FLARE ONE

The Flare One Computer System

Hardware Reference Guide

Version 1.05
20 February, 1989 3:12pm

by Martin Brennan, Ben Cheese and John Mathieson,
Copyright 1988, Flare Technolcgy Limited

Flare Technology Limited,
Unit O,
The Paddocks Trading Estate,
347 Cherry Hinton Road,
Cambridge CB1 4DH.

Copyright 1988

SYSTEM OVERVIEW 5

1. INTRODUCTION 5
   1.1. This Document 5

2. ARCHITECTURE 6
   2.1. System Clocks 6
   2.2. Bus Arbitration 6
   2.3. Bus Latency and Speed 7
   2.4. Interrupts 10

3. THE MEMORY MAP 11

4. INPUT/OUTPUT PORT MAP 12

INTERNAL INPUT/OUTPUT CONTROLLERS 14

1. KEYBOARD INTERFACE 14

2. GENERAL PURPOSE PORTS 16
   2.1. Joystick Input 16
   2.2. Floppy Disk Interface 16
   2.3. Screen Bank Select 16
   2.4. Audio Interface Muting 16
   2.5. Spare Lines 16

THE VIDEO CONTROLLER 18

1. INTRODUCTION 18

2. THE SCREEN MAP AND DISPLAY MODES 18
   2.1. Low Resolution Pixels 19
   2.2. High Resolution Pixels 19
   2.3. Colour Hold Mode 20
   2.4. Border Colour 20

3. THE REGISTERS 21
   3.1. Bank registers 21
   3.2. Horizontal count register 21
   3.3. Vertical count register 22
   3.4. Baud rate register 22
   3.5. Interrupt register 22
   3.6. Command register 1 23
   3.7. Command register 2 23
   3.8. Border colour register 23
   3.9. Scroll registers 24
   3.10. Transparent register 24
   3.11. Palette registers 24
   3.12. Light-pen registers 24
   3.13. Interrupt Acknowledge 25

DIGITAL SIGNAL PROCESSOR 26

1. INTRODUCTION 26

2. DSP INSTRUCTION SET 27

2.1. Instruction Codes 28
2.2. Special Addressing Modes - OFFSET & INTRUDE 31

3. IO MAP 32

4. AUDIO INPUT AND OUTPUT CHANNELS 34
4.1. Audio Outputs 34
4.2. Audio Inputs 34

THE BLITTER 35

1. INTRODUCTION 35
1.1. This Section 35
1.2. The Blitter 35

2. ARCHITECTURE 37
2.1. The Data Path 37
2.2. The Address Generator 41
2.3. The Sequencer and Controller 44

3. MODES OF OPERATION 51
3.1. Simple Memory Fill and Copy Operations 51
3.2. Line Drawing 53
3.3. Character Painting 55

4. BLITTER COMMAND FORMAT 57
4.1. Command Register 59
4.2. Mode Control and Extension Bits Register 60
4.3. Step Register 61
4.4. Comparator and Logic Function Unit Control Register . . . 62
4.5. Source Address Register 63
4.6. Destination Address Register 64
4.7. Program Address Register 65

CONNECTORS 66

1. EXPANSION CONNECTOR 66

2. VIDEO EXPANSION CONNECTOR 66

3. FLOPPY DISK DRIVE CONNECTOR 67

INDEX 68

Copyright 1988

SYSTEM OVERVIEW

1. INTRODUCTION

Flare One is a high-performance, low-cost computer designed for leisure activities such as computer music, arcade games and graphics creation.

To this end, the central processor, a 6 MHz Z80, has two support processors; a graphics co-processor or Blitter, and a 16-bit 6 MIPs Digital Signal Processor (DSP). These two devices provide the power to produce the required graphics and sound performance, respectively.

The system has up to One Megabyte of main memory. This is all available to the Z80, which views it as 64 **pages** of 16K, any of which may be paged into one of its 4 16K **banks**. Both the co-processors view this memory linearly.

The main memory is controlled by the video generator. This provides the screen image in high colour and pixel resolution, with a variety of useful modes. It also provides support for a light-pen, for external video capture, and for video incrustation onto an external source.

The Blitter can perform a wide range of graphics operations on the screen, from simple memory and rectangle moves, to line-drawing, character painting, and compressed screen data expansion. It acts as a processor in that it reads sets of commands from memory to perform an arbitrary number of screen operations without any Z80 intervention.

The DSP is a sixteen bit R.I.S.C. processor, executing micro-program instructions at 6 MHz. It is a Harvard architecture device, in that it has separate program and data RAM, which are both of 2K words. It has an integral hardware multiply/accumulate unit, and a separate general purpose ALU. It is directly connected to a 14-bit compact disk DAC, and has facilities for stereo operation, and for A to D conversion.

1.1. This Document

This document is intended for serious technical users of Flare One, writing code that drives the hardware directly. Less technical users should consult the operating system and other documentation, when available.

Copyright 1988

2. ARCHITECTURE

The Flare computer system has the following major components :-

1 Z80 CPU clocked at 6 MHz.
2 Up to 1 Megabyte of memory.
3 High-speed graphics co-processor (blitter).
4 16-bit Digital Signal Processor (DSP) with separate program and data RAM.
5 High colour and pixel resolution video output, with facilities for video capture, locking to external video source, and mixing the output with an external source (incrustation).
6 High-quality stereo audio input and output channel controlled by the DSP, utilising a 14-bit CD DAC.

A common system bus of 20 address lines and 8 data lines is shared by the CPU, Blitter and DSP. This allows any one of these devices at a time to access system memory. Memory cycles on this bus occur at up to 3 MHz.

2.1. System Clocks

All the clocks used by the main system elements are derived from a single crystal operating at four times the European PAL chrominence subcarrier frequency, namely 17.73 MHz.

This is divided by one and a half to provide the main co-processor clock, at 11.82 MHz (normally referred to as 12 MHz); and half this frequency, 5.91 MHz (or 6 MHz), is used as the Z80 clock, and is the DSP instruction execution rate.

2.2. Bus Arbitration

The bus is shared between the three devices that may use it using a handshake based Bus Request/Bus Grant system. Priority on the bus is as follows, in descending order :-

DSP
Z80 Interrupt Handlers
Blitter
Z80 Normal Operation

The DSP may only use the bus to read or write byte pairs, and will not use a very high percentage of the bus band-width, therefore allowing interrupt handlers a fast response time.

The Blitter will normally take over the bus completely from the Z80 CPU during operation, using 100% of the available bus band-width, but it will suspend its operation and return the bus to the Z80 while the Z80 Interrupt line is active. Blitter operation resumes when the peripheral retracts the interrupt line.

Copyright 1988

2.3. Bus Latency and Speed

Exact calculations of execution time/speed by Z80 and Blitter programs may not be made as the bus is subject to video and RAM refresh contention, resulting in the injection of wait-states in certain memory cycles. However this effect will be slight, not slowing down operations by more than a few per cent.

The bus timing is all referenced to the 12 MHz clock. This is twice the Z80 clock rate. In the calculations below clock cycles refer to this clock (a cycle being 84.6 nanoseconds)

Wait cycles occur during memory accesses because of the following:-

1 5 RAM Refresh cycles occur per video line (64 microseconds). These may be of two sorts:

   a) Opportunistic refresh. This occurs up to 5 times per video line when the bus is not immediately requested at the end of a memory cycle, and will occupy five clock cycles. Therefore the worst case delay due to one of these will be four clock cycles.

   b) Enforced refresh. These take place when insufficient opportunities have occurred for type a refreshes. Up to five of these will occur in sequence, immediately following a video fetch cycle. Therefore the current cycle will be held up while the video fetch takes place, immediately followed by all of the enforced refreshes, at four clock cycles each. These therefore give the longest period for which a cycle can be held up.

2 One video fetch cycle occurs per line. This takes five clock cycles, and therefore can hold up a cycle by up to this amount.

2.3.1. Z80 Execution Speed

The Z80 is clocked at 6 MHz. Therefore one Z80 T-state corresponds to two of the clock cycles being referred to in this section.

All Z80 memory cycles are subject to one additional T-state in wait. Therefore instruction execution times given by Z80 data sheets should be increased by 1 T-state per memory cycle (also referred to as machine cycles). This includes opcode-fetch cycles, memory read cycles and memory write cycles. Input/Output cycles are not affected.

The Z80 will also be slowed to some extent by the RAM refresh and video fetch mechanism, however the effects are unpredictable, and will be slight (probably about 1-2%).

2.3.2. Blitter Execution Speed

Blitter memory cycles will normally occur in four cycles. However, it is subject to wait as follows.

During blitter operations the bus is in almost constant use, therefore the blitter can expect to incur either enforced refresh cycles or to be

Copyright 1988

held in wait for four cycles by opportunistic refreshes. Therefore it will be held up by twenty cycles due to refresh and five cycles due to video fetch per line. There are approximately 750 cycles per line, so this is an overhead of about 3.3%.

It should also be noted when calculating blitter execution rates that during certain blitter operations, such as outer loop updates and inhibited write cycles, the bus is idle for one or more cycles. Refer to the blitter section below.

Copyright 1988

2.3.3. DSP DMA Latency

The program execution of the DSP is separate to the system bus, and therefore its speed is constant at 6 MHz (6 MIPs). However, it is necessary when creating DSP micro-programs to know the latency of DMA transfers with main system memory.

It should also be noted that each DSP DMA transfer can impose an overhead of eight clock cycles on any Z80 or Blitter activity, and that this should be taken into account when calculating the execution speed of Z80 or Blitter programs.

There are two cases here, depending on whether the bus is currently owned by the blitter or the Z80. These are given as follows :-

Operation	Time
Z80 owns bus	
Longest Z80 machine cycle. This is 6 T-states plus the associated wait state.	14
Two memory cycles	8
Two opportunistic wait cycles occurring during the cycles above.	8
Total =	30
Blitter owns bus	
Blitter current cycle completes	4
Two memory cycles	8
One of these three cycles is subject to wait from a video fetch cycle and five enforced refreshes.	25
Total =	37

The blitter case is dominant. [1]

---

[1] Certain assumptions may sometimes be made in the calculation of wait states, particularly when the blitter is running. At each DSP DMA bus request/acknowledge event, an opportunity for a refresh occurs. If sufficient of these occur per video line (64 us), then an enforced refresh will not occur, and the latency can be reduced accordingly- leaving the Z80 running case as the dominant one.

Copyright 1988

2.4. Interrupts

The Flare system allows a variety of devices to interrupt the CPU according to their particular needs. All these devices drive the Z80 maskable interrupt line directly. This allows for real-time responses to various events.

However, no hardware support is provided to prioritise these interrupts, and so it is up to the interrupt response routine to poll the hardware that can cause interrupts in an appropriate order.

All interrupts are held active until the software makes some sort of response to them. This is a necessity of the hardware, and any external interrupt generation hardware should obey this.

2.4.1. Interrupts and the Blitter

As the blitter can hold the bus for considerable periods, and prevent any Z80 operation, a mechanism is provided where the blitter will suspend operation when the interrupt line goes active, and allow the Z80 to process the interrupting device.

Note that this mechanism will come into effect whether the Z80 has interrupts enabled or not. This implies that using the blitter with interrupts disabled is dangerous, as the blitter will not start operation if the interrupt line is active; and this blitter command will get over-written if another occurs before interrupts.

