A Description of the Bally Professional Arcade Video Hardware and Associated Coin-Operated Hardware

Introduction

This description of how the BPA video hardware works was produced in an effort to dispel the rumor that a Bally Professional Arcade (BPA) can display commercial resolution video. Entire contents Copyright © 2001 by Anthony J. Miller.

Sources of Information

In the preparation of this document I have mainly relied on four information sources:

The Nutting Manual	This paper borrows heavily from the Nutting Manual, and will include references to certain pages of it. The Nutting Manual was originally written by employees of Dave Nutting Associates, a research and development group for Midway Manufacturing, Inc., (a subsidiary of Bally Manufacturing and the company which actually produced the Bally Arcade) as a guide for internal and external BPA game developers. This group actually designed the BPA, and programmed the majority of game cassettes offered.
U.S. Patents 4,301,503 and 4,475172	Both patents cover the chip set. I have read the former extensively, and looked at the latter enough to realize that it is a re-issue of the former, with additional claims (which are totally irrelevant to this document). The inventor is listed as Jeff Frederiksen, and the assignee is Bally Manufacturing.
Midway Mfg. SeawolfII	This is the first Midway coin-operated game using the custom chips. You can download the schematic from: http://www.hypertech.com/arcade/manuals.asp along with manuals and data schematics of several other Midway games using the chip set. This schematic is the easiest to follow, since everything is on one sheet of paper, unlike the card rack system developed later.
The Bally Service Manual	which is available from: http://www.classicgaming.com/ballyalley/misc_docs/bally_service_manual.pdf This manual contains the schematic of the BPA

I have made every attempt to make this description as accurate as possible.

Dave Nutting Associates was purchased by Bally and became an internal development group. Dave Nutting Associates consisted of four groups:

Industrial Design:	Headed by Dave Nutting, who did the console and handgrip design. Dave was also responsible for the design of the Excalibur kit automobile, for another company. Dave had at least one other engineer working for him (Ken ….) , who was responsible for the mechanical design of the BPA Add-on cabinet.
Hardware Design:	Headed by Jeff Frederiksen, who was Dave's partner, Vice President of Engineering, and Main Associate. Jeff was an Electrical Engineer, and along with Terrence Coleman, did the design of the three custom chips used in the BPA. Coleman also designed the circuitry on the BPA PC Board. I (Tony Miller) also worked in this group, and designed the early version of the Add-on.
Software Design:	Headed by Jay (now Jamie) Fenton, whose primary responsibility was the implementation of the operating system, called HVGSYS (**H**ome **V**ideo **G**ame **SYS**tem) internally. I believe Jay also did the Scribbling game. There were several people in the Software group, each was responsible for a separate game cassette. The following individuals had primary responsibility for the 'bundled' games:

Gunfight:	Alan McNeil
Checkmate:	Lou ...
Calculator:	Jeff Frederiksen

PC Design:	Headed by Jim Hemmer, who did the PC design for the BPA.

The design was sold to a company by the name of Astrovision, Inc., some time in 1982. I believe that the game sold under this name was identical to that sold as the Bally Professional Arcade.

Block Diagram

A block diagram of the BPA appears below. This diagram is based on the one shown on page 108 of the Nutting Manual (hereafter referred to as NM108, etc).

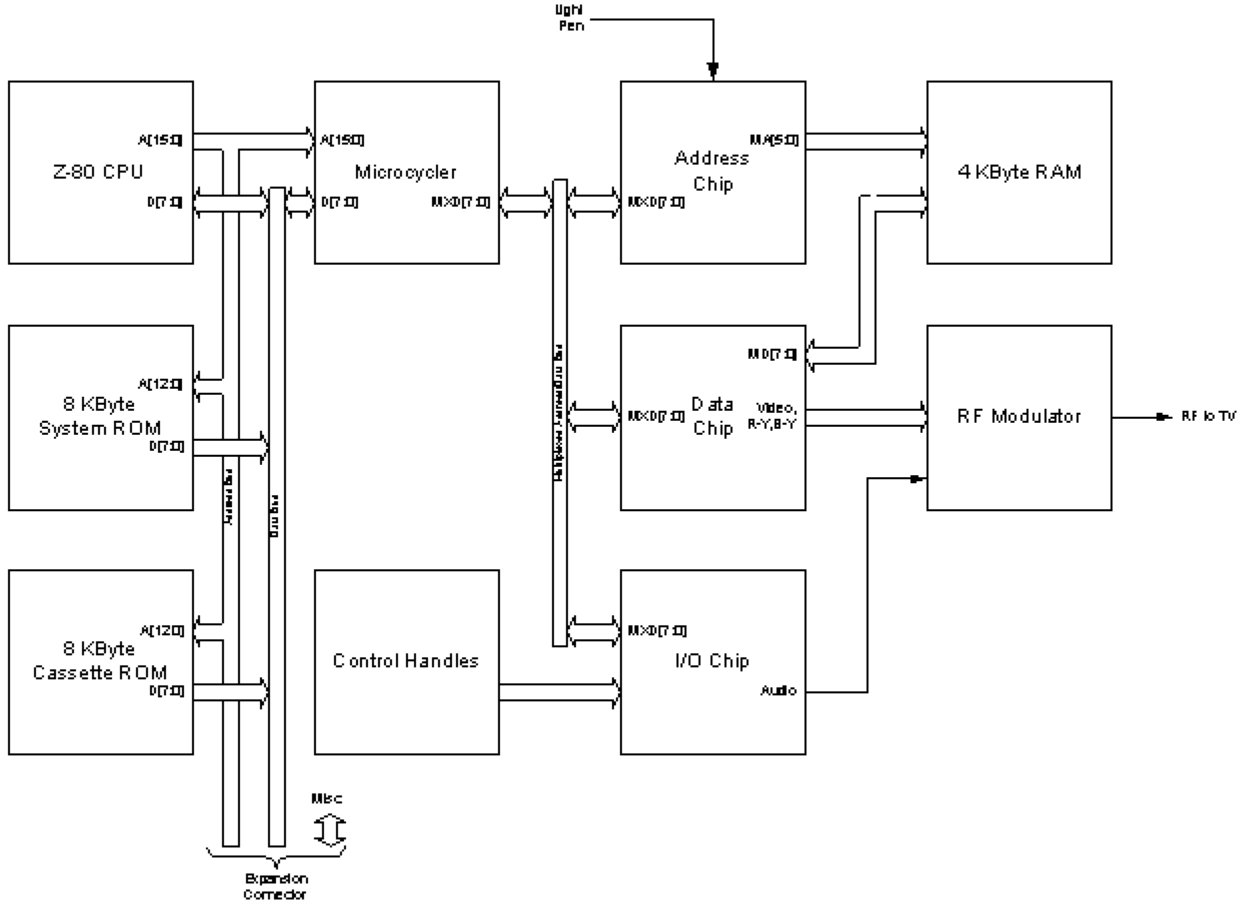


Figure 1: **BPA Block Diagram**

Other circuits which control the various blocks are not shown in the diagram above. Note, however, that the Z80 Address and data busses are shown as separate busses connecting the Z80, System ROM, Cassette ROM, and Expansion Connector. The Microcycler Block is used to combine the 16 bits of address and 8 bits of data from these busses into a time-multiplexed Microcycle bus, which then connects to the three custom chips. This was done (NM109) to save pins on the custom chips.

The Address Chip is used to generate Screen Refresh, memory refresh, and Z80 Access address cycles to the Screen RAM, which also doubles as ScratchPad memory for the system. This part also captures the screen refresh address when a light pen signal is detected, to generate the Light Pen Interrupt to the Z80. What is meant by the terms Screen Refresh, Memory Refresh, and Z80 Access cycles will be described in a forthcoming paragraph.

The Data chip handles data functions for Z80 Screen and Scratch memory accesses, Magic Memory writes, and Parallel to Serial formatting of screen data (in consumer mode) to be output to the video modulator. This chip also handles color look-up table functions to convert serial pixel data to color/brightness index data, and the generation of Video, sync, and the color components (R-Y, B-Y) for the video modulator.

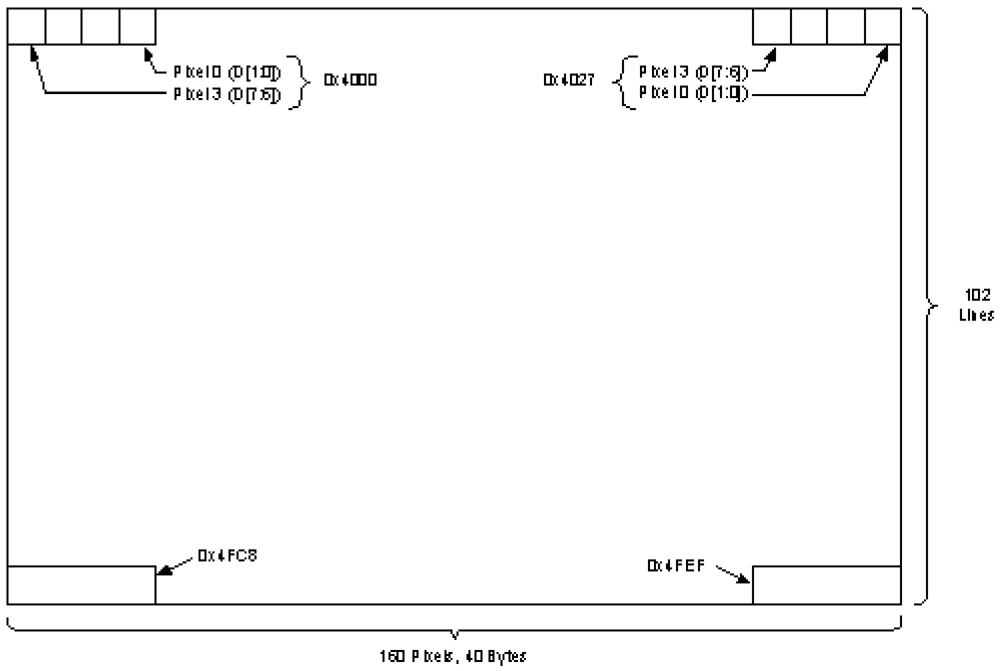
The I/O chip is responsible for the sound generation function, and control handle interface. It will not be described in this article.

Screen Memory

For the BPA, eight 4096 (4K) x 1 Dynamic Random Access Memories (DRAMs) were used. This gives a total of 4096 Bytes of Read/Write memory, which was organized as a 160 pixel wide by 102 pixel tall frame buffer. Each pixel occupied two bits, for a total of four simultaneous colors on the screen. Four pixels occupy one byte of screen RAM. The Data chip implemented several features to make it appear as through there were many more simultaneous colors available. How this was accomplished will be covered later in the Data Chip section.

This was one of the first implementations of the frame buffer architecture in a consumer (or coin-operated, for that matter) game. The contemporaneous Atari 2600 used very little memory - a line or two of pixels in memory, and required the microprocessor to generate video just ahead of it being displayed. The BPA frame buffer uses an entire frame of pixel memory, which the processor can update ahead of time for display.

