ATARI®

PANTHER

16-BIT

Atari Proprietary and Confidential

ATARI

Atari Corporation
330 N. Eisenhower Lane
Lombard, Illinois 60148
(708) 629-6500
Fax (708) 629-6699

Panther features

Processor	16 MHz 68000, 1.4 MIPS!
Memory	32 KBytes fast static-RAM 64 KBytes ROM on-board 8 KBytes Sound RAM
Cartridge	Up to 6 Mbytes (48 Mbits)
Video	320 x 240 pixels (programmable) non-interlaced 32 colours/line 7,680 colours/screen Palette of 262,144 colours RF, RGB and S-VHS outputs Genlock option
Sound	8 MIPs, 29 bit Digital signal processor 16 bit, stereo PCM sound 25 voices, each with independent volume, envelope, 4 pole digital filter, frequency control and pan 8K Bytes PCM RAM Twin, stereo headphone sockets
Graphics	32MHz, 32 bit, object oriented, graphics processor About 2,000 sprites can be displayed (and manipulated) simultaneously Object processor provides hardware scrolling, horizontally and vertically. Hardware scaling, (zoom and shrink) horizontally and vertically DMA. Run-length-decoding in hardware Fast hardware addition, for object manipulation Pixel-programmable interrupts
Joysticks	Two, multi-function joystick ports X-Y controller, 3 fire button, 12 key joystick as standard
Options	Paddles, lightgun, infra-red remote control CD-ROM, Modem, Genlock, Comlynx interface

RGM 7 Jan '91

PANTHER

Display Sub-system Specification

Contents

I. Overview

This document describes the display sub-system of Panther, a high performance games machine based on the Motorola 68000 microprocessor. This document does not cover the sound processor or the I/O capabilities.

The display is generated by an object list processor. For each display line, a list of objects is executed by this processor and used to build that display line in an internal buffer. The objects in the list may specify bit-map images or run-length encoded images. Other objects in the list provide a variety of useful functions. This mechanism allows a 'sprite' based display to be generated, with considerable flexibility. The object list processor can also perform hardware zooming on bit-map data.

The number of sprites is limited only by the number of objects that can be processed in one video line. To enhance the performance of the object list processor, the system contains 32-bit wide fast static RAM, from which the object list is read by the object list processor.

II. Object List Processor

A. Overview

Everything visible on the screen, with the exception of the border, is an object. Objects are of variable size and have a data format described by an object header. Each object on the screen has a unique object header, although two or more objects may share the same data. The object header also specifies the position of the object and an offset into the colour palette.

The display is built into the line buffer, on a line-by-line basis, by the object list processor. Each line is built while the previous line is being displayed, so this building process must be completed during one display line.

The object list processor acquires the bus and starts executing the object list from the address given by the OLP register automatically. It will continue to execute objects from the list until an interrupt object is executed.

Object headers are a whole number of longwords in length and are stored in the object list, which must lie in 32-bit RAM on a longword boundary. Objects are painted in the order they appear in the object list. The first object therefore has the lowest priority (usually the background), and the last object has the highest priority. The object headers form a linked list, with a link address pointing to the next object, with the exception of the memory move objects which are executed sequentially as they contain no link (for faster execution). A branch object performs no action other than to provide a link address.

The first object header in the object list (OL) is pointed to by the object list pointer register (OLP) in the object list processor. This register is word wide and specifies the offset in longwords into RAM. That is:

OL address = OLP * 4

In addition to objects used to display data, there are special objects which can be used to manipulate data in memory in a way that is synchronized with the display. These may be used for such tasks as modifying the object list itself, for moving data to the sound controller, or changing the palette.

B. Objects

Objects are specified by object headers. The first byte of the header specifies the object type. The meaning of the remaining bits depends on the object type.

Objects are made up from one or more longwords, which must lie on longword boundaries. The diagrams below show how the data is arranged within these longwords.

1. **Scaled bit-mapped object** (0x01 - 0x7F)

31 ... 24 23 ... 16 15 ... 8 7 ... 0

type 0x01-0x7F	Y position	Y size	?	depth
Y scale	unused	link		
data width	drawn width	X position		
palette	data address			R

Object type — is a byte specifying the type of object, 0x20 for bit-mapped object with no scaling (see discussion below).

