Specification for

a High Performance 68000 Based Game Machine

Overview

This document describes a high performance game machine based on the Motorola 68000 microprocessor. This machine will have the following features:

- 68000 16/32 bit microprocessor running at 16MHz
- 32K bytes fast static RAM, externally expandable to 8M bytes
- ROM cartridges of up to 7.5M bytes
- Cartridges capable of saving game state and high score
- Cartridges require electronic key
- Custom 32 bit display processor
- 320 pixel maximum horizontal resolution
- 32 colors per line from a palette of 262,000 (6 8-bit DACs)
- Complete software control over screen position and resolution
- Hardware bitwise horizontal and vertical scrolling
- Hardware control of an unlimited number of two dimensional objects
- Split screen operation
- Interrupt on any arbitrary screen position
- Support for 4 standard joysticks or 2 enhanced (ST+ style) controllers
- Support for 4 console buttons
- Support for 2 pairs of paddles
- Support for 2 Light Guns/Pens
- Superior 6 voice, 8 bit sound with DMA to DAC capability
- Modulator output
- ST style RGB connector
- Composite video, Stereo and Monural sound supplied on phono jacks
- Two independent stereo headphone jacks
- Software determination of machine configuration (NTSC, PAL, etc.)

This machine can be divided into several main subsections; The Programmable Display Generator, The Display Processor, The Color Palette, The Sound Generator, and The Controller Interfaces. Briefly, the programmable display generator is responsible for generating horizontal and vertical synchronization as well as all the other timing information required by the display processor. The display processor handles the manipulation of all video data and related structures. The color palette provides the mapping from the 5 bit internal data representation to the 6 bits of red, green, and blue actually displayed. The sound generator and controller interfaces are self explanatory.

Move Long to Memory Indirect (0xE8-0xEF) (RAM to RAM transfers only)
Move Word to Memory Indirect (0xD8-0xDF)
Move Byte to Memory Indirect (0xC8-0xCF)

The object header for a move to memory indirect has the following format:

Object type	Destination Address
Count	Source Address

When the display processor encounters this object it will move "Count" words or bytes from the specified source address to the destination address. If bit 2 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 transfered (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, the destination increment must either be zero or must be greater than or equal to the size of the data being transfered.

Interrupt Display Processing (0xF0-0xF7)

The object header for an interrupt display processing object has the following format:

Object type	Vector	

When this object is encountered, display processing will stop and the CPU will regain control. A vectored interrupt will immediately be generated. The level of the interrupt will be seven for type 0xF7, six for type 0xF6, ... one for type 0xF1 and no interrupt for type 0xF0. The vector that will be supplied is the one specified in the object header. This byte is meaningless for type 0xF0. If this interrupt conflicts with another pending interrupt of the same level, this interrupt will be serviced first. Display processing can be resumed by reading *LALA*.

Memory Considerations

To display 100 four bit objects 16x16 bits in size on a 200 line screen the following amount of RAM is required:

```
200 x (101 + 32) / 32 =  800        For OLEV list
101 x 8               =  808        For OL
                        ----
                        1608 bytes
```

Note that 101 was used in the calculations to account for a background.

Controllers

All controllers, with the exception of the four console buttons, connect through one of two 15 pin controller connectors. These connectors, through the use of an adaptor, can each accommodate two standard 9 pin joysticks or one standard 9 pin paddle pair or one standard 9 pin light gun/pen. Since there are two connectors capable of handling four standard joysticks, we will refer to the connectors as JOY0 and JOY1 and we will refer to the "sub" joysticks as A and B. Thus the four joystick connections are JOY0A, JOY0B, JOY1A, and JOY1B.

Console and Fire Buttons

The console and fire buttons both appear in the same byte register (*FB*). These lines are normally high and go low when the button is depressed. They are arranged in the byte as follows:

```
MSB  Console button 4
     Console button 3
     Console button 2
     Console button 1
     JOY1B Fire button
     JOY1A Fire button
     JOY0B Fire button
 LSB JOY0A Fire button
```

FB may be read at any time and always reflects the current state of the buttons. In addition to the four console buttons there will be a hardware reset button.

Joysticks

A 16 bit register (*JOY*) provides the direction bits for the four joysticks. All bits are normally high and go low when the controller is moved in the appropriate direction. The bits are arranged in the word as follows:

```
MSB  JOY1B Up
     JOY1B Down
     JOY1B Left
     JOY1B Right
     JOY0B Up
     JOY0B Down
     JOY0B Left
     JOY0B Right
     JOY1A Up
     JOY1A Down
     JOY1A Left
     JOY1A Right
     JOY0A Up
     JOY0A Down
     JOY0A Left
 LSB JOY0A Right
```

JOY may be read at any time and always reflects the current state of the joysticks.