The screen origin is the upper left corner of the video screen, at Z80 address 0x4000. There are actually four pixels at this address. Pixel 3 (the left-most of the four) occupies bits[7:6], pixel two bits[5:4], pixel one bits[3:2], and pixel 0 (the rightmost of the four) occupies bits[1:0]. This relationship is used throughout the rest of the screen. The figure following is for the consumer game and is identical to the one on NM86:



Therefore, the first (left) pixel on the second line (line 1, where the first line is line 0) occupies bits 7:6 of address 0x4028, the last pixel (right) on line 1 occupies bits 1:0 of 0x404F, the first pixel on line 100 occupies bits 7:6 of 0x40A0 and the last pixel on line 100 is in bits 1:0 of 0x4FC7.

Figure 2: Consumer Resolution Screen Map

The commercial version has a resolution of 320 by 204. See the following illustration, identical to the one on NM87:

Here, the leftmost pixel of line 1 (i.e., the second line of video) occupies bits[7:6] of 0x4050, the rightmost pixel of line 1 occupies bits[1:0] of 0x 409F, the leftmost pixel of line 202 occupies bits[7:6] of 0x7F20, and the rightmost pixel of line 202 occupies bits[1:0] of 7F6F. Since the commercial screen resolution is doubled in each direction, the amount of memory required is increased by a factor of four, to 16,384 bytes. Due to the architecture of the system, this required the use of 32 4K x 1 DRAMs, in four banks of eight each.

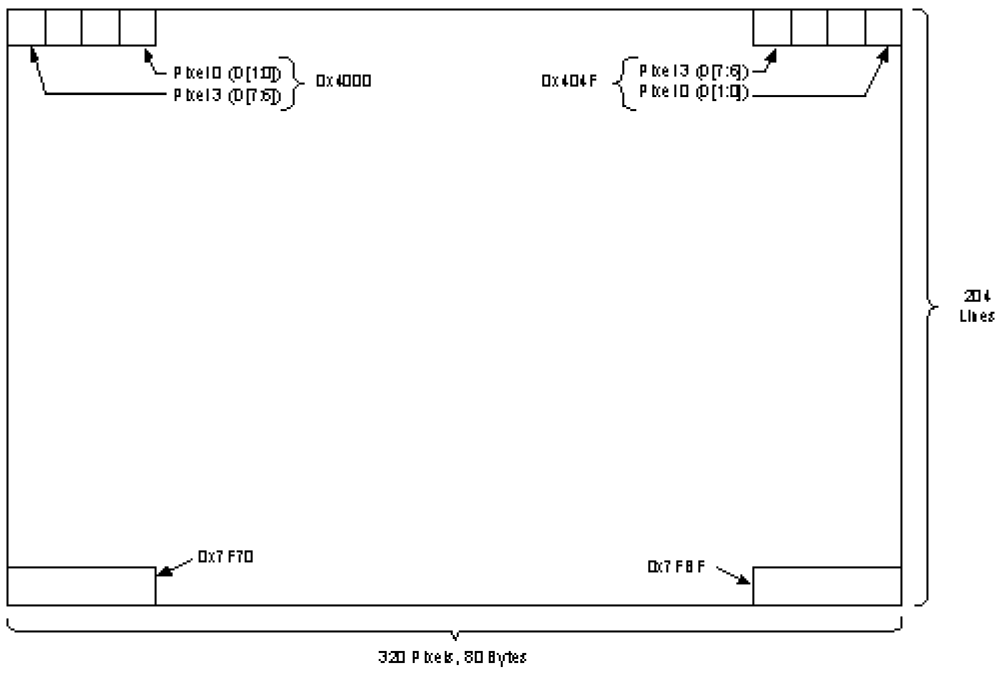


Figure 3: **Commercial Resolution Screen Map**

In consumer mode, each line is scanned twice per field (the second time immediately after the first), to give 204 active lines of video. The remaining 58 ½ lines of video (each NTSC field is 262 ½ lines) is made up of blanking and sync lines. In commercial mode, each line is scanned once per field, with 58 ½ blanking and sync lines. Both systems output video at the same timing, conforming to NTSC specifications. Each frame consists of two fields, scanned in an interlaced format, at 60 fields (or 30 frames) per second. NTSC stands for **N**ational **T**elevision **S**tandards **C**ommittee, which sets the standards for video broadcasting in this country, but some would say that it actually means Never Twice the Same Color.

Dynamic RAMs

I believe the first DRAM (**D**ynamic **R**andom **A**ccess **M**emory) to use the multiplexed address approach was the 4K x 1 DRAM, used in the BPA. DRAMs have a major advantage over the SRAM or **S**tatic **R**andom **A**ccess **M**emory. That is, a memory cell can be made of a single transistor and capacitor. The capacitor is the actual memory element. Depending upon design, a charged capacitor may signify a logical 1 and a discharged capacitor may signify a logical 0, or vice-versa. The transistor is used to connect or isolate the capacitor from the rest of the circuitry. An equivalent SRAM cell requires at least six transistors, so a DRAM uses about six times less chip area than an equivalent-sized SRAM. One of the main cost factors in a memory is the amount of chip area used. The smaller the area, the cheaper the chip, since more chips can fit on a single wafer. The cost of processing a wafer is roughly constant, so more die (chips) on a wafer, the cheaper each chip is. Another factor affecting the cost of a chip is the package. More pins on a package means a larger package, and more connections between the die inside and the package pins. For these reasons, a DRAM is cheaper than the same-sized SRAM.

To save on package pins and to make it possible to get more memories on a board, the multiplexed address technique was invented. To address 4096 bits of memory, 12 address lines ($2^{12} = 4096$) are required. Along with the required Chip Select (CS# where # means active low), Write Enable (WE#), Data In (DI), Data Out (DO) and the four power supplies (+5 Volts, -5 Volts, +15 Volts, and ground), this means a 20 pin package. At the time, 14- and 16-pin DIPs (Dual Inline Packages) were the cheapest, so the multiplex technique was used to get this memory into a 16-pin package. Under this scheme, six pins are used for addressing. First, the most significant addresses (called the Row Addresses) are applied, and the Row Address Strobe (or RAS#) is asserted to logical 0 to latch these addresses into an internal register. Next, the least significant addresses (called the Column Addresses) are applied, and the Column Address Strobe (or CAS#) is asserted to latch these addresses into another internal latch. If the CS# input is logical 0 at the falling edge of CAS# , the chip is selected for either a write or a read, as determined by the state of the WE# input. If the WE# pin is at logical 0 during this time, the operation is a write, and the data appearing on the DI line is saved in the memory. Otherwise the operation is a read and the data at the addressed location appears on the DO line. If the CS# line is at logical 1 during this process, the memory is not selected: No location will be written and no location will be read, and the DO line will be off.

RAM Read, Write, and Refresh Timing

Typical Read, Write, and Refresh timing waveforms appear in the figure following:

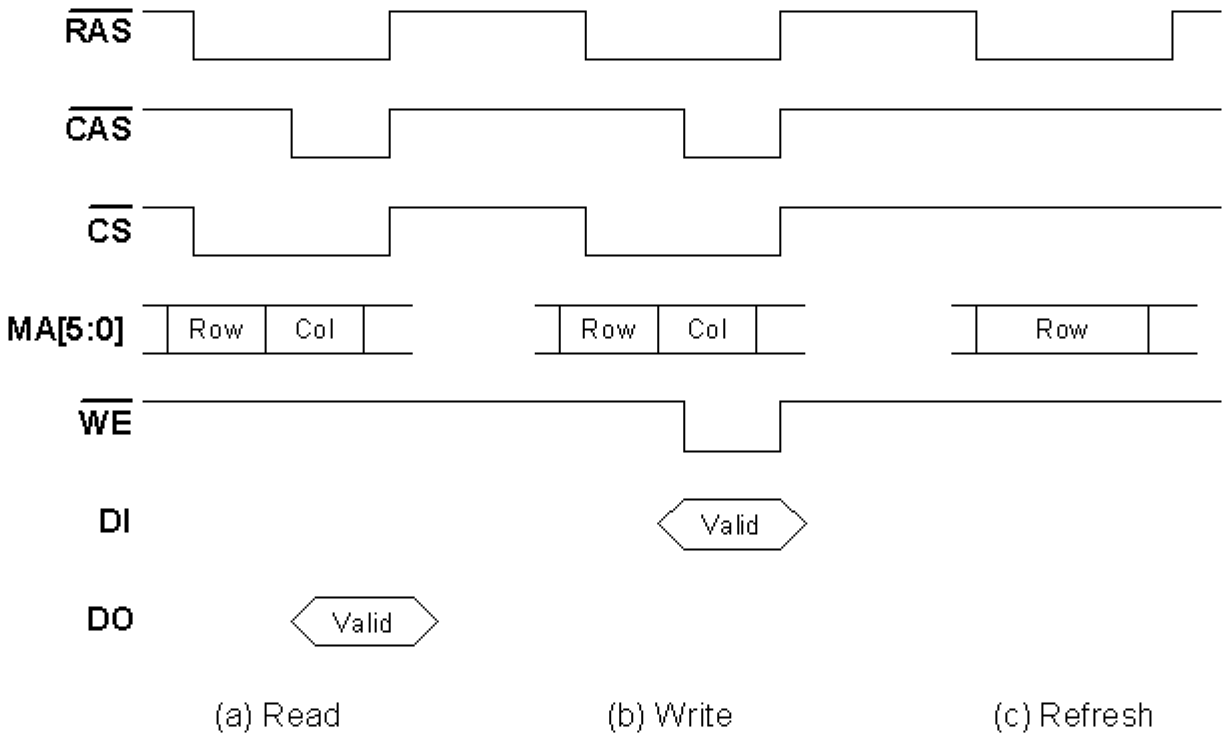


Figure 4: **DRAM Read, Write, and Refresh Timing**

DRAMs have some disadvantages as compared to SRAMs. They are slower, which means that it takes longer to read or write them, primarily due to the multiplexed address scheme. They also need to be refreshed. Since the storage element is a capacitor, the charge will dissipate over a period of time, usually measured in milliseconds. This means that the capacitor charge (or lack thereof) has to be re-established periodically. Any read or write will cause the capacitor to be recharged, so if the application regularly reads or writes ALL row addresses, the memory will retain valid data. This is a fairly time-consuming process, since for this memory usually 64 rows have to be refreshed every 2 milliseconds, and the refresh process usually takes around a microsecond per row.

All of the RAMs made by a given supplier must conform to that supplier's published specifications, otherwise, they cannot be sold as premium product. One of the biggest problems with these early RAMs was that they could net meet the refresh specification. If, however, they were refreshed more often that all 64 rows every 2 milliseconds, they would function perfectly normally. As production continued, each RAM manufacturer gathered their refresh reject parts in barrels on the factory floor. Periodically, they would salvage these parts, since they contained small amounts of gold, which could be recovered. But the BPA regularly reads from each row of each memory, since it must read the memory many times each second to refresh the screen. This fact makes it possible to use these otherwise refresh-reject parts. So, Midway was able to get large quantities of very cheap DRAMs, since they were all officially reject parts. But they were perfectly suited for use in the BPA frame buffer.

