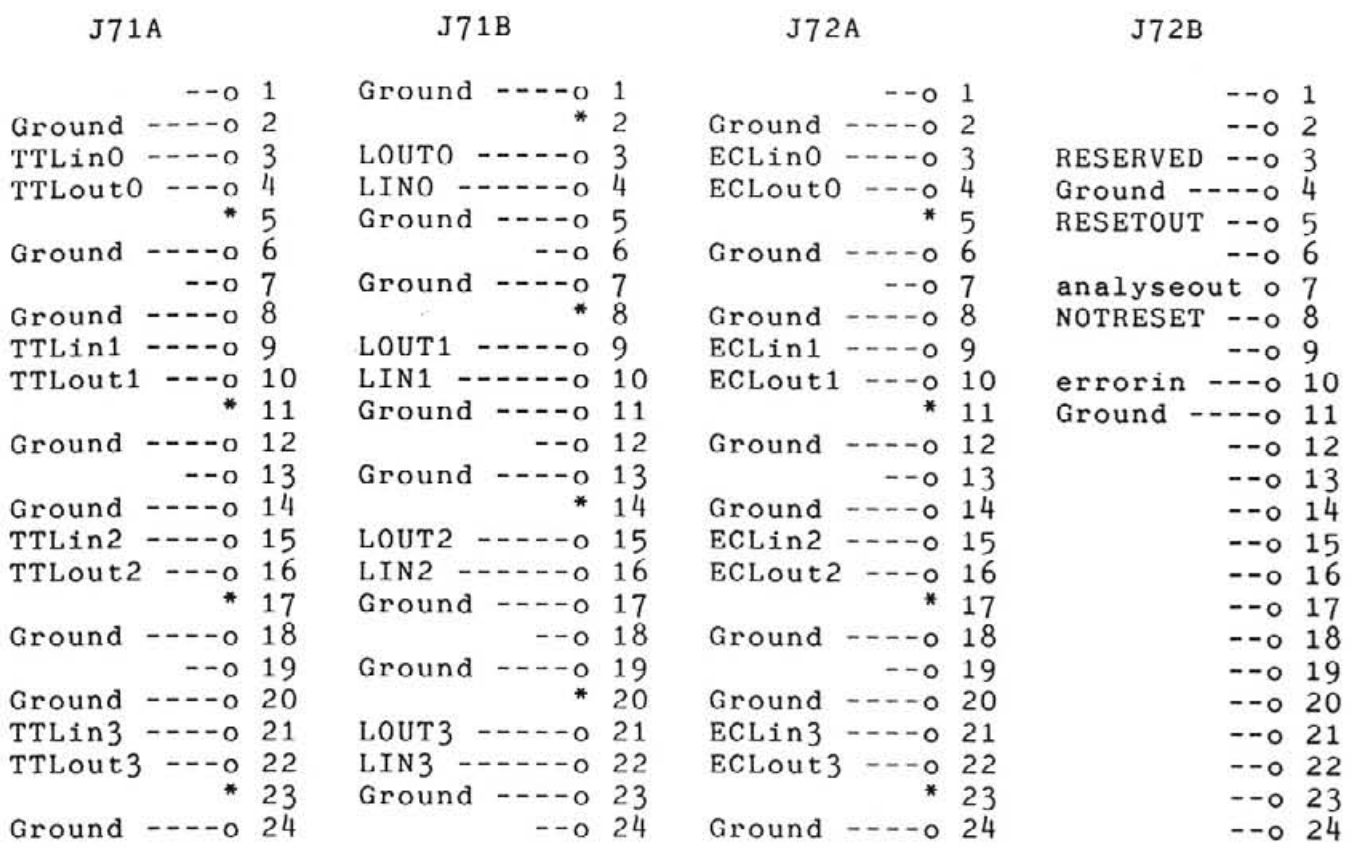
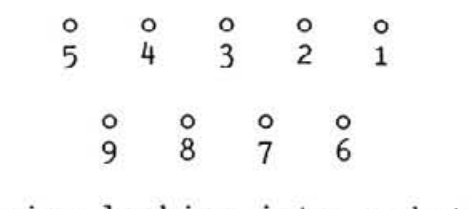


Fig 2.2 ABAQ Issue 3 ECL-buffered Link Connectors

J4 (9-way 'D' female socket)

