USER'S MANUAL

FOR THE RASTER MEMORY SYSTEM

VERSION 3.00

APRIL 15, 1984

MOTOROLA SEMICONDUCTOR PRODUCTS SECTOR
CONSUMER STRATEGIC MARKETING
PHOENIX, ARIZONA

(602) 244-6381

SUBJECT TO CHANGE WITHOUT NOTICE

The figures have been grouped together at the back of this manual. This, combined with the loose-leaf manual format, allows the user to remove and look at the figures while referring to the text.

The information in this manual has been carefully checked and is believed to be correct. If differences exist between this manual and the data sheets, the data sheets have precedence. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others.

TABLE OF CONTENTS

APPENDICES

LIST OF TABLES

1.Ø INTRODUCTION

The Raster Memory System provides sophisticated video display and memory management for microprocessor-based systems. It is really two systems in one chip set: Machine 1 provides the full RMS power for new designs, and machine 2 provides backward compatibility with MC68Ø9E systems using the MC6883-MC6847 combination.

This chapter of the Raster Memory System User's Manual contains an overview of the RMS' capabilities, to allow evaluation by system designers. Hardware design considerations are discussed in Chapters 2 (Microprocessors), 3 (Memory Organization), 4 (Hardware Interface), 5 (Reset and Initialization), and 13 (Pin Drive and Loading). Hardware and software issues are covered in Chapter 6 on General Video. Machine 1 software features are detailed in Chapters 7 (Display Modes), 8 (True Objects), 9 (Control Registers), 1Ø (System Memory Map), 11 (Video Overlay), and 12 (Real Time Software). Machine 2 is discussed in Chapter 14. As discussed there, some parts of Chapter 9 apply to Machine 2 operation.

1.1 RMS Highlights

The Raster Memory System (RMS) is a video display generator system with the following characteristics:

o Intended for personal and home computers and teletext/videotex.

o Compatible with NTSC and PAL displays with or without interlace.

o Supports up to 1 Mbyte of dynamic RAM (DRAM) including refresh.

o Low parts count with MC68Ø9E and MC68ØØØ family MPU's.

o Horizontal resolution from 64 to 64Ø pixels.

o Vertical resolution from 64 to 5ØØ pixels.

o Bit-plane mode and 6 character/object-oriented list modes.

o 32 available colors from a palette of 4Ø96.

o ASCII and mosaic characters in internal ROM.

o From 32 to 32K user-definable characters in the list modes.

o Text oriented attributes: underline, flash, invert, color, double height and width.

o Game oriented attributes: collision, priority, color offset.

o Virtual screen much larger than visible screen, smooth scrolling.

o 2 chips - Raster Memory Controller (HCMOS) MC68487
Raster Memory Interface (MOSAIC 1.5 bipolar) MC68486

o 8 hardware objects positioned by XY registers.

o 2 machine modes allow MC6847/MC6883 emulation or full feature operation.

1.2 Discussion of Terms

It is based on the Texas Instruments TMS52ØØ speech processor chip. The unit makes use of its own 65Ø2 microprocessor, and interfaces as if it were a printer. It was available in RS-232 serial or Centronics parallel versions. Upon power up, the Echo unit responds with the phrase Echo Ready,, to let you know all is well. One of the first points the user will notice is that the Echo is capable of intoning a sentence. Rather than speaking in a monotone, the pitch of the voice is dynamic. This makes for a more intelligible and less grating speech quality. You can use the internal speaker of the unit or route the sound to an external speaker. This section provides an orientation to the RMS capabilities and defines some of the terms used to describe them.

The first terms, pixel and pel, both mean picture element. Traditionally, pixel refers to the smallest physical picture element, which is the resolution of the CRT being used, and pel refers to the smallest logical picture element, which is set by the rest of the video system. This manual uses pixel to mean the element set by the particular horizontal and vertical screen resolutions selected by the user; these set the limit on system resolution, and it is assumed that they will be picked to match the CRT resolution. Pel is used to refer to all other picture elements.
The RMS uses a block of DRAM as screen memory to contain the display information. The user locates this block using RMS registers as pointers. The data may be larger than the user's display can show at one time; the full set of data is the virtual screen, and the data currently being displayed is the displayed screen. The user may scroll smoothly through the virtual screen, moving the displayed screen as little as one pixel at a time vertically or horizontally or both by changing at most four RMS registers, while leaving the screen memory unchanged.

The user can display individual pels in the bit-plane mode, or characters and fixed objects in any of the six list modes. The screen memory in the bit-plane mode is arranged in scan lines. Within each scan line, the color of the first pel is followed by the color of the second, and so on. The characters and fixed objects used in the list modes are defined in image tables which contain their pel-by-pel descriptions. The list mode screen memory is arranged as a display list of pointers to entries in the image tables, character row by character row. The list modes allow the display list to include attributes for the images; these allow each individual occurrence of an image to be altered, for example by underlining or flashing. Characters and fixed objects differ from each other in the attributes that they may use, but the main difference is that fixed objects may interact with true objects and characters may not.

The true object follows few of the rules of the other images. Its many unique features are explained in its own chapter, but its primary distinction is that it is designed to move. Each of the eight true objects is placed on the screen using pointers and XY registers in the RMS. The pointer indicates the object's location in its image table in memory, and the XY registers locate the object on the displayed screen. Flags indicate when each object has been completely displayed; the MPU may then change the XY registers to either move it smoothly on the screen or to create another object further down the screen. In this way, by changing only RMS registers, many more than 8 true objects may appear on the screen, but only 8 can appear in any single scan line. Registers also report overlap of true objects and fixed objects or of two true objects, and priorities allow control of the third dimension: objects can pass in front of or behind other objects. True objects may also be used in the bit-plane mode.

The RMS offers the user a variety of character types. It has 96 ASCII alphanumerics and two types of mosaics in internal ROM and allows a Dynamically Redefinable Character Set (DRCS) to be user defined. The DRC's come in several varieties, depending on the list mode. Some are 8 pels wide and use exactly the same attributes as the alphanumerics, some are 8 pels wide but with different attributes, and others are 16 pels wide.

Independent of the display mode, the user can set HRES, the displayed screen's horizontal resolution, and VRES, the vertical resolution.

The Color Mapping RAM (CMR) consists of 32 registers in the RMS. Each register can be set to any one of 4Ø96 colors. The bits that select colors in the bit-plane screen memory and the list mode image tables are used to select CMR registers, rather than actual colors.

Other terms are defined when they are introduced.

1.3 System Features

A simple transistor reset circuit indicates to the RMS on reset whether the MPU is an MC68Ø9E or an MC68ØØØ family member, and whether the display uses 525 or 625 scan lines. Everything else is software-controllable. The user can set the RMS registers, load the DRAM with image tables, fill the screen memory, and go. Once the system is running, the RMS allows flexible dynamic screen operation; the user can save the screen XY position at a light pen or other TTL input, change the Color Mapping RAM in the RMS to slightly or completely change colors partway through a screen, detect and act on collisions, and indicate that the CRT beam has reached a selected screen XY position. These events (except the CMR changing) may be individually selected to merely set a flag for the MPU, or to cause an MPU interrupt.

The bit-plane mode suits graphics applications and mixed text and graphics; characters can be drawn on the screen (using simple, quick routines) at random locations, with none of the list modes' row or spacing restrictions.

The six list modes offer a variety of attributes; modes Ø, 2, and 5 are intended for text and word processing applications using mode Ø's minimum 12 lines of 32 characters (NTSC noninterlace) up to mode 2's maximum of 5Ø lines of 8Ø characters (PAL interlace). Modes 1, 3, and 4 are intended for video games of varying complexity. All modes trade memory against performance, from mode Ø's minimum of 384 bytes per displayed screen to mode 4's maximum of 756Ø.

The RMS provides many game-related features. Fixed objects can flag or ignore true objects when they overlap, under user control. A fixed object, which always occupies a rectangular block, can be partly background color and partly "solid" object. The collision reporting to the MPU is made only when the true object overlaps the solid part of the fixed object. The true object can appear in front of the background color, but behind the fixed object's solid area. Since the arrangement of colors in the CMR registers is arbitrary, this puts no restriction on what colors are solid. Only the CMR address, not its contents, affects this.

Shading is a related feature. Part of a fixed object can represent an area in shadow; selecting the shading attribute allows a true object to change color as it passes through the shade. Once again, the shade can be any color, since only the CMR address determines what gets shaded.

The true objects are positioned relative to the displayed screen, while characters and fixed objects are positioned relative to the virtual screen; this allows fully independent movement. A true object can be held at the center of the screen while the rest of the scene is scrolled around it, and it seems to pass behind some parts and in front of others; alternately, the fixed objects could move over a stationary background. Each could move independently and simultaneously, or some could move while other true objects remained attached to the scenery. Four registers control visible screen movement and two control each true object, so very little MPU effort is required to make the changes.

1.4 System Hardware and System Performance

The video performance of the system is affected by the choice of MPU and the amount and organization of DRAM.

1.4.1 System Performance and the MPU

Any of the MPU's can get full video performance from the RMS, but non-video system processing throughput will be higher for the MC68ØØØ family than for the MC68Ø9E, so displays that require more processing may need higher MPU capability. The RMS must supply the MPU clocks so that it can synchronize DRAM access. This has a different effect on the MC68Ø9E's synchronous bus than on the MC68ØØØ family's asynchronous bus.

Depending on the user-selected horizontal screen resolution, the RMS allows access to its registers and its DRAM at a 745 KHz to 994 KHz rate (NTSC). When used with an MC68Ø9E, the RMS does this by providing the Q and E clocks at that frequency, which means that all MC68Ø9E bus transactions take place at that frequency even when they are not using the RMS. The MC68ØØØ family's bus requires handshaking, which the RMS provides at the same 745 KHz to 994 KHz, but it also provides a 7.95 MHz clock (NTSC) which allows the MPU to perform internal operations and non-RMS bus activity at a higher rate. For PAL systems, the frequencies are 739 KHz to 985 KHz and 7.88 MHz.

The user can supply a separate clock for MC68ØØØ family MPU's and leave the RMS-supplied 7.95 MHz clock disconnected, which would allow using a higher speed MPU and a clock to match. The MPU's internal operations would then run at the higher frequency, but the bus cycles would still be controlled by the speed of the memory or I/O handshaking. RMS handshaking would be unaffected by this clock change, but the system could use high speed memory or I/O with separate handshaking to take advantage of the faster clock.

1.4.2 System Performance and DRAM Use

The RMS uses Dynamic Random Access Memory (DRAM) organized in banks Each bank consists of identical DRAM's connected in a byte-wide data bus (data inputs tied to data outputs) with a multiplexed address bus of 14, 16, or 18 lines (for 16K, 64K, or 256 Kbyte banks).

The RMS supports up to four banks of DRAM, each of which can have up to 256 Kbytes. With 8-bit MPU's (MC68Ø9E and MC68ØØ8), one, two, or four banks may be used; with the 16-bit MPU's two or four banks are possible. The independent banks allow time division multiplexing of the DRAMs onto the RMS data bus as well as DRAM refresh and MPU access in the same cycle.

Because of this, 2 and 4-bank systems have more performance available than 1-bank systems; they can display more colors in bit-plane and list modes, and the list modes can have fewer scan lines in each character row. See Table 3-1. Figuring the size required for a display is complicated by the fact that the DRAM may be used to hold a variable number of user-defined display objects, as well as a virtual screen much larger than the displayed screen, or even several independent screens. 16 Kbytes is the smallest possible bank because of the refresh methods supported by the RMS. With one 16K bank, the user could display all the list modes (the densest noninterlace visible screen requires 378Ø bytes; each user-described image takes from 8 to 128 bytes in its image table) or in bit-plane mode a 4-color screen 32Ø pixels wide by 2ØØ pixels high.

Because the RMS is designed to provide complete control of DRAM with its data bus and control signals, it does not allow access by the RMS to any other memory type. If a particular application requires that a display or image tables be present at power up, the data can be stored in ROM and transferred to DRAM by the MPU before the RMS' video output is enabled.

1.5 Raster Memory Controller (RMC)

The RMC is designed in Motorola's HCMOS process. It allows the LSI RMC to pack a lot of function into a low cost part, and be capable of operating at the high speeds associated with video. Figure 1-1 shows the RMC's block diagram, and Figure 1-2 shows its functional pinout.

The RMC provides all of the display address generation for RMS. The address generator is essentially a dedicated MPU whose architecture and instructions have been optimized for display address generation calculations.

The RMC also processes the video data. Raw data is received from DRAM and decoded into pixels. The high performance of HCMOS allows the pixel data to be processed at rates greater than 14 MHz.