The normal DRAM refresh cycle shown above is not used in the BPA, since CAS# runs all the time and RAS# and CS# are connected together. Therefore what passes as a memory refresh cycle for the BPA is actually a memory read, with the Row and Column addresses set to the same value, a value determined by the Z80 lower address bus bits while the Z80 RFSH# output is asserted to logical 0. The pinout of a typical DRAM used in the BPA appears following:

Notice that for this generation of DRAM, three power supplies were used. The BPA used a +15 VDC supply for the RAMs, while the later coin-op games used +12 VDC. Later BPAs may have used the +12 VDC supply, since the DRAM suppliers later moved over to a fabrication process which could use a +12 VDC supply. Later generations of DRAMs eliminated both the positive high voltage supply and the negative supply, so only a single +5 VDC supply was required. The next generation 16 K x 1 DRAM replaced the CS# pin with a seventh address line (A6). This line was used during both the RAS and CAS accesses, so that it, too, brought in two address lines.

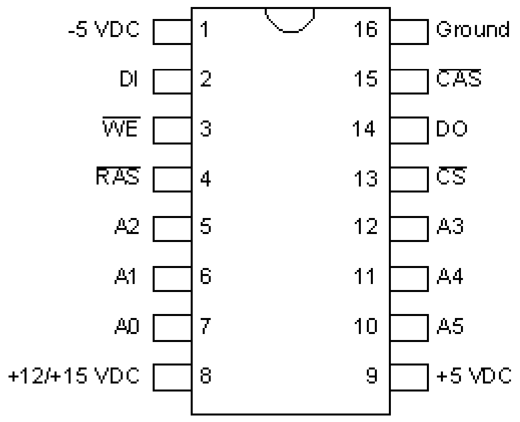


Figure 5: **4K x 1 DRAM Pinout**

RAM Addressing

As touched on above, the DRAMs require a multiplexed Row/Column Address bus to select the correct memory location. These addresses come from one of three sources, in order of priority:

Memory refresh: As detailed above, all 64 rows of the memory must be refreshed every 2 milliseconds. This process has the highest priority, but only during blanking time (when the video signal is blanked for retrace, at the end of each line and for a longer period at the end of each video field).

Screen Refresh: In order to keep an image on the screen, the Screen memory must be continuously read out and the data serialized for output to the RF modulator. Horizontal and vertical counters are implemented in the Address chip, which keep track of the current position of the screen address in the Screen memory. These are used to provide the memory address for each Screen refresh cycle.

Z80/Magic Access: For the BPA, the Screen RAM is the only RAM which exists, so Z80 Scratch memory also resides in the Screen RAM. The coin-op machines used separate memories for Scratch, so the entire DRAM (except for the small portion left unused from 0x7FBF through 0x7FFF) is available to display screen images. Read or Write addresses are provided by the Z80, through the Microcycler Bus.

The primary function of the Address chip is to generate the Screen Refresh address, gather together all addresses for Screen memory access, and multiplex them to generate the Row/Column Addresses to the Screen RAM. In addition, the Light Pen, Line interrupt, memory Read/Write control, and Z80 Wait# control are implemented n the Address chip.

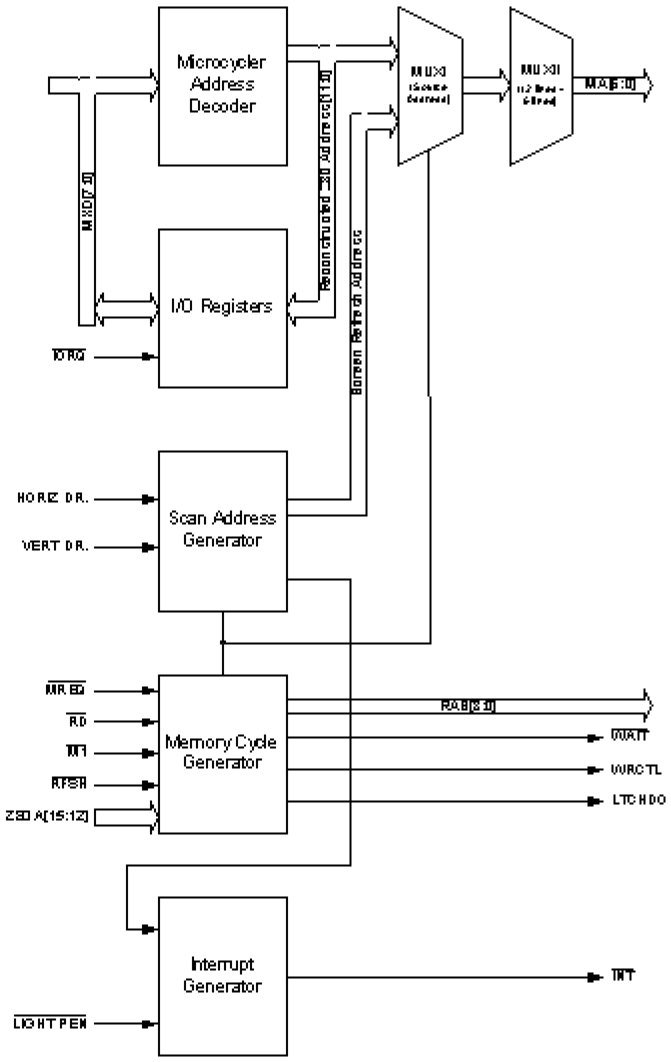


Figure 6: Address Chip Block Diagram

The Address Chip

A block diagram of the Address chip is at left. Each block will be described below.

Microcycle Address Decoder

The Microcycle Address Decoder reconstructs the twelve least significant bits (i.e., A[11:0]) from the multiplexed Z80Address Bus. The four most significant Address bits (A[15:12]) are input to the chip directly, due to timing requirements. As stated in the text (NM111), they would arrive too late in the memory cycle if brought in through MXD[7:0]. The 12 Address LSBs are directly input to MUX I, and are also used to select the appropriate I/O register during an I/O cycle. The six LSBs of this bus are also used in memory refresh cycles.

I/O Registers

Address chip registers are tabled below. The column labeled **I/O** defines whether the register is an input or an output of the Z80.

Address	I/O	Name/Description
0x08	O	Consumer/Commercial. The Address chip needs this bit to determine the address organization of the memory, and how often to perform Screen Refresh memory accesses. This register is duplicated in the Data chip.
0x0A	O	Vertical Blank. The Address chip needs this register to determine the scan line below which Screen Refresh is no longer needed. The Data chip must duplicate this register to determine when to turn on the border color.
0x0C	O	Magic Register. The Address chip needs this register to determine what kind of Magic Memory access is in progress: Write, or Read/Modify/Write. The Data chip must duplicate this register to control the data path for magic writes.

Address	I/O	Name/Description
0x0E	I/O	Interrupt Enable and Mode/Vertical Line Feedback. On write, this register enables the two interrupts and determines the interrupt mode of each. On read, this register contains the vertical line number that a Light Pen Interrupt was received on.
0x0F	I/O	Interrupt Line/Horizontal Address Feedback. On write, this register sets the scan line number on which to generate a line interrupt back to the Z80. On read, this register contains the horizontal address (i.e., position in the line) that a Light Pen Interrupt was received on.

MUX I

MUX I is controlled by the Memory Cycle Generator, and determines which set of twelve addresses will be applied to the Screen Memory. During a Z80/Magic Memory access, the addresses come from the Microcycler Decoder. During Screen Refresh memory reads, the addresses come from the Scan Address Generator.

MUX II

MUX II takes the output of MUX I and time division multiplexes the address lines from 12 lines down to 6, to be output to the Screen Memory.

Scan Address Generator

This block contains the counters used for Screen Refresh Address generation. It consists of a 12-bit counter which is reset to 0x000 each time the Address chip input Ver Dr. is asserted to logical 1, at the beginning of a frame of video. Starting a few clocks after the Address chip input Hor Dr. is asserted to logical 1, this counter will increment to point to the next memory location to read for Screen Refresh. The counter operates differently when the chip is operating in Commercial (hi-res) mode.

Consumer Mode: In Consumer Mode, each Screen Refresh cycle reads four pixels (one byte) at a time. Thus, the counter will increment every fourth PX (pixel) Clock, and a Screen Memory Refresh read is performed to access the next four pixels. This means that the memory is unavailable to the Z80 half the time when active video (i.e., when the screen is not blanked and not in the border color area). Signal PX runs at twice the rate of the Z80 clock Φ. A Z80 access consists of at least four (T1, T2, Tw, T3) and as many as 5 (T1, T2, Tw, Tw, T3) Φ clock cycles. Since the Screen Refresh has higher priority, if the Z80 timing ends up such that the Screen Refresh is currently using memory, an extra wait state will be generated to the Z80. A Z80 access cycle to this memory always results in at least one wait state. The Scan Address counter will increment forty times (bytes) per line. If this line is the first of the two line duplicate display, the counter will reset to its current count minus 40 at the next Hor Dr. Otherwise it will increment to the next address.

Consumer Mode: In Commercial Mode, each Screen Refresh cycle reads sixteen pixels (four bytes) at a time. These pixels are loaded into an external shift register and shifted at the 7M (7 MHZ) clock rate into the Data chip Ser[1:0] inputs. Thus, the counter will increment every eighth PX Clock (recall that pixels are shifted at twice the {consumer} PX clock rate), and a Screen Memory access read is performed to access the next sixteen pixels. In this case, the memory is unavailable to the Z80 only one fourth the time. The Scan Address counter will increment twenty times per line. An external counter is used to calculate the correct time to load the shift register. This counter is synchronized by the Hor Dr. signal.

In order to facilitate Memory refresh using the internal Z80 refresh address counter, the least-significant Z80 and Screen refresh counter addresses become Row addresses to the memory, and the most-significant addresses become the column address. This is contrary to the 'normal' way DRAMs were connected. The fact that sixteen pixels are read simultaneously in Commercial mode means that the memories are organized in a non-obvious manner. This will be discussed in detail later. The Scan Generator counter contents are made available to the I/O Registers section and also the Interrupt generator.

Interrupt Generator

The Interrupt Generator can be programmed to generate two interrupts, the line and Light Pen Interrupts. Bit 3 of the Interrupt Enable port (0x0E) when set enables line interrupts, while bit 1 of the same register enables the Light Pen interrupt. The Scan address counter contents are compared with the content of the Interrupt Line register. If the line interrupt is enabled, an interrupt to the Z80 will occur when the line portion of the counter contains the line number stored in the Interrupt Line register. If the Light Pen Interrupt Address chip input is asserted to logical 1, the current Scan Generator contents are latched into the Vertical Line and Horizontal Address registers, which can be read by the Z80 following a Light Pen interrupt. The Z80 responds to an interrupt by simultaneously asserting its IORQ# and M1# outputs.