As soon as the Interrupt line goes inactive, the blitter resumes operation. This also has certain implications:

1 Interrupt service routines may not have completed at this point, so there is a distinct possibility that stack overflow could occur as successive interrupts pile up return addresses on the stack.

2 Hardware that pulses the interrupt line will not be serviced properly.

3 Interrupt service routines may not use the blitter. If the blitter has suspended operation they will not be able to access it, and if it is inactive the underlying software may be in the middle of reading or writing its I/O ports.

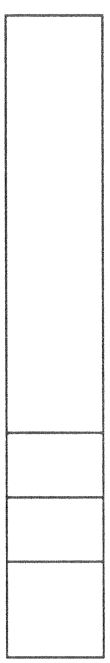
2.4.2. Use of Blitter under Interrupt

It would in fact be possible for the blitter to be used under interrupt, under certain conditions; by means of a semaphore between the underlying software and the interrupt service routine. This semaphore would indicate to the interrupt code whether or not the blitter was free to be used. This may be necessary to allow things like pointer animation under interrupt.

Copyright 1988

3. THE MEMORY MAP

The system has a One Megabyte address space. This divides into four major areas as far as the hardware is concerned, thus :-



This memory is addressed directly by the co-processors. For Z80 access the memory is divided into 64 x 16K byte pages and the Z80 address space is divided into four 16K byte banks. Any page can appear in any bank. Four bank registers determine which page appears in which bank. The page number is given by the top six bits of the physical (20 bit) address. After a reset the four bank registers point to page zero (in ROM).

Copyright 1988

4. INPUT/OUTPUT PORT MAP

The Z80 IO space is allocated as follows :-

Hex	Usage
00-0F	Video Controller
10-17	Digital Signal Processor
18-1F	Glue Gate Array
20-23	Blitter
24-27	UART - RS232 variable baud rate
28-2B	UART - RS232 fixed baud rate
2C-2F	UART - MIDI
30-33	Floppy disk controller
34-3F	reserved
40-4F	Auxiliary Chip One
50-5F	Auxiliary Chip Two
60-7F	reserved
80-FF	Free

All IO is fully decoded to eight bits internally, and all external IO should be similarly decoded. Locations marked as reserved may be used by Flare for future modifications or undocumented functions, and therefore must not be used by external hardware or be written to by applications software.

Address	Write Register	Read Register
00h	Bank 0	Bank 0
01h	Bank 1	Bank 1
02h	Bank 2	Bank 2
03h	Bank 3	Bank 3
04h	Horizontal count	Light-pen register 1
05h	Vertical count	Light-pen register 2
06h	Baud rate	Light-pen register 3
07h	Interrupt register	Interrupt acknowledge
08h	Command register 1	- reserved -
09h	Command register 2	- reserved -
0Ah	Border colour	- reserved -
0Bh	Scroll register 1	- reserved -
0Ch	Scroll register 2	- reserved -
0Dh	Transparent colour	- reserved -
0Eh	Palette register 1	- reserved -
0Fh	Palette register 2	- reserved -
10h	DSP intrude data	DSP intrude data
11h	DSP post-inc intrude data	DSP post-inc intrude data
12h	DSP intrude address	DSP intrude address
13h	DSP microprogram data	DSP microprogram data
14h	DSP run register	DSP status register
15h	DSP program counter	DSP program counter
16h	- reserved -	- reserved -
17h	- reserved -	- reserved -
18h	Blitter Program Address 0	Keyboard data

Copyright 1988

19h	Blitter Program Address 1	- reserved -
1Ah	Blitter Program Address 2	- reserved -
1Bh	Keyboard control	- reserved -
1Ch	- reserved -	Keyboard status
20h	Blitter Command Register	Destination Address 0
21h	- reserved -	Destination Address 1
22h	General Purpose Output	General Purpose Input
23h	- reserved -	- reserved -
24h	UART 1 Control	-
25h	UART 1 Transmit data	-
26h	-	UART 1 Status
27h	-	UART 1 Receive data
28h	UART 2 Control	-
29h	UART 2 Transmit data	-
2Ah	-	UART 2 Status
2Bh	-	UART 2 Receive data
2Ch	UART 3 Control	-
2Dh	UART 3 Transmit data	-
2Eh	-	UART 3 Status
2Fh	-	UART 3 Receive data
30h	FDC Command Register	FDC Status Register
31h	FDC Track Register	FDC Track Register
32h	FDC Sector Register	FDC Sector Register
33h	FDC Data Register	FDC Data Register

Copyright 1988

INTERNAL INPUT/OUTPUT CONTROLLERS

This section describes the Input and Output control devices fitted to Flare One internally. Note that some of these are optional.

1. KEYBOARD INTERFACE

The Keyboard Interface is intended to work with any IBM PC standard keyboard, of either the PC or the AT type, although different software drivers will be necessary for these as the codes returned are different.

The keyboard interface is accessed via a 5-pin DIN socket, with the following pin-out :-

Pin	Function
1	Clock
2	Data
3	Reset (optional)
4	Ground
5	+5 Volts

Keyboard Interface Socket

The keyboard interface controller may optionally generate interrupts whenever a complete byte has been read from the keyboard. Alternatively this condition may be polled for. The keyboard control register controls this, as follows :-

Bit	Function
0	Keyboard interface reset
1	Keyboard clock drive, active low
2	Keyboard interrupt enable

Keyboard Control Register

Copyright 1988

The keyboard interface status may be read from the status read port, this is as follows :-

Bit	Function
0	Keyboard data valid (interrupt active)
3	Comparator 0 [2]
4	Comparator 1
5	Comparator 2
6	Keyboard interrupt enable

Keyboard Status Read Port

The keyboard interface is reset whenever a 1 is written to bit 0 of the control register. This is necessary to clear the keyboard interrupt, and to allow the interface to receive the next character, whether interrupts are enabled or not. It is not necessary to write a zero to it afterwards.

The keyboard clock drive bit, if set to a zero, is used to drive the keyboard clock line (which is normally an input) low. This is used to initialise the keyboard. If it is set high the keyboard clock comes from the keyboard.

Note that the keyboard interface has no protection against or detection of overrun, and therefore it is essential that the keyboard data is read and the interface cleared as quickly as possible.

---

[2] The three comparator bits present here are the same comparators that are readable by the DSP. Their outputs are duplicated here for convenience.

Copyright 1988

2. GENERAL PURPOSE PORTS

The system contains general purpose input and output ports to control various functions. These are :-

	Bit	Function
1	0	Audio interface Muting
2	1	Floppy Disk drive select 0
4	2	Floppy Disk drive select 1
8	3	Floppy Disk side select
16	4	Spare output 0
32	5	Spare output 1
64	6	Screen bank select

General purpose output register

	Bit	Function
1	0	Joystick up
2	1	Joystick down
4	2	Joystick left
8	3	Joystick right
16	4	Joystick fire
32	5	Floppy Disk ready
64	6	Spare input 0
128	7	Spare input 1

General purpose input port

2.1. Joystick Input

A standard Atari type joystick input is provided. The Z80 may poll the current state of this input on bits 0-4 of the General Purpose Input. Bits will be 0 if the relevant switch is closed and 1 if it is open.

2.2. Floppy Disk Interface

The floppy disk interface is controlled by a WD1770 Floppy Disk Controller (FDC). However this does not provide all the interface lines, and so the remainder are provided by the General Purpose ports. These are, as shown above, two drive select and one side select outputs, and the drive ready input.

2.3. Screen Bank Select

Selecting between the two screen banks must be programmed in two locations, one here and the other in the Video Chip Command 1 Register.

2.4. Audio Interface Muting

The audio interface muting control is provided here.

2.5. Spare Lines

Copyright 1988

Two inputs and two outputs are not currently assigned functions. These are available on the expansion connector, and may be used by external hardware if they have not been assigned uses by any internal options.

Copyright 1988

THE VIDEO CONTROLLER

1. INTRODUCTION

This document describes the FLARE video chip. The video chip has the following functions :-

1) To extend the addressing range of the Z80 to 1Mbyte
2) To provide DRAM timing on behalf of the processors
3) To generate video and refresh cycles
4) To generate video timing and support gen-locking
5) To convert video memory contents to pixel data.
6) To generate an interrupt at a given display line.
7) To latch the presence and position of a light-pen signal
8) To generate one fixed and one variable baud rate.

2. THE SCREEN MAP AND DISPLAY MODES

There are two physical screen locations in memory. Data is stored in these in the order of the raster scan, i.e. form left to right for each lines, and with lines in the order top to bottom.

One screen map at any time is converted into the displayed picture. This is normally a non-interlaced display of 312 lines at 50 Hz. Horizontally, pixels are displayed for 43us of the 64 us line time, and vertically for 256 lines.

Each byte in memory is converted into one pixel in lo-res mode or two pixels in hi-res mode. There are 256 lo-res or 512 hi-res pixels per screen line. The lowest address in screen memory determines the top left pixel, the next address gives the next (lo-res) pixel on that line. Pixels on the next line are at an address 256 higher.

Note that this simplistic description ignores the effect of the scroll registers which can move the display origin to any part of the screen memory.

Note also that provision is made for generating an interlaced display of 512 lines vertically, by using both screen maps alternately, and generating the necessary half line at the end of each frame by software intervention with the horizontal counter. See below.

Copyright 1988

2.1. Low Resolution Pixels

The pixel colour is defined by an eight bit physical colour. The physical colour has three red bits, three green bits and two blue bits. Each byte in lo-res mode defines a logical colour which may have 6, 7 or 8 bits. In eight bit mode the mapping is:

MSB									LSB
	r2	r1	r0	g2	g1	g0	b1	b0	

In seven bit mode one bit of red is sacrificed

MSB									LSB
	--	r1	r0	g2	g1	g0	b1	b0	

The four physical levels of red which correspond to the four logical values are 0, 2, 4 & 7.

In six bit mode one bit of green is also sacrificed

MSB									LSB
	--	--	r1	r0	g1	g0	b1	b0	

The mapping of the green bits to the physical levels is as before for red.

The unused (or masked) bits may be used elsewhere, e.g. by the blitter for depth information. Bit 7 may also be used by the video chip:

It can be made available on the incrustation output to embed the locally generated picture onto an external video source. (Bit 7 set selects the local picture).

It can be used in variable-mode to determine whether the current byte is displayed as one lo-res pixel or two hi-res pixels. (Bit 7 set two hi-res pixels are generated).

The incrustation output may also be used to strobe frame grabbing AD convertors.

2.2. High Resolution Pixels

In hi-res mode each byte defines two pixels. The low nibble defines the first pixel displayed (the left hand one), the high nibble the second. Either three or four bits of the nibble are used to determine the physical colour. If three bits are used the mapping is:

MSB					LSB
	--	r0	g0	b0	

Each bit is mapped onto all the corresponding bits of the physical colour.

Copyright 1988

If four bits are used the most significant bit adds brightness to the others.

MSB | Br | r0 | g0 | b0 | LSB

The mapping to physical colour is summarised by the following tables:

b0	Br	B0	B1
0	0	0	0
0	1	1	0
1	0	0	1
1	1	1	1

g0	Br	G0	G1	G2
0	0	0	0	0
0	1	0	1	0
1	0	1	0	1
1	1	1	1	1

r0	Br	R0	R1	R2
0	0	0	0	0
0	1	0	1	0
1	0	1	0	1
1	1	1	1	1

The most significant bit in each nibble may be passed to the incrustation output as before.