The video data can be broken down into bit-plane pixels, or routed to the one of the RMC's internal character generators. Character generators are available for either high quality alphanumerics or mosaic graphics.

Dedicated logic collects true object data during horizontal inactive video and then presents it at the correct pel time. The user gets to select that time by means of screen XY coordinates.

The Color Mapping RAM (CMR) allows the user to change colors quickly and easily. There is no need to have the MPU process all of the data in DRAM, when changing a single RMC register will change how the DRAM data is interpreted and generate an entirely new color or intensity.

The video outputs are analog RGB. Special interface parts are available for most users, but it is also simple to buffer the outputs with transistors.

1.6 Raster Memory Interface (RMI)

The RMI is an LSI bipolar digital part designed with Motorola's MOSAIC 1.5 process. This process allows it to be a complex part and still have the same speed as 74ALS logic. It needs this speed since it must interface with dynamic RAM and provide clocks for the entire system. Figure 1-3 shows the RMI's block diagram, and Figure 1-4 shows its functional pinout.

The RMI is the interface between the DRAM and the MPU's address and control lines. It translates the MPU's address bus into the information needed on the DRAM address bus. It also provides all of the timing signals required by the DRAM. The outputs of the RMI are designed to drive up to 32 DRAM parts directly; there is no need for additional buffering.

The RMI also determines how DRAM accesses are made so that both the MPU and RMC can access DRAM at high speed without interfering with each other. It makes extensive use of page mode to utilize the DRAM as efficiently as possible. See Section 3.3.

It must also serve as a memory management device. The MC68Ø9E MPU can only access 64 Kbytes of memory, but the RMS can support up to 1 Mbyte of DRAM. The RMI provides the control the MC68Ø9E needs to work with this much memory.

The RMI also provides address decoding for devices other than RMS. Several different memory map options are provided that allow for ROM and I/O as well as the RMS and DRAM. Signals are made available so that chip selects can be generated as simply as possible.

RMI also provides MPU handshaking signals for the MPU's. These signals are provided for both the RMS and for the other sections of the overall system for which RMI decodes addresses.

RMI contains an interface for a crystal so that it can be the master oscillator for the entire system. The master oscillator is approximately 36 MHz; see Section 4.1.1. RMI generates all of the timing signals needed by the RMC for video generation and also generates clocks for the MPU.

2.Ø THE MICROPROCESSOR

The Raster Memory System is designed to work with several members of the Motorola family of 8 and 16 bit MPU's. The user has a choice of the MC68Ø9E 8-bit MPU, the MC68ØØ8 8-bit MPU, or the MC68ØØØ 16-bit MPU. The user's choice of MPU has a significant affect on the architecture and performance of the total system.

This chapter examines the features that should be considered as the user chooses an MPU for the system.

2.1 The MC68Ø9E Microprocessor

The MC68Ø9E is Motorola's most sophisticated midrange microprocessor. It has a 16-bit address bus, an 8-bit data bus, and operates synchronously. The RMS contains several features so that the MC68Ø9E can work efficiently with the RMS as a peripheral.

2.1.1 The Address Bus

The MC68Ø9E's 16-bit address bus must be connected to the RMS X bus by means of two packages of 74ALS257 or equivalent logic. All of the control signals to operate the 74ALS parts are generated by the RMS. See Figure 2-1.

There are two lines from RMI used to control the 74ALS logic: the first is ADEN, which is used as an enable for any of the MPU address lines to be presented to the X bus; and the second is ADSEL, which is used to choose between either the most significant or the least significant address lines. ADEN is assumed to be active for the following table.

X Bus Bit	ADSEL Low	ADSEL High
X9	Not Used	Not Used
X8	Not Used	Not Used
X7	A7	A9
X6	A6	A8
X5	A5	A15
X4	A4	A14
X3	A3	A13
X2	A2	A12
X1	A1	A11
XØ	AØ	A1Ø

Table 2-1 MPU Address to X Bus Connections for the MC68Ø9E

These are all of the necessary address bus connections, but there are a few additional connections to the MPU address bits that can be useful. If the user wants to take advantage of the chip select decode capability of RMI, then MPU A7 should be connected to RMI pin UDS(A7), MPU A6 to RMI pin AS(A6), and MPU A5 to RMI pin LDS(A5). These connections are in addition to the connections listed above and are only required if the user plans to use the RMI-encoded chip select lines (S2, S1, and SØ). See Chapter 1Ø.

The MPU's 16 address lines are not enough to access the RMS' full address range. The RMS is designed to operate with up to 1 Mbyte of memory, which requires 2Ø address lines. The remaining four lines are supplied as a paging register in the RMS memory map. This paging register is easily programmed by the MPU and provides a simple form of memory management for the MC68Ø9E. A detailed discussion of the operation of the paging register is provided in Sections 9.3.6 and 9.3.7.

2.1.2 The Data Bus

The MC68Ø9E's 8-bit data bus is connected to the RMC 8-bit B port. The B port is a bidirectional port designed to interface directly to the MPU data bus with no additional logic required. It provides all of the data bus connections needed between the MC68Ø9E and the RMS. The RMC provides the buffering and separation needed between the MPU data bus and the dynamic RAM data bus, as well as the RMS control registers.

In addition to the data bus, the MPU's R/W line must be connected to the R/W lines of both the RMI and RMC.

2.1.3 Control and Timing

In addition to the address and data buses, there are several other connections between the MC68Ø9E and the RMS. These are control and timing signals.

The RMI generates the clocks needed by the MPU. The MPU's E clock is available on the RMI's CLK(E) pin. The MPU's Q clock is available on the RMI's DTACK(Q) pin. These clock signals must be used by the MPU and any other parts of the system that normally connect to the MPU clocks.

In standard MC68Ø9E configurations the RMS chip select (CS) is tied low, which permanently enables the RMS, but special applications can use CS to disable the RMS' bus control. The RMS generates the video display independent of the chip select's state.

The MPU's RESET line should also be tied to the RMS. This line is connected to the RMS via an additional transistor that connects it to the X bus. The way in which it is connected to the X bus informs the RMS, at system reset, what kind of MPU is in use in the system. See Chapter 5 for more information about resetting and initializing the system.

2.1.4 MPU Speed

The MC68Ø9E is a synchronous MPU, so the RMS maintains its clocks at a constant frequency from 745 KHz to 994 KHz (NTSC), depending on the display's horizontal resolution. During clock synchronization at the end of each scan line, one cycle is stretched. See Sections 4.2.7 and 4.3.1.

The MC68ØØ8 Microprocessor

The MC68ØØ8 is the lowest cost member of the Motorola MC68ØØØ MPU family. It is code-compatible with the MC68ØØØ, but uses an 8-bit data bus and a smaller address bus. This makes it possible to build a low cost system that can be upgraded at a later date and will still be able to use the existing software.

2.2.1 The Address Bus

The MC68ØØ8's 2Ø-bit address bus allows it to address 1 Mbyte of memory, which is the same range as the RMS. The RMS also provides chip selects for other devices in its memory range, but if they are used, these addresses subtract from the 1 Mbyte available for DRAM. The MC68ØØ8's 2Ø-bit address bus is connected to the RMS X bus via three packages of 74ALS logic, such as 74ALS257's. All of the control signals required to control the 74ALS parts are generated on the RMI. See Figure 2-2.

There are two signals used to control the 74ALS logic. The first is ADEN, which is used to enable the outputs of the 74ALS logic. The second is ADSEL, which is used to select which 1Ø-bit address word will be presented to the X bus. The following chart shows how to arrange the connections through the 74ALS logic.

X Bus Bit	ADSEL Low	ADSEL High
X9	A9	A19
X8	A8	A18
X7	A7	A17
X6	A6	A16
X5	A5	A15
X4	A4	A14
X3	A3	A13
X2	A2	A12
X1	A1	A11
XØ	AØ	A1Ø

Table 2-2 MPU Address to X Bus Connections for the MC68ØØ8

These are all of the connections required from the MPU address bus. All of the MPU address lines have been accounted for, so it is recommended to tie the chip select (CS) pin on RMI low so that it is permanently enabled.

2.2.2 The Data Bus

The MC68ØØ8's 8-bit data bus must be connected to the RMC's 8-bit B port. The B port is bidirectional and designed to interface directly to the MPU data bus with no additional logic. The RMS provides all of the buffering and separation required between the MPU data bus and the DRAM data bus.

Most of the control and all of the timing information required to operate the B port correctly is generated inside the RMS. The one additional line required from the MPU is its R/W line, which must be connected to the R/W lines on both the RMI and RMC.

2.2.3 Control and Timing

Several control and timing signals must be connected between the RMS and the MPU in addition to the address and data buses.

There are several timing and handshaking lines located on RMI. These include CLK(E), AS(A6), LDS(A5), and DTACK(Q). These pins should be connected to the MPU pins with similar names. Note that the RMS provides the MPU with its master clock and is also able to operate on the MPU's asynchronous data bus.

It is possible to use the RMS chip select (CS), located on RMI. This is not necessary since the RMS occupies the same address space that the MC68ØØ8 can address. However, it is possible to use it. The level on CS has no effect on the video display, which is maintained at all times.

The MPU's RESET line is also connected to the RMS. This is done via an external transistor. The way in which the transistor is connected to RMS provides important initialization information; see Chapter 5 for more information on reset and initialization.

2.2.4 MPU Speed

The RMS provides a 7.95 MHz clock for the MC68ØØ8, which is an asynchronous MPU. The MC68ØØ8 uses at least 4 clock cycles to perform each bus transaction. If the device being accessed is unable to respond quickly enough, the MPU inserts wait states in the bus cycle until the device responds. With a 7.95 MHz clock, the MC68ØØ8 has a maximum bus frequency of 1.988 MHz. If the need exists, a high speed MPU can be used with a user-supplied high frequency clock. High speed memory and I/O with their own address decoding and handshaking can then be used to take advantage of this speed.

When the MPU is accessing the RMS, the DRAM controlled by RMS, or some device whose MPU handshaking is performed by the RMS (using the S bus and RMS' DTACK), the average bus frequency is between 745 KHz and 994 KHz, depending on the display mode in use. One cycle may be stretched for resynchronization during horizontal retrace.

2.3 The MC68ØØØ Microprocessor

The MC68ØØØ's 16-bit data bus offers the user a very high level of performance. It uses an asynchronous memory bus, which allows it to operate with a wide variety of peripherals, regardless of their speed of operation.

The RMS requires slightly more external logic to interface to the MC68ØØØ than it does to interface to the other MPU's, but a very cost-effective system is still possible. See Figure 2-3.

2.3.1 The Address Bus

The MC68ØØØ is capable of directly addressing 16 Mbytes of memory. This is accomplished through a combination of 23 address lines (A1 to A23) and two data strobes (LDS and UDS). There is no AØ line, because the MC68ØØØ's address bus is set up to address 16-bit words. The two data strobes resolve to the byte level.

The interface between the MC68ØØØ address bus and the RMS X bus can be simply made via three packages of 74ALS257's, although it is possible to use other parts if desired. The RMS generates ADEN and ADSEL, the two signals required to control the 74ALS parts. ADEN is a general enable signal for the 74ALS outputs. ADSEL is a select signal to determine which address lines will be used. The 74ALS parts should be wired according to the following table.

X Bus Bit	ADSEL Low	ADSEL High
X9	A9	A19
X8	A8	A18
X7	A7	A17
X6	A6	A16
X5	A5	A15
X4	A4	A14
X3	A3	A13
X2	A2	A12
X1	A1	A11
XØ	UDS	A1Ø

Table 2-3 MPU Address to X Bus Connections for the MC68ØØØ

In addition to these connections, the RMS chip select is needed in MC68ØØØ systems. The MC68ØØØ can address 16 Mbytes of memory, but the RMS only occupies 1 Mbyte of its address space. Therefore the four most significant address lines of the MPU should be decoded and connected to the RMS chip select located on the RMI. This informs the RMS which 1 Mbyte block, out of a possible 16 blocks, it occupies. The display is maintained by the RMS regardless of the state of its CS pin.

2.3.2 The Data Bus

The MC68ØØØ has a 16-bit data bus, so it is necessary to have a 16-bit wide DRAM organization in order to have proper system operation. The RMS requires some additional 74ALS logic to control the connection between the MPU and DRAM data buses.

The signals to control the 74ALS logic are developed from the control signals already available in the RMS system, processed by 3 two-input NAND gates. Figure 2-4 shows the bus control hardware but not the other system connections. The DRAM data bus connects to the RMC's A and B busses, and the rest of the data bus connections are made directly to the MPU's data bus.