Memory Cycle Generator

This block is responsible for arbitrating between memory refresh, screen refresh, and Z80/Magic memory accesses. It also controls the timing of the resulting memory access, along with the generation of the RAS and Multiplexed Address (MA[5:0]) bits. Z80 inputs to this block include: MREQ#, RD# M1# and RFSH#, along with the four Most Significant Address bus bits. Also input to this block is a signal originating in the Scan Address Generator to request a memory access every four PX clock cycles to refresh the screen in consumer mode. This block outputs the four RAS signals to memory, the row/column Memory Address Multiplexer control signal, WAIT# to the Z80, and the WRCTL# and LTCHDO control signals to the Data chip. This circuit operates quite differently as determined by the setting of the consumer/commercial bit.

In consumer mode, the screen organization is straightforward. There is only one bank of memory. Pixels 0 through 3 occupy the first address of the Screen RAM, pixels 4 through 7 occupy the second address, etc. From the Z80's perspective, the screen is a contiguous set of addresses, starting from 0x4000 at the upper left corner and incrementing as you go to the right across the screen. Once you reach the right side, increment again to move to the left side at the second line. The table below maps the Z80 addresses into Screen addresses:

Memory Address	**Row**	**Column**
RAS#/CS#	Asserted when A[15:12] = 0x4	
MA5	A5	A11
MA4	A4	A10
MA3	A2	A9
MA2	A2	A8
MA1	A1	A7

Memory Address	Row	Column
MA0	A0	A6

As can be seen, the maximum address (lower right corner of the screen, but below the blanking line) is 0x4FFF.

In commercial mode, the screen is organized such that (starting from the left-most pixel of a line) pixels 0 through 3 come from the RAMs controlled by RAS0, pixels 4 through 7 come from the RAMs controlled by RAS1, pixels 8 through 11 come from the RAMs controlled by RAS2, and pixels 12 through 15 come from the RAMs controlled by RAS3. For these 16 pixels, the row and column addresses applied to all 32 RAMs are identical. Pixel 16 starts over again from RAS0, at the next greater address. Therefore the address applied to RAMs for these first 16 pixels at upper left corner of the screen is row = 000000_b, column = 000000_b, with RAS0 asserted to logical 1 for pixels 0 through 3, RAS1 for pixels 4 through 7, etc. But from the Z80's perspective, the line is still a contiguous set of addresses, starting from 0x4000 for the upper left corner and incrementing from there. So, how is the Z80 address mapped into Screen RAM addresses? The table below shows the mapping for Commercial mode:

Memory Address	Row	Column
RAS0#	Asserted when A[15:14] = 0x1 and A[1:0] = 0x0	
RAS1#	Asserted when A[15:14] = 0x1 and A[1:0] = 0x1	
RAS2#	Asserted when A[15:14] = 0x1 and A[1:0] = 0x2	
RAS3#	Asserted when A[15:14] = 0x1 and A[1:0] = 0x3	
MA5	A7	A13
MA4	A6	A12
MA3	A5	A11
MA2	A4	A10
MA1	A3	A9
MA0	A2	A8

Screen Refresh memory addressing is simpler, since all RAS#s are simultaneously asserted, and the address counter outputs simply increment at one-fourth the rate of consumer mode.

Referring to the Seawolf II schematic, all 32 DRAMs are located at the upper right, in four columns of eight DRAMs each. The 32 Data out (D0[7:0], D1[7:0], D2[7:0], and D3[7:0]) are connected to a set of four, 4/1 data multiplexers (74LS253 at chip locations U41 through U44). These parts are used to multiplex the read data from the selected DRAM (via RAS*X* where *X* is 0 to 3) onto an eight-bit bus so that it can be read by the Data chip. This path IS NOT used for the screen refresh. A simultaneous read of 16 pixels (i.e., all 32 DRAMs) is performed for screen refresh, and the data is latched into the four eight-bit parallel-in/serial-out shift registers at the bottom right of the page. These are 74LS166 shift registers. The shifters are shown in screen pixel order, so odd-numbered Data out lines (D07, D05, D03, etc.,) connect to the left two shifters (U37 and U38) and even-numbered Data out lines connect to the two right shifters (39 and U40). The rightmost shifter of each pair o(U38 and U40) outputs data on its Qh pin, which is connected to the SI pin of the leftmost shifter of the pair (U37 and U39). The Qh pin of the leftmost shifter is connected to a D flip-flop (U26, type 74LS74) which delays the shifted data by a clock. The outputs of these flip-flops connect to the Ser[1] (D07, D05, etc.,) and Ser[0] (D06, D04, etc.,) inputs of the Data chip. This is how commercial resolution data is input to the Data chip, since the MD[7:0] data path of the data chip is not wide enough to handle all 32 bits simultaneously. Note that these same pins (Serial1/0) are connected to ground in the BPA. Because of this, the BPA cannot display at commercial resolution without serious changes to the PC board. The shifters and D flip-flop are clocked by the 7 MHZ clock, therefore commercial resolution data is shifted into the Data chip at twice the rate of the consumer version. This makes sense, since the horizontal resolution of the commercial version is twice that of consumer, but the line timing is the same. The shifters are loaded with data whenever their S/L# pin is at logical 0, which occurs once every 16 7 MHZ clocks, as determined by U25, a 74LS161 hexadecimal counter. Otherwise, data is shifted to the left, from A toward H. Thus, a screen refresh reads 16 horizontally adjacent pixels (2 bits per pixel from 32 DRAMs) simultaneously, loads this data into two 16-bit shift registers, and shifts this data into the Data chip Serl[1:0] pins at a 7 MHz rate. This means that all four RAS signals are asserted simultaneously for a screen refresh. Since the Z80 treats the sixteen pixels as occupying four successive addresses, it follows that RAS0 corresponds to A[1:0] = 0x0, RAS1 corresponds to A[1:0] = 0x1, etc. It follows, then, that if a system were to be designed for commercial resolution using the Data chip, it would require 32-bit wide RAMs, or a narrower but significantly faster RAM and additional hardware to capture the successive screen refresh reads and serialize them for input to the Data chip Ser[1:0] pins.

Memory Timing

The diagrams below detail the timing of read and write (with and without extra WAIT states) accesses. They are based on the Nutting diagrams, with more waveforms included to (hopefully) clarify timing issues left uncertain by the original timing diagrams.

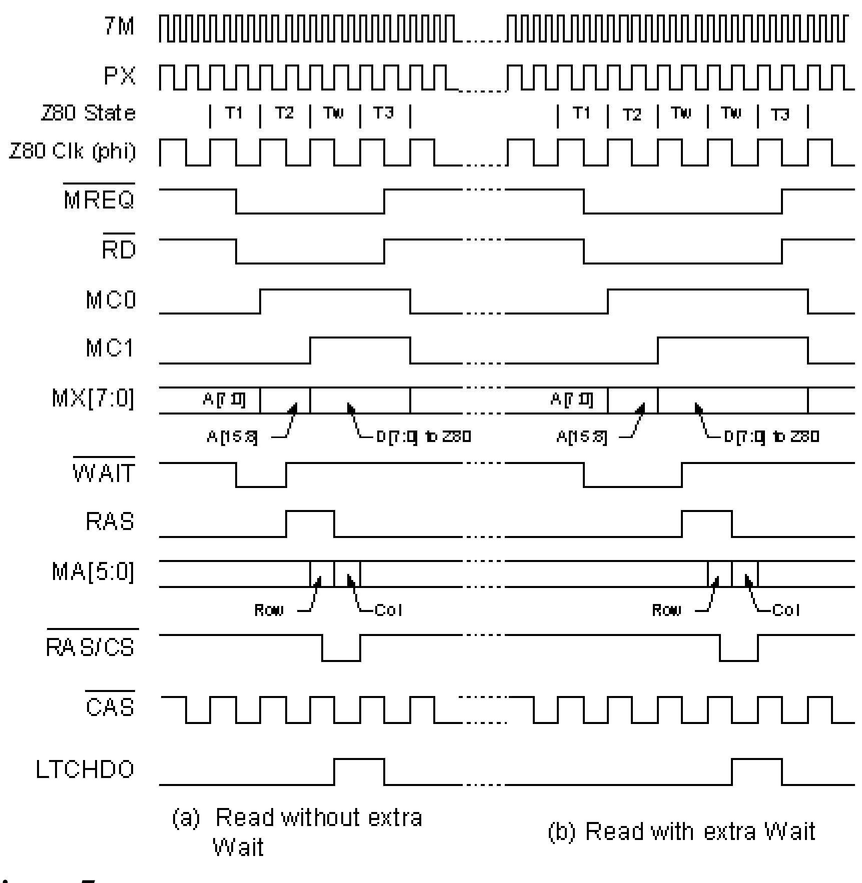


Figure 7: Memory Read Cycle Timing Diagrams

Memory Reads

The diagram at left details both types of memory read accesses, using a single Wait state (a) and using two Wait states (b):

The illustration is similar to those on NM126 and NM127, except that several more waveforms of interest are included. These waveforms show the relationship between clocks 7M, PX, and Φ, and also show the contents of the Microcycle bus (MX[7:0]) at each instant of the access cycle. The important relationship between the RAS output of the Address chip and the RAS/CS# signal to the DRAMs is also shown, along with the vital relationship between RAS/CS#, CAS#, MA[5:0], and LTCHDO. Note that CAS# runs all the time. This adds to the power dissipated by the power supply regulators, helping to keep the cabinet nice and warm. Since CAS# is free-running, memory accesses are required to be in specific time slots as determined by the CAS# timing. A memory access time slot starts and ends with the falling edge of Φ. The control signals MC[1:0] go from 0x0 to 0x1 to 0x3 during a read, connecting first the Z80 A[7:0] (which will become row addresses on MA[5:0] during the row time), followed by Z80 A[15:8] (which will become column addresses on MA[5:0] during the column time), and finally, the data read from the DRAMs. The DRAM read data becomes valid at the DO pin of each DRAM slightly after the falling edge of CAS#. Since DATEN# is logical 1 during this time, the MD[7:0] bus buffer (BPA chip U23) is on, U11 inverts DATEN#, applying a logical 0 to U23), driving this data back toward the Data chip. This data will be valid on MX[7:0] around halfway through the time when signal LTCHDO (**LaTCH D**ata **O**ut) is at logical 1, and will remain valid (since it is latched in the Data chip) until the cycle ends when MC[1:0] return to 0x0.The Data chip also latches data read from the DRAMs for a possible forthcoming Magic write. Since in consumer mode, Screen Refresh cycles occur on every other possible RAS timeslot, there is a 50% chance that an extra Wait state will be inserted for any Z80 Screen memory access cycle during active video time.

Memory Writes

The diagram below details both types of memory read accesses, using a single Wait state (a) and using two Wait states (b):

Again, this illustration is similar to NM124 and NM125, with additional waveforms included. Close inspection of the diagram will reveal several important differences as compared to the read cycle. First, the write to the DRAMs occurs a full Z80 clock cycle later. Second, the states of the Microcycler are different. The first and second states are identical in that the Z80 A[7:0] is applied first followed by Z80 A[15:8]. But now, the data *from* the Z80 appears on the Microcycle bus, instead of data to the Z80. The Address chip asserts signal WRCTL# to logical 0 as shown. The Data chip responds by sending data out on the MD[7:0] bus, and asserting its DATEN# output signal to logical 0. DATEN# at logical 0 turns the MD bus buffer off. The inverted DATEN# is also applied to NAND gate U12 which combines this signal with the inversion of CAS# to produce WE#, the write enable signal to the DRAMs. The data appearing on MD[7:0] is written into the DRAMs on the falling edge of WE#.