In order to expand the choice of colours in hi-res mode two colours are replaced by palette colours. Magenta is replaced by the physical colour in palette register 1. Yellow is replaced by the physical colour in palette register 2. The actual bit pattern corresponding to magenta and yellow depends on the number of bits used as follows:

	magenta	yellow
3 bits	x101b	x011b -X110b
4 bits	0101b	0011b -0110b

2.3. Colour Hold Mode

In colour hold mode one physical colour is elected to be 'transparent'. This colour is replaced by the previously displayed non-transparent colour. This display scheme may be used to fill large areas with colour by simply drawing their outline.

Note that the left side should be drawn in the fill colour, and the right side should be drawn in the required background colour. If the first pixel displayed on a line is transparent, then the colour held is the border colour.

The colour hold mechanism works AFTER all the previous transformations.

2.4. Border Colour

A border register determines the physical colour of picture outside the computer screen.

Copyright 1988

3. THE REGISTERS

There are 16 I/O locations used by the video chip

Address	Write Register	Read Register
00h	Bank 0	Bank 0
01h	Bank 1	Bank 1
02h	Bank 2	Bank 2
03h	Bank 3	Bank 3
04h	Horizontal count	Light-pen register 1
05h	Vertical count	Light-pen register 2
06h	Baud rate	Light-pen register 3
07h	Interrupt register	Interrupt acknowledge
08h	Command register 1	- reserved -
09h	Command register 2	- reserved -
0Ah	Border colour	- reserved -
0Bh	Scroll register 1	- reserved -
0Ch	Scroll register 2	- reserved -
0Dh	Transparent colour	- reserved -
0Eh	Palette register 1	- reserved -
0Fh	Palette register 2	- reserved -

3.1. Bank registers

BANK0	00h
BANK1	01h
BANK2	02h
BANK3	03h

These four six bit registers contain the page accessed by the Z80 in each of its four banks. The registers are readable. Zeroes should be written into the top bits for future compatibility.

3.2. Horizontal count register

HCNT	04h

When a byte is written into this register the horizontal time-base counter is loaded with this value, and the value in bits 3 & 4 of command register 1. This is mainly useful for test purposes but could be used to synthesize an interlaced display.

The counter counts up from 0 (just after hsync) to 756 (the end of hsync). The active display area is from 120 to 631 inclusive. The counter counts in units of one hi-res pixel (11.82MHz).

Copyright 1988

3.3. Vertical count register

VCNT 05h

When a byte is written into this register the vertical time-base counter is loaded with this value, and the value in bit 5 of command register 1. Again this is mainly useful for test purposes but could be used to reduce the display height.

The counter counts up from 0 (just after vsync) to 311 (the end of vsync). The active display area is from 33 to 288 inclusive. The counter counts in display lines (15.625Khz).

3.4. Baud rate register

BAUD 06h

The bottom three bits of this register determine the programmable baud rate. If the UART is programmed to use a 16x clock the baud rates will be as follows:

D2	D1	D0	Rate
0	0	0	38400
0	0	1	19200
0	1	0	9600
0	1	1	4800
1	0	0	2400
1	0	1	1200
1	1	0	600
1	1	1	300

Bit 3 of this register opens the light pen registers so that they reflect the current value of the time-base counters. This allows the current beam position to be read for beam synchronous screen update.

Bit 4 of this register is used for diagnostic purposes and should be programmed with a zero.

Bits 5..7 are not used but should be programmed with zeroes.

3.5. Interrupt register

INT 07h

The eight bits in this register and bit 2 in command register 1 determine the screen line at which an interrupt is generated. The interrupt is generated two hi-res pixels after the end of hsync. The interrupt remains until acknowledged by reading the interrupt acknowledge port.

If there is an interrupt pending this is indicated by bit 4 of light-pen register 3.

Copyright 1988

3.6. Command register 1

CMD1 08h

This register is used to set up the most significant bits of HCNT, VCNT and INT. In addition it selects screen 2 and enables gen-locking. The various bits have the following functions:

Bits 0..1	reserved, write 0
Bit 2	the most significant bit of INT
Bits 3..4	the most significant bits of HCNT
Bit 5	the most significant bit of VCNT
Bit 6	Screen 2. N.B. The blitter must also be programmed to enable the output from the appropriate VRAM.
Bit 7	Enables gen-locking. When enabled hsync and vsync become inputs and the video time-base generators are reset by these inputs. N.B. these must be synchronous.

3.7. Command register 2

CMD2 09h

This register is used to control the display mode as follows :-

Bit 0	sets hi-res mode. NB the mode is only changed at the end of the current line.
Bit 1	enables incrustation. If zero the local display is enabled. If one the local pixels are enabled if the most significant bit of each pixel is set.
Bit 2	controls incrustation of the border. If incrustation is enabled this bit enables the local border colour.
Bit 3	enables colour hold mode
Bit 4	reserved, write 0
Bit 5	masks most significant bit of each pixel. In lo-res mode bit 7 of each byte is ignored. In hi-res mode bit 3 of each nibble is ignored.
Bit 6	masks two most significant bits of each pixel. In lo-res mode bits 6 & 7 of each byte are ignored
Bit 7	enables variable resolution mode. The mode is determined by bit 7 of each display byte

3.8. Border colour register

BORDER 0Ah

This register gives the border physical colour as follows:

MSB

r2	r1	r0	g2	g1	g0	b1	b0

LSB

Copyright 1988

3.9. Scroll registers

SCROLL1	0Bh
SCROLL2	0Ch

These registers determine the address of the top left of the screen as an offset from the bottom of the screen memory. If both registers contain zero the top left of the display will be at address 20000h (assuming screen 1 is being used). If 01 is written to SCROLL1 the display will start at 20001h and the pixels will have moved left by one. If 08 is now written to SCROLL2 the display will start at 20801h and the pixels will have moved up by eight.

NB pixels shifted off the screen appear at the other side and must be cleared in a separate operation.

3.10. Transparent register

TRANS	0Dh

This register gives the physical colour elected to be 'transparent' in colour hold mode.

3.11. Palette registers

PAL1	0Eh
PAL2	0Fh

These registers determine the physical colours to replace magenta and yellow in hi-res mode.

3.12. Light-pen registers

LPEN1	04h
LPEN2	05h
LPEN3	06h

These registers determine the current position of the light-pen or electron beam depending on the state of bit 3 in BAUD.

LPEN1 gives the lowest eight bits of the horizontal count. The top two bits are given by bits 0 & 1 of LPEN3.

LPEN2 gives the lowest eight bits of the vertical count. The top bit is given by bit 2 of LPEN3.

Light-pens produce well defined rising edges but slow decays and respond to several display lines. The value latched is the value at the leading edge of the first pulse each frame. A latch is cleared at the start of each frame (at the end of vsync) and is set by the arrival of the light-pen pulse. The state of this latch can be read at bit 3 of LPEN3 (a one indicates pulse arrived).

Bit 4 of LPEN3 indicates that a video chip interrupt is pending.

The top three bits of LPEN3 give the top three bits of the vertical counter. This can be used to get a rough idea of the beam position.

Copyright 1988

3.13. Interrupt Acknowledge

ACK 07h

Reading this port clears the video chip interrupt

Copyright 1988

DIGITAL SIGNAL PROCESSOR

1. INTRODUCTION

The DSP system in Flare One is a general purpose arithmetic processor with sufficient power to implement a high performance music synthesizer. A 14 bit audio DAC is provided for signal output with sample and hold circuits for generation of stereo signals and a comparator circuit to enable the DAC to be used in a successive approximation analogue to digital conversion routine, thus providing the ability to transfer sound or other analogue information into the computer.

The DSP is micro-programmable from the host CPU and the instruction set is sufficiently flexible to enable the user to program the device to fulfill many different functions that are considerably at variance with that of 'music synthesizer'. Such applications might include algorithmic speech generation and recognition, audio analysis using fast fourier transform techniques, or use as a numeric processor for graphics translation.

The DSP uses Harvard Architecture for maximum data throughput (separate program and data buses).

It provides addition, subtraction and logical functions with a 16 bit parallel ALU incorporating a 16 bit full adder / subtracter. The carry bit from the adder / subtracter is stored in a separate latch and can be either used either to propagate carry for multiple precision arithmetic or can be used for the conditional instruction ADD IF CARRY NOT SET. This instruction will perform the addition if the carry has not been set by the last addition, otherwise the result register will contain the Y input to the ALU with the carry bit cleared.

It has multiplication functions with a 12 x 16 fast serial multiplier / accumulator ALU. This allows a series of products to be summed, e.g. for a digital filtering algorithm. Multiply operations take several instruction cycles to complete, but other instructions may be executed while a multiply is taking place. The multiplier produces a 28 bit 2's complement result.

Data transfers within the device (with the exception of internal transactions within the multiplier / accumulator) are all 16 bit.

Each ALU can address 2048 x 16 bit words of internal Data SRAM.

Copyright 1988

2. DSP INSTRUCTION SET

Each instruction (for both ALU's) has a 5 bit opcode and an 11 bit address vector. The five bit opcode is encoded in the top five bits of the instruction. The values are given below. The low eleven bits contain a data address where appropriate.

This data address gives the operand of an instruction. Most instructions will either read data from the operand address in data RAM or write data to it.

All arithmetic instructions operate on 2's complement numbers, as does the DAC.

All micro-coded instructions (with the exception of OFFSET and INTRUDE instructions) are completed in one 169.162 ns cycle. All instructions are memory to register transfers or vice versa.

A carry/borrow flag is maintained. This is set or cleared only by the ADD, SUB, ADC or SBC instructions, as appropriate; and the carry or borrow result of one of these may be used by as subsequent ADC, SBC or ADINC instrunction.

Copyright 1988

2.1. Instruction Codes

00	MOV (nn), MZ0	Move data from the least significant word of the multiplier result register to the operand.
01	MOV (nn), MZ1	Move data from the most significant word of the multiplier result register to the operand. Reading this word clears the entire result register, and therefore the result must not be read during a multiply and accumulate operation. As the multiplier is 12x16, the result is 28 bits. Therefore MZ1 is only a 12 bit register. However this operation sign extends the value in it to 16 bits.
02	MOV MX, (nn)	Move 12 bit data word from data SRAM to multiplier X register. Only the bottom 12 bits of the word are used. The top 4 bits are ignored, but the number is effectively sign extended to 16 bits.
03	MOV MY, (nn)	Move data word from data SRAM to multiplier Y register. This transaction starts the multiplier sequence. The result register must not be read for seven instruction cycles after this instruction has executed.
04	MOV (nn), CMPR	Move data from comparator inputs to data SRAM.
05	MOV DMA0, (nn)	Move data from SRAM to the least significant word of the DMA address register. This action triggers the DMA sequence. Once this instruction has been completed it is important that none of the DMA registers (DMA0, DMA1 and DMD) are written to for at least the latency time of the host system, for more information on this time refer to the discussion of latency earlier. For the same reason the DMD register must not be read. As the DSP view of the host DRAM is word wide the least significant bit of DMA0 is ignored.
06	MOV DMA1, (nn)	Move data from SRAM to the most significant word of the DMA address register.
07	MOV DMD, (nn)	Move data from SRAM to DMA data register.
08	MOV (nn), DMD	Move data from the DMA data register to SRAM.
09	NOP	

Copyright 1988

0A	MOV (nn), INTRA	Move data from the INTRUDE address register to SRAM.
0B	OFFSET	See explanation below
0C	MOV (nn), PC	Move data from Program counter to SRAM. The value transferred is the address of the instruction following this one. (This is useful for relative jumps, computed GOTOs etc).
0D	MOV AX, (nn)	Move data from SRAM to ALU x register.
0E	MOV (nn), AX	Read data from ALU x register to SRAM.
0F	MOV (nn), AZ	Read data from ALU result register to SRAM
10	ADD (nn)	Add data in SRAM to data in ALU x register and place result in the ALU z (result) register. Ignore carry flag but set it afterwards according to result.
11	SUB (nn)	Subtract data in SRAM from data in ALU x register. Ignore carry flag but set it if the result overflows.
12	AND (nn)	Perform logical AND between SRAM data and ALU x register. Carry flag is undisturbed.
13	OR (nn)	Logical OR between SRAM and ALU x register. Carry flag not changed.
14	ADC (nn)	Add the SRAM data, the ALU x register and the carry bit together. Carry flag will be updated.
15	SBC (nn)	Subtract operand and the carry flag from the ALU x register. Carry flag is updated.
16	ADinC (nn)	If the carry bit is not set then the operand data is added to the x register, the result is passed to the result register, and the carry flag reflects the result of this operation. If the carry bit is set then the operand contents are passed unchanged to the result register and the carry bit is cleared. NOTE - it is the parameter of ADinC that is passed to AZ if carry is set, and not the contents af AX. This is a common mistake!
17	MOV AZ, (nn)	Load the result register directly with the operand.