The DRAM is still organized in byte-wide banks. Banks Ø and 1 are used together to address 16-bit words, and separately to address bytes. Banks 2 and 3 are used in a similar fashion. Banks 1 and 3 are the least significant bits of the 16-bit data words. They must be connected to the A port. See Figure 2-5.

The RMS control registers must be read and written as bytes. The results of accessing the control registers as 16-bit words are undefined. The DRAM may be accessed as bytes, 16-bit words, or 32-bit long words.

The read/write (R/W) line must be connected to both RMI and RMC as well as the 74ALS logic used to interface the MPU data bus to the DRAM and RMC data buses.

2.3.3 Control and Timing

The MC68ØØØ requires several timing and handshaking signals in order to interface with the RMS. All of these signals are directly supplied by RMS.

First, the MC68ØØØ requires a clock (CLK). RMI supplies a 7.95 MHz clock.

There are also some handshaking signals required to control memory accesses. These include address strobe (AS), upper and lower data strobes (LDS and UDS), and data transfer acknowledge (DTACK). These signals should be attached to the RMI pins with similar names. In addition, UDS must also be attached to the 74ALS logic required to interface the MPU address bus to the X bus.

The MPU reset line should be connected to the RMS X bus via an external transistor. The way in which it is connected to the X bus will tell RMS, at system reset, that it is connected to an MC68ØØØ family MPU. See Chapter 5.

2.3.4 MPU Speed

The RMS provides a 7.95 MHz clock for the MC68ØØØ, which is an asynchronous MPU. The MC68ØØØ, like the MC68ØØ8, uses at least 4 clock cycles to perform each bus transaction. If the device being accessed is unable to respond quickly enough, the MPU inserts wait states in the bus cycle until the device responds. With a 7.95 MHz clock, the MC68ØØØ has a maximum bus frequency of 1.988 MHz. If the need exists, a high speed MPU can be used with a user-supplied high frequency clock. High speed memory and I/O with their own address decoding and handshaking can then be used to take advantage of this speed.

When the MPU is accessing the RMS, the DRAM controlled by the RMS, or some device whose MPU handshaking is performed by the RMS (using the S bus and RMS' DTACK), the average bus frequency is between 745 and 994 KHz, depending on the display mode in use. One cycle may be stretched for clock synchronization during horizontal retrace.

3.Ø MEMORY ORGANIZATION

This chapter describes the dynamic memory that must be connected to the Raster Memory System. It lists the memory parts that can be used with the RMS and it also describes the different memory configurations that can be attached to the RMS, and their features.

3.1 Uses of the Memory

The dynamic memory attached to the RMS is a multiple-use resource. It is used for the RMS screen memory and it is available for use by the MPU for other tasks.

The amount and organization of the DRAM used for screen memory affects the performance of the system. For example, it is not possible to display a full screen of high resolution 4 color graphics without sufficient DRAM in the system.

The DRAM is used to hold all of the data required to generate a video screen. In bit-plane mode all of the data for every pel displayed is stored in DRAM. In list mode all of the data in the display list is stored in DRAM. In addition, all of the image tables required to define DRCs, fixed objects, and true objects are stored in DRAM.

The parts of DRAM in use as screen memory are accessible to the MPU. The MPU may read or write these parts of memory at any time without causing any flickering or other undesirable effects on the video screen.

The MPU can use the parts of DRAM not in use as screen memory for data or programs that it is executing. These uses do not interfere with the video display operation in any way. They do not cause undesirable effects on the video screen.

It is also possible to use the DRAM as a source or destination for DMA operations. The restriction on DMA use is that data may not be transferred any faster than the MPU memory access rate even though the speed rating of the DRAM's might lead the user to believe that a faster rate is possible.

This memory access rate is no less than 745 Kbytes per second and no greater than 994 Kbytes per second in an NTSC system. The rate depends on the display mode in use.

It is possible for the MPU to use the memory at the same time RMS is using it for display because access to the memory is time division multiplexed. When the MPU requests a byte of memory, it is accessed within a few hundred nanoseconds. In the case of a read the data is held until the MPU expects to receive it. In the meantime the RMS display process is accessing the memory to retrieve up to four bytes that will be used to generate the video display.

It might appear that the RMS has a very large share of the memory access time, while the MPU has to settle for a small percentage. In practice this is not true. The display process makes extensive use of page mode accesses to DRAM, which allow very high speed memory throughput, as long as the required data is stored in the required pattern of addresses. During each memory cycle, the MPU and RMS both get exactly one random access to DRAM.

3.2 Recommended Memory Parts

The RMS supports a wide variety of dynamic RAM types. Regardless of their organization, all memory parts used with the RMS should have an access time of no more than 150 nanoseconds from Row Address Strobe. Faster parts may be used without any problems.

Memory parts which are known to work are listed below. These parts are all industry standards, so it should be possible to substitute similar parts. However, only these parts have been tested, so only these parts are guaranteed to work.

Any users who would like to use other parts are directed to Section 3.3 which describes the signals generated by RMS and their timing.

Part Number	Organization
MCM4516-15	16Kx1
MCM6665-15	64Kx1
TMS4416-15	16Kx4
MCM6256-15	256Kx1

Table 3-1 RMS-Compatible Dynamic RAM Types

These parts are supported and others may not be because of two parameters. The first is memory timing, which is covered in Section 3.3. The second is the way in which signals are routed to the RMI Z bus.

The RMI's Z bus is connected directly to the DRAM address inputs. Only a limited number of configurations are available. The configuration used will depend on the organization and type of DRAM the user selects. The user informs the RMS of this choice by means of a control register in the RMS memory map, whose details are discussed in Section 9.3.17. In all cases, Z0 (the LSB) is connected to the DRAM's LS address pin (A0), Z1 to A1, and so forth. If less than 1 Mbyte of DRAM is used, the most significant Z bus pins must be unterminated.

3.3 Memory Timing

The following data is listed in the same way as it would be listed on a Motorola memory device data sheet. Timing diagrams are also shown (Figures 3-1, 3-2, and 4-6) in case the definition of some terms is in doubt.

Parameter	Symbol	Nominal Time
Random read or write cycle time	t_{RC}	335
Read-modify-write cycle time	t_{RWC}	Not Applicable
Access time from row address strobe	t_{RAC}	168
Access time from column address strobe	t_{CAC}	84
Output buffer and turn off delay	t_{OFF}	$<t_{CAC}$
Row address strobe precharge time	t_{RP}	140
Row address strobe pulse width	t_{RAS}	196
Column address strobe pulse width	t_{CAS}	112
Row to column strobe lead time	t_{RCD}	84
Row address setup time	t_{ASR}	>0
Row address hold time	t_{RAH}	28
Column address setup time	t_{ASC}	56
Column address hold time	t_{CAH}	56
Column address hold time referenced to RAS	t_{AR}	140
Transition time (rise and fall)	t_{T}	>3
Read command setup time	t_{RCS}	196
Read command hold time	t_{RCH}	168
Read command hold time referenced to RAS	t_{RRH}	168
Write command hold time	t_{WCH}	140
Write command hold time referenced to RAS	t_{WCR}	223
Write command pulse width	t_{WP}	196
Write command to row strobe lead time	t_{RWL}	168
Write command to column strobe lead time	t_{CWL}	168
Data in setup time	t_{DS}	17
Data in hold time	t_{DH}	140
Data in hold time referenced to RAS	t_{DHR}	223
Column to row strobe precharge time	t_{CRP}	140
RAS hold time	t_{RSH}	112
Refresh period	t_{RFSH}	1.64
Write command setup time	t_{WCS}	56
CAS to WRITE delay	t_{CWD}	Not Applicable
RAS to WRITE delay	t_{RWD}	Not Applicable
CAS hold time	t_{CSH}	196
CAS precharge, non page mode	t_{CPN}	140
RMW cycle RAS pulse width	t_{RRW}	Not Applicable
RMW cycle CAS pulse width	t_{CRW}	Not Applicable
Page mode cycle time	t_{PC}	223
Page mode cycle time (read-modify-write)	t_{PCM}	Not Applicable
CAS precharge time (page mode cycle only)	t_{CP}	112
RAS pulse width (page mode cycle only)	t_{RPM}	531

Table 3-2 DRAM Timing Requirements

All units are in nanoseconds except tRFSH, which is in milliseconds. The RMS provides 5 refresh cycles per video line, for a worst-case 128-cycle time of 1.64 milliseconds (against a specification of 2.0 msec maximum) and a 256-cycle time of 3.28 milliseconds (4.0 msec maximum). Some parameters are not applicable because read-modify-write cycles are not possible in the RMS system.

3.4 Memory Organization

The overall organization of the DRAM attached to RMS must fall into one of two general classifications: 8 or 16 bits wide. The 8-bit organization is used with the MC68Ø9E and MC68ØØ8 MPU's, and the 16-bit organization is used with the MC68ØØØ. MPU type is the only parameter involved in making this choice. Either organization can be built from any of the recommended memory parts listed in Section 3.2.

After the width of the memory organization has been chosen, one more choice remains: how many banks of memory will be used. Banks are equal-length, byte-wide blocks of memory that share a data bus. Their use of the data bus is time-division multiplexed. The user may elect to use 1, 2, or 4 banks of memory with an 8-bit MPU, or 2 or 4 banks with a 16-bit MPU. Only 2 and 4 are possible in the 16-bit systems, because banks are 8 bits wide. This is done to support the byte-oriented instructions in the MC68ØØØ. The RMS is informed of the user's choices for number of banks by means of an MPU-addressable control register. Detailed information about that register is in Section 9.3.17. All of the banks must be the same size and built from the same device type.

There are two benefits from the use of multiple banks. The first is that it increases the total amount of memory the RMS can support. A single bank system can support a maximum of 256 Kbytes of DRAM, while a 4 bank system allows RMS to support up to 1 Mbyte of DRAM.

The second benefit from multiple bank systems is increased video performance. The level of video performance is heavily influenced by memory throughput. The time-division multiplexed data bus of the multiple bank allows access to two banks in one memory cycle, giving twice the single-bank memory throughput, with a corresponding increase in video performance.

Multiple-bank systems can display more colors in both bit-plane and list modes, and they can have fewer scan lines per character row in the list modes (and therefore more character rows displayed at one time). The performance limits are shown in Table 3-3.

Both 2 and 4 bank systems offer all of the video performance possible with the RMS. The single bank system allows lower cost, but it restricts performance.

Horizontal Resolution Number	Horizontal Resolution In Pixels	Number of Memory Banks	Maximum Bits Per Pixel	List Mode	Minimum Number of Scan Lines Per Character Row	
					(Note 1)	(Note 2)
0	64	1	2	0	8	8
1	128			1	8	10
2	256			4	8	16
		2, 4	4	0, 1, 4	8	8
3	256	1	2	0	8	10
				1	10	--
				4	16	--
		2, 4	4	0	8	8
				1	8	10
				4	8	16
4	320	1	2	0	8	8
				1	8	12
				4	8	16
		2, 4	4	0, 1, 4	8	8
6 7	512 640	1	1	2	10	--
				3, 5	16	--
		2, 4	2	2	8	10
				3, 5	8	16

Note 1: Noninterlace sync and data, interlace sync with noninterlace data
Note 2: Interlace sync and data

Table 3-3 Effect of Memory Banks and Horizontal Resolution

4.Ø HARDWARE INTERFACE

This section deals with the use and function of the pins of RMI and RMC. The pinouts of the two parts are shown in Figures 4-1 and 4-2. Because of the limitations of the word processor, signal names in the text do not show the polarity of signals. The pinout diagrams show the correct polarity of the signals using the bar convention to indicate active-low signals.

4.1 Clocks

All of the clocks required in the RMS are generated from one master oscillator. This includes several clocks that are supplied by RMI for the use of RMC, as well as color sub-carrier and MPU clock(s).

4.1.1 Master Oscillator

The master oscillator for the Raster Memory System is located on RMI. The external circuit required is shown in Figure 4-3.

For NTSC applications with a 3.579545 MHz color sub-carrier, the master oscillator frequency is ten times the sub-carrier, or 35.79545 MHz. For applications involving PAL television with a 4.43361875 MHz subcarrier, the master oscillator is eight times color sub-carrier, or 35.46895 MHz.

The master oscillator is used to generate all of the internal clocks required by the RMS. It is also the base for deriving the color sub-carrier and the MPU clock(s).

4.1.2 VTCLK (Video Timing Clock)

VTCLK is generated by RMI and supplied to RMC, which uses it to generate horizontal and vertical sync pulses and blanking, as well as for internal timing.