Paddles

Two standard paddle pairs can be supported. One plugged into JOY0A and one in JOY1A. Four 8 bit registers provide the current paddle position, *PAD0X*, *PAD0Y*, *PAD1X* and *PAD1Y*. The current paddle positions will be assessed during the active portion of the screen and is guaranteed to remain stable during vertical blanking. Due to tolerances in parts and paddles, the minimum and maximum value for the paddle registers will vary from machine to machine and paddle to paddle. In all cases the maximum value will occur with the paddle in a full counterclockwise position and the minimum value will occur with the paddle in a full clockwise position. The fire buttons are normally high, low to fire and are mapped as follows:

Paddle 0 X fire button - JOY0A Left
Paddle 0 Y fire button - JOY0A Right
Paddle 1 X fire button - JOY1A Left
Paddle 1 Y fire button - JOY1A Right

Light Gun / Pen

Two standard light guns can be supported. One plugged into JOY0A and one in JOY1A. Four 10 bit registers provide the current light gun position, *LP0X*, *LP0Y*, *LP1X* and *LP1Y*. The current light gun positions will be assessed during the active portion of the screen and is guaranteed to remain stable during vertical blanking. The X position is the 10 bit signed offset in pixels from *HDB*. The Y position is the 10 bit signed offset in lines from *VDB*. The fire buttons are normally low, high to fire and are mapped as follows:

Light gun 0 - JOY0A Up
Light gun 1 - JOY1A Up

Note: These registers report the pixel being displayed at the time the light gun detects the beam. Because of latencies caused by CRT phosphors and light gun optics, this may be 30 to 70 pixels late. For precision games it might be wise to have a "sighting" procedure to determine the proper constant to subtract from the X position register. The Y position should not be affected.

Super Controllers

Actual super controllers are yet to be designed. However, they will be designed to take advantage of the 6 input and 4 I/O lines (and possibly paddle inputs) available on the 15 pin connector.

The "A" joystick direction lines are bidirectional. They will assume the direction of the last access to them. If read they will be inputs, if written to they will be outputs until next read. They can be used in conjunction with the "B" direction lines to scan a matrix. This is accomplished by writing the low byte of *JOY* which contains the "A" lines then reading the high byte of *JOY* which contains the "B" lines.

Sound

The sound generator has the following features:

- 6 Tone generators (0 - 8kHz) with linear frequency control
- Each tone generator is capable of generating Sawtooth, Triangle, Variable Pulse Width and Noise. (One noise generator not six)
- Programmable Attack, Decay, Sustain and Release segments of the envelope generator
- Exponential envelope capability
- Waveform syncronization between voices
- Ring modulation capability
- 48 dB dynamic amplitude range
- Programmable Low pass / High pass / Band pass / Notch filter
- Four output channels, any or all voices can be switched to any or all channels

There are six voices. Each voice consists of a tone generator which produces either a sawtooth, triangle or pulse waveform or can be replaced by a random noise output. The envelope is shaped by a programmable attack, decay, sustain, release (ADSR) sequence. These six outputs can be individually switched onto any or all for four output channels. Each output channel is high and low pass filtered and has its own volume control.

Registers per voice (6 voices):

```
xxxx xxxx Control register
00xx xxxx Frequency (high)
xxxx xxxx Frequency (low)
00xx xxxx Pulse width (high)
xxxx xxxx Pulse width (low)
0000 xxxx Attack
0000 xxxx Decay
0000 xxxx Sustain
0000 xxxx Release
xxxx xxxx Oscillator output (read only)
xxxx xxxx Envelope output (read only)
```

Control - xxxx xxxx

```
Gate
Sync
Ring Mod
Reset (Exponential decay?)
Triangle
Sawtooth
Pulse
Noise
```

Noise - When set, this voice will output pseudo-random noise at the specified frequency.

Pulse - When set, this voice will output a pulse of the specified width and frequency.

Sawtooth - When set, this voice will output a sawtooth waveform of the specified frequency.

Triangle - When set, this voice will output a triangle waveform of the specified frequency.

Reset - When set, performs a software reset of the voice.

Ring Mod - When set, the triangle waveform of this voice is replaced with a ring modulated combination of this voice and the next voice.

Sync - When set, will synchronize the frequency of this voice to the frequency of the next voice.

Gate - When set, the attack, decay, sustain portion of the envelope is initiated. When cleared, the release portion of the envelope will begin.

Frequency - The frequency registers are a pair of registers which contain a 12 bit number that linearly specify the frequency of the tone generator according to the formula:

Frequency out = (N x ?_____?) Hz

Where N is the value in the frequency register.

Pulse Width - The pluse width registers are a pair of registers which contain a 12 bit number that specifies the pulse width of the pulse waveform according to the formula:

Pulse width = (N x ?_____?) %

Where N is the value in the pulse width register.

Attack - A four bit register specifying the attack rate of the envelope.

Value	Time	Value	Time
0	2 mS	8	100 mS
1	8 mS	9	250 mS
2	16 mS	A	500 mS
3	24 mS	B	800 mS
4	38 mS	C	1 S
5	56 mS	D	3 S
6	68 mS	E	5 S
7	80 mS	F	8 S

Decay - A four bit register specifying the decay rate of the envelope.

Value	Time	Value	Time
0	6 mS	8	300 mS
1	24 mS	9	750 mS
2	48 mS	A	1500 mS
3	72 mS	B	2400 mS
4	114 mS	C	3 S
5	168 mS	D	9 S
6	204 mS	E	15 S
7	240 mS	F	24 S

Sustain - A four bit register specifying the sustain amplitude of the envelope.

Release - A four bit register specifying the release rate if the envelope. Values specify same times as the decay register.

Osc Output - An eight bit read only register which reports the current output of the waveform generator. This register will count from 0 to 255 for sawtooth, from 0 to 255 and back down to 0 for triangle, and will oscillate between 0 and 255 for pulse.