Copyright 1988

18	MOV DAC0, (nn)	The operand is passed to the D to A converter with neither sample and hold switched on. Note at least 16 machine cycles must elapse before another DAC instruction is executed or else the data will become corrupted.
19	MOV DAC1, (nn)	The operand is passed to the D to A converter with the LEFT sample and hold switched on. See instruction 18 for restrictions on use.
1A	MOV DAC2, (nn)	The operand is passed to the D to A converter with the RIGHT sample and hold switched on. See instruction 18 for restrictions.
1B	MOV DAC12, (nn)	Move the operand data to the D to A converter with both sample and holds switched on. See instruction 18 for restrictions.
1C	NOP	
1D	MOV PC, (nn)	Jump instruction. Note that because instructions are pipelined the instruction immediately following the jump instruction in program memory will be executed before the one at the location specified by the contents of nn. Therefore this instruction will normally be followed by a NOP.
1E	NOP	
1F	INTRUDE	This instruction MUST be followed by NOP. The address field of the NOP instruction is supplanted by the intrude address register and the data direction is taken from recent history of host transactions with the intrude register. At least one INTRUDE instruction must be put into the dsp program if any data transfer is to take place between host and dsp (see section below). NOTE - the Flare DSP Assembler automatically inserts the following NOP instruction after an INTRUDE.

Copyright 1988

2.2. Special Addressing Modes - OFFSET & INTRUDE

Because instructions are pipelined and the data and address buses are physically separate the way that the DSP generates addresses for:

a) data transfer to and from host under host control and
b) indexed instructions

becomes a little contrived.

2.2.1. INTRUDE instruction

When an intrude instruction is encountered in the DSP code the address field from the program static RAM is inhibited and replaced with the contents of the Intrude Address Register. This is then delayed through the address pipeline latch and appears as the address field for the following instruction. This instruction will be a transfer of data between this new "INTRUDE" address and he intrude data register. The direction of data flow depends on recent history of host activity with this register. A sequence of events might be as follows.

1. Host writes data to INTRUDE register.
2. INTRUDE instruction detected in DSP code:
   DSP transfers data to Data SRAM.
3. Next INTRUDE instruction detected:
   DSP transfers data from SRAM to intrude register.

In other words, the direction of data transfer is always FROM data SRAM TO the intrude register except for during the INTRUDE instruction immediately following the host writing new data to the intrude register.

2.2.2. OFFSET instruction

In order that tables may be used there is a mechanism for the DSP generating its own address.

This instruction takes three cycles and might look something like this:

```
MOV AX, (nn1)      ;  The data which is to become the address
                      for the third instruction is fetched from
                      nn1.
OFFSET (nn2)       ;  Offset instruction uses a dummy address
                      nn2 which is left unused and un-
                      corrupted.
MOV AX, (nn3)      ;  The address of nn3 is supplanted with
                      data nn1 used as an address. The data in
                      nn3 is left undisturbed.
```

Copyright 1988

3. IO MAP

The DSP is allocated IO addresses in the range 10-18h. This area is used as follows :-

10h INTRUDE data register.

Data is written or read to this register and the DSP then transfers this data to and from the internal data SRAM locations defined by the intrude address counter. Data in the DSP is 16 bits wide so this register needs to be written or read from TWICE before a successful transfer can take place. The lowest order byte is taken first.

11h INTRUDE data register; post increment

Data is transferred between host and DSP as above but in this case the intrude address is automatically incremented after the data has been successfully transferred.

12h INTRUDE address register

Target address for data transfers (under host control) between DSP and host. On the next INTRUDE instruction after this register is updated, data from the new SRAM location is loaded into the INTRUDE register ready for the host to read. This register is 11 bits long so TWO read or write operations need to be executed before the transaction is completed. The 8 lower order bits are transferred first.

13h MPROG data

Microprogram data is loaded via this port. It is only successful when the DSP is in STOP mode. Note also that programs must be loaded before data as the program executes as it loads, so data may become corrupted. The target address for the microprogram must also be loaded into the program counter. Microprogram data is 16 bits wide and so TWO write or read operations need to be executed to transfer one word of microprogram data. The program counter is automatically incremented after every word of program transferred.

Copyright 1988

14h RUN/Status register.

Bit 0 in the write register controls RUN/STOP of the DSP. Writing a ONE into this bit set DSP to RUN. Writing any other value to this port causes the DSP to single step. Bits 0-2 of the READ register are the bits of the state machine controlling the INTRUDE mechanism. All three bits need to be zero in order for the host to initiate any INTRUDE transaction. If bit 0 is set then the host has only completed one byte of a two-byte transfer. Bit 3 reads the RUN/STOP condition

15h PROGRAM counter.

The value of the program counter is read / written to this address. The program counter is 11 bits long so each transaction requires two inputs / outputs. The lowest order byte is transferred first.

16h reserved

17h reserved

Copyright 1988

4. AUDIO INPUT AND OUTPUT CHANNELS

4.1. Audio Outputs

The system has two audio output channels. The level of these is given by a sample-and-hold circuit for each channel, connected to the output of the DAC. Either or both of these can be loaded with a given value, with DSP opcodes 19-1B.

4.2. Audio Inputs

Audio input value may be determined from the comparators. There are up to three audio input channels, and any of these may be compared with the output of the DAC. The result is given by the comparator input word read by instruction 04. Bit 0 is comparator 0, bit 1 is comparator 1, and bit 2 is comparator 2.

Analogue to Digital conversion may be performed using a technique of successive approximation. Opcode 18 would be used for this to avoid spurious noises appearing on the outputs.

It should be noted that the state of the comparator outputs may also be read by the Z80 on the Keyboard status read port.

Copyright 1988

THE BLITTER

1. INTRODUCTION

1.1. This Section

This section describes the graphics co-processor, or blitter, of the Flare Technology product Flare 1. It explains its modes of operation and how these are controlled by the programmer.

The intention of this section is to allow the advanced programmer to extract maximum performance from the system by driving the blitter hardware directly. For most purposes this will not be necessary, and most users will never need to come closer than the operating system graphics calls.

The blitter is a complicated beast, and to drive it properly requires a pretty clear understanding of how the hardware works. This is all explained here, but a knowledge of some aspects of digital hardware design is assumed.

1.2. The Blitter

The blitter is a processor in the sense that it has the ability to execute programs. It can perform arbitrarily long sequences of graphics operations by reading new command sets from memory. It does not have the ability to perform jump or branch operations, but can perform conditional stops, so that the CPU can abort the current operation. It is not a Turing machine.

While it is performing any graphics operations, the blitter becomes a bus master, and denies the CPU any bus activity whatsoever. This is reasonable behaviour because the blitter is being used to perform operations that the CPU would otherwise have performed, and is therefore speeding up program operation. This also removes the need for any asynchronous control programming for blitting operations, and the need for any interrupt generation hardware in the blitter.

However, to allow real-time programming of either of the other two processors, the blitter will grant the bus to the DSP DMA channel, and to the CPU if an interrupt occurs. During either of these the current operation is suspended but will restart when the interrupt goes away or when the DSP DMA access completes.

The blitter may be operated in a variety of modes to perform graphics and block move operations. These are described in the sections that follow.

Copyright 1988

WARNING - as a consequence of the blitter granting the bus to the Z80 during interrupts, two problems can arise :-

1 Interrupt service routines should not attempt to use the blitter, as the underlying software may be in the middle of a blitter operation.

2 The blitter should not be used with interrupts disabled, as the interrupt line may be active preventing the blitter operation taking place - and therefore interfering with subsequent blitter operations. This particularly applicable to attempts to use the blitter twice within a piece of code in which interrupts are disabled.

Copyright 1988

2. ARCHITECTURE

The internal architecture of the blitter is best considered as three, largely separate, blocks. These are the Data Path, the Address Generator, and the Sequencer and Controller.

2.1. The Data Path

The Data Path contains two data registers, and two data manipulation blocks. These are the Logic Function Unit, which can combine the contents of the data registers in a number of ways to produce the output data; and the comparator, which can perform certain comparisons on the data to inhibit write operations, and optionally stop operations.

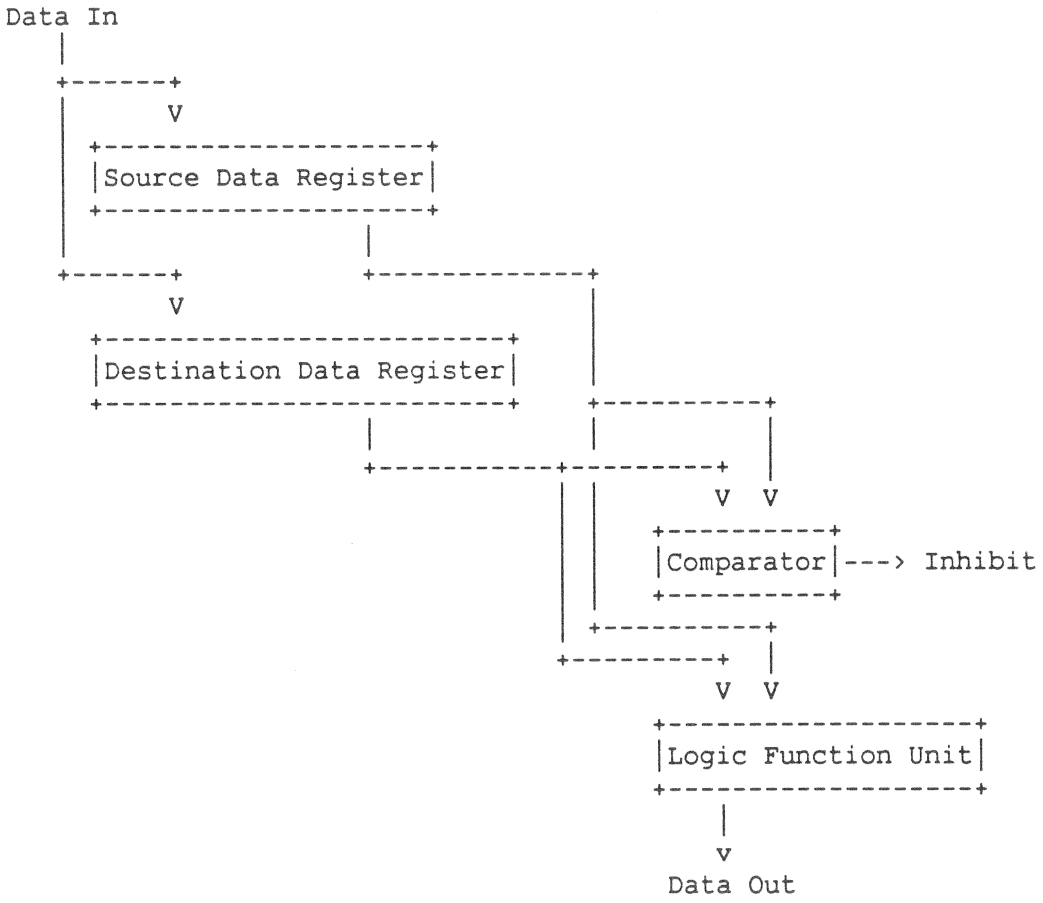


Figure - Data Path

Copyright 1988

2.1.1. Registers

There are two eight bit wide data registers. These are the Source and Destination Data registers.