VTCLK is always equal to the master oscillator divided by 5. When the master oscillator is 35.79545 MHz, VTCLK is 7.159Ø9 MHz. When the master oscillator is 35.468944 MHz, VTCLK is 7.Ø9379 MHz.

VTCLK is a free-running clock with a 5Ø% duty cycle. Unlike other clocks, it is never resynchronized.

4.1.3 PCLK (Picture Element Clock)

PCLK clocks each picture element's video data out of the RMS' video outputs. It is generated by RMI and used by RMC. Depending on the user-selected horizontal resolution, PCLK runs between 2.5 and 6 times slower than the master oscillator. Vertical resolution has no effect on PCLK. During one cycle in the horizontal retrace it is stretched to allow resynchronization with HSYNC. It is stretched less than one memory cycle (see Section 4.1.4). For more information on resynchronization, see Sections 4.1.4 and 4.1.5.

When the user selects the horizontal display resolution (by setting the HRES mode in an RMS register - see Section 9.3.18), the RMS sets the PCLK speed to match.

The following table summarizes the PCLK's available and how they are used. The table assumes a master oscillator of 35.79545 MHz. If the master oscillator is actually 35.46895 MHz, the PCLK's will be less than 1% slower than those listed. Memory cycles are defined in the next Section.

HRES Mode	Horizontal Resolution in Pixels	PCLK Ratio to Master Osc.	PCLK Frequency in MHz	Pixel Duration in Nano-seconds	Memory Cycle Duration in Micro-seconds
7	64Ø	2.5	14.32	69.8	1.117
6	512	3	11.93	83.8	1.341
4	32Ø	4.5	7.95	125.7	1.ØØ6
2,1,Ø	256 (Narrow),128,64	5	7.16	139.7	1.117
3	256 (Wide)	6	5.97	167.6	1.341

Table 4-1 Picture Element Clock

4.1.4 MTCLK (Memory Timing Clock)

MTCLK is used to keep track of memory cycles. It is generated by RMI and used by RMC. A memory cycle is 1.ØØ6 to 1.341 microseconds in duration (except during resynchronization, Section 4.1.5) and provides one memory access opportunity each to the MPU and RMC. A memory cycle is 16 PCLK cycles long when the horizontal resolution is 512 or 64Ø pixels and 8 PCLK cycles long for all the lower horizontal resolutions. Each memory cycle is made up of 9 MTCLK cycles. The first 8 MTCLK cycles are MASTER OSC divided by 4 with a 5Ø% duty cycle, and the ninth is stretched, if necessary, to match the length of a memory cycle as defined by PCLK.

MTCLK is also stretched near the trailing edge of horizontal sync in order to resynchronize the memory cycle to HSYNC on a video line basis.

Table 4-1 and Figure 4-4 show MTCLK relationships.

4.1.5 CLK(E) (MPU Clock or E Clock)

CLK(E) is an RMI output used to provide the MPU with a basic clock. The type of clock provided depends on whether an MC68Ø9E or MC68ØØØ family MPU is in use.

For MC68ØØØ family MPU's, CLK(E) is the master oscillator frequency divided by 4.5, which is 7.95 MHz for NTSC and 7.88 MHz for PAL. It is a free-running clock.

For the MC68Ø9E, CLK(E) is used as the MPU's E clock, and it runs at the memory frequency. E's duty cycle depends on the horizontal resolution; in addition, it is stretched at the end of each horizontal line the same way that PCLK or MTCLK is, to resynchronize it to HSYNC. When the MPU is used with a single bank of DRAM, eight of the E cycles in each video line are stretched to allow true object data gathering.

The MC68Ø9E also requires a Q clock. This signal comes from RMI's DTACK(Q) output, which is described in Section 4.2.5.

4.1.6 CSC (Color Subcarrier)

CSC is an RMI output made available for use by other parts of the system, such as a video modulator. CSC is equal to the master oscillator divided by 8 (PAL) or 1Ø (NTSC). This makes it easy to derive the common color subcarriers of 3.579545 MHz (NTSC) or 4.43361875 MHz (PAL).

The method of selecting either a divide by 8 or a divide by 1Ø color sub-carrier is defined in Section 5.2, which describes the strap reader used during initialization.

4.2 Handshaking

The handshaking signals, primarily generated by RMI, ensure that data gets passed between the different parts of the RMS, or between the RMS and the MPU.

4.2.1 CS (Chip Select)

The CS pin on RMI must be low to enable the RMS. Systems that take advantage of the RMS' address range and device-select bus to perform all addressing may ground this pin to permanently enable the RMS. This is especially helpful for MC68Ø9E and MC68ØØ8-based systems, whose address range is less than or equal to that of the RMS.

MC68ØØØ-based systems, or systems not using the RMS' full 1 Mbyte addressing range, must decode the most significant address lines for CS. The simplest case is decoding the most significant four bits to select a single 1 Mbyte block from the MC68ØØØ's 16 Mbyte address space. The RMS generates its video output no matter what signal is applied to CS.

When CS is disabled the RMS' data bus is disabled and the S bus (Section 4.4) outputs all 1's. In MC68ØØØ family systems, RMS' DTACK (Section 4.2.5) remains high.

The RMS does not provide MC68ØØØ-peripheral interrupt handshaking, so if interrupts are to be used with an MC68ØØØ family MPU, hardware that detects a function code value of 7 (interrupt acknowledge) must be used to disable the RMS chip select. RMS-generated interrupts must use an Autovector interrupt.

4.2.2 AS(A6) (Address Strobe)

AS(A6) is an RMI input. If an MC68ØØØ family MPU is used, its Address Strobe output must connect to this pin.

When the MC68Ø9E is used, AS(A6) is an extra performance option. If the user plans to make use of the three-bit device select bus (SØ, Sl, S2), then MPU address bit 6 must be connected to AS(A6). If the user does not plan to use the device select bus, it is not necessary to make a connection to AS(A6).

Connecting address bit 6 to AS(A6) allows the RMS to detect its state in time to generate the device select output; the A6 input via the X bus arrives too late to do this.

4.2.3 UDS(A7) (Upper Data Strobe)

The UDS(A7) line is similar to the AS(A6) line, in that it is optional for use with the MC68Ø9E. If the user plans to make use of the RMI device select bus (SØ, Sl, S2), then UDS(A7) must be connected to address bit 7 of the MPU. If the user does not plan to use the device select bus, then it is not necessary to connect address bit 7 to UDS(A7).

If the system is using the MC68ØØ8 MPU, then there is no connection to the UDS(A7) pin.

If the system is using the MC68ØØØ MPU, then UDS(A7) must be connected to the UDS output of the MPU.

4.2.4 LDS(A5) (Lower Data Strobe)

The LDS(A5) line is similar to the AS(A6) and UDS(A7) lines, in that it is optional with the MC68Ø9E MPU. If the user plans to use the RMI's device select bus (SØ, Sl, S2), then LDS(A5) must be connected to the MPU's address bit 5. If the user does not plan to use the device select bus, then this connection is not required.

If the system is using the MC68ØØ8 MPU, then LDS(A5) must be connected to the MPU's data stobe (DS) line. This connection is required in all MC68ØØ8 applications.

If the system is driven by the MC68ØØØ MPU, then LDS(A5) must be connected to the MPU's lower data strobe (LDS) pin. This connection is required in all MC68ØØØ applications. This pin is used by the MC68ØØØ, in conjunction with UDS(A7), to define whether the MPU wishes to access an entire 16-bit word in memory, or just its upper or lower byte.

4.2.5 DTACK(Q) (Data Transfer Acknowledge)

DTACK(Q) is an RMI output pin. It is used as a clock output in MC68Ø9E systems, and as a handshaking line in MC68ØØØ family systems.

In MC68Ø9E systems DTACK(Q) is the output for the MPU's Q clock. The relationship between the MC68Ø9E's E and Q clocks is well defined in the MC68Ø9E data sheet, and the signals generated by RMI conform to that specification. The frequency of the Q clock is the same as for the E clock. Clock stretching is performed on the Q clock in the same way, and under the same circumstances, as it is performed on the E clock (Section 4.1.5).

In MC68ØØØ family systems, DTACK(Q) should be connected to the MPU's DTACK pin. DTACK is used to inform the MPU that the external device it is accessing has completed the requested task. The RMS DTACK(Q) line responds when the access is to DRAM, an RMS control register, or possibly another device; the RMS can perform handshaking for some other blocks. Chip Select must be low for DTACK to be active. See Chapter 1Ø.

4.2.6 R/W (Read/Write)

R/W pins are located on both RMI and RMC. They should both be connected to the MPU's R/W pin. There is no difference in the connections for different MPU types.

R/W is used to control the direction of data flow for MPU accesses to either DRAM or the RMS control registers.

4.2.7 HSYNC (Horizontal Sync)

HSYNC is used between RMI and RMC to maintain synchronization between the various clocks in the system. HSYNC will occur during each horizontal video line. HSYNC's active time matches display horizontal sync and may be used for synchronization of external hardware. The trailing (rising) edge of HSYNC is the event that resynchronizes the various clocks in the system. VTCLK is the master clock, since all others are resynchronized to it. HSYNC's trailing edge occurs one VTCLK period before the end of the horizontal line. RMI detects this event and waits until the next rising edge of VTCLK to restart all of the other clocks. These other clocks include PCLK, MTCLK and (if the MPU is an MC68Ø9E) E and Q, as well. Therefore, the first rising edge of VTCLK after the rising edge of HSYNC is the start of a pixel and the start of a memory cycle.

HSYNC's period and duty cycle conform to the specifications for video horizontal sync, except for the one-VTCLK skew (approximately 14Ø nanoseconds). Most applications that require separate horizontal and vertical sync may use vertical sync from the RMC SYNC pin and use HSYNC for horizontal sync.

The amount of clock-stretching on PCLK, MTCLK, and the MPU clock(s) depends on the display mode in use. It is always less than one memory cycle. Clock stretching is begun by the LSTCYC bit of X bus control word 1 (see Section 4.3.1).

4.2.8 DBEN (Data Bus Enable)

DBEN is generated by RMI and used by RMC. It is used with R/W to determine when, and in which direction, RMC should enable the data bus to the MPU. It is also used during an MPU read to latch data from DRAM or RMC onto the MPU data bus until the MPU is ready for it.

This allows the RMS to read or write data to DRAM or an RMS control register with the correct timing, so that there is no conflict with other devices on the MPU data bus.

In the case of an MC68ØØØ MPU, with its 16-bit data bus, DBEN must be connected to the external 74ALS logic that is used to connect the RMS and DRAM data bus to the MPU data bus and RMC. See Section 2.3.2.

4.3 The X Bus and Its Control

The X bus is used as the major means of passing information from RMC to RMI. It is also used to bring MPU addresses into the RMS system, and to generate reset in RMC and RMI.

4.3.1 The X Bus

The X bus is a 1Ø-bit bus that connects to RMI, RMC, and to some 74ALS family logic that allows it to connect to the MPU's address bus.

The X bus is time-division multiplexed. Eight different words are passed on the X bus during each memory cycle. The time-division multiplexing is controlled by MTCLK, ADEN, and ADSEL. Depending on the horizontal resolution in use, there may be a pause after the eight words are passed, and before the next memory cycle starts. See Figures 4-5 and 4-6.

There are nine cycles of MTCLK during a memory cycle. They are named MTØ to MT8. The only way to tell one cycle of MTCLK from another is to count MTCLK cycles beginning immediately after the trailing edge of HSYNC. A memory cycle lasts from the beginning of MTØ to the end of MT8. The 9 possible phase relationships between the MPU clock and the RMS memory cycle are shown as case 1 through case 9 in Figure 4-6.

The data that is passed on the X bus begins during the previous memory cycle's MT7, so that it will be available at the beginning of the memory cycle (MTØ).

The first word passed on the X bus contains the most significant bits of the MPU address. The MPU address comes from the MPU, via 74ALS logic, and is read by both RMI`and RMC. It is used in conjunction with the MPU least significant bits to determine where the MPU is accessing.

During the last half of MT7 and the first half of MT8 the RMI holds ADEN low and ADSEL high so the 74ALS address multiplexers will put the MPU's most significant address bits onto the X bus. During MTØ the RMI keeps ADEN low and pulls ADSEL low so the 74ALS multiplexers will put the MPU's least significant address bits on the X bus. The MPU being used affects which address lines are presented; see Chapter 2. These two sets of address bits are the first two words passed on the X bus in each memory cycle.