Env Output - An eight bit read only register which reports the current value of the envelope generator.

Registers per output channel (4 output channels):

```
00xx xxxx Voice enable
00xx xxxx Filter enable
00xx xxxx High pass cutoff frequency (high)
xxxx xxxx High pass cutoff frequency (low)
00xx xxxx Low pass cutoff frequency (high)
xxxx xxxx Low pass cutoff frequency (low)
00xx xxxx Volume
```

Voice enab - A six bit register, each bit corresponds to a voice. When the bit is set, that voice is enabled onto this output channel. Bit 0 corresponds to voice 0, bit 1 to voice 1, etc.

Filter enab - A six bit register, each bit corresponds to a voice. When the bit is set, that voice is passed through the filters before being enabled onto this output channel. Bit 0 corresponds to voice 0, bit 1 to voice 1, etc.

HP cutoff - A pair of registers which together make a 12 bit register which specifies the 3 dB cutoff point for the high pass filter. The frequency is specified according to the formula for the fundamental frequency.

LP cutoff - A pair of registers which together make a 12 bit register which specifies the 3 dB cutoff point for the low pass filter. The frequency is specified according to the formula for the fundamental frequency.

Volume - A four bit register which specifes the volume for the channel. A value of 15 is full volume and a value of 0 is minimum volume.

Misc:

```
xxxx xxxx Random number (high)
xxxx xxxx Random number (low)
```

This random number (also used for noise) is generated by a 31 bit shift register. The outputs of the 28th and 31st stages are XNORed together and fed back to the input of the shift register. The shift register is clocked at 16MHz. These registers present the current value of the lower 16 bits of the shift register.

Memory Map

Overview

000000-000007	Reset vector Maps to 800000-800007 on supervisor program accesses only
000000-007FFF	Internal RAM
008000-7FFFFF	External RAM (Potential)
800000-FF7FFF	Cartridge ROM (Potential)
FF8000-FFFFFF	Devices (May be accessed in supervisor mode only) (User mode accesses cause bus error)

Registers

FF9200	---- ---- xxxx xxxx	ro	*FB*	Fire / Console buttons
FF9202	xxxx xxxx ---- ----	ro	*JOY*	Joystick directions
FF9202	---- ---- xxxx xxxx	rw	*JOY*	
FF9210	---- ---- xxxx xxxx	ro	*PAD0X*	Paddle 0 X position
FF9212	---- ---- xxxx xxxx	ro	*PAD0Y*	Paddle 0 Y position
FF9214	---- ---- xxxx xxxx	ro	*PAD1X*	Paddle 1 X position
FF9216	---- ---- xxxx xxxx	ro	*PAD1Y*	Paddle 1 Y position
FF9220	---- --xx xxxx xxxx	ro	*LP0X*	Light Pen 0 X position
FF9222	---- --xx xxxx xxxx	ro	*LP0Y*	Light Pen 0 Y position
FF9224	---- --xx xxxx xxxx	ro	*LP1X*	Light Pen 1 X position
FF9226	---- --xx xxxx xxxx	ro	*LP1Y*	Light Pen 1 Y position
FFB000	---- ---- xxxx xxxx	rw	*CR0*	Voice 0 Control register
FFB002	---- ---- 00xx xxxx	rw	*FH0*	Voice 0 Frequency (high)
FFB004	---- ---- xxxx xxxx	rw	*FL0*	Voice 0 Frequency (low)
FFB006	---- ---- 00xx xxxx	rw	*PWH0*	Voice 0 Pulse width (high)
FFB008	---- ---- xxxx xxxx	rw	*PWL0*	Voice 0 Pulse width (low)
FFB00A	---- ---- 0000 xxxx	rw	*ATK0*	Voice 0 Attack
FFB00C	---- ---- 0000 xxxx	rw	*DK0*	Voice 0 Decay
FFB00E	---- ---- 0000 xxxx	rw	*STN0*	Voice 0 Sustain
FFB010	---- ---- 0000 xxxx	rw	*REL0*	Voice 0 Release
FFB012	---- ---- xxxx xxxx	ro	*OO0*	Voice 0 Oscillator output
FFB014	---- ---- xxxx xxxx	ro	*EO0*	Voice 0 Envelope output
FFB020	---- ---- xxxx xxxx	rw	*CR1*	Voice 1 Control register
FFB022	---- ---- 00xx xxxx	rw	*FH1*	Voice 1 Frequency (high)
FFB024	---- ---- xxxx xxxx	rw	*FL1*	Voice 1 Frequency (low)
FFB026	---- ---- 00xx xxxx	rw	*PWH1*	Voice 1 Pulse width (high)
FFB028	---- ---- xxxx xxxx	rw	*PWL1*	Voice 1 Pulse width (low)
FFB02A	---- ---- 0000 xxxx	rw	*ATK1*	Voice 1 Attack
FFB02C	---- ---- 0000 xxxx	rw	*DK1*	Voice 1 Decay
FFB02E	---- ---- 0000 xxxx	rw	*STN1*	Voice 1 Sustain
FFB030	---- ---- 0000 xxxx	rw	*REL1*	Voice 1 Release
FFB032	---- ---- xxxx xxxx	ro	*OO1*	Voice 1 Oscillator output
FFB034	---- ---- xxxx xxxx	ro	*EO1*	Voice 1 Envelope output
FFB040	---- ---- xxxx xxxx	rw	*CR2*	Voice 2 Control register
FFB042	---- ---- 00xx xxxx	rw	*FH2*	Voice 2 Frequency (high)
FFB044	---- ---- xxxx xxxx	rw	*FL2*	Voice 2 Frequency (low)
FFB046	---- ---- 00xx xxxx	rw	*PWH2*	Voice 2 Pulse width (high)
FFB048	---- ---- xxxx xxxx	rw	*PWL2*	Voice 2 Pulse width (low)
FFB04A	---- ---- 0000 xxxx	rw	*ATK2*	Voice 2 Attack
FFB04C	---- ---- 0000 xxxx	rw	*DK2*	Voice 2 Decay