1	Ground
2	ECL0out+
3	ECL0in+
4	ECL1out+
5	ECL1in+
6	ECL0out-
7	ECL0in-
8	ECL1out-
9	ECLin1-



J5 (9-way 'D' female socket)

1	Ground
2	ECL2out+
3	ECL2in+
4	ECL3out+
5	ECL3in+
6	ECL2out-
7	ECL2in-
8	ECL3out-
9	ECL3in-

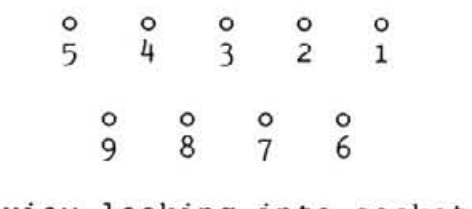
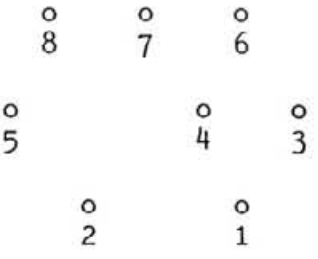


Fig 2.3 ABAQ Issue 3 TTL-buffered Link Connectors

J6a (8-pin mini-DIN)

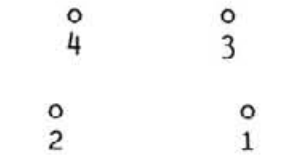
1	XSRST
2	XHRST
3	TTL0out
4	analysein
5	TTL0in
6	Ground
7	errorout
8	Ground



view looking into socket

J6b (4-pin mini-DIN)

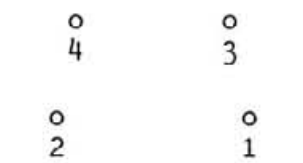
1	TTL1out
2	TTl1in
3	Ground
4	Ground



view looking into socket

J6c (4-pin mini-DIN)

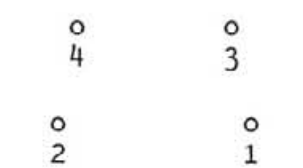
1	TTL2out
2	TTl2in
3	Ground
4	Ground



view looking into socket

J6d (4-pin mini-DIN)

1	TTL3out
2	TTl3in
3	Ground
4	Ground

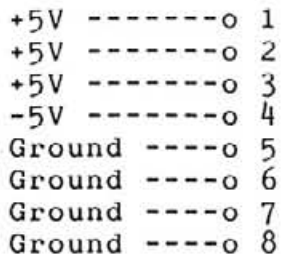


view looking into socket

Factory Settings: jumper J71A p3 - J71B p3
J71A p4 - J71B p4
J71B p9 - J71B p10
J71B p15 - J71B p16
J71B p21 - J71B p22