The MPU's MS and LS address bits are used by both RMI and RMC to determine where in the RMS system the MPU is accessing. If the access is to DRAM, the RMI is primarily responsible for the access. If it is to a control register, then RMC is primarily responsible for the access.

It may also be that the access is not to the RMS system. This type of access can fall into two categories. The first is that the access has nothing to do with the RMS at all. In this case the RMS' chip select (CS) should be false, so that ADSEL never changes. In this situation the MPU address information never comes onto the X bus.

In the second situation the user is using the RMS' device select capabilities. This logic is located in RMI, and RMI has enough information after seeing the MPU MS address word to generate the proper information on the S bus. RMC's only involvement is that it decodes the address and recognizes that it does not have to take any action.

Most of the rest of the X bus cycles are concerned with communication from RMC to RMI.

The third word is passed during MT1 and contains the MSB's of the display address. The display address is the location in screen memory to access next in order to get video data to put on the screen.

The fourth word is passed during MT2 and contains the LSB's of the display address.

MT3, MT4, MT5 and MT6 are used to pass control words 1, 2, 3, and 4, respectively. The control words contain a wide variety of information that RMI must receive from RMC. A bit-by-bit listing follows. All bits are active high.

Bit #	Name	Function
Ø	HRESØ	Horizontal resolution bit Ø
1	HRES1	Horizontal resolution bit 1
2	HRES2	Horizontal resolution bit 2
3	LSTCYC	Last memory cycle before the end of the line
4		Reserved
5		Reserved
6	MTYPØ	Memory type bit Ø
7	MTYP1	Memory type bit 1
8	MTYP2	Memory type bit 2
9	MTYP3	Memory type bit 3

Table 4-2 X Bus Control Word 1

HRESØ, 1 and 2 are located in an RMS control register. They select the horizontal screen resolutiuon and must be passed to RMI so that the proper pel clock is generated. See Section 9.3.18.

LSTCYC affects clock resynchronization. It is a signal to RMI that the current memory cycle is the last one on this horizontal line. After it is complete, RMI stretches clocks until one VTCLK time after the trailing edge of HSYNC.

The MTYP bits inform RMI what type of dynamic RAM in use, and how it is organized in the system. An explanation of how the bits are coded follows.

MTYP3	MTYP2	MTYP1	MTYPØ	Meaning
Ø	Ø	Ø	Ø	16Kx1 DRAM's, 8 bits wide
Ø	Ø	Ø	1	16Kx4 DRAM's, 8 bits wide
Ø	Ø	1	Ø	64Kx1 DRAM's, 8 bits wide
Ø	Ø	1	1	Reserved
Ø	1	Ø	Ø	256Kx1 DRAM's, 8 bits wide
Ø	1	Ø	1	Reserved
Ø	1	1	Ø	Reserved
Ø	1	1	1	Reserved
1	Ø	Ø	Ø	16Kx1 DRAM's, 16 bits wide
1	Ø	Ø	1	16Kx4 DRAM's, 16 bits wide
1	Ø	1	Ø	64Kx1 DRAM's, 16 bits wide
1	Ø	1	1	Reserved
1	1	Ø	Ø	256Kx1 DRAM's, 16 bits wide
1	1	Ø	1	Reserved
1	1	1	Ø	Reserved
1	1	1	1	Reserved

Table 4-3 Memory Type Bits

Bit #	Name	Function
0		Reserved
1		Reserved
2		Reserved
3		Reserved
4		Reserved
5	SWAP	MC6809E paging bit
6	PG0	MC6809E paging bit
7	PG1	MC6809E paging bit
8	PG2	MC6809E paging bit
9	PG3	MC6809E paging bit

Table 4-4 X Bus Control Word 2

The RMS is based on a 20-bit address bus, which allows it to operate with up to 1 Mbyte of DRAM. The MC6809E only has a 16-bit address bus. The remaining bits the MC6809E needs to completely utilize RMS are available as a paging register in the RMS memory map. These four bits are passed from RMC, where the register is located, to RMI via the X bus. The SWAP bit is also a paging bit. It is used to exchange the two halves of the 64 Kbyte DRAM section selected by the other four bits. It is the same as SWAP in the paging register, while the other bits in this control word are the same as PG0 to PG3. See Section 9.3.6. The MC68000 family MPU's do not use SWAP and PG0-PG3, but these values are passed here for those processors, as well.

Bit #	Name	FUNCTION
0	LPI0/VEC	Lower page independent block bit 0
1	LPI1	Lower page independent block bit 1
2	LPI2	Lower page independent block bit 2
3	LPI3	Lower page independent block bit 3
4	UPI0	Upper page independent block bit 0
5	UPI1	Upper page independent block bit 1
6	UPI2	Upper page independent block bit 2
7	UPI3	Upper page independent block bit 3
8	DB0	Number of memory banks bit 0
9	DB1	Number of memory banks bit 1

Table 4-5 X Bus Control Word 3

The upper and lower page independent blocks are for the use of the MC6809E MPU. They provide a method for the MC6809E to select different small sections of the total 1 Mbyte of memory in the RMS, and have them decoded so that they appear to be part of the MC6809E's 64 Kbyte address range, regardless of their physical address.

The page independent blocks are selected by the user through the RMS control registers. For more information see Section 9.3.7.

The same bits are passed for MC68000 family MPU's, but only LPI0/VEC has any significance, as discussed in Section 9.3.7.

The number of memory banks is taken from another register, as discussed in Section 9.3.1.7.

Bit #	Name	Function
Ø		Used only at reset
1		Used only at reset
2		Used only at reset
3		Used only at reset
4	MODEØ	Memory cycle type bit Ø
5	MODE1	Memory cycle type bit 1
6	MODE2	Memory cycle type bit 2
7	UF	Unfolded control register map
8	MAPA	Memory map select
9	MACH2	Machine 2 mode

Table 4-6 X Bus Control Word 4

In control word four only six bits are defined in the normal fashion. The name of bit 9 is MACH2. It refers to the fact that RMS can appear to have two completely different control register maps.

Machine 2 mode is in effect when MACH2 is true. This memory map is backwards compatible with the MC6883 and MC6847 combination. See Chapter 14.

MAPA is used with the RMS external device select feature. It is an option that selects between two different memory maps. For more information see Chapter 1Ø.

UF is true if the RMS control registers are set to the unfolded map option, rather than the folded map option. See Section 9.3.1.

The mode bits define what operations will take place in the next memory cycle. The coding of those bits is shown below. The type of cycle used depends on where RMS is in the current video line and what display mode is in use. The user has no direct control over these bits. They are generated automatically by RMS and are dependent on a variety of choices the user has already made regarding display mode.

Mode2	Model	ModeØ	Cycle Type
Ø	Ø	Ø	MPU & single CAS display access
Ø	Ø	1	MPU & double CAS display access
Ø	1	Ø	MPU & four CAS display access
Ø	1	1	MPU & two refresh cycles
1	Ø	Ø	Triple refresh cycles
1	Ø	1	Four CAS display only
1	1	Ø	MPU only
1	1	1	Reserved

Table 4-7 Memory Cycle Mode Bits

Each memory cycle is normally divided into two parts. The first part is used for the MPU to access DRAM. The second part is used for the video process to access DRAM. The MPU's share uses about one third of the time in a memory cycle, while the display process uses the remainder. The reason for this is that the MPU always accesses only one byte (or 16-bit word), while the display process may access several. The display process is able to do this by using page mode to access up to four bytes quickly.

There are different types of memory cycles. It might be that the MPU will perform one access while the display process accesses one, two or four bytes. It might also be that the MPU performs an access, and the rest of the cycle is used to perform two DRAM refresh cycles. Its also possible that only the MPU will access memory and the rest of the cycle will be unused, because there is nothing that currently has to be done for display or refresh.

Some types of cycles do not allow the MPU to access memory. If this is true and the MPU is an MC68Ø9E, then the MPU's clocks are stretched so it is not aware of the missing cycle. The memory cycles are arranged so that the MC68Ø9E clocks are never stretched longer than the MPU data sheet recommends.

If the MPU is a member of the MC68ØØØ family, then DTACK is witheld. Since the MC68ØØØ family has an asynchronous bus this does not cause a problem.

There are two types of cycles that can exclude the MPU. The first is a four CAS display only cycle. This type of cycle is only used to fetch data for true objects, and only happens when the system is using a single bank memory organization.

A triple refresh also excludes the MPU. This is only used in horizontal resolutions 3, 6, and 7 and cannot occur more than once per video line.

The four least significant bits in the Control Word 4 time slot are used by the reset circuitry. See Chapter 5 for more information.

4.3.2 ADEN (Address Enable)

ADEN is an active low signal generated by RMI. It enables the outputs of the multiplexers that place the MPU address information onto the X bus. It does not control which information is placed on the X bus. It is active for the first two words on the X bus for each memory cycle.

4.3.3 ADSEL (Address Select)

ADSEL is used with ADEN. It determines which data to place on the X bus. When it is high and ADEN is low, the most significant MPU address bits are placed on the X bus. When it is low and ADEN is low, the MPU low bits are placed on the X bus. ADSEL is active only during machine cycles that allow an MPU access, and only if CS is low.

4.4 S Bus (Device Select)

The S bus is a three-bit bus that originates at RMI and is intended for use by parts other than RMC and RMI. The RMS offers a few different memory maps to the user, who can use the S bus to perform the device selects.

The S bus should be decoded by a three-to-eight line decoder to provide the individual chip selects. The decode of 7 should not be used, since it means that an RMS control register or the DRAM attached to the RMS is selected.

For more information see Chapter 1Ø, which discusses the system memory map and the S bus.

4.5 Dynamic RAM Interface

The RMS supports the use of dynamic RAM directly without additional parts. This section describes the signals that are used to interface to the DRAM.

4.5.1 Z Bus (DRAM Address Bus)

DRAM's use a multiplexed address bus. The translation to the correct format of multiplexed address bus is performed automatically by the RMI, and the result is made available on the RMI's Z bus.

The Z bus is a nine-line bus designed to connect directly to the address lines of the DRAM parts. Not all DRAM's require nine address lines; when fewer lines are required, the least significant Z bus lines should be used and the most significant lines should be unterminated.

4.5.2 RAS (Row Address Strobe)

The RAS line is a control line required by DRAM's. It is available on the RMI and may be used to drive the DRAM's directly.

It is used to strobe in the most significant bits of address from the Z bus.

4.5.3 CAS (Column Address Strobe)

The CAS lines are control lines available from the RMI and able to drive DRAM directly. They are used to strobe in the least significant bits of address from the Z bus to the DRAM.

There are five different CAS lines: CASØ, CAS1, CAS2, CAS3, and CASTB. Each line except CASTB is used to drive a different bank of DRAM. Therefore RMS can support up to four banks of DRAM.

CASTB (CAS Strobe) is a composite of CASØ-3 and changes state with each of them during the display portion of the memory cycle. It is used to strobe display data from the DRAM into the RMC.

4.5.4 WE (Write Enable)

Write enable is a control line provided by RMI. It is used to write data to the DRAM, as opposed to reading data from the DRAM. WE is able to drive DRAM directly.

4.5.5 Data Bus

The DRAM's also need to connect to a data bus. They are connected either to the RMC's A data bus or to both its A and B data buses. Regardless of how the DRAM is connected to RMC, the DRAM's data in and data out pins must be connected to each other, so that they use a common bus. As a result, the RMS does not support read-modify-write cycles.

4.6 Data Buses A and B

There are two eight-bit data buses on the RMC. The way in which they are used depends upon the MPU type in the system. The function they perform is always to tie together the data buses of the RMS, DRAM, and MPU.

4.6.1 Data Bus A

When the RMS is used with an eight-bit MPU, either the MC68Ø9E or MC68ØØ8, the A bus is connected to the DRAM data bus. It is a bidirectional bus that provides for both reading from and writing to memory.

When the 16-bit MC68ØØØ MPU is used, the A bus is also connected to DRAM. It should be connected to the least significant bits of the 16-bit MPU data bus and to DRAM bank 1 (and bank 3 if used).

The A bus is capable of directly driving the data bus of the dynamic RAM without any additional logic.

Figures 2-4 and 2-5 show the A bus connections for the MC68ØØØ.

4.6.2 Data Bus B

The B bus is connected to the MPU's data bus when an eight-bit MPU is used. It is also a bidirectional bus.

When the 16-bit MC68ØØØ is used, the B bus is connected to DRAM. It should be connected to the most significant bits of the 16-bit MPU data bus and to DRAM bank Ø (and bank 2 if used).

The B bus is capable of directly driving the DRAM data bus, and should be capable of directly driving the MPU data bus in most applications. See Chapter 13 for its drive capability.