FFB04E	----	----	0000	xxxx	rw	*STN2*	Voice 2 Sustain
FFB050	----	----	0000	xxxx	rw	*REL2*	Voice 2 Release
FFB052	----	----	xxxx	xxxx	ro	*OO2*	Voice 2 Oscillator output
FFB054	----	----	xxxx	xxxx	ro	*EO2*	Voice 2 Envelope output
FFB060	----	----	xxxx	xxxx	rw	*CR3*	Voice 3 Control register
FFB062	----	----	00xx	xxxx	rw	*FH3*	Voice 3 Frequency (high)
FFB064	----	----	xxxx	xxxx	rw	*FL3*	Voice 3 Frequency (low)
FFB066	----	----	00xx	xxxx	rw	*PWH3*	Voice 3 Pulse width (high)
FFB068	----	----	xxxx	xxxx	rw	*PWL3*	Voice 3 Pulse width (low)
FFB06A	----	----	0000	xxxx	rw	*ATK3*	Voice 3 Attack
FFB06C	----	----	0000	xxxx	rw	*DK3*	Voice 3 Decay
FFB06E	----	----	0000	xxxx	rw	*STN3*	Voice 3 Sustain
FFB070	----	----	0000	xxxx	rw	*REL3*	Voice 3 Release
FFB072	----	----	xxxx	xxxx	ro	*OO3*	Voice 3 Oscillator output
FFB074	----	----	xxxx	xxxx	ro	*EO3*	Voice 3 Envelope output
FFB080	----	----	xxxx	xxxx	rw	*CR4*	Voice 4 Control register
FFB082	----	----	00xx	xxxx	rw	*FH4*	Voice 4 Frequency (high)
FFB084	----	----	xxxx	xxxx	rw	*FL4*	Voice 4 Frequency (low)
FFB086	----	----	00xx	xxxx	rw	*PWH4*	Voice 4 Pulse width (high)
FFB088	----	----	xxxx	xxxx	rw	*PWL4*	Voice 4 Pulse width (low)
FFB08A	----	----	0000	xxxx	rw	*ATK4*	Voice 4 Attack
FFB08C	----	----	0000	xxxx	rw	*DK4*	Voice 4 Decay
FFB08E	----	----	0000	xxxx	rw	*STN4*	Voice 4 Sustain
FFB090	----	----	0000	xxxx	rw	*REL4*	Voice 4 Release
FFB092	----	----	xxxx	xxxx	ro	*OO4*	Voice 4 Oscillator output
FFB094	----	----	xxxx	xxxx	ro	*EO4*	Voice 4 Envelope output
FFB0A0	----	----	xxxx	xxxx	rw	*CR5*	Voice 5 Control register
FFB0A2	----	----	00xx	xxxx	rw	*FH5*	Voice 5 Frequency (high)
FFB0A4	----	----	xxxx	xxxx	rw	*FL5*	Voice 5 Frequency (low)
FFB0A6	----	----	00xx	xxxx	rw	*PWH5*	Voice 5 Pulse width (high)
FFB0A8	----	----	xxxx	xxxx	rw	*PWL5*	Voice 5 Pulse width (low)
FFB0AA	----	----	0000	xxxx	rw	*ATK5*	Voice 5 Attack
FFB0AC	----	----	0000	xxxx	rw	*DK5*	Voice 5 Decay
FFB0AE	----	----	0000	xxxx	rw	*STN5*	Voice 5 Sustain
FFB0B0	----	----	0000	xxxx	rw	*REL5*	Voice 5 Release
FFB0B2	----	----	xxxx	xxxx	ro	*OO5*	Voice 5 Oscillator output
FFB0B4	----	----	xxxx	xxxx	ro	*EO5*	Voice 5 Envelope output
FFB0BC	----	----	xxxx	xxxx	ro	*RNH*	Random number (High)
FFB0BE	----	----	xxxx	xxxx	ro	*RNL*	Random number (Low)
FFB0C0	----	----	00xx	xxxx	rw	*VE0*	Channel 0 Voice enable
FFB0C2	----	----	00xx	xxxx	rw	*FE0*	Channel 0 Filter enable
FFB0C4	----	----	00xx	xxxx	rw	*HPH0*	Channel 0 High pass cutoff (high)
FFB0C6	----	----	xxxx	xxxx	rw	*HPL0*	Channel 0 High pass cutoff (low)
FFB0C8	----	----	00xx	xxxx	rw	*LPH0*	Channel 0 Low pass cutoff (high)
FFB0CA	----	----	xxxx	xxxx	rw	*LPL0*	Channel 0 Low pass cutoff (low)
FFB0CC	----	----	00xx	xxxx	rw	*VOL0*	Channel 0 Volume
FFB0D0	----	----	00xx	xxxx	rw	*VE1*	Channel 1 Voice enable
FFB0D2	----	----	00xx	xxxx	rw	*FE1*	Channel 1 Filter enable
FFB0D4	----	----	00xx	xxxx	rw	*HPH1*	Channel 1 High pass cutoff (high)
FFB0D6	----	----	xxxx	xxxx	rw	*HPL1*	Channel 1 High pass cutoff (low)
FFB0D8	----	----	00xx	xxxx	rw	*LPH1*	Channel 1 Low pass cutoff (high)
FFB0DA	----	----	xxxx	xxxx	rw	*LPL1*	Channel 1 Low pass cutoff (low)
FFB0DC	----	----	00xx	xxxx	rw	*VOL1*	Channel 1 Volume