Y position — is a byte in the range 0-199 specifying the top line of the object (register **VDB** specifies where line zero falls).

Y size — is an eight bit integer with a five bit fraction, which specifies the 'data height' of the object. This is decremented by Y scale on every display line, and must be initialised every field. When this is decremented past zero the value zero is written back to the object, and this indicates that the object data has been exhausted.

? — is an unused bit.

Depth — is a two bit field which specifies the number of data bits per pixel. 0 = 1 bit, 1 = 2 bits, 2 = 4 bits, 3 = 8 bits.

Y scale — is a five bit integer with a five bit fraction which determines the vertical scaling. The data address is advanced by Y scale data lines for every one display line. This number is the reciprocal of the horizontal scaling factor (in the object type).

Link — is the 22 bit address of the next object to be processed. Objects are aligned to longwords.

Palette — is a 8 bit field specifying the starting offset into the palette.

Data address — is the full 23 bit word address of the object data.

R — is a bit specifying if the object should be reflected in X

Data width — is a 10 bit field which specifies the offset in words to the beginning of the next line.

Drawn width	is a 10 bit number specifying the number of words of pixel data to be drawn. This may be less than the data width where the software wishes to accelerate clipping.
X position	is a 12 bit field specifying the signed starting offset into the line buffer (-2048 to 2047 to allow clipping).

The data is stored in packed format where the N most significant bits of the word are the first pixel. The next N bits are the second pixel and so on. N has a value of 1, 2, 4, or 8 and is specified by the depth field. Note that only the least significant 5 bits will be used in 8 bit mode. For compatibility with future machines the upper three bits should be zero.

Bit-map data in thirty-two bit memory (RAM) must be long aligned, and must have a width which is an even number of words. In sixteen bit memory (ROM) the data should be word aligned.

A data value of zero is always interpreted as transparent and will not modify the line buffer. Data values other than zero will be added to the value of the palette field, and will replace the appropriate pixels of the line buffer.

The display processor's procedure for handling a bit-mapped object is:

```
        Fetch data address
        Current X = X position
loop:   DATA = bit N of data
        Increment N
        If DATA != 0 then Line buffer[Current X] = DATA + Palette
        Increment Current X (or Decrement if reflected)
        If all data is used then goto done
        If Not Reflected
            If Current X <= 319 then goto loop
        Else
            If 0 <= Current X then goto loop

done:   Compute new data address
        Compute new Y size
        Write back to object
        Process next Object
```

The new data address is computed as follows:

i) Add the data width to the data address N times where N is the integer part of the Y scaling factor.

ii) Subtract the Y scaling factor from the Y size, and if there is a borrow out of the fractional part, then add the data width to the data address once more.

Note that the data address in the object header is modified to point to the start of the next data line. Since the data address and Y size are being modified during the display, they should be reset at the end of the frame. This can be done by the CPU or by one of the special data manipulating objects.

X direction scaling is specified in the object type. Bits 6-0 provide a 7 bit scale factor. The decimal point is after bit 5. Thus, 0x20 is 1, 0x01 is 1/32nd, 0x60 is 3, etc. Scaling is accomplished by dropping or repeating pixels as appropriate.

Reflected objects behave just like unreflected ones except that the data is placed in the line buffer starting at the X position and moving to the left instead of the right. The net result is that the object is reflected in X and the X position specifies the right side of the object instead of the left.

2. **Run length object** (0x00)

31 – 24	23 – 16	15 – 8	7 – 0		
type 0x00	Y position	Y size		B	depth
Y scale	unused	link			
unused			X position		
palette	data address				R

This object is similar to the bit-mapped object, the differences being that there are no width fields, the data format is different, and there is an additional flag bit - B.

The run length data is a word count specifying the number of words in the run (including the count itself), followed by the data which consists of words containing colour/length pairs. Note that in thirty-two bit memory the run count is contained in the top word of a longword whose bottom word is unused, and that the run data must be on a longword boundary. In sixteen bit memory the count is just one word, and word alignment is required.

Each run item is a word with the more significant byte containing the width of the run (0-255), and the less significant byte containing the colour. In thirty-two bit memory the word at the lower address in the longword is executed first.