Figures 2-4 and 2-5 show the B bus connections for the MC68ØØØ.

4.7 Video Outputs

There are five lines located on the RMC that are directly involved in outputting the video information. This section describes their functions.

4.7.1 R (Red)

The R output varies from a low voltage, which represents blanking, to a high voltage, which represents peak luminance. The difference between blanking and peak luminance is 1.Ø volts.

The R output is intended to drive a high impedance load. It requires buffering when the load impedance is less than 1Ø Kohms.

4.7.2 G (Green)

The G output is very similar to the R output. The voltage difference between blanking and peak luminance is the same, and so is its drive capability.

The one difference between the R and G outputs is that the G output can also have a sync level output voltage, which is Ø.4 volts below blanking. Sync level is a user selectable option controlled by a register bit in an RMS control register. See Section 9.3.19.

4.7.3 B (Blue)

The B output represents the blue component of the video. It is the same as the R output in voltage level and drive capability. The B output has no special options.

4.7.4 VIDEN (Video Enable)

The video enable signal is an output of the RMC that is intended for use in overlay applications. A high level on VIDEN indicates that the current output on the R, G, and B lines is programmed to be transparent in the Color Mapping RAM (see Section 9.3.25). This does not mean that there is no signal on R, G, and B. The transparency bit is separate from the twelve bits that define a color. VIDEN is provided to the user as a control signal for an external video multiplexer that would select between the outputs of the RMS and another video source. Since VIDEN is readily available, the user may choose to overlay or not overlay by combining VIDEN with another signal.

4.7.5 SYNC

SYNC is a dual purpose pin. It can be used either as an output for a sync signal or as a frame sync input.

As an output, SYNC provides TTL level composite sync, vertical sync, or horizontal sync. Composite sync will include equalizing pulses if RMS is being operated in interlace mode. There will not be equalizing pulses if RMS is being operated in non-interlace.

When SYNC is used as an input, a falling edge will cause the RMC's internal video timing generator to be reset to the trailing edge of vertical sync.

The user controls SYNC using the RMS's SYNC MODE register. See Section 9.3.19.

RMC's HSYNC output can also be used for horizontal sync. It is identical to the SYNC output except that HSYNC occurs one VTCLK time ahead of SYNC.

4.8 RTI (Real Time Input)

RTI is an active low TTL input to RMC. A falling edge on RTI causes the current value of the X and Y counters used to position true objects to be loaded into registers that can be read by the MPU in the RMS control register memory map. See Chapter 8 on true objects for a discussion of screen XY coordinates and Section 9.3.3 for a discussion of the Interrupt Status register, which this also affects. RTI must be held low for at least 25Ø nanoseconds.

RTI may be used as an input for a light pen.

4.9 INT (Interrupt)

INT is generated for use by the MPU. It is controlled by a register in the memory map that is described in Section 9.3.3.

4.1Ø REN (Reset Enable)

REN coordinates an RMS reset with any MPU reset that occurs after power-up. REN is discussed in Chapter 5.

4.11 Tl, T2

Tl and T2 are used for testing during manufacture. They should be connected to ground.

5.Ø RESET AND INITIALIZATION

The RMS requires reset and a certain amount of initial information before the MPU can talk to it. It must know what type of MPU is driving the system before it can configure itself to operate with that MPU.

5.1 System Reset

The RMI has power-up and reset circuits, and the RMC has reset circuits. The RMI's power-up circuit is entirely self contained and requires no external components. A system reset must be generated before the RMS can operate; the reset line must be held low for at least 64 microseconds, once Vcc is established.

The following conditions are established after system reset. Any parameter that is not listed here is undefined after system reset.

1. Machine 1

2. Unfolded memory map.

3. MAPA set to Ø.

4. Interrupt output and reporting disabled.

5. Video display disabled.

6. UPI and LPI disabled. Paging register set to all 1's.

7. Noninterlace sync and data.

8. Horizontal resolution 4

5.2 Reset Enable

System resets are performed by means of the reset enable (REN) pin on RMC and the X bus. See Figure 5-1. REN is an active high RMC output designed to drive the base of an NPN transistor whose emitter is connected to the MPU's reset line. The collector must be tied to XØ, X1, X2, or X3 of the X bus. This circuit is arranged so that if the MPU reset line is active, and REN is active, then one of the four X bus lines will be low. This is the indication to RMS that a reset has been requested.

REN becomes active during every MT6 time. Both RMC and RMI sample the X bus then, and if any of the four X bus lines goes low, they reset themselves.

The transistor is connected to one of four different X bus lines. The RMS initialization depends on which X bus line is used, as explained below.

Line	Initialization Type
XØ	MC68Ø9E MPU and 625 line video timing.
X1	MC68Ø9E MPU and 525 line video timing.
X2	MC68ØØØ family MPU and 625 line video timing.
X3	MC68ØØØ family MPU and 525 line video timing.

Table 5-1 X Bus Initialization Types

The selection of 625 or 525 line video timing also selects the proper divide circuit for generating color subcarrier (CSC). CSC is an output pin located on RMI. If 625 line timing is selected, then CSC is equal to the master oscillator divided by 8. If 525 line timing is selected, then CSC equals the master oscillator divided by 1Ø.

This is also the way in which the the RMS finds out what kind of MPU is in the system. From this information, the RMI can arrange its MPU handshaking lines so that the MPU can provide the RMS with the additional information required.

The RMS requires this information immediately after power-up reset, as well as at other times when reset occurs. The MPU reset must last at least one video line time (64 microseconds), which is enough time to guarantee that the REN type of reset can occur.

Since the RMS may be involved in providing data for the MPU during its power up routine, many systems will have to guarantee that the RMS reset ends before MPU reset ends.

5.3 Video Reset

There is a third type of reset associated with the RMS: video reset. This type of reset does not have an effect on any of the control registers. Its function is to reset the video timing counter chain in the RMC to a known state. This is very important when trying to synchronize with an external video signal. Chapter 11 covers this subject, but it is summarized here.

In order to use video reset, RMC's sync pin must be programmed to act as an input. When the input signal appears on the sync pin, the vertical timing chain in the RMC is reset. The effect is to place the video timing chain into the same state it would be in if the RMS had just finished vertical sync.

6.Ø GENERAL VIDEO

This chapter deals with some of the general parameters of the video signal generated by the RMS. For example, it discusses the available screen resolutions and color sets. However, it does not get into the details of how the data is interpreted in the different display modes.

6.1 Video Timing

Video timing refers to the timing of the CRT control signals such as horizontal and vertical sync, blanking, and equalizing pulses. The RMS offers two basic options for video timing: 525 line, 6Ø Hz System M (NTSC) and 625 line, 5Ø Hz System B (PAL). The common notation in this manual is NTSC and PAL. While this is not technically accurate, it agrees with the way these systems are normally used and is familiar to a wider audience. Either can be used in an interlace or noninterlace mode.

The choice of 525 or 625 line timing is made during initialization, as described in Section 5.1. The choice of interlace or noninterlace sync is made in the same RMS control register that selects screen resolution.

Both 525 line and 625 line timing permit using all horizontal resolutions (HRES modes) and display modes (bit-plane mode and the list modes), but the higher vertical resolutions (VRES modes) are limited to 625-line timing (see Section 6.2.2). HRES Ø and 1, and VRES Ø and 1, are available only in Machine 2 mode. See Chapter 14.

6.1.1 525 Line Timing

525 line timing is available as either 525 lines per frame in full interlace mode or 262 lines per field in noninterlace mode. The following table is based on a 35.79545 MHz master oscillator.

Lines per frame	interlace	525	
	noninterlace	524	
Field frequency	interlace	59.94 Hz	
	noninterlace	6Ø.19 Hz	
Line frequency	interlace	15,734.26 Hz	
	noninterlace	15,768.92 Hz	
Line period	interlace	63.5555	microseconds
	noninterlace	63.4159	microseconds
Line blanking interval	interlace	11.873	microseconds
	noninterlace	11.733	microseconds
Front porch		1.816	microseconds
Sync pulse duration		4.749	microseconds
Back porch		5.3Ø8	microseconds
Field blanking period		21 H	
Duration of first series of equalizing pulses		3 H	
Duration of synchronizing pulses		3 H	
Duration of second series of equalizing pulses		3 H	
Duration of equalizing pulse		2.375	microseconds
Duration of field sync pulse		27.Ø29	microseconds
Interval between field sync pulses		4.749	microseconds

Note: H equals one horizontal line time.

Table 6-1 525 Line Timing

6.1.2 625 Line Timing

625 line timing is available either as 625 lines per interlace frame or as 312 lines per noninterlace field. The following table is based on a 35.46895 MHz master oscillator.

Lines per frame	interlace	625
	noninterlace	624
Field frequency		5Ø.ØØ Hz
Line frequency		15,625.Ø8 Hz
Line period		63.999 microseconds
Line blanking interval		11.982 microseconds
Front porch		1.833 microseconds
Sync pulse duration		4.793 microseconds
Back porch		5.216 microseconds
Field blanking interval		27 H
Duration of first series of equalizing pulses		2.5 H
Duration of synchronizing pulses		2.5 H
Duration of second series of equalizing pulses		2.5 H
Duration of equalizing pulse		2.396 microseconds
Duration of field sync pulse		27.2Ø7 microseconds
Interval between field sync pulses		4.793 microseconds

Note: H equals one horizontal line time.

Table 6-2 625 Line Timing

6.2 Screen Resolution

The user can choose from a large selection of screen resolutions. These choices are made entirely by software in the RMS control registers, and do not involve changing master oscillator components or other external devices. The display device that the RMS drives, however, may set some limits on which resolution choices are practical.

Horizontal and vertical resolution are separate choices. Section 6.2.4 shows the allowable combinations of horizontal and vertical resolution.

6.2.1 Horizontal Resolution

Horizontal resolution is chosen in the Video Operation register (see Section 9.3.18). Three bits of this register are referred to as HRESØ, HRES1, and HRES2. The resulting three-bit value is called an HRES mode.

By selecting an HRES mode, the user selects a horizontal resolution. Other changes that must be made to the RMS hardware, such as generating the proper picture element clock, are made automatically as a result of choosing an HRES mode.

HRES Mode	Resolution in Pixels	Active Display in Microseconds
0	64	35.76
1	128	35.76
2	256 (Narrow)	35.76
3	256 (Wide)	42.91
4	320	40.23
5	Reserved	
6	512	42.91
7	640	44.70

Table 6-3 Horizontal Resolutions

When the HRES mode is changed, the user should allow one full line time for the new HRES mode to take effect.

HRES modes 0 and 1 are backwards compatible with the MC6847 and MC6883. They can only be used with Machine 2.

HRES modes 2 and 3 are both 256 pixels wide, but mode 3 is 20% wider. Mode 2 is intended for use with televisions, which have large, unadjustable overscan, while mode 3 is intended for use with monitors.

6.2.2 Vertical Resolution

Vertical resolution is chosen in the Video Operation register. The resulting value is referred to as a VRES mode, and it is located in the same control register as the HRES mode (see Section 9.3.18).

The VRES mode allows the user to directly select the number of active video lines to display in each field. The relationship between the start of active video and field sync is controlled automatically when a VRES mode is selected, so that the display is centered with respect to sync.

VRES Mode	Lines per Field
0	64
1	96
2	192
3	200
4	210
5	240
6	250
7	Reserved

Table 6-4 Vertical Resolutions

The selections of 64 and 96 video lines are actually 192 lines high. The video data is repeated for either 3 or 2 video lines in a row. VRES modes 0 and 1 provide backward compatibility with the MC6847.

They can only be used with HRES modes Ø and 1 in Machine 2 operation. See Section 6.2.4.

VRES modes 5 and 6 are for use with 625 line 5Ø Hz display devices.

When a new VRES mode is selected, the user should allow one full video frame time for the new VRES mode to take effect.

6.2.3 Interlace Data

If the interlace sync option has been selected for use with a list mode, the user has the choice of interlace or non-interlace data. Interlace sync with non-interlace data is essentially the same as non-interlace sync (which must use non-interlace data): the same information fits on the screen in both cases. The only difference is that the data is presented twice per frame, with the second field being one scan line lower than the first.

Interlace sync with interlace data doubles the resolution of the screen. For example, with VRES mode 3 the user may display 4ØØ scan lines instead of 2ØØ, or a maximum of 5Ø character rows instead of 25. The user's memory organization is unchanged except that the displayed screen is twice as high (displayed screen is defined in Section 6.4). The same image tables of characters or objects can be used in both forms, but they will be half as high when used as interlace data. See Section 9.3.2 for a discussion of the RMS' Display Data Mode control register.