FFB0E0	---- ---- 00xx xxxx	rw	*VE2*	Channel 2 Voice enable
FFB0E2	---- ---- 00xx xxxx	rw	*FE2*	Channel 2 Filter enable
FFB0E4	---- ---- 00xx xxxx	rw	*HPH2*	Channel 2 High pass cutoff (high)
FFB0E6	---- ---- xxxx xxxx	rw	*HPL2*	Channel 2 High pass cutoff (low)
FFB0E8	---- ---- 00xx xxxx	rw	*LPH2*	Channel 2 Low pass cutoff (high)
FFB0EA	---- ---- xxxx xxxx	rw	*LPL2*	Channel 2 Low pass cutoff (low)
FFB0EC	---- ---- 00xx xxxx	rw	*VOL2*	Channel 2 Volume
FFB0F0	---- ---- 00xx xxxx	rw	*VE3*	Channel 3 Voice enable
FFB0F2	---- ---- 00xx xxxx	rw	*FE3*	Channel 3 Filter enable
FFB0F4	---- ---- 00xx xxxx	rw	*HPH3*	Channel 3 High pass cutoff (high)
FFB0F6	---- ---- xxxx xxxx	rw	*HPL3*	Channel 3 High pass cutoff (low)
FFB0F8	---- ---- 00xx xxxx	rw	*LPH3*	Channel 3 Low pass cutoff (high)
FFB0FA	---- ---- xxxx xxxx	rw	*LPL3*	Channel 3 Low pass cutoff (low)
FFB0FC	---- ---- 00xx xxxx	rw	*VOL3*	Channel 3 Volume
FFC000	xxxx xxxx ---- ----	rw	*OLEVL*	Object list enable vector length
FFC004	xxxx xxxx xxxx xxxx	rw	*OLEVP*	Object list enable vector pointer
FFC012	xxxx xxxx xxxx xxxx	rw	*OLP*	Object list pointer
FFC020	x000 0000 000x xxxx	rw	*BINIT*	Line buffer initialization value
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
FFC11A	---- --xx xxxx xxxx	rw	*HDE*	Horizontal display stop
FFC11C	---- --xx xxxx xxxx	rw	*HI*	Horizontal interrupt
FFC11E	---- --xx xxxx xxxx	ro	*HCP*	Horizontal current position
FFC120	---- ---- 0000 000x	rw	*VIDEN*	Video enable
FFC130	---- ---------- ----	ro	*LALA*	Lala - Video synchronization
FFC401	xxxx xx00	rw	*PALETTE*	Palette - Entry 0 - Red
FFC402	xxxx xx00	rw		Palette - Entry 0 - Green
FFC403	xxxx xx00	rw		Palette - Entry 0 - Blue
FFC405	xxxx xx00	rw		Palette - Entry 1 - Red
FFC406	xxxx xx00	rw		Palette - Entry 1 - Green
FFC407	xxxx xx00	rw		Palette - Entry 1 - Blue
.				
.				
.				
FFC47D	xxxx xx00	rw		Palette - Entry 31 - Red
FFC47E	xxxx xx00	rw		Palette - Entry 31 - Green
FFC47F	xxxx xx00	rw		Palette - Entry 31 - Blue

FFC800	000x xxxx	wo	*LBUF*	Line buffer - 0 (Byte accesses only!)
FFC801	000x xxxx	wo		Line buffer - 1
FFC802	000x xxxx	wo		Line buffer - 2
.				
.				
.				
FFC939	000x xxxx	wo		Line buffer - 319

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, number of pixels per line (320 maximum), and number of active lines per screen. All of these parameter can be modified on a frame by frame basis if desired.

The display generator is actually two separate, identical display generators. One produces horizontal timing while the other produces vertical timing. Each is controlled by a set of six registers, Period, Sync Start, Border Start, Border Stop, Display Start, and Display Stop. These registers will be referred to as *HP*, *HS*, *HBB*, *HBE*, *HDB*, *HDE* for horizontal and *VP*, *VS*, *VBB*, *VBE*, *VDB*, and *VDE* for vertical. Horizontal times are specified in pixels and vertical times are specified in lines. The diagram on the next page shows the relationship between these registers and the display timing and gives recommended values for NTSC and PAL. Although sync width and horizontal and vertical period are programmable to many values they must be set to either the correct value for NTSC or PAL in order for ordinary television sets to operate properly. A one bit register (*SMODE*) allows the programmer to determine the television system for which the host machine is configured. If the bit is low, the machine is configured for NTSC. If the bit is high, the machine is configured for PAL.