The Y scale should be given the value 1.0 unless all runs are the same length, in which case it can be used for vertical scaling.

Bit B will cause the run to be expanded from the end of the run to the front. This feature can be used to prevent the object processor wasting cycles expanding runs which do not fall on the screen.

3. **Load palette object** (0x80)

31 – 24	23 – 16	15 – 8	7 – 0
type 0x80	Y position	unused	offset
count	data address		

Y position	is a byte in the range 0-199 specifying the line of display on which the command is executed. A position of 255 will cause the command to be executed every line.
Offset	is the number of bytes from the palette base at which the transfer starts.
Data address	is the full 24 bit address of the first byte to be transferred.

When this object is encountered the display processor will 'remember' about it and at the next Hsync will copy **count** bytes from the data address to the palette starting **offset** bytes into the palette. Placing this object anywhere in the object list will cause the new palette to be in effect for the line in which this object is enabled. The data address is a 24 bit byte address.

The palette can be reloaded during the line using one of the following special objects. However, cycle counting will have to be done to determine where the reload will happen (i.e. there is no hardware support to aid in reloading the palette at a particular pixel), and there is a potential for causing flicker if the colour currently being displayed is changed.

4. Branch object (0x81)

31 – 24	23 – 16	15 – 8	7 – 0
type 0x81	Y position	unused	
unused		link	

Y position	is a byte in the range 0-199 specifying the line of display on which the branch command is executed. A position of 255 will cause the command to be executed every line.
Link	is the address of a longword containing the object to be branched to if the command is executed.

The branch object directs the object processor to the object at the link address if the Y position matches the current line. Otherwise execution continues with the following object in memory.

5. Move Longword to Memory Immediate (0xE0)
6. Move Word to Memory Immediate (0xD0)
7. Move Byte to Memory Immediate (0xC0)

31 – 24	23 – 16	15 – 8	7 – 0
type 0xE0	Destination address		
Longword data			

31 – 24	23 – 16	15 – 8	7 – 0
type 0xD0	Destination address		

unused	Word data

type 0xC0	Destination address
unused	Byte data

These objects cause the Longword, Word or Byte data specified in the object header to be written to the destination address.

8. Add Longword to Memory Immediate (0xE1)
9. Add Word to Memory Immediate (0xD1)
10. Add Byte to Memory Immediate (0xC1)

31 ... 24	23 ... 16	15 ... 8	7 ... 0
type 0xE1	Destination address		
Longword data			

type 0xD1	Destination address	
unused		Word data

type 0xC1	Destination address	
unused		Byte data

These objects cause the Longword, Word or Byte data specified in the object header to be added to the destination address.

11. Move Longword to Memory Indirect (0xE8-0xEF)
12. Move Word to Memory Indirect (0xD8-0xDF)
13. Move Byte to Memory Indirect (0xC8-0xCF)

31 ... 24	23 ... 16	15 ... 8	7 ... 0
type	Destination address		
Count	Source address		

When the display processor encounters this object it will move **count** longwords, words or bytes from the **source** address to the **destination** address.

If bit 2 in the type byte is set the source address will be increased (internal to the display processor, not in the object header) by the size of the data being transferred (i.e. 1 for bytes, 2 for words and 4 for longs). The destination address will be increased according to the value of bits 1 and 0 (00 adds 0, 01 adds 1, 10 adds 2 and 11 adds 4). Note that the destination increment should either be zero or be greater than or equal to the size of the data being transferred.

14. Interrupt Display Processing (0xF0-0xF7)

31 – 24	23 – 16	15 – 8	7 – 0
type 0xF0-0xF7	Y position	Vector	unused

These objects terminate the display list processing for the current line (processing for the next line will start automatically). The interrupt level is given by the type. Level zero does not cause an interrupt. Levels 2 and 6 are used by the video interrupt and the external interrupt, and should not be used.

Vector specifies which vector is supplied to the 68000 when it acknowledges the interrupt. If this interrupt conflicts with another pending interrupt of the same level, then this interrupt will be serviced first.

III. Programmable Display Generator

The programmable display generator produces all display related timing. It allows the programmer to select NTSC or PAL, the absolute screen position, and number of active lines per screen. All of these parameters can be modified on a frame by frame basis if desired.