The Source and Destination data registers are normally loaded from the Source and Destination addresses in memory, but are also loaded before the start of a blitter operation with the pattern data, and may be used as an additional source of data either in producing the output data or in the comparator. For example it may be a mask, a pattern for writing, a reference value, and so on.

2.1.2. Logic Function Unit

The Logic Function Unit generates the output data, which is written to the destination in memory. It can perform any logical combination of the source and destination data.

It does this by selecting any of the four boolean minterms[3] of the two sets of input data from the data registers, and generating the logical OR of the selected terms. This allows any logical combination of the of input data, there are therefore sixteen possibilities.

2.1.3. Comparator

The comparator can perform a variety of comparisons on the data in the data registers. If its comparison conditions are met then it generates an inhibit signal. This can be used to inhibit a write operation, and optionally to stop blitting operation.

The comparator performs the comparison between the source data register and the destination data register.

2.1.3.1. Multiple Plane Comparator Operation

The comparator may be used to provide a multiple plane effect, to give transparent colours, for memory search operations, and as an aid to character painting.

Multiple plane operation is supported by assigning a plane number to every pixel. In low resolution mode one or two of the eight bits are used giving two or four planes, in high resolution mode one of the four bits is used giving two planes.

---

[3] The minterms of a set of logical inputs are the set of AND terms given by each of the inputs combined together in all combinations of complimented and un-complimented. Therefore the minterms of two inputs A and B are:

A AND B, A AND (NOT B), (NOT A) AND B and (NOT A) AND (NOT B)

Any boolean combination of A and B can be given by choosing a suitable set of minterms to OR together.

Copyright 1988

The bits are allocated within the byte thus:

Bit	Plane Function	
	Lo-res	Hi-res
0		
1		
2		
3		Plane bit
4		
5		
6	Plane bit 0	
7	Plane bit 1	Plane bit

The comparator can produce an inhibit output if the plane number of the source data is less than, equal to, or greater than the plane number of the destination data, or any combination of these. This means that data being written onto the screen can be masked by data already present in a different plane.

The user may choose to allocate one or two bits as plane bits in the video controller. The same mechanism is used, however if only two planes are being used then the control bits for greater than and less than still have the same effect, but priority equal then requires knowledge of the state of the source plane bit.

2.1.3.2. Comparator Byte Comparison

The comparator can produce an inhibit output if the entire source byte is the same as the destination byte. This may be used for searching memory for a particular value; and, more importantly, for designating a colour to be transparent, and holding the transparent colour value in a data register.

2.1.3.3. Comparator Bit to Byte Expansion

This comparator operation allows bit to pixel expansion of data, used for example for character painting. In this mode the comparator selects a bit of the source byte based on the value of the inner counter (refer to the Sequencer Description), and inhibits the write operation if this bit is zero.

Copyright 1988

2.1.4. Handling of Data in Low and High resolution mode

The blitter makes provision for handling two screen resolution modes. These are low-resolution (lo-res), where each byte corresponds to one pixel, and high-resolution (hi-res), where each nibble corresponds to one pixel.

In lo-res the data path is handling one pixel at a time, and this makes its operation straightforward. In hi-res, however, only half the current byte is the current pixel so certain additional requirements are placed on the data path.

In a hi-res write operation unchanged destination data is written to the half of the data byte that does not correspond to the current pixel. This means that write-only blitter operation will not work properly in hi-res mode.

It is also possible that the source hi-res address and the destination hi-res address point at different halves of a byte. If this is the case, a shifter swaps the two halves of the source data.

This is all handled transparently to the user. However certain caveats should be noted:

1 - In hi-res mode, destination reads must always be performed. If loop type 2 is selected then the pixel in the other half of the destination byte will be corrupted.

2 - In hi-res mode, the two halves of the pattern register should normally be set to the same value.

Note, however, that the blitter may be set to lo-res mode when the display is in hi-res mode, to perform block moves, area fills, etc., and this will speed up certain operations. The only restriction is that all operations will then occur on multiples of two pixels, and on even pixel boundaries.

Copyright 1988

2.2. The Address Generator

The address generator contains two address registers, an increment or step register, an address adder, and an address output multiplexer.

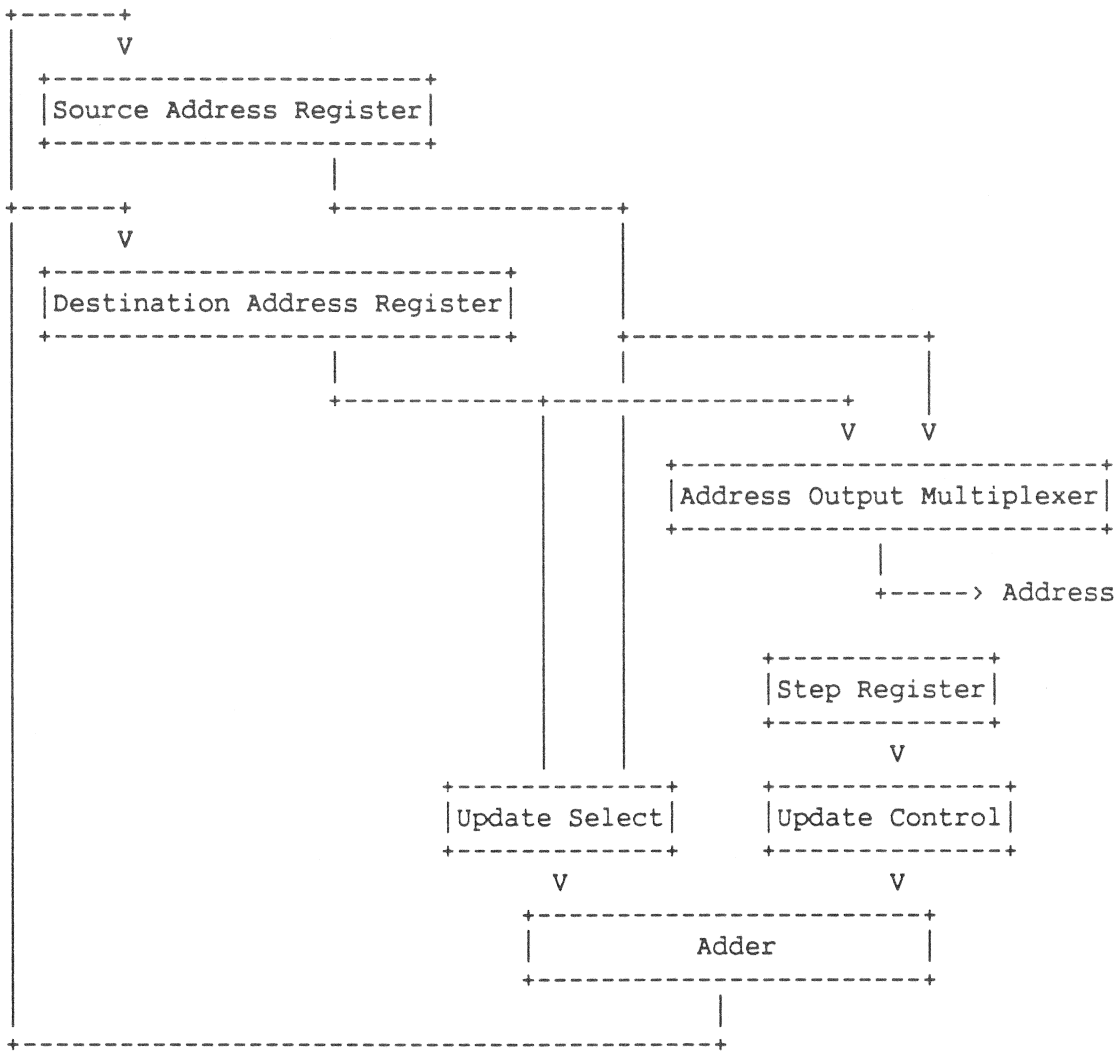


Figure - Address Generator

Copyright 1988

2.2.1. The Address Registers

There are three address registers, the source address, the destination address, and the program counter.

Each of these registers contains twenty address bits, referred to as bits 0 to 19, allowing the blitter to address up to One Megabyte. In addition to this the source and destination address registers contain a nibble bit used in high resolution mode, designated bit -1.

The Program Counter is as its name implies, a counter that is incremented by one each time a memory cycle is performed using it. The program counter is quite separate to the remainder of the address generation mechanism.

The Source and Destination Address registers are updated after each cycle, and at other times, using an adder that allows them considerable flexibility in the objects to which they refer.

All address updating is only performed on the bottom sixteen bits of the address register. This means that the blitter effectively operates in 64K pages. If an address overflows in a page, it will wrap, and the overflow (or underflow) lost.

2.2.2. The Address Adder

The address adder is a seventeen bit wide adder used to update addresses. It allows either a constant or a variable stored in the step register to be added to an address value. It can also subtract the same values. The seventeenth bit is the nibble part of the address.

Note that an increment of one pixel has a different effect on the address depending on the screen resolution set, and that this is automatically catered for.

All address registers are updated automatically at the end of the appropriate memory cycles.

2.2.3. The Address Output Multiplexer

The address output multiplexer provides the external address to the system memory. It provides three types of address, these are:

Source Address
Destination Address
Indexed Address

The two basic addresses are the source address and the destination address. These are derived directly from the corresponding address registers.

The other form of address is the indexed address, or table look-up address. This can be used to give a form of palette mapping. In this mode the source address register provides bits eight to twenty of the output address from its corresponding bits, and the destination data register provides bits zero to seven of the output address. This mode

Copyright 1988

of operation is explained more fully in the section on table look-up operation.

2.2.4. Line Drawing Address

When a line draw is being performed, the address registers are used in a different way from normal. The Destination Address register is used as the line draw address, and the Source Address register and Step register are used as Delta One and Delta Two respectively. During line drawing Delta Two is subtracted from Delta One, and the borrow output produced is used to determine what is added to the Destination Address register. See the section on line-draw.

Copyright 1988

2.3. The Sequencer and Controller

The sequencer and controller section controls the operation of the other sections of the blitter, and therefore governs the overall operation of the device.