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 setup with a valid object list (see following sections) and then the video subsystem should be enabled by writing a one to *VIDEN*. Writing a zero to *VIDEN* at any time will cause sync and display processing to stop and the screen to go black.

The display start registers (*HDB* and *VDB*) serve several purposes. First, they indicate the beginning of the active portion of the screen. Second, they are the base count from which actual screen positions are measured. And finally, they initiate display processing. On each line starting with the line one before *VDB* and ending with *VDE*, when the internal counter reaches *HDB*, the display processor will take control of the machine, disabling the CPU. The display processor will first load the current line buffer into the shifter to be sent to the display. Then it will initialize the line buffer 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 the next line will be built on top of the existing line. Finally, it will process the object list to 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. In order to synchronize the CPU with this process, it may read the lala register (*LALA*). A read of *LALA* will cause the CPU to hang until the display processor has finished processing the next line.

Two interrupts are provided to help in synchronizing the CPU and display. One will cause a level 4 autovectored interrupt at the beginning of vertical blanking (*HDE* and *VDE*). The other can provide a level 2 autovectored 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 (*HDB* and *VDB*). These registers may be reloaded during the frame to produce several interrupts. Two notes of caution. One, if the interrupt happens while the CPU is hung by either the *LALA* register or

Interface

Controllers:

DM15S

```
 1  ooooo  5
 6 ooooo  10
11  ooooo 15
```

1 - "A" Up
2 - "A" Down
3 - "A" Left
4 - "A" Right
5 - X Pot
6 - "A" Fire
7 - +5V
8 - Reserved
9 - Ground
10 - "B" Fire
11 - "B" Up
12 - "B" Down
13 - "B" Left
14 - "B" Right
15 - Y Pot

Video:

DIN13

```
        o   13
   9 o o o o 12
   5 o o o o  8
   1 o o o o  4
```

1 - Audio output
2 - Composite Video
3 - Not used
4 - Not used
5 - Not used
6 - Green output
7 - Red output
8 - +12V Pullup
9 - Hsync
10 - Blue output
11 - Not used
12 - Vsync
13 - Ground

Cartridge:

TBD

23 Address lines
32 Data lines
 4 AS, R/W, DTACK, Interrupt
 4 ROM Chip select
 1 RAM Chip select
 2 Clock and data for key
 1 Strap
 6 Power and ground

73

What's Missing?

This is the reader participation part of the document. Thing's that I know are missing (and will be added) are listed below. If there are other things you feel should be added, or if something is unclear, please jot it down here and give this sheet back to me. It's important that you do this NOW! because I know, from past experience, that if you don't you never will. Then when it's really important that a good specification exists. Someone (and we all know who) will go around and complain that this document is a piece of shit and totally useless. So ... HELP!

To be added:

- Display processor timing information relating display list organization and size to actual processing time.
- Recommended PAL timing values
- Sound chip definition.
- Safe min and max values to use for the paddles
- Super controller definition

Questions:

- Volume control?

the display processor, the interrupt will not be serviced until the display processor finishes. Two, only negative offsets up to minus *HDB* or minus *VDB* will be recognized. That is, -(*HDB* + 1) does not necessarily equal *HP* and similarly *HP* + 1 does not necessarily equal -*HDB*.

Color Palette

The color palette provides the mapping between the 5 bit internal data representation and the 6 bit red, 6 bit green and 6 bit blue color 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 color palette. Data of value zero is considered transparent and is therefore never loaded into the line buffer by the display processor. A zero data value is only output under three circumstances. First, the border is zero. Second, if the display processor is setup to initialize the line buffer to zero (with *BINIT*) and no objects are written over it. And last, if the CPU writes the data in the line buffer to zero.

The palette entries are arranged as a byte of each red, green and blue. Only the upper six bits of each byte are used. For compatibility with future products the lower two bits should be zero. The three bytes are arranged as the least 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.

	MSB				
PALETTE	Unused	Red	Green	Blue	Entry 0
PALETTE + 4	Unused	Red	Green	Blue	Entry 1
.					
.					
.					

Object List Format

Overview

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

Object headers are two longwords in length and are stored in the object list. Objects are painted down in the order they appear in the object list. The first object has the lowest priority (usually the background) and the last object has the highest priority. The object list is pointed to by a register in the display processor.

Entries in the object list may be individually enabled on a line by line basis. This is accomplished through the use of the object list enable vector. This is a bit vector whose length in bits is at least equal to the number of entries in the object list. There is one object list enable vector per line. The list of object list enable vectors is pointed to by a register in the display processor.

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. Some possible uses for these objects might be:

- Reload the color palette (or a portion) on specified lines of the screen.
- Reload the object list pointer on specified lines of the screen.
- Move data to the sound chip.
- Modify object position/data/color for the next frame.
- Reassign object list priorities.

The Object list

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

OL address = *OLP* * 4

This means the OL can be located on any longword boundary in the first 256K bytes of RAM.

Each entry of the OL is an object header which is two longwords in length. Entries in the OL are sequential. The number of entries (length of the list) is determined by the object list enable vector.