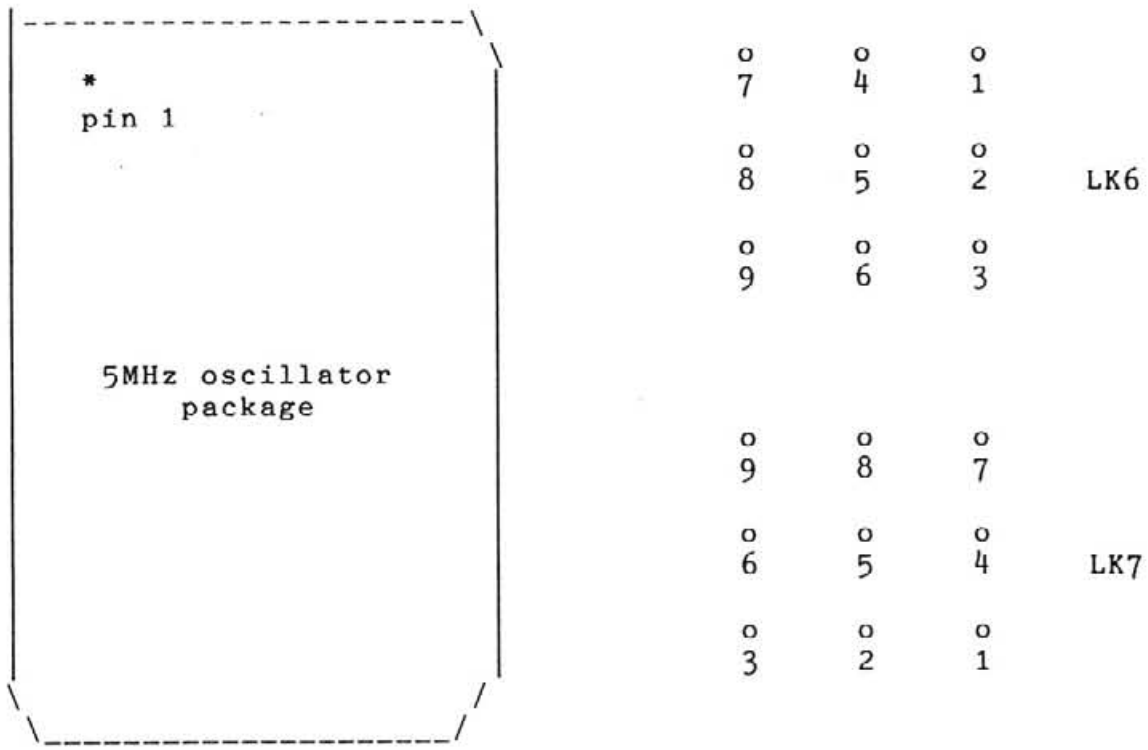
3. Power Input Connector

The power input connector J11 is an eight-pin 0.15" header. Note that there are three unused holes in the board at the pin 8 end of J11.



4. T800 Configuration Links

The two jumper blocks LK6 and LK7 beside the 5MHz xtal oscillator package are used to configure processor clock speed (LK6) and link speed (LK7). The jumpers on LK6 should not be changed from the Issue 3 factory setting.



LK6 (Processor Speed Select)

1	+5V
2	+5V
3	+5V
4	ProcSpeedSelect0
5	ProcSpeedSelect1
6	ProcSpeedSelect2
7	Ground
8	Ground
9	Ground

LK7 (Link Speed Select)

1	+5V
2	+5V
3	+5V
4	LinkSpecial
5	Link0Special
6	Link123Special
7	Ground
8	Ground
9	Ground

Refer to the T800 data sheet for the meaning and uses of these options. Factory settings are as follows:

Clock speed 17.5MHz:	jumper LK6	p4 - p7
		p5 - p2
		p6 - p3
Link speed 10MHz:	jumper LK7	p4 - p7
		p5 - p8
		p6 - p9

5. The Video Daughterboard Connectors

Pinouts of the video daughterboard connectors are shown below. Both connectors are 96-way DIN41612 female sockets.

Video Connector J1

J1A	J1B	J1C
pixbit0 ---o 1	-5V -------o 1	-5V -------o 1
pixbit1 ---o 2	-5V -------o 2	-5V -------o 2
pixbit2 ---o 3	-5V -------o 3	-5V -------o 3
pixbit3 ---o 4	-5V -------o 4	-5V -------o 4
pixbit4 ---o 5	-5V -------o 5	-5V -------o 5
pixbit5 ---o 6	--o 6	blank -----o 6
pixbit6 ---o 7	Ground ----o 7	Ground ----o 7
pixbit7 ---o 8	Ground ----o 8	hsync -----o 8
pixbit8 ---o 9	Ground ----o 9	Ground ----o 9
pixbit9 ---o 10	Ground ----o 10	vsync -----o 10
pixbit10 --o 11	Ground ----o 11	Ground ----o 11
pixbit11 --o 12	Ground ----o 12	pixoutclk -o 12
pixbit12 --o 13	Ground ----o 13	Ground ----o 13
pixbit13 --o 14	Ground ----o 14	syncscan --o 14
pixbit14 --o 15	Ground ----o 15	Ground ----o 15
pixbit15 --o 16	Ground ----o 16	wordclk ---o 16
pixbit16 --o 17	Ground ----o 17	Ground ----o 17
pixbit17 --o 18	Ground ----o 18	--o 18
pixbit18 --o 19	Ground ----o 19	--o 19
pixbit19 --o 20	Ground ----o 20	--o 20
pixbit20 --o 21	Ground ----o 21	+5V -------o 21
pixbit21 --o 22	Ground ----o 22	+5V -------o 22
pixbit22 --o 23	Ground ----o 23	+5V -------o 23
pixbit23 --o 24	Ground ----o 24	+5V -------o 24
pixbit24 --o 25	Ground ----o 25	+5V -------o 25
pixbit25 --o 26	Ground ----o 26	--o 26
pixbit26 --o 27	Ground ----o 27	--o 27
pixbit27 --o 28	--o 28	--o 28
pixbit28 --o 29	+5V -------o 29	+5V -------o 29
pixbit29 --o 30	+5V -------o 30	+5V -------o 30
pixbit30 --o 31	+5V -------o 31	+5V -------o 31
pixbit31 --o 32	+5V -------o 32	+5V -------o 32