The flow of control is best considered at two main levels. There is an outer loop governing the overall flow of control, and within that three subsections, the inner loop, the parameter read loop and the command read loop.

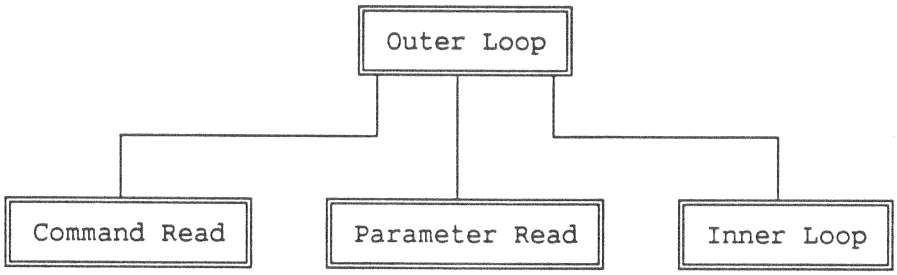
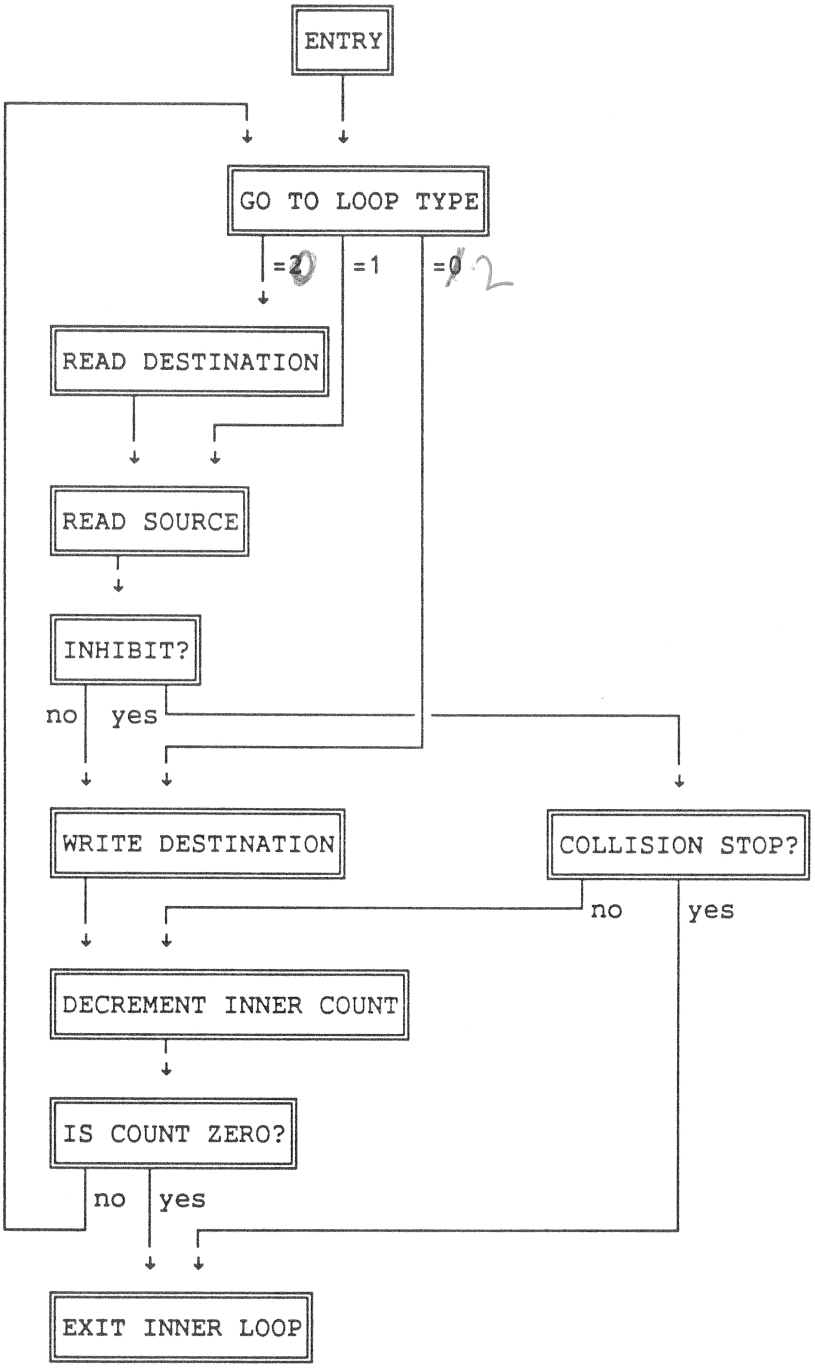


Figure - Control hierarchy

Copyright 1988

2.3.1. The Inner Loop

The inner loop performs the actual blitting or line-drawing operation. Its basic form is:



Blitter Inner Loop - Flow Diagram

From the entry point control can pass to three points. This corresponds to the three basic loop types. These are write destination only, used for area fill, etc.; read source then write destination, used for data move operations; and read destination then read source then write destination, used for combining the source and destination data, or plane priority based write operations.

Copyright 1988

A fourth type of loop, type 3, behaves like loop type 1 on its first pass round the inner loop, and like loop type 0 on second and subsequent repeats of the inner loop.

If the loop type includes a source read, or a source read and a destination read, then the comparator inhibit mechanism is tested before the destination write occurs. This allows the write cycles to be bypassed when the comparator inhibit conditions are met. Under these circumstances it is possible to have the current operation cease and control returned to the CPU. The program may then examine the address registers to determine where the inhibit has occurred, so that collision detection may be performed.

The Inner Loop is repeated until the inner counter reaches zero. This is an nine bit counter, and so can be any number of times from 1 to 512.

It is also possible to alter the address sources of the inner loop memory cycles to give line-drawing and a table look-up mode. Refer to the corresponding sections.

Copyright 1988

2.3.2. The Parameter Read Loop

The parameter read loop is a very straightforward sequence that loads up a new set of parameters to the inner loop.

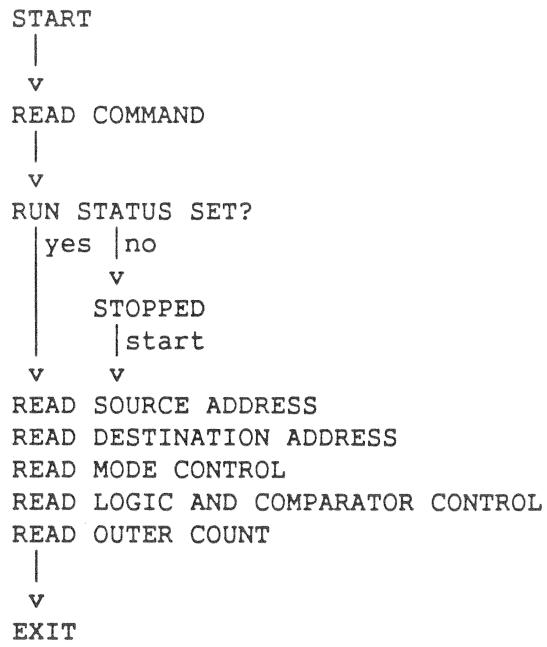
It reads, in sequence, the inner loop counter value, the step register value, and the pattern value - used to preset the source and destination data registers.

The inner count is the number of times the Inner Loop is repeated, the step register is used for address incrementing, and the pattern is used for data manipulation.

This loop is used as part of the command read operation at the start of a blitter operation; and also during a blitting operation between passes through the Inner Loop to allow various parameters of the loop to be altered. This allows such operations as irregular shape painting and run-length encoded data expansion.

2.3.3. The Command Read Loop

The command read loop is used to start a new blitting operation. The inactive, reset state of the blitter of the blitter is part of this loop. It looks like this:



Blitter Command Read Loop - Flow Diagram

The Stopped state of the Command read loop represents the normal inactive state of the blitter. From this state a Command register write is performed, usually preceded by a write of the program address register, and blitting operation starts.

The full set of operational parameters is loaded from the program count address, which is auto-incremented, and control passes out of the command read loop.

Copyright 1988

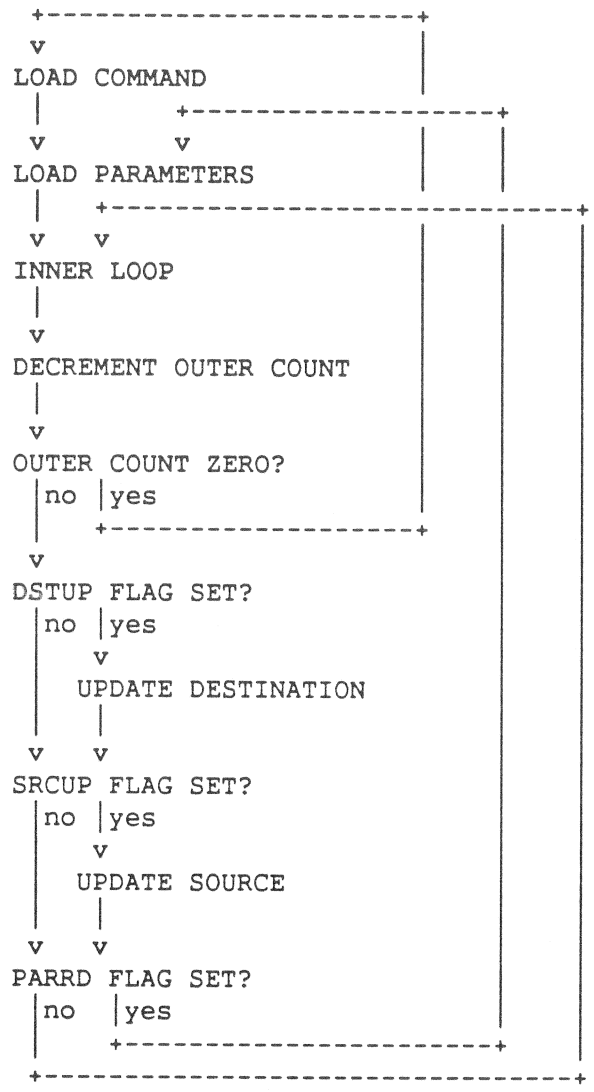
When a blitting operation is completed, a new command is read from the program count address, and if this command leaves the blitter in Run mode, then a new set of parameters is loaded and another operation is started. Otherwise the blitter enters its Stopped state and returns the bus to the CPU.

This mechanism allows the blitter to perform arbitrarily long sequences of graphics commands without requiring any processor intervention. This is extremely useful, as processor I/O write cycles are very slow in comparison to blitter memory reads.

Copyright 1988

2.3.4. The Outer Loop

The Outer Loop controls the overall operation of the blitter, including starting off the three inner loops described above. Its flow of control looks like this:



Outer Loop - Flow Diagram

Normal operation starts on exit from the command read loop. The parameter read loop then entered to read the first set of parameters, and then the inner loop is entered.

The presence of both an inner loop counter and an inner loop count register allows the inner counter to be reloaded with the same initial value on every pass round the outer loop.

The Inner Loop is then run, with the inner counter being loaded with this initial value before the start of operation.

The outer counter is then decremented, and if it is zero, then the command read loop is entered.

Copyright 1988

Then either or both of the source address and destination address register may be updated with the contents of the step register.