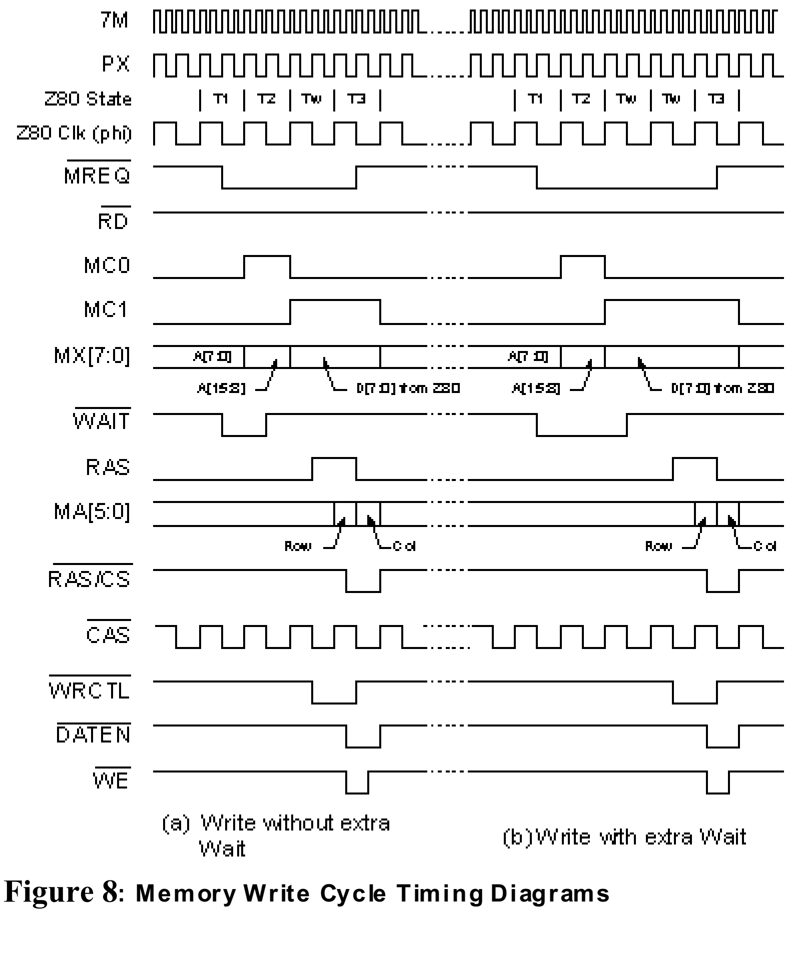


Figure 8: **Memory Write Cycle Timing Diagrams**

Screen Refresh Reads

The timing of a Screen Refresh read is identical to any other read. A Memory refresh cycle is identical to a Screen Refresh read cycle, except that the data is discarded. There are several differences between Consumer and Commercial Screen Refresh cycles, as shown in the timing diagram below:

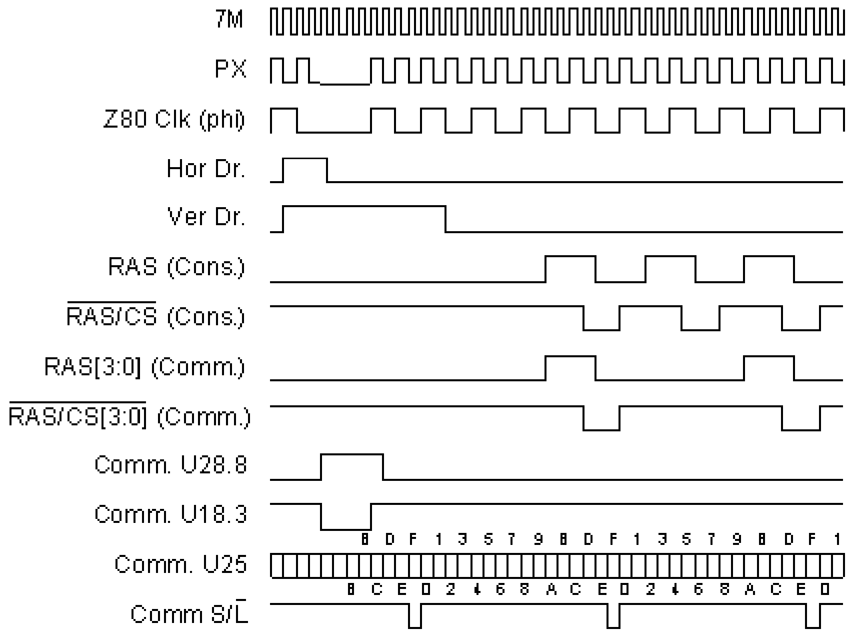

7M
PX
Z80 Clk (phi)
Hor Dr.
Ver Dr.
RAS (Cons.)
$\overline{\text{RAS/CS}}$ (Cons.)
RAS[3:0] (Comm.)
$\overline{\text{RAS/CS[3:0]}}$ (Comm.)
Comm. U28.8
Comm. U18.3
B D F 1 3 5 7 9 B D F 1 3 5 7 9 B D F 1
Comm. U25
B C E 0 2 4 6 8 A C E 0 2 4 6 8 A C E 0
Comm S/$\overline{\text{L}}$

Figure 9: **Screen Refresh Cycle Timing Diagram**

This diagram is similar to NM132, except that it shows RAS/CS# to the DRAMs, and the waveforms for a Commercial resolution system. As can be seen, the PX and Φ clocks stall at the beginning of each line, to re-synchronize Φ to make up for the non-integer relationship between it and 7M. This diagram, like NM132, shows Hor Dr. and Ver Dr occurring simultaneously. This only happens once per field, since Ver Dr occurs only once per field. Hor Dr and the clock stall occur at the beginning of every horizontal line. As you can seen, the Consumer Screen Refresh RAS occurs on every other Φ clock. Screen refresh data for the consumer system is latched into an internal Data chip shift register, then shifted through the Data chip color lookup table at the PX clock rate.

For the commercial system, the first Screen Refresh RAS occurs at the same time as the one for consumer, but it occurs only half as often, making three times as many RAS cycles available for Z80 accesses. Each commercial Screen Refresh access reads 16 pixels, and loads them into an external (to the Data chip) shift register. For Seawolf II, D flip-flop U28 detects the PX clock stall at the beginning of the line. Pin 8 of this flip-flop goes to logical 1 at this time as shown above. The U28 output is combined with PX# in NAND gate U18 to generate the logical 0 pulse (U18.3), which loads hexadecimal counter U25 to value 0xB. This counter is used to generate an S/L# pulse to the shift registers every 16 7M clocks, to load the data read from the DRAMs. The counter outputs a logical 1 pulse on its TC pin (not shown above) which is inverted by U22 to form signal Comm S/L#. The numbers/letters above and below the Comm. U25 waveform are the hexadecimal values in the counter at each instant of time. Loading value 0xB when Comm. U18.3 is at logical 0 will insure that Comm. S/L# goes to 0 at the correct time, during the Screen Refresh RAS# cycle, when read data is valid. The left-most pixel of the data read will be valid at the shift register output when S/L# returns to logical 1, followed by the next pixel, etc., at each successive 7M positive edge. The serialized pixel data is delayed by one more 7M clock by D flip-flop U26 before being applied to the Ser[1:0] inputs of the data chip. This data is then shifted through the Data chip color lookup table like the consumer version. The difference here is that the data is shifted at twice the rate (7M).

The Data Chip

The following is a block diagram of the Data chip. It is similar to, but more detailed than the one on NM116. As can be seen, the Block Diagram is rearranged somewhat, which hopefully makes it much easier to understand.

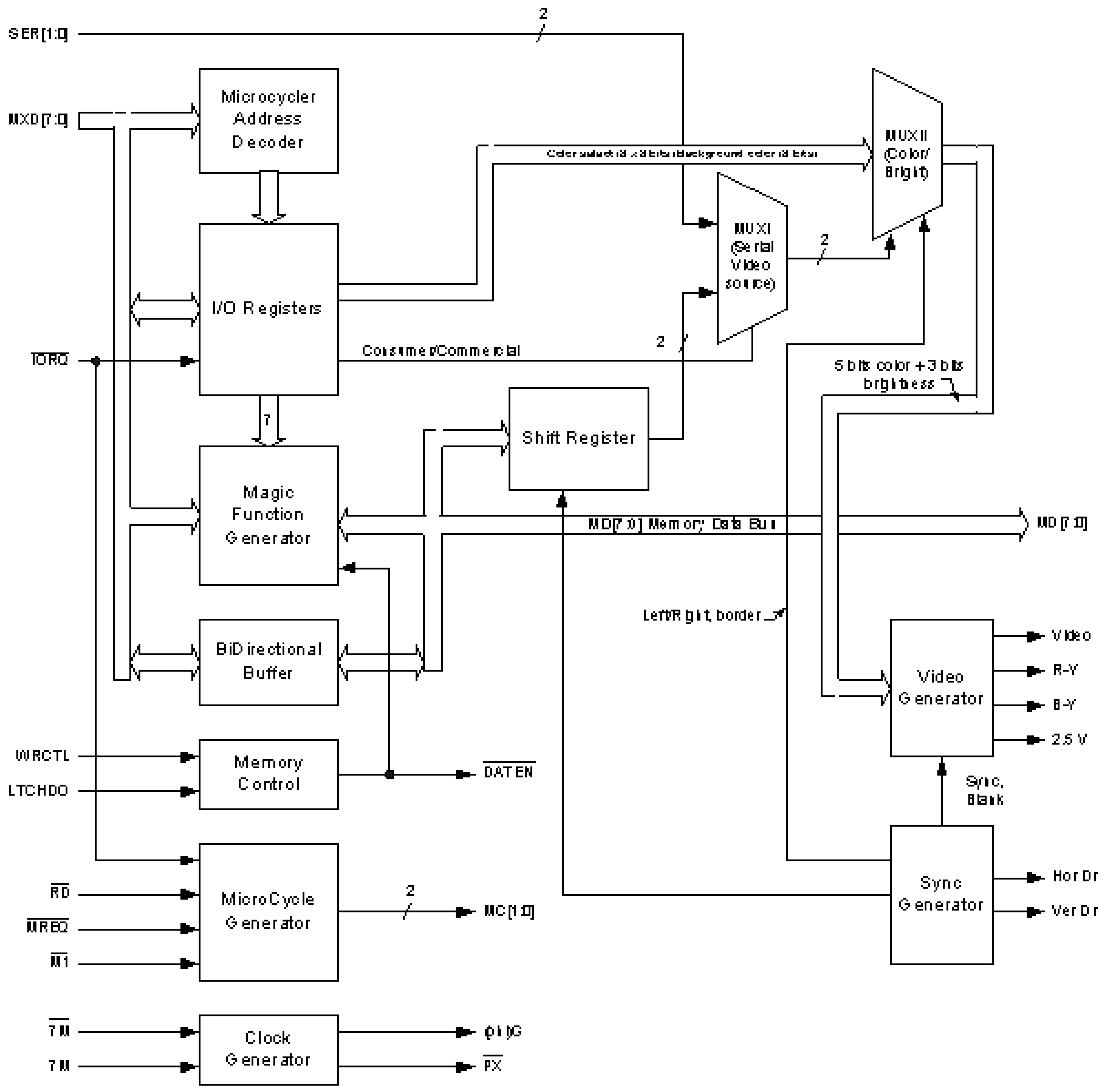


Figure 10: **Data Chip Block Diagram**

Microcycle Address Decoder

Similar to the Address chip, the Microcycle Address Decoder reconstructs the Z80 address from the multiplexed Z80Address Bus. These addresses are used to select the appropriate I/O register during an I/O cycle.