Video Connector J3

J3A	J3B	J3C
Ground ----o 1	SD7 -------o 1	Ground ----o 1
Ground ----o 2	SD6 -------o 2	Ground ----o 2
Ground ----o 3	SD5 -------o 3	Ground ----o 3
Ground ----o 4	SD4 -------o 4	Ground ----o 4
Ground ----o 5	SD3 -------o 5	Ground ----o 5
Ground ----o 6	SD2 -------o 6	Ground ----o 6
Ground ----o 7	SD1 -------o 7	Ground ----o 7
Ground ----o 8	SD0 -------o 8	Ground ----o 8
Ground ----o 9	Ground ----o 9	Ground ----o 9
IOwrite ---o 10	--o 10	IOselect --o 10
+5V -------o 11	+5V -------o 11	+5V -------o 11
+5V -------o 12	+5V -------o 12	+5V -------o 12
--o 13	IOstrobe --o 13	--o 13
-5V -------o 14	-5V -------o 14	-5V -------o 14
-5V -------o 15	-5V -------o 15	-5V -------o 15
NOTRESET --o 16	--o 16	--o 16
+5V -------o 17	+5V -------o 17	+5V -------o 17
+5V -------o 18	+5V -------o 18	+5V -------o 18
--o 19	--o 19	--o 19
Ground ----o 20	ADDR0 -----o 20	Ground ----o 20
Ground ----o 21	ADDR1 -----o 21	Ground ----o 21
Ground ----o 22	ADDR2 -----o 22	Ground ----o 22
Ground ----o 23	ADDR3 -----o 23	Ground ----o 23
Ground ----o 24	ADDR4 -----o 24	Ground ----o 24
Ground ----o 25	ADDR5 -----o 25	Ground ----o 25
Ground ----o 26	ADDR6 -----o 26	Ground ----o 26
Ground ----o 27	ADDR7 -----o 27	Ground ----o 27
Ground ----o 28	Ground ----o 28	Ground ----o 28
Ground ----o 29	Ground ----o 29	Ground ----o 29
Ground ----o 30	Ground ----o 30	Ground ----o 30
Ground ----o 31	Ground ----o 31	Ground ----o 31
Ground ----o 32	Ground ----o 32	Ground ----o 32

6. The Expansion Memory Addressing Jumpers

Jumper areas LK9 - LK12 are provided to allow the four decoded CAS lines to be routed either to expansion slots J101-103 or to the 4Mbytes of on board main RAM (CASA only).

The pinouts of these links are given below. Note that LK9 is incorrectly laid out on the Issue 3 board, producing a slightly nonsensical arrangement of pins.

```
LK12:   slot1CASD --o 1   3 o---)
        slot3CASD --o 2   3 o---)--- CASD
        slot2CASD --o 4   3 o---)

LK11:   slot1CASC --o 1   3 o---)
        slot3CASC --o 2   3 o---)--- CASC
        slot2CASC --o 4   3 o---)

LK10:   slot1CASB --o 1   3 o---)
        slot3CASB --o 2   3 o---)--- CASB
        slot2CASB --o 4   3 o---)

LK9:    slot3CASA --o 2   1 o---)
        CASA -------o 3   1 o---)
        slot2CASA --o 4   1 o---)--- slot1CASA
        MRCAS ------o 5   1 o---)
```

Factory setting is for CASA to select the on-board RAM. This places the on-board RAM at #80000000 (excluding space overlayed by T800 internal RAM).

Issue 3 factory settings: jumper LK9 p3 - p1
p5 - p1

LK10-12 not jumpered.

Appendix 6 Memory timing requirements and expansion memory

1 User RAM

1.2 Timing specification

The memory interface of ABAQ will cycle at 250 nS, requiring 120nS access time parts. Static column mode is not used.