The display generator has two timing generators, horizontal and vertical. Each is controlled by a set of registers giving Period, Sync Start, Border Start, Border Stop, Display Start, and Display Stop. The table below shows the gives recommended values for NTSC and PAL. Horizontal times are specified in pixels and vertical times are specified in lines. They must be set to the correct value for either NTSC or PAL in order for ordinary television sets to operate properly. Note that there is no horizontal display end register, the display width is fixed (at 320 pixels).

A one bit register (SMODE) allows the programmer to find out the television system for which the machine is configured (0=NTSC, 1=PAL).

Display values	Horizontal			Vertical		
	name	NTSC	PAL	name	NTSC	PAL
Period	HP	480	480	VP	260	312
Sync start	HS	450	450	VS	240	292
Border Start	HBB	420	420	VBB	235	287
Border Stop	HBE	30	30	VBE	5	5
Display Start	HDB	90	90	VDB	25	25
Display Stop				VDE	1023	1023

The above horizontal parameters are proportional to the crystal frequency; which the table assumes to be 30MHz.

When the machine is powered up, the programmable display generator and the display processor are disabled. The display generator should be set up for the correct television system, the display processor should be set up with a valid object list (see following sections) and then the video subsystem should be enabled by writing a one to **VIDEN**.

Bit zero of **VIDEN** controls sync and display processing (1=enable). Bit one can be used to disable the object list processing (1=enable), the screen will display the background colour while this is disabled.

The display start registers (**HDB** and **VDB**) serve several purposes:

- They indicate the beginning of the active portion of the screen.
- They are the base count from which actual screen positions are measured.
- They initiate display processing.

On each line starting with **VDB** and ending with the line before **VDE**, when the internal counter reaches **HDB**, the object list processor will take control of the machine, disabling the CPU. As this happens the current line buffer is loaded into the shifter to be sent to the display, and the line buffer may be initialised with the value in the **BINIT** register if the most significant bit of the **BINIT** register is set, if the most significant bit is cleared, then the next line will be built on top of the existing line.

The object list processor will build the line buffer for the next line to be displayed. When it finishes with the object list the CPU will regain control. This means that the object list must be able to be handled in one scan line.

An interrupt is provided to help in synchronizing the CPU and display. This provides a level 2 auto-vectored interrupt on any pixel of the screen (visible or not).

This interrupt is controlled by the horizontal and vertical interrupt registers (**HI** and **VI**). **HI** and **VI** are specified in number of pixels and number of lines relative (signed) to the start of the screen. These registers may be reloaded during the frame to produce several interrupts. One note of caution. If the interrupt happens while the CPU is hung by the display processor, the interrupt will not be serviced until the display processor finishes.

IV. Colour Palette

The colour palette provides the mapping between the 5 bit internal data representation and the 6 bit red, 6 bit green and 6 bit blue colour outputs. Entry 0 in the palette corresponds to an internal data value of zero. Entry 1 corresponds to an internal data value of 1, and so on. There are 32 entries in the colour palette.

Because data of value zero is considered transparent by the object list processor, it is never written into the line buffer by the object list processor. A zero data value can only be output under three circumstances:

- The border is zero.
- The display processor can be set up to initialize the line buffer to zero, using register BINIT, and no objects are written over it.
- The CPU can write zero to the line buffer.

The palette entries are arranged as a byte of each red, green and blue. Only the lower six bits of each byte are used. For compatibility with future products the upper two bits should be zero. The three bytes are arranged as the most significant three bytes of a longword. Entry 0 is at the palette base address (PALETTE) with entry 1 at next longword, then entry 2 and so on. Only byte cycles may be performed on the palette.

	MSB			LSB	
PALETTE	Unused	Red	Green	Blue	Entry 0
PALETTE + 4	Unused	Red	Green	Blue	Entry 1
	|	|	|	|	
PALETTE + 124	Unused	Red	Green	Blue	Entry 31

V. Memory Map

A. Overview

000000-000007	Reset vector Maps onto 800000-800007 on supervisor program accesses only
000000-017FFF	Internal RAM
018000-7FFFFF	External RAM (Potential)
800000-8FFFFF	Cartridge ROM 1 - bootstrap
900000-9FFFFF	Cartridge ROM 2
A00000-FF7FFF	Cartridge ROM (Potential)
FF8000-FFFFFF	Devices May be accessed in supervisor mode only, user mode accesses cause bus error.