I/O Registers

Address chip registers are tabled below. The column labeled **I/O** defines whether the register is an input or an output (or both) of the Z80.

Address	I/O	Name/Description
0x00	O	Color Register 0. This register defines the color (bits [7:3]) and Brightness (bits [2:0]) of pixels whose content is 0x0 and which are to the right of the color boundary as defined by the register at I/O address 0x09.
0x01	O	Color Register 1. Similar to Color Register 0, except that this register affects pixels whose content is 0x1.
0x02	O	Color Register 2. Similar to Color Register 0, except that this register affects pixels whose content is 0x2.
0x03	O	Color Register 3. Similar to Color Register 0, except that this register affects pixels whose content is 0x3.
0x04	O	Color Register 4. Similar to Color Register 0, except that this register affects pixels whose contents is 0x0 and are to the left of the color boundary as defined by the register at I/O address 0x09.
0x05	O	Color Register 5. Similar to Color Register 4, except that this register affects pixels whose content is 0x1.
0x06	O	Color Register 6. Similar to Color Register 4, except that this register affects pixels whose content is 0x2.
0x07	O	Color Register 7. Similar to Color Register 4, except that this register affects pixels whose content is 0x3.
0x08	I	Intercept Feedback. See text for definition.
0x08	O	Consumer/Commercial. The Data chip needs this bit to determine how the Screen Refresh Data is handled. This register is duplicated in the Address chip. When written to 0x0, the system is in consumer resolution. Writing this register to 0x1 puts the system into commercial resolution mode, assuming there is sufficient Screen memory and other hardware (shift registers, multiplexers, etc.) to support this mode.
0x09	O	Background color (bits [7:6]), Horizontal color boundary (bits 5:0]). The color boundary is byte (not pixel) addressed, and defines the byte immediately to the left of the boundary. For Bytes 0 to (color boundary -1), Color registers 4 through 7 contents will be used. For bytes color boundary to 0x27 (consumer) or 0x4F (commercial), Color registers 0 through 3 will be used. The Background color uses the same register definition as above for left and right of the color boundary.
0xA	O	Line number below which the background color is displayed. This makes it possible to use the underlying Screen RAM as Scratch. This register is duplicated in the Address chip.
0xB	O	A block transfer to this register will load Color registers 7 through 0 (in that order) with the data written to this address.
0xC	O	Magic register. See text for explanation.
0xD	I/O	Interrupt feedback register. Upon Interrupt Acknowledge by the Z80, the previously-written contents of this register are driven back to the Z80 through the Microcycle bus.
0x19	O	Expand register. See text for explanation.

MUX I and the Shift Register

In consumer mode, a Screen Refresh access is made to Screen memory for one out of every two Z80 clocks during active video time. The data, input on the MD[7:0] bidirectional Data bus between the Data chip and the Screen RAMs, is latched into the shift register, where it is shifted out, two bits at a time at the PX clock rate to one input pair of MUX I. Bits [7:6] are shifted out first, followed by bits [5:4], etc. Just prior to the fourth pixel (bits [1:0]) shifted out, another Screen Refresh Access occurs to reload the Shift Register. In commercial mode, the serialized pixels are routed from input chip pins Ser[1:0] through MUX I. The output of MUX I is a two bit pixel, applied to MUX II.

MUX II

MUX II uses the serialized pixel, the left/right boundary and blank signals coming from the Sync Generator to determine which 8-bit pixel value to output. These values come from the Color and Background registers as determined by the table below:

In Border area of Screen?	Left of Horizontal Color Boundary?	Border register bits[7:6]	Pixel contents [1:0]	Color register contents output from MUX II
Yes	Yes	0x0	don't care	0x4
Yes	Yes	0x1	don't care	0x5
Yes	Yes	0x2	don't care	0x6
Yes	Yes	0x3	don't care	0x7
Yes	No	0x0	don't care	0x0
Yes	No	0x1	don't care	0x1
Yes	No	0x2	don't care	0x2
Yes	No	0x3	don't care	0x3
No	Yes	don't care	0x0	0x4
No	Yes	don't care	0x1	0x5
No	Yes	don't care	0x2	0x6
No	Yes	don't care	0x3	0x7
No	Yes	don't care	0x0	0x0
No	Yes	don't care	0x1	0x1
No	Yes	don't care	0x2	0x2
No	Yes	don't care	0x3	0x3

Video Generator

The output of MUX II is an 8-bit pixel, with bits [7:3] defining one of 32 colors, and bits [2:0] defining 1 of

8 intensities. This 8-bit bus is connected to the Video Generator, which combines these signals with the Sync and Blank signals output by the Sync Generator to generate the video output signals Video, R-Y, B-Y, and 2.5 V.

B-Y, R-Y and +2.5 V Outputs

A simplified schematic of the R-Y, B-Y, and 2.5 V output generation circuitry appears in the following diagram:

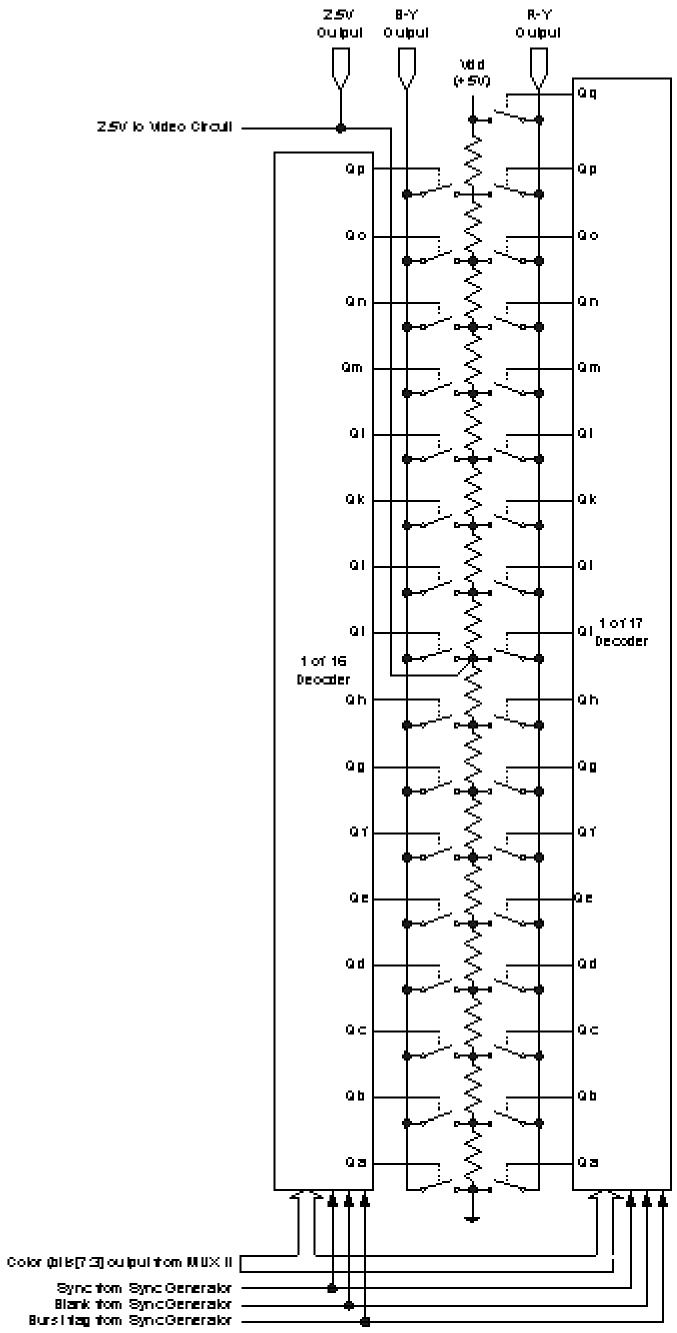


Figure 11: **B-Y, R-Y, and 2.5 V output Circuitry**

As can be seen, the five color bits output from MUX II connect to both decoders, along with sync, blank, and burst flag inputs from the sync generator. These last three inputs are used to force the B-Y and R-Y to specific voltages during the time when these signals are asserted, which is during horizontal blanking. The sequence of syng signal assertions are as follows:

End Active Video->Start Blank->Start Sync->End Sync->Start Burst->End Burst->End Blank->Start Active Video

B-Y and R-Y are forced to specific voltage levels during burst flag, to inform the video modulator that color burst must be inserted into the signal at this time.

The color output circuitry is based on a resistor ladder network, consisting of a series connection of 16 resistors, between +5 Volts and Ground. The resistors are not of equal value, but are symmetrical about the centerconnection, which is at 2.5 VDC. This connection is brought out of the chip to act as a reference voltage for the video modulator. It is also used as the center reference for the Video output, which will be covered later. The resistor values are weighted such that there is little voltage change going up from ground or down from +5 Volts, and the voltage difference increases as you approach the center of the network from either end. The choice of resistor values gives a reasonable distribution of colors which can be displayed on a typical TV set. Each decoder will activate one and only one of its outputs for any combination of input values. The selected output turns on the appropriate switch, connecting the selected voltage from the resistor network to the output pin.

Video Output

As shown at right, the Video Output is similar to the R-Y, B-Y outputs, but consists of only eight resistors. It functions in a similar manner to that described for the B-Y, R-Y, and +2.5 V outputs.

Sync Generator

The Sync Generator consists of counters and flip-flops which generate standard NTSC composite sync and blank waveforms. These are used to force the outputs of the Video Generator to specific voltages which can be decoded by a TV set into the correct horizontal and vertical scanning waveforms required to display a picture. The Sync Generator also outputs Hor Dr and Ver Dr, which are used to synchronize a similar counter running in the Address chip. This latter counter is used to determine when to perform Screen Refresh cycles, along with generating the proper Screen RAM addresses to refresh the Screen from. The Sync Generator circuitry also generates the correct Load/Shift controls to the Data chip Shift Register, used to shift pixels through MUX I and MUX II in consumer resolution mode, and to control which sets of color and border registers are used in MUX II in both resolution modes.