The parameter read loop may then be optionally entered to update various inner loop parameters, before the inner loop is entered again.

The two-level loop nature of the blitter allows it to cope with two-dimensional screen structures, with the source and destination address register updates moving screen pointer onto the start of the structure on the next line. The parameter read loop adds flexibility by allowing the screen structure parameters to be altered on a line by line basis.

Copyright 1988

3. MODES OF OPERATION

This section describes the normal modes of operation of the blitter, and how they should be programmed. An understanding of the previous section describing the mechanisms of the blitter is assumed.

3.1. Simple Memory Fill and Copy Operations

The simplest, and probably the most widely used operations are those involving copying one block of memory to another, and filling a block of memory with a pre-defined value. These operations can be performed on linear parts of memory, and on arbitrary screen rectangles.

The destination data register is used as the address of the memory being modified, and the source address register is used as the address of the data being copied, if it is a copy operation.

3.1.1. Handling of Data

The output data being given by the Logic Function Unit will be selected as Pattern data in the case of a memory fill, or as Source data in the case of a memory copy. For simple operations, the comparator control register will be set to zero.

3.1.2. Addressing and Counters

When the operation is to be performed on linear areas of memory, most of the address control bits - SRCUP, SRCINH, DSTUP, LINDR, YFRAC and LKUP - will be set to zero. The step register is not used, and the only requirement is to determine whether the copy will be made with the address incrementing or decrementing, and setting DSIGN and SSIGN appropriately. Note that the initial value placed in the address register should be the bottom of the area to be operated on if the sign bit is not set, and at the top if it is. In both cases, the first byte read or written will be the address first. The length of the operation will be placed in the inner counter, and the outer counter set to one.

If the block being operated on is very large, then the inner and outer counters may both have to be used, and the number of bytes operated on will be given by the product of the inner and outer counter values.

When either or both of the source and destination data are rectangles rather than linear areas, then the inner counter will contain the rectangle width, and the outer counter the rectangle height. The step register is set to the address increment from the right hand side of the rectangle round to the left hand side on the next line. The SRCUP and DSTUP bits are set according to whether the source or destination are rectangles.

3.1.3. Mode control

In low-resolution mode, loop type 2 will be used for memory fill, and loop type 1 for memory copy. In high-resolution mode, the same loop types can be performed, without the HIRES bit set, with the restriction

Copyright 1988

that all operations are in multiples of two pixels. If the HIRES bit is set, then loop type 0 must always be used, so that a destination read is performed to avoid corrupting the other pixel. Note that this will be slower.

Copyright 1988

3.2. Line Drawing

Line drawing is based in a simple D.D.A. (Digital Differential Analyzer) algorithm. The basis of this is that one of the X-address or Y-address is incremented for every pixel drawn, while the other one is incremented if a suitable arithmetic condition is met.

The algorithm used by the blitter computes this arithmetic condition by repeated subtraction - effectively division to give the gradient.

If a line is being drawn from (x_1,y_1) to (x_2,y_2), then:

$$\begin{aligned} Delta_x &= |(x_1-x_2)| \\ Delta_y &= |(y_1-y_2)| \end{aligned}$$

$Delta_1$ is given by the larger of $Delta_x$ and $Delta_y$, $Delta_2$ by the smaller. Then, for each pixel drawn, $Delta_2$ is subtracted from a working value, initially $Delta_1 / 2$, and the sign of the result of this is the arithmetic condition for the conditional screen address update. If this condition is met, then the original value of $Delta_1$ is added back to the result value.

It can be seen that the ratio of $Delta_x$ to $Delta_y$ will give the frequency with which this underflow and adding-back occurs. The ratio between them is, of course, also the gradient of the line.

These values are used to set the blitter command as follows:

The start point of the line (x_1,y_1) is the destination address

$Delta_1$ is placed in the middle byte of the source address register - this is the stored original value, and $Delta_1 / 2$ is placed in the low byte - this is the working value. Delta 1 also gives the inner counter value.

$Delta_2$ is placed in the step register.

If $Delta_x > Delta_y$ then the YFRAC flag is set, otherwise it is cleared.

SSIGN gives the sign of X-address updates, DSIGN the sign of Y-address updates.

In low-resoltion mode $Delta_1$ and $Delta_2$ are eight-bit values. However, in high-resolution mode it will be necessary to extend them to nine-bit values to get the necessary resolution of line-position. In this case, a further, less-significant, bit is used which would otherwise be set to zero. This is the so-called "minus one" bit.

The inner counter value, given by $Delta_1$, is also a nine-bit value, but in this case the **most** significant bit is place in ILCNT8, and the lower eight bits in the inner counter byte.

The least significant bit of $Delta_2$ is placed in STEP8, the least

Copyright 1988

significant bit of $Delta_1$ is placed in Source Address bit 16, and the least significant bit of $Delta_1$-1 is placed in Source Address bit -1.

Note that as $Delta_y$ is never more than 256, $Delta_2$ is similiarly restricted, so that in high-resolution mode its top bit (of nine) is always clear.

3.2.1. Handling of Data

As all the registers in the address section are occupied in computing the line address, the blitter has no ability to move data when drawing lines. Therefore the data written at the line address has to be given either directly by the pattern data or by a combination of the pattern register and the data already there, according to the L.F.U.

In hires or XOR modes, where loop type 1 is used, the read data goes to the destination data register, and the pattern data is held in the source data register.

3.2.2. Addressing and Counters

The addressing has largely been discussed above. The inner counter is set to the length of the line, given by $Delta_1$, and the outer counter will be set to one.

3.2.3. Mode control

In low-resolution mode, loop type 2 will be used for paint with pattern, and loop type 1 for read - modify - write operation. In high-resolution mode, loop type 1 must always be used, so that a destination read is performed to avoid corrupting the other pixel.

Copyright 1988

3.3. Character Painting

The blitter has the ability to paint characters on the screen as a single operation. Character painting as far as the blitter is concerned involves painting a rectangular area up to 8 pixels wide and of arbitrary height. The pixels in this area are either written to or left unchanged according to a bit pattern.

This mode of use is not restricted to character painting, but may be used for expanding any graphics stored as a monochrome bit plane.

3.3.1. Character Font

The source register addresses the bit pattern, normally part of the font, where each byte corresponds to one row of the character. Therefore blitter fonts may be up to 8 pixels wide. Wider fonts may of course be used, but these will require more than one blitter operation to paint a character.

The data is arranged with the bit corresponding to the left-most pixel in the least significant bit, and the top of a character at the lowest address. If the data is less than 8 pixels wide, then the least significant bits of the font data are not used. For example, the font entry for the letter R in a 6x10 font would be something like this:

Byte	Pixels set	Binary	Hex
0	xxxx	01111000	78
1	x x	10001000	88
2	x x	10001000	88
3	x x	01001000	48
4	xxx	00111000	38
5	x x	00101000	28
6	x x	01001000	48
7	x x	10001000	88
8	x x	10001000	88
9		00000000	00

3.3.2. Mode Control and Screen Addressing

The destination address register is used to address the area of the screen to which the character is to be painted. Normally this area will have been cleared to the required background colour by a previous blitter operation. The destination address is initialised to the top left hand corner of the character.

The character to be painted is a rectangle, and therefore the destination addressing is programmed correspondingly. The Inner Counter is set to the width of the character and the Outer Counter to its height. The step register is set to be the screen width less the width of the character. The DSTUP bit is set to allow the destination address to be updated between passes through the inner loop.

Inner loop type 3 is set in the control bits LT0 and LT1, character painting being the reason for its existence. This allows the font byte for each row to be read just once.

Copyright 1988

The comparator is used to control the painting of pixels, and so the CMPBIT control bit is set, to enable its bit to byte expansion mechanism.

The colour to be painted is set as the pattern, and held in the destination data register.

Copyright 1988

4. BLITTER COMMAND FORMAT

A blitter operation command is given as a table of data in memory. The blitter loads the contents of this table into its internal registers and performs the specified operation. The blitter will read successive sets of commands until a stop instruction is read into the command register.

The blitter program address must be set up before the command byte is issued. The blitter program address is given by three writeable registers, which contain an absolute system address.

WARNING - The blitter program address must ALWAYS be written in the order Byte 0 then Byte 1 then Byte 2. If this is not observed then the blitter program address may become corrupted.

A full table of blitter command data starts with a command byte. However the first blitter command in a sequence has its command byte written into the command register by a CPU I/O cycle, and so starts reading the command data from the second byte.

The blitter has four CPU I/O write registers, three of which are the programm address, and the fourth is the command register. See the table below.

A full blitter command instruction takes the form:

```
COMMAND
SOURCE ADDRESS LOW BYTE
SOURCE ADDRESS MIDDLE BYTE
SOURCE ADDRESS HIGH BYTE
DESTINATION ADDRESS LOW BYTE
DESTINATION ADDRESS MIDDLE BYTE
DESTINATION ADDRESS HIGH BYTE
MODE CONTROL BYTE
COMPARATOR AND LOGIC CONTROL BYTE
OUTER COUNT
INNER COUNT
STEP
PATTERN
```

The contents of the individual parts of the command bytes are described in the sections that follow. Here is a summary of the bit allocations:

Copyright 1988

0	1	2	3	4	5	6	7
RUN	COLST	PARRD	SRCUP	DSTUP	LT0	LT1	LINDR
---- Source address byte 0 ----							
---- Source address byte 1 ----							
Source address top nibble							SRCA-1
---- Destination address byte 0 ----							
---- Destination address byte 1 ----							
Destination address top nibble							DSTA-1
STEP8	ILCNT8	CMPBIT	HIRES	LKUP	YFRAC	DSIGN	SSIGN
CMPLT	CMPEQ	CMPGT	CMPBEQ	LOG0	LOG1	LOG2	LOG3
---- Outer loop counter value ----							
---- Inner loop count - low byte ----							
---- Step register value ----							
---- Pattern byte ----							

Figure - Blitter command summary

The blitter has the following registers in the CPU I/O space:

	I/O Address	Register
Write	20	Command Register
	18	Program Address - low byte
	19	Program Address - middle byte
	1A	Program Address - high byte
Read	20	Destination Address - low byte
	21	Destination Address - middle byte

See the warning above on blitter program address writes.

Copyright 1988

4.1. Command Register

0	1	2	3	4	5	6	7
RUN	COLST	PARRD	SRCUP	DSTUP	LT0	LT1	LINDR

RUN causes the blitter to start operation. It is used both to start the blitter reading a command, and to resume operation after a collision stop. If the blitter loads a command with RUN set to zero, as part of a command read, then operation ceases.

COLST causes operation to stop if a collision (write inhibit) occurs. From this point the current operation can be continued or aborted, and various internal registers may be read.

PARRD requires the blitter to read a new parameter set (that is the step register, the inner count and the pattern register) from the program counter address, every time the inner loop is exited.

SRCUP requires the contents of the step register to be added to the source address on exit from the inner loop.

DSTUP requires the contents of the step register to be added to the destination address on exit from the inner loop.

LT0-1 control the inner loop type.

> 00 - loop type 0 indicates that the inner loop will perform two read cycles and a write cycle. This is used either as read source and destination and write destination or in the table look-up mode.
>
> 01 - loop type 1 is composed of a read cycle and a write cycle. This is used either as read source and write destination or as a read and a write at the line draw address.
>
> 10 - loop type 2 is composed of only a write cycle. This is used for painting areas or blocks, and for line drawing.
>
> 11 - loop type 3 consists of a source read cycle on first entry to the inner loop, followed by only destination write cycles.