The Object List Enable Vector List

The object list enable vector (OLEV) list is a list of bit vectors (one vector per line) that specifies which of the OL entries to be processed on each given line. The OLEV list is described by two registers in the display processor; the OLEV list pointer register (*OLEVP*) and the OLEV length register (*OLEVL*). The *OLEVP* register is word wide and specifies the offset in longwords into RAM. The *OLEVL* register is byte wide and specifies the number of longwords of vector per line. That is:

OLEV list address = *OLEVP* * 4
OLEVL = int((# of OL entries + 31) / 32)

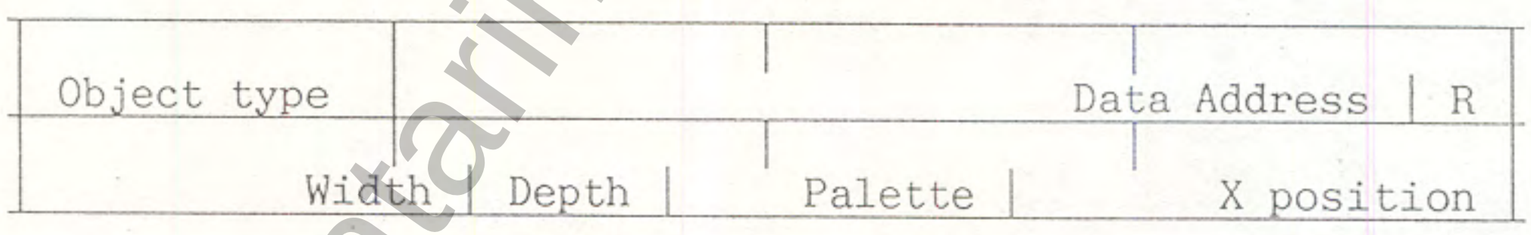
Each vector contains one bit per OL entry. If the bit is set, that object is processed. If the bit is cleared, the object is ignored. The most significant bit of the vector is used to enable the first entry of the object list, the next to most significant bit corresponds to the next entry and so on.

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.

Scaled bitmapped object (0x01 - 0x7F)

The object header for a bitmapped object has the following format:



Where:

Object type	is a byte specifying the type of object (0x20 for bitmapped object with no scaling).
Data address	is the full 23 bit address of the object data.
R	is a bit specifying if the object should be reflected in X
Width	is a 10 bit field which specifies the offset in words to the beginning of the next line.
Depth	is a two bit field which specifies the depth of the object data.
Palette	is a 8 bit field specifying the starting offset into the pallette.
X position	is a 12 bit field specifying the starting offset into the line buffer.

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 (0 = 1 bit, 1 = 2 bits, 2 = 4 bits, 3 = 8 bits). 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.

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 replace the appropriate pixel of the line buffer.

The display processor's procedure for handling a bitmapped object is:

```
          Fetch data address
          Write new data address = data address + 2 * Width
          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:     Process next Object
```

Note that the data address in the object header is modified to point to the start of the next data line. Two dimensional objects are handled by setting the corresponding bit in the OLEV in each line that the object appears. This also serves as the Y position and Y length. These lines do not necessarily need to be contiguous lines but could be every second (or third or fourth, etc.) line for special effects, exploding objects, etc. Since the data address is being modified during the display, it will need to be reset at the end of the frame. This can be done by the CPU or by one of the special data manipulating objects.

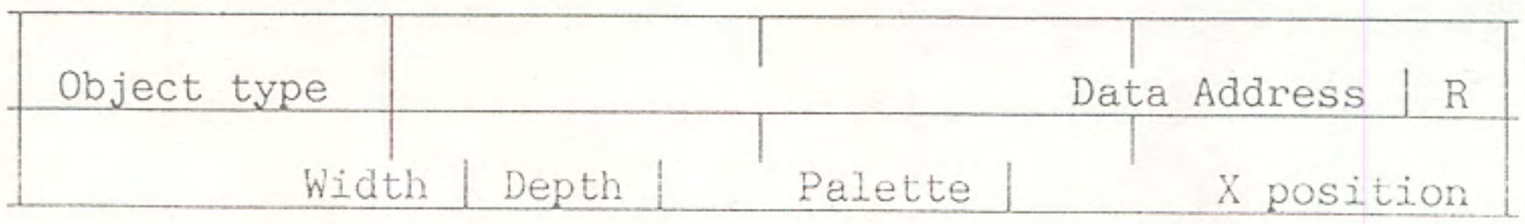
Scaling (X direction only) is specified in the object type. Bits 6-0 provide a 7 bit scale factor. The decimal point is after bit 5. Thus, 0x20 is 1, 0x01 is 1/32nd, 0x60 is 3, etc. Scaling is accomplished by dropping or repeating pixels as appropriate.

Scaling in Y can be handled by a similar algorithm but must be handled by the programmer. Special objects can be used to modify the data address on a line by line basis for this purpose.

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 postion specifies the right side of the object instead of the left.