6.2.4 Total Screen Resolution

The following table shows the combinations of vertical and horizontal resolution that are supported by the RMS. The valid combinations are marked with a "1" if they work in Machine 1 mode and a "2" if they work in Machine 2 mode.

					HORIZONTAL					
	MODE		Ø	1	2	3	4	5	6	7
	O									
	D	RES	64	128	256	256	32Ø	*	512	64Ø
	E	E								
		S								
V	Ø	64	2	2				R		
E	1	96		2				e		
R	2	192		2	1,2	1	1	s	1	1
T	3	2ØØ			1	1	1	e	1	1
I	4	21Ø			1	1	1	r	1	1
C	5	24Ø (PAL)			1	1	1	v	1	1
A	6	25Ø (PAL)			1	1	1	e	1	1
L	7	Reserved						d		

Table 6-5 Horizontal and Vertical Resolution Combinations

When the RMS is operated with interlace sync and data, the vertical resolution is double the values shown in this table.

6.2.5 Interlace Data and List Modes

When noninterlace sync is used, each list mode character row can be 8, 1Ø, 12, or 16 scanlines high. When interlace sync and noninterlace data are used, the number of scanlines per character row doubles, but because the total scanline count also doubles the apparent character height remains unchanged. When interlace sync and data are used, the number of scanlines per screen is twice that of a noninterlace screen using the same vertical resolution, but the number of scanlines per character row is the same as in a noninterlace screen. The interlace sync and data screen has twice as many character rows as the noninterlace screen, and each row is half as high.

For example, a noninterlace display using VRES 3 has 2ØØ scan lines of active video. The screen can display 25, 2Ø, 16.6, or 12.5 character rows if the character height is 8, 1Ø, 12, or 16 scanlines, respectively. This is shown in the "Noninterlace" column of Figure 6.1.

If interlace sync with noninterlace data is selected, the number of scan lines doubles (to 4ØØ, in this example), but the number of character rows does not. Instead, each scan line's data repeated in each field of the frame, as shown in the "Interlace Sync" column in Figure 6-1.

If interlace sync and data are both selected, the number of scan lines is twice the noninterlace count (again, 4ØØ in this example), but each scan line has its own data, and the number of character rows doubles, as well. This is shown in the "Interlace Sync & Data" column in Figure 6-1.

6.3 Color Selection

RMS offers the user a very wide selection of colors that may be used to generate the display. In order to keep the bits per pel requirements as low as possible, a Color Mapping RAM (CMR) is used. This section discusses both the possible color set and the in-use color set.

6.3.1 Color Palette

The RMS is capable of generating 4Ø96 different colors. A twelve bit word is required to select one of these colors.

The twelve-bit word is divided into three four-bit sections. Each of the four-bit sections defines the magnitude of one of the primary colors: red, green, and blue. By choosing the proper amount of each of the primary colors and mixing them together, it is possible to generate any color.

The four-bit value for each primary is used as the input to a 4-bit D/A converter. The output of the D/A becomes an output of the RMS. The RMS has an output pin for each of the primary colors.

The twelve-bit values that define a color are not stored in DRAM. They are stored in a special RAM inside the RMC referred to as the Color Mapping RAM (CMR). Access to the CMR is available through the control register memory map.

Each CMR register has 13 active bits. The 13th bit does not affect the R, G, or B outputs. It is used to drive the VIDEN output. Since the 13th bit does not have a direct effect on R, G, or B, the user may elect to use VIDEN to control a transparency feature. External logic is required to utilize this feature. See Chapter 11.

6.3.2 In-use Colors

The CMR provides the user with a wide selection of colors without having to use a large number of bits per pel. The CMR consists of 32 twelve-bit words. Each twelve-bit word defines one of 4096 colors, but there are only 32 words stored in the CMR. These 32 words constitute the in-use color set. Only 32 different colors can be displayed on the screen at once, unless the MPU modifies the CMR contents in real time.

The 1, 2, or 4 bits per pel of color information from DRAM are used with leading zeroes to get a 5-bit CMR address. Some characters in some list modes can have attributes that alter the pel data to get different CMR addresses, and the true objects can use up to 24 of the CMR registers. The selected CMR value goes to the red, green, and blue D/A converters to generate the video output.

In some display modes, there are fewer than 5 bits per pel of video data. In these cases it is not possible to make use of all 32 colors stored in the CMR.

6.3.3 Using the CMR

The CMR is a powerful software tool for manipulating the video display. The user can change the color of a large portion of the video screen by changing the value of a single twelve-bit word, without having to modify every pixel of video data in DRAM.

The user can also make one area of the screen "disappear" into another by giving both of them the same twelve-bit value in their respective CMR registers. The area can be made to reappear by changing it back again.

It is also possible to display more than 32 different colors on the screen at once by changing the contents of the CMR in real time. As a simple example, the entire contents of the CMR could be changed midway through a video field to display 64 different colors during a single field. The upper limit on how many colors can be displayed in one field is determined by the application and the software.

Changing the data in the CMR is a powerful tool, but there are some restrictions on its use. If the MPU and the RMS both try to access the same CMR register at the same time, the MPU will gain access to the CMR, but the video process will be locked out. Therefore the color that appears on the screen during the 2ØØ nanoseconds or so that the MPU is accessing the CMR is indeterminate, if it is the same CMR location that is being accessed. The screen will not be affected if the MPU is accessing a CMR register that is not being accessed by the video data. The recommended approach is to write to the CMR when video is not active, during vertical and horizontal retrace. Vertical blanking is provided in the Interrupt Status Register (see section 9.3.3) and the RTO interrupt can be used to select a horizontal retrace (see Sections 9.3.3 and 9.3.15).

The restrictions on accessing the CMR apply to the 13th bit as well as the twelve color bits.

6.4 Virtual and Displayed Screens

The amount of video data that can be displayed at one time is controlled by the horizontal and vertical resolution of the displayed screen (see Section 9.3.18). The user can define a virtual screen that is larger than the displayed screen, using three registers: the Virtual Screen Start Address register, the Virtual Screen Size register, and the Virtual Screen Width register. They are discussed in detail in Section 9.3. The screen memory is organized as a series of scanlines in the bit-plane mode (or character rows in the list modes), and these three registers permit defining a rectangular block of screen memory with more pixels (or characters) than can fit across the displayed screen, and with more scanlines (or character rows) than can fit down the displayed screen.

The displayed screen, whose size is defined by the HRES and VRES modes, is located within the virtual screen by the Horizontal Offset and Vertical Offset registers, which indicate where the upper left corner of the displayed screen is with respect to the virtual screen. The details of their computations are covered in the control register discussion.

6.5 Scrolling

The displayed screen can be moved within the virtual screen one pixel at a time vertically and horizontally using four registers: the Horizontal and Vertical Offset registers, and the Horizontal and Vertical Scroll registers.

Coarse scrolling is accomplished using the offset registers and fine scrolling uses the scroll registers. The distinction between fine and coarse is that the offset registers must point to data that begins a memory cycle, while the scroll registers point within a memory cycle.

Each memory cycle deals with 16 pixels (horizontal resolution of 512 or 640) or 8 pixels (horizontal resolution of 320 pixels or less). The user can then set the Horizontal Scroll register to the number of pixels into the memory cycle data that the display is to begin.

The Vertical Offset register measures scan lines in bit-plane mode and character rows in the list modes. The Vertical Scroll register is ignored in bit-plane mode, since the Vertical Offset register is already at the smallest unit; in the list modes the scroll register is used to move up and down within a character row. The user sets the height in pixels of the character rows in a separate register (see Section 9.3.2); the Vertical Scroll register must be limited to values less than the selected row height. Its maximum value is 15.
The offset and scroll registers are discussed in Chapter 9. Section 12.4 has more details on the mechanics of scrolling.

7.Ø DISPLAY MODES

This chapter describes the capabilities, limitations and requirements for using each of the display modes available with the RMS in Machine 1 mode. Machine 2 is discussed in Chapter 14.

The RMS' display modes fall into two general categories; bit-plane and list modes. Bit-plane is a direct mode. Data is taken from DRAM and broken into pieces. Each piece represents one pixel, so data flows directly from DRAM to the video outputs.

List mode is indirect. Data is taken from DRAM to select a character. The pel pattern of the character needs to be defined in a second step that accesses either DRAM or an internal ROM. The results of this second access, possibly modified by character attributes, are sent to the video outputs.

The choice between bit-plane and list mode is made in an RMS control register; see Section 9.3.2.

7.1 Bit-plane Mode

Once bit-plane mode has been chosen, the user must select how many bits will be used to describe each pel in the screen memory. With a horizontal resolution of 512 or 64Ø pixels, the user may choose 1 or 2 bits per pel, and with less resolution 1, 2, or 4 bits per pel are available. This allows the user to display 2, 4, or 16 colors at the cost of doubling the required screen memory for each increase in color range.

The data in DRAM is used in a very straightforward manner. It is accessed sequentially to create a screen, and each byte is broken into 2, 4, or 8 pels. The most significant bits of a byte are the first to be turned into video. For example, if the user has selected a 4 bits per pel mode and the visible screen start address is $1ØØØ, the first data that RMS will pull in to create video will be at $1ØØØ, followed by $1ØØ1, $1ØØ2, etc. The byte at $1ØØØ will be broken into two 4-bit pels. The first pel will be made up of bits 7, 6, 5, and 4 of the byte. Bit 7 will be the most significant bit of the pel. The second pel will be made up of bits 3, 2, 1, and Ø, with bit 3 being the most significant. The other bytes will be broken up in the same fashion. Leading zeros are added to the bits for each pel to get the required 5-bit CMR address.

The amount of memory required to display one screen may influence the display mode, and possibly the screen resolution, chosen by the user. The following table lists the amount of memory required for the displayed screen for Machine 1 in many modes. If the virtual screen is larger than the displayed screen, it will require more memory, of course.

Horizontal Resolution	Vertical Resolution	1 Bit per Pel	2 Bit per Pel	4 Bit per Pel
256	192	6144	12,288	24,576
256	200	6400	12,800	25,600
320	200	8000	16,000	32,000
256	210	6720	13,440	26,880
320	210	8400	16,800	33,600
256	240	7680	15,360	30,720
320	240	9600	19,200	38,400
256	250	8000	16,000	32,000
320	250	10,000	20,000	40,000
512	384	24,576	49,152	NA
640	384	30,720	61,440	NA
512	400	25,600	51,200	NA
640	400	32,000	64,000	NA
512	420	26,880	53,760	NA
640	420	33,600	67,200	NA
512	480	30,720	61,440	NA
640	480	38,400	76,800	NA
512	500	32,000	64,000	NA
640	500	40,000	80,000	NA

Table 7-1 Bit-plane Screen Memory Requirements

7.2 List Mode

In list mode the screen is made of rows of characters or fixed objects which are register-selectable to be 8, 10, 12, or 16 pels high. List mode uses memory two ways: screen memory containing a display list of references to characters (and attributes), and image tables giving pel-by-pel descriptions of the characters to be displayed. The RMS contains image tables for alphanumerics and mosaics, and the user can define other types.

Memory Requirements in Bytes for One Displayed Screen
8 Scan Lines/Character Height, Noninterlace Data

List Mode	HRES	VRES				
		2 (192)	3 (200)	4 (210)	5 (240)	6 (250)
0	2, 3 (256)	768	800	864	960	1008
	4 (320)	960	1000	1080	1200	1260
1	2, 3 (256)	1536	1600	1728	1920	2016
	4 (320)	1920	2000	2160	2400	2520
2	6 (512)	1536	1600	1728	1920	2016
	7 (640)	1920	2000	2160	2400	2520
3	6 (512)	2304	2400	2592	2880	3024
	7 (640)	2880	3000	3240	3600	3780
4	2, 3 (256)	2304	2400	2592	2880	3024
	4 (320)	2880	3000	3240	3600	3780
5	6 (512)	2304	2400	2592	2880	3024
	7 (640)	2880	3000	3240	3600	3780

Table 7-2 Sample List Mode Screen Memory Requirements

For 10, 12, and 16 scan lines/character height, divide the above values by 1.25, 1.5 and 2, respectively. Some character heights have a non-integer number of lines in some VRES modes; these partial lines use a full character row's worth of memory. Double any memory size when using interlace sync and data, but see Table 3-3 for available combinations. The choice of 1, 2, or 4 bits per pel affects the size of the image table entries but not the display list size.

The rest of this section describes the character types, the attributes that they may take on, and the list modes that use them.