Main RAM timing requirements:

Parameter	Symbol	MIN	MAX	Units
Access time from RAS	tRAC		120	ns
Access time from CAS	tCAC		60	ns
Refresh period	tREF		8	ms
Random read / write cycle time	tRC	230		ns
Operating current	Icc1		50	mA
Input Capacitance (Address, data)	Cin		5	pF
Output Capacitance (Data)	Cout		7	pF

1.3 Expansion of main RAM

Four CAS lines are provided for main memory, only one of which is used on the mother–board. Extra RAM can be added, for which three CAS lines are available. The buffering provided on the main board is sufficient for three further banks on the data lines, although the address lines will have to be buffered again for each bank of 32 chips. The specification for expansion RAM is otherwise the same as for on–board RAM.

The following table summarises the total RAM possibilities with this system.

	1 Mbit parts	4 Mbit parts
Standard system	4 Mbytes	16 Mbytes
Fully expanded system	16 Mbytes	64 Mbytes

1.4 Refresh

In anticipation of larger DRAM parts, 10 bit refresh is provided, cycling at least every 16 mS.

This is equivalent to 512 refresh cycles every 8 mS, or 1024 cycles every 16 mS. RAS only refresh is used, with sequential row addresses (ie not hidden refresh).

2 Timing specifications for video RAM

The timing requirements for the random access port of the video RAM are at least that of the user RAM. The Hitachi part (HM53461P–12) appears to satisfy these requirements. It is important that the SAM port has a static RAM shift register; in other words, that there is no minimum cycle time for the video port.

Timing requirements for video RAM:

Parameter	Symbol	MIN	MAX	Units
Access time from RAS	tRAC		120	ns
Access time from CAS	tCAC		60	ns
Refresh period	tREF		4	ms
Random read / write cycle time	tRC	230		ns
Serial clock cycle time	tSCC	40		ns
Operating current	Icc2		100	mA
Input Capacitance (Address, data)	Cin		5	pF
Output Capacitance (Data)	Cout		7	pF

Appendix 7 Link Speed Selection

The Atari Transputer Workstation issue 3 has its link speed set at the default of 10 Mbits/sec. This default was chosen because all Transputer devices will work at 10 Mbits/sec.

Section Appendix 4–10 and Appendix A–2 give details on how to change the link speed on the Atari Transputer Workstation motherboard.

Details follow on how to change the speed selection on the SCSI/Link I/O PCB:

1. Cut track to IC19 pin 16 (TOPSIDE).
2. Fit 3 pin header to IC38 pins 13,14 and 15.
3. Link IC19 pin 16 to IC38 pin 14
4. Link IC19 pin 15 to IC 38 pin 13
5. Link IC11 pin 14 to IC38 pin 15

This modification will allow link speed jumper selectable on SCSI/Link PCB.

For 10 mbits/sec fit jumper IC38 pin 14 to pin 15
For 20 mbits/sec fit jumper IC38 pin 14 to pin 13

Appendix A: Atari Abaq Development System Start Up

The Atari Abaq Iss3 Development System requires the following hardware:

1 Mega ST (Mega 2 or Mega 4)
1 SH205
1 SM124 1 Monitor for Abaq (eg NEC multisync plus)

1. With the Abaq disconnected from mains supply, remove top cover by removing four screws in side of case.

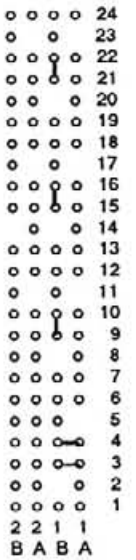
2. Identify the Video PCB below the PSU unit and ensure that the video PCB is seated correctly

3. Ensure the PSU voltage settings on the PSU side is set for your local voltage.

4. Ensure that the Video cables are connected to the video output of your choice. Factory selection hasvideo cables connected to Mode 1. Additional video cables can be manufactured or purchased direct from Atari UK.

5. Ensure the following jumpers are present on the mother PCB:

5.1 Links terminated on J7 as follows:



J7

Connects LOUT3 to LIN3

Connects LOUT2 to LIN2

Connects LOUT1 to LIN1

Connects TTLout0 to LIN0
Connects TTLin0 to LOUT0

For a full description of this patch panel please refer to Hardware Manual Appendix 4 section 2.