Run length object (0x00)

The object header for a run length object has the following format:

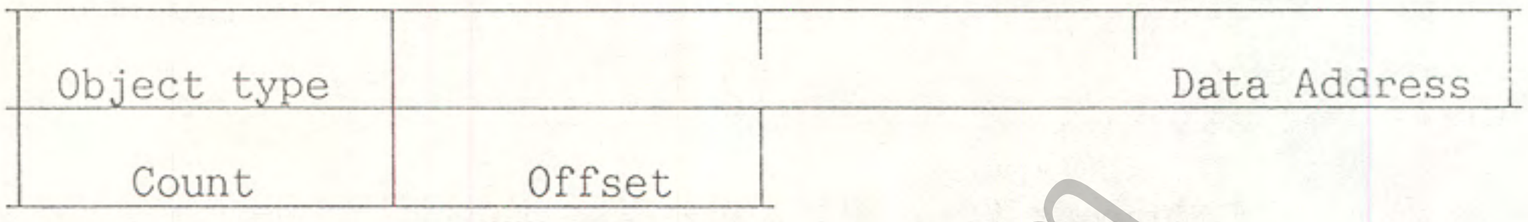


This is the same as for the bitmapped object. These objects differ only in data format. The data for the run length object is stored as a word which specifies the number of length/data pairs followed by a byte length followed by a byte of data followed by another byte of length followed by a byte of data and so on. The length (0x00 = 0, 0xff = 255) is the number of times to

repeat the data. The data, although it takes up a byte, only uses the N least sigificant bits, where N is specified by the depth field. Note that the width field is ignored and the data address is modified according to the word that specifies the number of pairs.

Load palette (0x80)

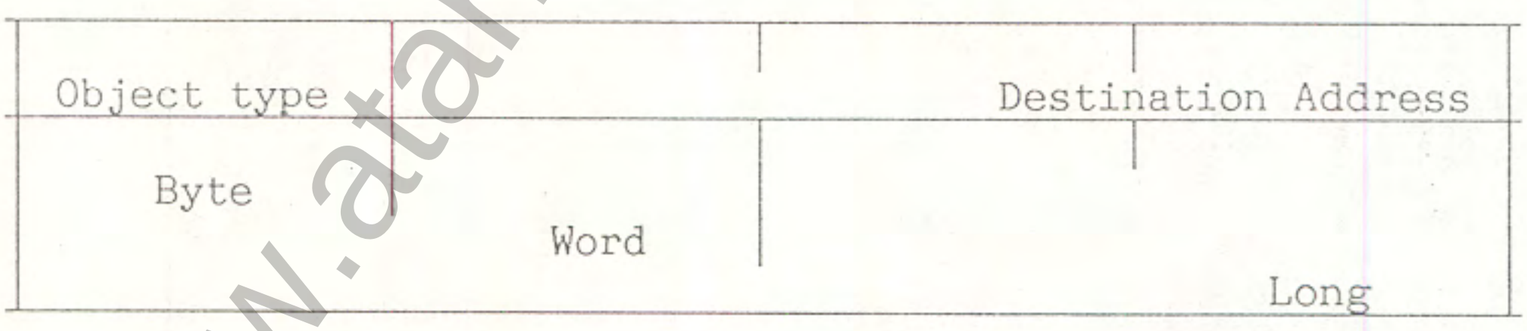
The object header for a palette load has the following format:



When this object is encountered the display processor will "remember" about it and at the next Hsync will copy "Count" longwords from the data address to the palette starting "Offset" entries into the palette. Placing this object anywhere in the OL will cause the new palette to be in effect for the line in which this object is enabled. 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).

Move Long to Memory Immediate (0xE0)
Move Word to Memory Immediate (0xD0)
Move Byte to Memory Immediate (0xC0)

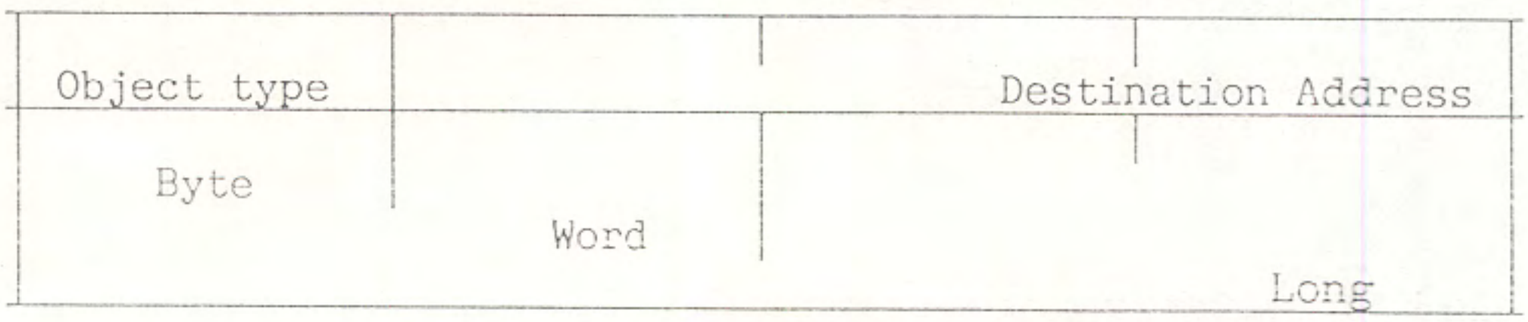
The object header for a move to memory immediate has the following format:



When the display processor encounters this object it will move the Long, Word or Byte specified in the object header to the destination address.

Add Long to Memory Immediate (0xE1)
Add Word to Memory Immediate (0xD1)
Add Byte to Memory Immediate (0xC1)

The object header for an add to memory has the following format:



When the display processor encounters this object it will add the Long, Word or Byte specified in the object header to the Long, Word or Byte at the destination address.