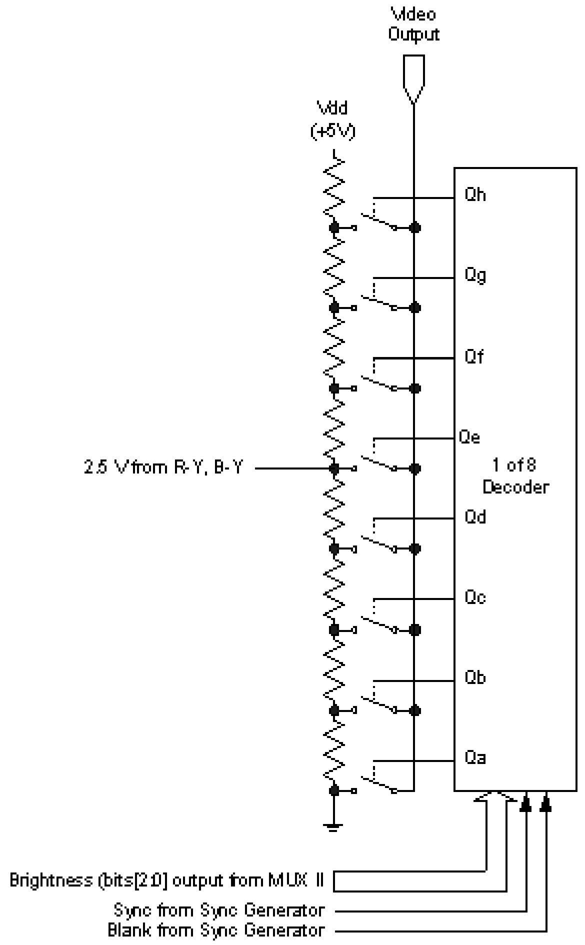


Figure 12: **Video Output Circuitry**

Clock Generator

The Clock Generator takes the 7M, 7M# inputs and divides them by two to generate the PX clock output. 7M and 7M# are again divided by two to generate ΦG, the clock to the Z80. Both PX and ΦG are stalled for three cycles of 7M at the beginning of each horizontal line so that there will be an integer number (113) of ΦG clocks per horizontal line. PX is stalled to keep it synchronous with ΦG.

Microcycle Generator

The Microcycle Generator is used to generate the MC[1:0] signals at the correct times to control activity on the Microcycle bus. The following truth table comes directly from NM109 and NM123:

RFSH#	MC1	MC0	MX[7:0] Contents
0	0	0	A[7:0] from Z80
0	0	1	A[7:0] from Z80
0	1	0	A[7:0] from Z80
0	1	1	A[7:0] from Z80
1	0	0	A[7:0] from Z80

RFSH#	MC1	MC0	MX[7:0] Contents
1	0	1	A[15:8] from Z80
1	1	0	D[7:0] ***from*** Z80
1	1	1	D[7:0] ***to*** Z80

Not documented in the table above is a special cycle which occurs in response to a Z80 Interrupt Acknowledge. When the Z80 is acknowledging an interrupt, it simultaneously asserts its IORQ# and M1# outputs. This is detected by the data chip, which responds by asserting MC[1:0] to 0x3, and outputting the contents of its Interrupt Feedback Register (I/O address 0x0D) onto the Microcycle Bus. The control of MC[1:0] is done internal to the Data chip, but the additional control provided by the Z80 RFSH# signal is implemented externally. The following schematic details the Z80-side connections to the Microcycle Bus.

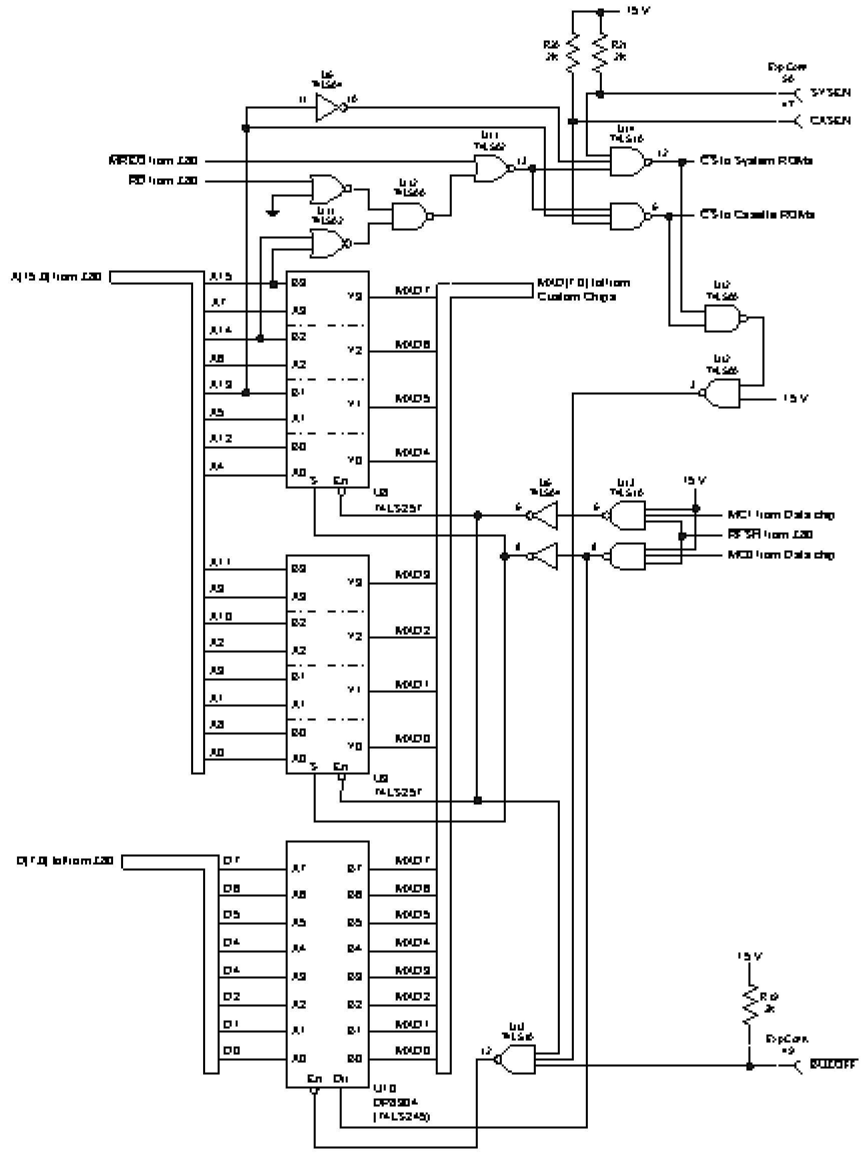


Figure 13: **Microcyle Bus Circuit Details**

The Microcycle bus was implemented to save on chip pins for the three custom chips. It time-multiplexes the 16-bit address and 8-bit data bus down into a bidirectional 8-bit multiplexed bus. What information the bus contains at any instant of time is controlled by the Data chip MC[1:0] outputs, and the Z80 RFSH# signal, as detailed in the table above. The preceding illustration gives the circuit details.

Two quad 2-1 multiplexers (BPA U8 and U9, which are type 74LS257) are used to switch in the Z80 address bus. Each has eight data inputs (B[3:0] and A[3:0]) which are connected to the Address bus, four data outputs (Y[3:0]) which are connected to the MX bus, and a pair of control inputs, En (**En**able) and S (**S**elect). Control input En, when at logical 0 turns the outputs on, so that they may drive the MX bus from the selected address input. The S input selects which set of inputs connect to the outputs. If S is logical 0, inputs A[3:0] are connected to outputs Y[3:0] (assuming En is logical 0). Otherwise with S at logical 1, inputs B[3:0] are connected to outputs Y[3:0], respectively.

The data connection between the Microcycle bus and the Z80 is provided by BPA U10 which originally was type DP8304, and was replaced by the 74LS245, a chip which is functionally and pin-for-pin compatible. This chip has eight identical bi-directional drivers, which are controlled by the En (**En**able) and Dir (**Dir**ection) inputs. Like the En input of the 74LS257, this chip is enabled or 'on' when En is logical 0. When En is logical 1, the chip will not drive data in either direction. With En at logical 0, the Dir pin controls the direction of data flow. When both En and Dir are at logical 0, data is driven from the B pins to the A pins. When En is logical 0 and Dir is at logical 1, data is driven from the A pins to the B pins.

The three-input NAND gate U13 and inverter U6 combine MC[1:0] and RFSH# to control the action of the Microcycle bus. When both MC1 and MC0 are at logical 0, both NAND gate outputs are forced to logical 1. Inverters U6 invert the signals to apply logical 0 signals to both U8 and U9 En and S pins. Therefore the multiplexers are on (En is 0) and the A[3:0] inputs connect to outputs Y[3:0]. Thus, Z80 Address bus bits A[7:0] connect to MX[7:0]. Note that if RFSH# is logical 0, the identical conditions exist.

When MC1is at logical 0 and MC0 is at logical 1, the multiplexers are still enabled (U6 pin 6 is at logical 0 and U6 pin 8 is at logical 1), but the S inputs are now at logical 1. This connects the B[3:0] inputs to the Y[3:0] outputs. Z80 Address bus bits A[15:8] now appear on MX[7:0].

With MC1 at logical 1and Z80 signal RFSH# at logical 1, U8 and U9 are disabled in that the En pin of each is at logical 1 (U6 pin 6 is now at logical 1). Under this condition, U10 may be enabled, too, since one of the inputs to the lower gate of U13 (the 74LS10 whose output is connected to the En pin of U10) is at logical 1. In order for U10 to be on, all three inputs of the U13 gate whose output is on pin 12 must be at logical 1. The gating shown at the top of the drawing controls this, as does signal BUZOFF#, which comes from the Expansion Connector. The gates at the top of the illustration will be covered first.

NOR gates U11 and one gate of NAND gate U12 combine Z80 signals A[15:14], MREQ# and RD# such that pin 13 of NOR U11 will go to logical 1 during a memory read when A[15:14] are 0x0. In other words, this output will go high when the Z80 reads from System or Cassette ROM. This signal is input to two gates of NAND U14. Other inputs to these gates include Z80 A13 and A13# (from inverter U6 pin 10), SYSEN and CASEN from the Expansion Connector. If A13 is logical 0, A13# is at logical 1. During a System memory read, U11 pin 1 will be 1 and A13# will be 1. If nothing is connected to SYSEN, resistor R21 will pull the other input to U14 to logical 1 and U13 pin 12 will go to logical 0, chip-selecting the System ROMs. The ROMs will now output the contents of the addressed location onto the Z80 Data bus. SYSEN, if pulled to a logical 0, will prevent the System ROM contents from appearing on the Z80 Data bus. CASEN woks similarly, except that A13 must be logical 1, and Expansion Connector signal CASEN is used to disable the Cassette chip select. During normal operation this pin, like SYSEN, is left open and R20 provides the logical 1 to permit the CS# to the Cassette ROMs to occur. Both chip selects are combined in NAND gate U12 such that pin 3 of this chip will go to logical 0 when either the System ROMs or the Cassette ROMs are selected. This input to U13 prevents pin 12 from going to logical 0, keeping U10 off. With U10 off, the current MC[1:0] state is irrelevant and the data read from ROMs will bot be disrupted by anything occurring on the Microcycle bus.