5.2 Expansion Memory Addressing Jumpers:

5.3 Clock speed select Jumper:

o–o o
o–o o LK6
o o–o

For a full description of this patch panel please refer to Hardware Manual Appendix 4 section 4.

5.3 Link speed select Jumper:

o o o
o o o LK7
o o o

For a full description of this patch panel please refer to Hardware Manual Appendix 4 section 4.

6. Ensure the molex power connector J11 is fitted correctly.

7. Replace Cover on Abaq

8. Fit the SCSI/Link adaptor card into a Mega ST as follows:

8.1 Ensure Mega ST disconnected from mains and remove top lid by removing 10 screws in base of unit.

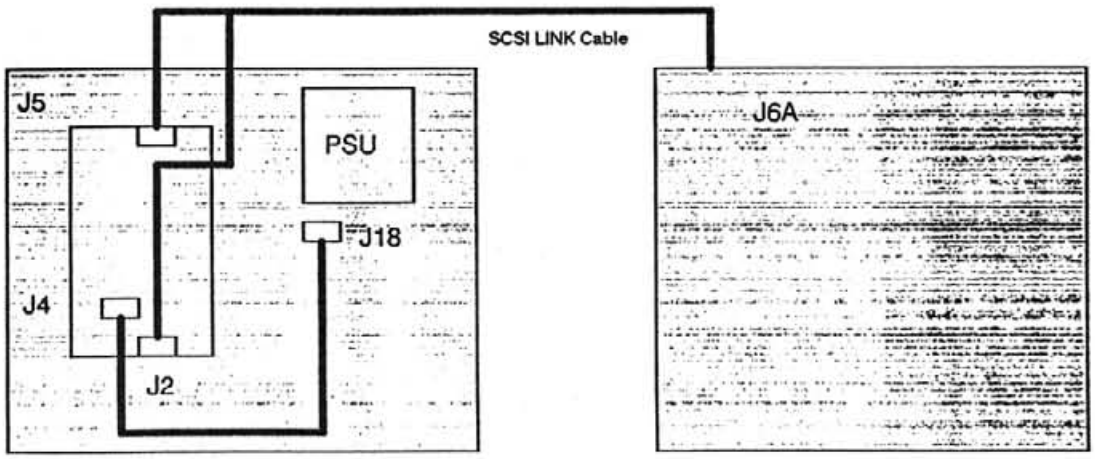
8.2 Remove internal shielding.

8.3 Fit SCSI/Link adaptor card into the internal expansion bus in Mega ST.

8.4 Identify SCSI poer lead and connect as shown in Fig App A.1

8.5 Identify Reset/Analyse/Link cable and shown in Fig App A.1

8.6 Refit Mega Lid. Shielding will not fit. US users should take FCC precautions.



Mega ST ABAQ–3

9. Connect up SH205 hard disc to Mega ST

10. Connect SM124 monitor to Mega ST

11. Plug Reset/Analyse/Link cable into J6a 8 pin socket in Abaq.

12. Apply power to Mega St system and format Hard Disc into 4 partitions C D E and F, each 5 megabytes.

13. Create a folder called AUTO on C.

Copy INSTALL.PRG from Helios distribution disc into AUTO folder.

14. If you have been supplied with the HELIOS distribution discs create a folder called HELIOS on C.
Copy contents of Helios distribution disc into HELIOS folder.

Helios should be obtained direct from Perihelion Software.

15. If you have been supplied with the BOUNCE demonstration discs create a folder called BOUNCE on D.
Copy contents of BOUNCE distribution discs into BOUNCE folder.

16. If you have been supplied with the DIAGNOSTIC floppy disc create a folder called DIAGNOSE on E.
Copy contents of the DIAGNOSTIC distribution discs into DIAGNOSE folder.