B. Registers

FF9200	xxxx xxxx xxxx xxxx	ro	JOY	Joystick inputs
FF9200	---- ---- xxxx xxxx	wo	JOY	Joystick outputs
FFB000 - FFB0FF				Sound generator registers
FFC012	xxxx xxxx xxxx xxxx	rw	OLP	Object list pointer
FFC020	x000 0000 000x xxxx	rw	BINIT	Line buffer initial value
FFC030	---- ---- xxxx xxxx	ro	PAD0X	Paddle 0 X position
FFC032	---- ---- xxxx xxxx	ro	PAD0Y	Paddle 0 Y position
FFC034	---- ---- xxxx xxxx	ro	PAD1X	Paddle 1 X position
FFC036	---- ---- xxxx xxxx	ro	PAD1Y	Paddle 1 Y position
FFC038	---- --xx xxxx xxxx	ro	LP0X	Light Pen 0 X position
FFC03A	---- --xx xxxx xxxx	ro	LP0Y	Light Pen 0 Y position
FFC03C	---- --xx xxxx xxxx	ro	LP1X	Light Pen 1 X position
FFC03E	---- --xx xxxx xxxx	ro	LP1Y	Light Pen 1 Y position
FFC0F0	---- ---- ---- ---x	ro	SMODE	Display configuration mode
FFC100	---- --xx xxxx xxxx	rw	VP	Vertical period
FFC102	---- --xx xxxx xxxx	rw	VS	Vertical sync start
FFC104	---- --xx xxxx xxxx	rw	VBB	Vertical blank start
FFC106	---- --xx xxxx xxxx	rw	VBE	Vertical blank stop
FFC108	---- --xx xxxx xxxx	rw	VDB	Vertical display start
FFC10A	---- --xx xxxx xxxx	rw	VDE	Vertical display stop
FFC10C	---- --xx xxxx xxxx	rw	VI	Vertical interrupt
FFC10E	---- --xx xxxx xxxx	ro	VCP	Vertical current position
FFC110	---- --xx xxxx xxxx	rw	HP	Horizontal period
FFC112	---- --xx xxxx xxxx	rw	HS	Horizontal sync start
FFC114	---- --xx xxxx xxxx	rw	HBB	Horizontal blank start
FFC116	---- --xx xxxx xxxx	rw	HBE	Horizontal blank stop
FFC118	---- --xx xxxx xxxx	rw	HDB	Horizontal display start

FFC11C	---- --xx xxxx xxxx	rw	HI	Horizontal interrupt
FFC11E	---- --xx xxxx xxxx	ro	HCP	Horizontal current position
FFC120	---- ---- 0000 00xx	rw	VIDEN	Video enable
FFC122	---- ---- ---- ----	rw	TEST1	Manufacturing test only
FFC124	---- ---- ---- ----	ro	TEST2	Manufacturing test only
FFC126	---- ---- ---- ----	ro	TEST3	Manufacturing test only
FFC128	---- ---- ---- ----	ro	TEST4	Manufacturing test only

The palette and line buffer locations may only be accessed in byte wide cycles.

FFD001	00xx xxxx	rw	PALETTE	Palette - Entry 0 - Red
FFD002	00xx xxxx	rw		Palette - Entry 0 - Green
FFD003	00xx xxxx	rw		Palette - Entry 0 - Blue
FFD005	00xx xxxx	rw		Palette - Entry 1 - Red
FFD006	00xx xxxx	rw		Palette - Entry 1 - Green
FFD007	00xx xxxx	rw		Palette - Entry 1 - Blue
.				
.				
.				
FFD07D	00xx xxxx	rw		Palette - Entry 31 - Red
FFD07E	00xx xxxx	rw		Palette - Entry 31 - Green
FFD07F	00xx xxxx	rw		Palette - Entry 31 - Blue
FFE000	000x xxxx	wo	LBUF	Line buffer - 0
FFE001	000x xxxx	wo		Line buffer - 1
FFE002	000x xxxx	wo		Line buffer - 2
.				
.				
.				
FFE139	000x xxxx	wo		Line buffer - 319