The final control signal which affects U10 Enable is the Expansion Connector signal BUZOFF#. This signal was added late in the design when it was realized that reads from the Expansion Connector address space had no way of preventing Microcycle bus contents from disrupting the data coming from the Expansion Connector. Any time BUZOFF# is asserted to logical 0, Data bus to Microcycle Bus buffer U10 is turned off. When enabled, the direction of data flow for U10 is as follows: when MC0 and RFSH# are both logical 1, data will flow from the B pins to the A pins, from the Microcycle bus to the Z80 data bus. When MC0 is logical 0, data flows from the Z80 data bus to the custom chips. Note: this is all signal BUZOFF# does.

Memory Control and Bidirectional Buffer

The Memory Control block takes the input signals WRCTL# and LTCHDO generated by the Address chip and uses them to control when data is latched internally or when the various buffers are enabled. The signal WRCTL#

asserted to logical 0 indicates that the memory access is a write. If a standard memory write is in progress (i.e., the address is between 0x4000 and 0x7FFF) the bidirectional buffer is enabled to drive data from the Microcycle bus to the memory data (MD[7:0]) bus. If a Magic memory write is in progress (i.e., the write address is between 0x0000 and 0x3FFF) the output of the Magic function generator is enabled on the MD bus. In either event, output DATEN# is asserted to logical 0 to perform a memory write. If a memory read is in progress, LTCHDO will be asserted. This will enable the bidirectional buffer to drive data from the MD bus to the Microcycle bus, so that it can be forwarded to the Z80. The read data will also be latched in the Magic Function generator, if the Magic register bits were set for a forthcoming OR or XOR function. Reads keep DATEN# at logical 1 so that the MD bus buffer (U23) will be enabled to drive memory data back toward the Data chip.

Magic Function Generator

The Magic Function Generator is used to perform the various manipulations to the data going to memory. Since the memory from address 0x0000 through 0x3FFF is Read-Only, memory writes to this address range writes data to the equivalent Screen Memory address (Address written + 0x4000), but with the magic function applied. The Magic Function generator uses a pair of registers, expand (I/O Address 0x19), and the Magic register itself (I/O Address 0xC). This latter register only contains 7 bits, which are defined below:

Bit[7]: Not used

Bit[6]: Flop. If set to logical 1, the data written to RAM will be flopped: Pixel 3 (bits[7:6]) will be swapped with Pixel 0 (bits[1:0]) and pixel 2 (bits[5:4]) will be swapped with pixel 1 (bits[3:2]).

Bit[5]: XOR. If set to logical 1, the data written to RAM will be the bitwise logical XOR of data just read from this memory location, and the data coming from the Z80. If the pixel content in one of the currently addressed pixels was non-zero (i.e., 0x1, 0x2, or 0x3) *and* the new pixel to be written is also non-zero, a bit corresponding to the pixel position in the Intercept register will be written to logical 1.

Bit[4]: OR. If set to logical 1, the data written to RAM will be the bitwise logical OR of data just read from this memory location, and the data coming from the Z80. An Intercept will be generated under the conditions stated above for XOR writes.

Bit[3]: Expand. If set to logical 1, an expand write will take place. This permits ROM savings by expanding single bits of ROM storage into two-bit pixels. The expand function is assumed to consist of at least two memory writes, after the set-up write of the Magic register. The first, third, etc., write will expand bits[7:4] of the incoming byte, and the second, fourth, etc., write will expand bits[3:0] of the incoming byte. The set-up write clears a flip-flop which is toggled be each successive Magic Address write. On the first Magic Address write following the set-up, bits[7:4] of the data written will be used as look-up values into the Expand register to determine what the final MD bus content will be for the write. A bit in the incoming nybble at logical 1 will result in bits contained in the Expand register bits [3:2] appearing in the equivalent pixel-position of the MD bus. Bits set to 0 in the incoming nybble use bits [1:0] of the expand register. For the first and each subsequent odd-numbered write, bit[4] of the incoming nybble maps to bits[1:0] of the data written, etc. For even-numbered writes, bit[0] of the incoming nybble maps to bits[1:0], etc.

Bit[2]: Rotate. When set to logical 1, a process is enabled which will rotate the data written to memory by 90 degrees in the clockwise direction. It takes a total of eight Z80 write cycles to write 16 rotated pixels, and this function only works in commercial mode. This function operates as follows:

1. Write the first byte to a location in Magic memory. This results in intermediate data written to memory, but this will be over-written in the fourth cycle by the rotated data. The first 4 of the 16 rotate registers internal to the Data chip are written with the properly-formatted data.

2. Write the second byte to the first address + 80 (0x50). This, too results in intermediate data written to memory which will be over-written in the fifth cycle. Now eight of the 16 rotate registers contain the properly-formatted data.

3. Write the third byte to the first address + 160 (0xA0). This, too results in the intermediate data write. Now twelve of the 16 rotate registers contain the properly formatted data.

4. Write the fourth byte to the first address + 240 (0xF0). Intermediate data is again written to memory, and all sixteen rotate registers now contain the correctly formatted data.

5. Write the fifth byte to the original address. The data written by the Z80 is ignored, and the rotated data is written to the Screen memory

6. Write the sixth byte to the first address + 80. The data written by the Z80 is ignored, and the rotated data is written to the Screen memory.

7. Write the seventh byte to the original address + 160. The Z80 data is again ignored, and the rotated data is written to the Screen memory.

8. Finally, write th eighth byte to the original address + 240. The Z80 data is ignored, and the rotated data is written to the Screen memory.

The following shows pre-and post rotated data:

P3	P2	P1	P0
P7	P6	P5	P4
P11	P10	P9	P8
P15	P14	P13	P12

Original

P15	P11	P7	P3
P14	P10	P6	P2
P13	P9	P5	P1
P12	P8	P4	P0

Rotated

Figure 14: **Before and After Rotated Data**

Bits[1:0] Shift. Writing data with bits[1:0] greater than 0x0 results in that data being written to Screen RAM but shifted to the right by one, two, or three pixels as determined by the contents of this field. Successive shifted writes must follow at incremental addresses. The following shows examples of shifted and flopped data:

Byte 0				Byte 1				Byte 2				
P3	P2	P1	P0	P7	P6	P5	P4	P11	P10	P9	P8	Original Data
0	P3	P2	P1	P0	P7	P6	P5	P4	P11	P10	P9	Shift 1
0	0	P3	P2	P1	P0	P7	P6	P5	P4	P11	P10	Shift 2
0	0	0	P3	P2	P1	P0	P7	P6	P5	P4	P11	Shift 3
P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	Flopped

Figure 15: **Examples of Shifter and Flopper Operation**

When performing Magic functions, Shifting and Flopping cannot be enabled at the same time, nor can OR and XOR be enabled at the same time.

The Intercept register (I/O Address 0x8) contains several flag bits which save the result of previous OR or XOR Magic Memory writes. If any of bits[3:0] are set, this indicates that the previous write to the corresponding pixel (bit[3] = pixel[3], etc.,) wrote a non-zero pixel with a non-zero value. Bits[3:0] track the last Magic memory write, while bits[7:4] track *any* write to the corresponding pixel (bit[7] = pixel[3], etc.,) since the last time the Intercept register was read. Thus, once any of bits[7:4] are set to logical 1, they will remain that way until the register is read, at which time these bits will be cleared to logical 0.

Summary of BPA Operation

In commercial mode, pixels read during Screen Refresh memory reads are latched into a shift register which is external to the Data chip, and shifted into Data chip inputs Ser[1:0] at the 7M clock rate, for twice the horizontal resolution of consumer mode. In the BPA, inputs Ser[1:0] are connected to ground (logical 0), and there is no external shift register. Therefore, the BPA will never display commercial resolution graphics, without major changes to the Printed Circuit board.

A Note about the Blue RAM Accessory

Several posts to the Bally Alley Astrocade Discussion Group have mentioned using the Perkins Engineering Blue Ram Accessory to somehow magically display commercial resolution graphics on an otherwise normal BPA. The schematics are available at:

http://www.classicgaming.com/ballyalley/perkins/blue_ram_schematics.pdf

These schematics are a bit difficult to follow, since they are split up into five pages, and the outputs of certain gates are shown connected to a three-bit bus, but nothing indicates what gate output drives what bit. The Blue Ram seems to be a 16K or 32K RAM/PROM Add-on which uses the BPA Expansion connector. This accessory also has a ZIF (Zero-Insertion-Force) socket into which an optional keyboard or other accessory can be connected, and a chip which implements a pair of eight-bit I/O ports to which other devices can be attached. Once loaded, it appears as though the expansion memory can be write-protected. There are several connections (or lack thereof) to the Arcade Expansion connector which will be covered below:

Video In/Video Out:	these are NOT CONNECTED (NC, as called out on the schematic) to anything on the Blue Ram board. Therefore there is no video generated by this board. Note that the Expansion connector pins Video In and Video Out connect to opposite ends of BPA R33, which connects between the Video output of the Data chip and the video input of the RF modulator. This pin only has video brightness information, and contains ***NO COLOR DATA***. This means that anything connected to these pins cannot generate color video, and could conceivably damage the Data chip if an incorrect level were applied to the Video Out pin.
7 M, PX Clk, Vert Dr, Hor Dr	these are also NOT CONNECTED to anything on the Blue Ram board. These signals would be necessary to generate any video on this board, since any video generated must be synchronized to that generated by the BPA in order to overlay any BPA video. It should now be obvious that this board does not generate any video.
BUSREQ#, BUSACK	these are also NOT CONNECTED, which means that nothing in the Blue Ram can take over control of the Z80 bus.
CASEN, SYSEN, BUZOFF#	these are connected to 74LS09 gates. A 74LS09 is an open-collector AND gate. The text above discusses how all three inputs work, but I'll repeat it here. A logical 0 on CASEN will stop any Cassette ROM from driving data onto the Z80 data bus. A logical 0 on SYSEN will stop a System ROM from driving data onto the Z80 data bus. A logical 0 on BUZOFF# will turn off BPA U10, preventing it from driving data onto the Z80data bus, or anything from driving data onto the Microcycle bus.

No amount of magic (software or otherwise) on the Blue Ram board will result in a BPA display commercial resolution graphics.

A Note about the R & L 64K RAM Accessory

This accessory functions similarly to the Blue Ram. Differences are that there is no I/O chip, and all 64 K of Z80 address space can be populated with RAMs or PROMs. The signals CASEN, SYSEN, and BUZOFF# are driven in a similar manner to the Blue RAM, except that since there are more sockets for memories, wider (i.e., more inputs) gates are used to generate the control signals, and a 74125 tri-state buffer actually drives the signals back to the BPA. Therefore, the R & L 64K RAM will not get the BPA to display commercial resolution video, either.