Appendix B Atari Transputer Development System Contents

The Atari Transputer Development System consists of the following:

1 Atari Abaq consisting of:

- 4 Mbyte main RAM
- 1 Mbyte video RAM
- 1 T800–20
- 1 Blitter
- 1 Video subsystem

1 SCSI/Link Interface PCB
1 set RGB cables 1 set Reset/Analyse/Link cables

1 Abaq Hardware Manual
1 Diagnostic Disc

Notes:

1. The following additional equipment is required to use the Atari Transputer Development System:

   1 Mega ST (Mega 2 or Mega 4)
   1 ST monitor (colour or monochrome) (SM124/SC1224)
   1 ST hard disc (SH205)
   1 Abaq Monitor (eg NEC Multisync Plus/XL)

2. The Helios operating system should be obtained directly from Perihelion Software Ltd:

   Perihelion Software Ltd
   24 Brewmaster Buildings
   Charlton Trading Estate
   Shepton Mallet
   Somerset
   BA4 5QE
   UK

   Tel: 0749 4203
   Fax: 0749 4977

3. Additional Farm Cards containing 4 transputers and 4 Mbytes RAM can be obtained direct from Perihelion Hardware Ltd:

   Perihelion Hardware Ltd
   33 Bridge St
   Cambridge
   CB2 1UW
   UK

   Tel: 0223 356555
   Fax: 0223 311475

4. The current version of the Atari Transputer Development System is running at 17.5 Mhz. This will be increased to 20 Mhz when the new blitter is avaialable.

5. The current version of the blitter (known as charity I) is being upgraded (to Charity II). If you intend to use the blitter please be aware that there are likely to be changes between the two revisions. I will confirm the changes as soon as they are known.

6. The Atari Transputer Development System is still in its development stage and should be treated as such.

Appendix C Atari Transputer System Developers Price List

Effective May 17th 1988 Ref:LP805172

The following prices show both the recommended retail prices (**RRP**) and the recomeneded developer price (**RDP**) after **20%** discount.

All prices exclude VAT.

Product	Description	RRP	RDP
ABAQ	Atari Transputer Development System Iss 3 4 M RAM. 1M Video RAM 1 T800 1 ST I/O Card	**2999.99**	**2399.99**
Mega ST2 MONO	2 Meg ST with High res Mono monitor, mouse	**899.99**	**719.99**
Mega ST 4 Mono	4 Meg ST with High res Monitor	**1199.99**	**959.99**
SH205	20 Meg Hard Disc	**521.73**	**417.38**
SLM804	Atari Laser Printer	**1199.99**	**959.99**
SF314	1Mb double sided disk drive	**139.99**	**111.30**
NEC MS PLUS	NEC Multisync Plus	**900.00**	**720.00**

Software

Helios Helios operating system Only available direct from Perihelion Software Ltd		**500.00**

The following software is for the Atari ST and may be useful to developers considering writing software for the Atari ST.

SSW 1202–001	ST Software Developers Toolkit	**295.00**
SSW 1300–001	GDOS Development Toolkit	**8.69**
TH 9304–001	Mark Williams C ST Development Toolkit	**99.00**

1. All prices are calculated on a payment with order basis. Price includes carriage within the UK.

Appendix D ABAQ DIAGNOSTICS

Pre Release June 13th 1988

This diagnostic can either be run from floppy disc or hard disc using COMMAND.TOS.

1. RUN INSTFAB3.PRG to initialise the Mega ST.
2. RUN COMMAND.TOS and wait for prompt.
3. ENTER test xtest
4. Diagnostic will load and offer a menu.
5. ENTER reset
   This will issue a soft reset to the Abaq.
6. Enter /
   This will now allow you to setup address for test numbers (sets current address to 8000 0900)
7. ENTER =
   This will allow you to enter test numbers

Bit	0	Video ON (Always required)
	1	Main RAM Test
	2	Video RAM Test
	3	VRAM scope loop
	4	Main RAM scope loop
	5	Transputer Blit Test
	6	Charity Blit Test
	7	Mode 3 Demo
	8	Draw Border
	9	Draw Stripes
	10	0
	11	0
	12	0
	13	0
	14	0
	15	0

   eg 207 (hex) will Draw stripes. test video and main RAM with Video ON.

8. ENTER GO
   GO command will run tests specified.
9. Tests are interrupted by ESC
   Commands Available:

QUIT	Back to Command.TOS
ANALYSE	Not yet implemented
BASE	Not yet implemented
BYTES	Not yet implemented
CMP	Not yet implemented
GO	Begin Testing
LOAD	Not yet implemented
RESET	Initialise communication link to Abaq
SYMBOLS	Not yet implemented
TEST	Variety of tests
TRACE	Not yet implemented
WORDS	Not yet implemented
FARM	Not yet implemented