LINDR sets line-drawing mode. This mode uses both the source and destination address registers to generate the line-draw address, which may be used for both reading and writing.

Copyright 1988

4.2. Mode Control and Extension Bits Register

0	1	2	3	4	5	6	7
STEP8	ILCNT8	CMPBIT	HIRES	LKUP	YFRAC	DSIGN	SSIGN

STEP 8 is the least significant bit of the step (think of it as STEP-1). It is used a the nibble bit of the step when in high resolution mode.

ILCNT8 is the most significant bits of the inner loop count. It is normally only used when drawing long lines, or in high-resolution mode.

CMPBIT is used to give a bit to byte expansion scheme. It causes the comparator to generate an inhibit by selecting a bit of the source operand using the inner counter, and generating an inhibit if the bit selected is a zero. The selection is given by 8 in the inner counter selecting bit 0, 7 selecting bit 1, 6 bit 2, and so on.

HIRES sets the blitter into high resolution mode, where each pixel is half of a byte (4 bits) as opposed to a whole byte (8 bits).

DSIGN is the sign when updating the destination address. Setting it causes the destination address to be decremented rather then incremented. It makes Y negative in line-drawing.

SSIGN is the sign used when updating the source address. Setting it causes the source address to be decremented rather then rather then incremented. It makes X negative in line-drawing.

YFRAC indicates which of the X and Y addresses has the fractional increment in line-drawing mode. It is set if the Y address has the fractional increment.

LKUP is used to give a table look-up mode. It is used with loop type 0, and changes the addressing function to:

```
LD dest_data,(dest_addr)
LD src_data,(src_addr+dest_data)
LD (destination),src_data
```

This is used to give animation effects similar to the use of a palette map. Data from the program counter address is used as an index into the table pointed at by the source address, and the data from the table is then written. By changing the table, the same source data will produce different results when written.

Copyright 1988

4.3. Step Register

0	1	2	3	4	5	6	7
STEP0	STEP1	STEP2	STEP3	STEP4	STEP5	STEP6	STEP7

STEP 0-7 are the eight most significant bits of the step. The step is used to increment an address by a number other then 1, so that non-contiguous screen data may be handled in single operations. The step register is also used in line drawing mode to hold Delta Two.

Copyright 1988

4.4. Comparator and Logic Function Unit Control Register

0	1	2	3	4	5	6	7
CMPLT	CMPEQ	CMPGT	CMPBEQ	LOG0	LOG1	LOG2	LOG3

CMPLT causes the comparator to inhibit an inner loop write if the plane priority of the destination operand is less than the plane priority of the source operand.

CMPEQ causes the comparator to inhibit an inner loop write if the plane priority of the destination operand is equal to the plane priority of the source operand.

CMPGT causes the comparator to inhibit an inner loop write if the plane priority of the destination operand is greater than the plane priority of the source operand.

CMPBEQ compares the entire destination operand with the source operand, and inhibits the write operation if they are equal. This may be used for searching memory for particular values.

LOG 0-3 The logic function unit (L.F.U.) controls the data that is written in a destination write cycle. The L.F.U. allows any logical combination of the source and destination data. This is achieved by each of the L.F.U. bits selecting one of the Minterms, with the output being given by the OR of the selected terms.

The correspondence between bits and Minterms is

LOG0	NOT source AND NOT destination
LOG1	NOT source AND destination
LOG2	source AND NOT destination
LOG3	source AND destination

There are therefore sixteen possibilities.

Copyright 1988

4.5. Source Address Register

Source Address Low Byte

0	1	2	3	4	5	6	7
SRCA0	SRCA1	SRCA2	SRCA3	SRCA4	SRCA5	SRCA6	SRCA7

Source Address Middle Byte

0	1	2	3	4	5	6	7
SRCA8	SRCA9	SRCA10	SRCA11	SRCA12	SRCA13	SRCA14	SRCA15

Source Address High Byte

0	1	2	3	4	5	6	7
SRCA16	SRCA17	SRCA18	SRCA19				SRCA-1

The Source Address Register is used as a pointer to a source of data for blitting operations.

Bits SRCA0 to SRCA19 correspond directly to address lines 0 to 19. The SRCA-1 bit is used as a nibble address in high resolution mode.

The source address register is also used as Delta One in line-drawing mode.

Copyright 1988

4.6. Destination Address Register

Destination Address Low Byte

0	1	2	3	4	5	6	7
DSTA0	DSTA1	DSTA2	DSTA3	DSTA4	DSTA5	DSTA6	DSTA7

Destination Address Middle Byte

0	1	2	3	4	5	6	7
DSTA8	DSTA9	DSTA10	DSTA11	DSTA12	DSTA13	DSTA14	DSTA15

Destination Address High Byte

0	1	2	3	4	5	6	7
DSTA16	DSTA17	DSTA18	DSTA19				DSTA-1

The Destination Address Register is used as a pointer to the destination of the data in blitting operations.

Bits DSTA0 to DSTA19 correspond directly to address lines 0 to 20. The DSTA-1 bit is used as a nibble address in high resolution mode.

Copyright 1988

4.7. Program Address Register

Program Address Low Byte

0	1	2	3	4	5	6	7
PRCA0	PRCA1	PRCA2	PRCA3	PRCA4	PRCA5	PRCA6	PRCA7

Program Address Middle Byte

0	1	2	3	4	5	6	7
PRCA8	PRCA9	PRCA10	PRCA11	PRCA12	PRCA13	PRCA14	PRCA15

Program Address High Byte

0	1	2	3	4	5	6	7
PRCA16	PRCA17	PRCA18	PRCA19				

The Program Address points to the source of blitting operation commands. Data is read from it sequentially upwards through memory.

Note that these bytes must always be written in the order low byte, middle byte, then high byte.

Copyright 1988

CONNECTORS

This section gives the pinout of every connector used on Flare One.

1. EXPANSION CONNECTOR

The expansion connector brings out the main system bus and control lines. This allows extra peripheral hardware to be added to expand the functionality of the system. Its pinout is:

A0	1	2	A1
A2	3	4	A3
A4	5	6	A5
A6	7	8	A7
A8	9	10	A9
A10	11	12	A11
A12	13	14	A13
A14	15	16	A15
A16	17	18	A17
A18	19	20	A19
D0	21	22	D1
D2	23	24	D3
D4	25	26	D5
D6	27	28	D7
MREQ\	29	30	IORQ\
RD\	31	32	WR\
MCLKOUT	33	34	WAIT\
MCLKIN	35	36	INT\
XCLKOUT	37	38	BUSREQ\
XCLKIN	39	40	NMI\
HALT\	41	42	BUSACK\
M1\	43	44	RFSH\
ZCLK	45	46	+5v
ZA14	47	48	ZA15
AUXCS2\	49	50	0v
GPO4	51	52	GPO5
RESET\	53	54	ADC4
GPI0	55	56	GPI1
DSPBRQ\	57	58	DSPBAK\
BLITPC\	59	60	
	61	62	
-12v	63	64	+12v

Copyright 1988

2. VIDEO EXPANSION CONNECTOR

The video expansion connector allows the flare video output to be manipulated, mixed with other video signals, and allows capture from external video. Its pinout is:

R0	1	2	0v
R1	3	4	0v
R2	5	6	0v
G0	7	8	0v
G1	9	10	0v
G2	11	12	0v
B0	13	14	0v
B1	15	16	0v
HSYNC\	17	18	0v
VSYNC\	19	20	0v
CSYNC\	21	22	0v
RED OUT	23	24	0v
GREEN OUT	25	26	0v
BLUE OUT	27	28	0v
RED IN	29	30	0v
GREEN IN	31	32	0v
BLUE IN	33	34	0v
INC	35	36	0v
	37	38	0v
	39	40	0v

3. FLOPPY DISK DRIVE CONNECTOR

The floppy disk drive connector is a standard 34-way pin header, intended to be connected via ribbon cable to the internal floppy disk drive(s). Its pinout is:

0v	1	2	DS1
0v	3	4	
0v	5	6	
0v	7	8	IPL
0v	9	10	DS0
0v	11	12	DS1
0v	13	14	
0v	15	16	DS0
0v	17	18	DIR
0v	19	20	STEP
0v	21	22	WD
0v	23	24	WG
0v	25	26	TR0
0v	27	28	WPRT
0v	29	30	RD
0v	31	32	SIDE
0v	33	34	READY

Copyright 1988

INDEX

2's complement 27
Address Adder 42
Address Range 42
Address Updating 42
Arbitration 6
Architecture 6, 37
Bank registers 11, 21
Baud 22
Beam synchronous 22
Bit Allocation 39
Bit to Byte Expansion 39
Blitter 35
Blitter command 57
Blitter command summary . . . 58
Blue 19
Border 20, 23
Bus 6
Bus master 35
Bus priority 6
Byte Comparison 39
Character Painting 55
Clocks 6
CMPBIT 56, 60
Collision 59
Collision detection 46
Colour hold mode 23
COLST 59
Command Read 47
Command Register 59
Comparator 15, 38, 62
Contention 7
Contents 3
Controller 44
Copying Memory 51
D.D.A 53
DAC 6, 26
Data Path 37
Data registers 38
Delta One 63
Destination Address Register 64
DMA Latency 9
Drive select 16
DSIGN 60
DSP 6, 26, 35
DSP INSTRUCTION SET 27
DSTUP 59
External Address 42
Filling Memory 51
Flare 1
Floppy disk controller . . . 12
Font 55
Frame grabbing 19
Genlocking 23
Green 19
Harvard Architecture . . . 26
High Resolution 40
HIRES 60
Hires Caveats 40
Horizontal timebase counter 21
I/O Register 58
IBM PC standard keyboard . 14
incrustation 6, 19, 23
Inhibit 38
Inner Count MSB 60
Inner counter 46
Inner Loop 45
Interrupt 22, 35
Interrupt Acknowledge . . . 25
Interrupt Priority 10
Interrupts 10
Joystick 16
Keyboard 14
Keyboard interrupt 14
Latency 7
Lightpen 24
LINDR 59
Line Drawing 53
Line Drawing Address . . . 43
Logic Function Unit . . 38, 62
Look-up 60
Look-up Address 42
Loop type 59
Loop type 3 55
Low Resolution 40
MEMORY MAP 11
Minterms 38
Mode Control 60
Multiple Planes 38
Multiplier/accumulator . . 26
Muting 16
Outer counter 49
Outer Loop 49
Page number 11
Palette 20, 24
Parameter Read 47
PARRD 59
Pattern 38
Physical screen locations 18
Plane number 39
Pointer animation 10
Program Address Register . 65
Program Counter 42
Ready 16
Red 19
Refresh 7
RUN 59
Screen Map 18

Copyright 1988

Screen rectangles 51
Scroll registers 24
Searching memory 39
Sequencer 44
Shifter 40
Side select 16
Source Address Register . . . 63
SRCUP 59
SSIGN 60
Stack overflow 10
Step LSB 60
Step Register 61
Stopped 47
Synthesizer 26
Transparent 24, 39
Two-dimensional screen
structures 50
UART 12
Variable resolution mode . . 23
Variable-mode 19
Vertical timebase counter . 22
Video chip 18
Video fetch cycle 7
Wait 7
YFRAC 60
Z80 6
Z80 Execution Speed 7
Z80 IO 12