7.2.1 Alphanumeric Characters

Alphanumeric characters are available from a character ROM in the RMC which has 96 ASCII characters.

The characters are 5 pels wide and 7 pels high; see Figure 7-1. The top row is always blank. Each character is displayed in an 8-pel-wide block with the two leftmost and single rightmost columns blank, as shown in Figure 7-2. Blank rows are added to the bottom of the character as the character height is increased from the normal 7 pel height, except for the seven ASCII characters that have descenders. These are ", ; g j p q y". When an 8-pel character height is chosen, these are displayed two pels above their normal position to avoid cutting off their tails. For 10, 12, and 16 pel heights, they are displayed in their proper positions, descending two pels below the bottoms of the nondescending characters. See Figure 7-2.

Alphanumerics are stored as one bit per pel characters which have foreground and background colors only, regardless of the bits per pel of the user-defined characters.

7.2.2 Mosaic Characters

Mosaic characters are block graphics characters that are defined in the RMC's internal ROM which allow simple graphics to be displayed without the need for image tables. Five list modes allow mosaics using 6 blocks per character, and two of these modes also allow 4-block mosaics. See Figure 7-3.

Each pel is numbered with the binary bit that selects foreground color (if a 1) and background color (if a Ø). Mosaic 6 characters range from Ø (all background) to 95 ($5F) (all foreground), and mosaic 4 characters range for Ø to 15 ($ØF). Mosaic 6 character numbers do not use the 5 bit, which is reserved for the separation attribute. When mosaic 4 is selected, the 4 least significant character bits are used, and the 2 most significant bits are ignored. For example, mosaic 6 characters 95 ($5F), 31 ($1F), and 15 ($ØF), all convert to mosaic 4 character 15 ($ØF) when the mosaic 4 attribute is selected.

All mosaics characters are 8 pels wide, so each block is 4 pels wide. The height of each block depends on the number of scan lines per character row that the user has selected. See Figure 7-3. Mosaic 4 characters divide evenly into four 4 x 4 blocks (8 lines/row), 4 x 5 blocks (1Ø lines/row) 4 x 6 blocks (12 lines/row), and 4 x 8 blocks (16 lines/row). Mosaic 6 characters' blocks have top and bottom blocks of the same size and middle blocks of a different size:

Character Height	Block Heights
8	3, 2, 3
1Ø	3, 4, 3
12	4, 4, 4
16	5, 6, 5

Table 7-3 Mosaic Block Heights

The separation attribute is always available when mosaic characters are available. It can apply only to mosaic characters. See Section 7.2.4.9.

7.2.3 Redefinable Characters

Redefinable characters are characters whose pel patterns are not defined by the RMS. The patterns are defined by the user and stored in image tables in DRAM like the screen display list itself. When the RMS logic encounters one of these characters in the display list, it accesses the image table to get the basic pel pattern. Any selected attributes are then applied to the pattern data. The RMS supports two types of user-defined characters: the Dynamically Redefinable Character Set (DRCS) and fixed objects.

The DRCS and fixed objects take up the same space on the displayed screen: in list modes Ø, 1, and 4 they are 8 pels wide; in list modes 2, 3, and 5 they are 16 pels wide. Both types must have the number of rows of pels specified in the lines-per-row bits (see Section 9.3.2). Both character types use image tables of the same form.

7.2.3.1 The Dynamically Redefinable Character Set (DRCS)

The DRCS are available in two variations. First, in list modes Ø, 1, and 4, there are 32 alphanumeric-type DRC's. Their image tables must contain the number of bits per pel specified in the Display Data Mode register, but only the least significant bit of each pel value is used. For the two DRCS types discussed so far, a final pel value of 1 selects the alphanumeric foreground color, and a Ø selects the background color. The final pel value is based on the image table pel value plus any changes caused by user-selected attributes (Section 7.2.4). The colors for foreground and background are list-mode dependent; see Section 7.2.5.

The second variation of DRCS is available in list modes 2, 3, 4, and 5. Its bits per pel must match the Display Data Mode register's value, but all bits are used to select pel colors, not just the LSB. This is a graphics-oriented, rather than a character-oriented, type. As discussed in Section 7.2.5.5, the two types of list mode 4 DRC's share the same DRCS image table.

All attributes that can be used with the DRCS affect their displayed appearance. Attributes available for the DRCS are flash, foreground and background colors, double height and width, CMR offset, invert, underline, and color/resolution. These are described in Section 7.2.4; the availability of these for each list mode is discussed in Section 7.2.5.

7.2.3.2 Fixed Objects

Fixed objects have three distinct characteristics: their image tables must always contain the number of bits per pel in the Display Data Mode register (Section 9.3.2), they can have appearance-altering attributes, and they can have true-object-interactive attributes. Fixed objects are available in list modes 1-5.

The bits per pel values allow a fixed object to display 4 colors (list modes 2, 3, and 5) or 16 colors (list modes 1 and 4), in contrast to the alpha-type DRC's two colors. Fixed objects can have these appearance-altering attributes: CMR offset, flash, color/resolution, underline, invert, and double height. They can have these true-object-interactive attributes: collision enable, color collision, priority, and shading. The attributes are discussed in Section 7.2.4, the list modes and their attribute availability are discussed in Section 7.2.5, and true objects are discussed in Chapter 8.

7.2.3.3 Image Tables

DRCS and fixed object image tables must contain the correct number of bits per pel, and each character in the tables must have enough pels to fill its area of the screen. The number of pels in a character depends on the character width (8 pels in list modes Ø, 1, and 4; 16 pels in list modes 2, 3, and 5) and the character height, which is the number of lines per character row. The number of pels in a character can range from 8 x 8 = 64 to 16 x 16 = 256.

The RMS has address registers for the DRCS and fixed objects (see Sections 9.3.1Ø and 9.3.12) that it uses to find the start of each image table. The user puts a character in the display list by selecting the character number (Ø is the first), using the formats described in Section 7.2.5. The RMS locates the pel pattern for the character by multiplying the character number times the character size and adding the result to the start address register contents.

The amount of memory needed to describe a character in the image table depends on the number of bits per pel (1, 2, or 4), the number of scan lines per character row, and the character width (8 or 16 pels).

CHARACTER SIZE = (Bits per Pel)*(Character Height)*(Character Width)

The table is filled in the same way as the bit-plane mode's screen memory: the upper left pel is described in the most significant bits of the first byte. The next bits in that byte describe the pel to its right, and so forth. For example, the smallest possible character description is one bit per pel, 8 scan lines high, 8 pels wide. Each entry in the image table is:

```
CHARACTER SIZE = (1 Bit per Pel)*(8 Scan Lines)*(8 Pels Width)
               = 8 Bytes
```

Since one byte describes 8 pels at 1 bit per pel, and since the character is 8 pels wide, the first byte describes the character's top row, the second byte describes the next-to-the-top row, and so on.

The largest possible character description is 4 bits per pel, 16 scan lines high, and 8 pels wide:

```
CHARACTER SIZE = (4 bits per Pel)*(16 Scan Lines)*(8 Pels Width)
               = 64 Bytes
```

Each byte describes two pels in this case, so the top row of 8 pels is described in the first 4 bytes.

The MPU must store these patterns in DRAM and set the RMS pointers if redefinable characters are to be used. The three parameters (bits per pel, character height and width) are fundamental to correct location and interpretation of the image tables, so if the color range or character height are changed, the current image table will be invalid. If redefinable characters are to be used in the new display, the user must provide an image table using the new values, and the pointer must be changed when the other parameters are. This may also be necessary if only the list mode is changed, since some character types have 8-pel width in some modes and 16-pel width in others.

7.2.4 Attributes

The display list that contains the identifiers for the characters to be put on the screen also contains flags that allow each of these characters to be individually modified. These individual features are called attributes, and every list mode allows some. The character type and the list mode in use affect the selection; attributes are available for the ROM-based alphanumerics and mosaics as well as the redefinable characters. Attributes are applied to the data once it is in the RMS, so they do not alter the image table contents.

Each entry in the display list contains its own character identifier and flags for that character's attributes. There are no attributes that can be set for more than one character at a time.

Some attributes are designed for videotex and word processing, and others are intended for games. The rest of this section discusses the attributes in detail, and the next section discusses where the attributes can be used. Where appropriate, each attribute discussion also describes what happens when the attribute is not available.

7.2.4.1 Foreground Color and Background Color

Foreground and background colors are selectable in list modes 1 and 4 for the ROM-based alphanumerics and mosaics, and for the first 32 characters of the DRCS when they are used with alphanumeric attributes. The 3 bits available for each color in list mode 1 allow selection of CMRØØ through CMRØ7, and the 4 bits in list mode 4 allow selection of CMRØØ through CMRØF. If a DRC is used as an alphanumeric (see Section 7.2.5.2 and 7.2.5.5), the least significant bit of each pel in the image table (which can have 1, 2, or 4 bits per pel) selects foreground color if it is a 1 and background color if Ø. This takes extra image table memory, but it may be necessary in order to get full color for fixed objects, which must have the same number of bits per pel.

In list modes Ø and 2 the ROM-based alphanumerics and mosaics have no color attributes. This is also true for the DRCS in list mode Ø. In these cases the foreground color is CMR address 15 (CMRØF) and the background color is CMR address Ø (CMRØØ). When no color attributes are available for redefinable characters, their bits per pel are used as the CMR addresses. Therefore, if one bit per pel is being used, the colors available are CMR addresses Ø and 1; if two bits per pel are being used, the CMR addresses can go from Ø to 3, and if four bits are in use, they can go from Ø to 15.

7.2.4.2 Color Mapping RAM Offset

Some characters can have a CMR offset that can be combined with the bits-per-pel data from the image table to get a new CMR address. When the attribute contains 4 CMR offset bits, they are the most significant bits, with an understood LSB of zero. When it contains 3 CMR offset bits, they are the middle bits, with an MSB of zero and an LSB of zero. When there is one CMR offset bit, it is the MSB, with 4 trailing zeros. These bits are inclusive or'ed with the bits-per-pel data, which forms the least significant bits. For example, in list mode 4 with 4 bits per pel, the CMR offset might be binary 1Ø1Ø and the image table data might be ØØ11:

	MSB				LSB
CMR OFFSET	1	Ø	1	Ø	(Ø)
IMAGE TABLE DATA	(Ø)	Ø	Ø	1	1
CMR ADDRESS	1	Ø	1	1	1

Or, as an extreme example, if the CMR offset had been 1111:

	MSB				LSB
CMR OFFSET	1	1	1	1	(Ø)
IMAGE TABLE DATA	(Ø)	Ø	Ø	1	1
CMR ADDRESS	1	1	1	1	1

When the CMR offset is 1111, only two colors can be displayed for that character, and only the image table's LSB has any effect. The colors available would be 1111Ø (CMR1E) and 11111 (CMR1F).

This attribute is simplest to use when the offset bits that overlap the image table data are zero. In that case, it acts as a true offset.

List modes Ø and 2 do not have CMR offsets; mode 1 has three bits, modes 3 and 4 allow four bits, and mode 5 allows one and three bits of offset. In addition, list mode 1's two fixed object types contain a single CMR bit that is fixed; see Section 7.2.5.2.

See Section 7.2.4.15 for attribute interaction.

7.2.4.3 Flash

When flash is used with one-bit-per-pel characters, it causes the foreground color to become the same as the background color momentarily, at a regular rate. When only one bit is available for the flash attribute, it is the Flash 1 bit in the table below, so the rate is about 2 Hz. When two bits are available, three different flash speeds may be selected. These are approximately 1, 2, and 4 Hz. The exact speeds are vertical sync rate divided by 64, 32, or 16.

The attribute bits to select flash are coded as follows.

Flash 1 (MSB)	Flash Ø (LSB)	Flash Rate
Ø	Ø	No Flash
Ø	1	1 Hz
1	Ø	2 Hz
1	1	4 Hz

Table 7-4 Flash Rates

Since the foreground color becomes the background color part of the time, while the background color remains the same, the effect is that the character appears and disappears at a regular rate.

List modes 3, 4, and 5 allow DRC's and fixed objects to flash. Regardless of the number of bits per pel, these characters alternate between their regular display and a full character of CMRØØ.

See Section 7.2.4.15 for interaction among attributes.

7.2.4.4 Invert

Invert causes the video pattern data to be inverted. If the character is an alphanumeric or mosaic character from the internal ROM, or an alpha-type DRC, the effect is to reverse the foreground and background colors. If the character has multiple bits per pel, all of the pel bits are inverted.

When a character has both CMR offset and invert, the invert attribute does not have any affect on the CMR offset bits.

For example, suppose a list mode 4 fixed object has a CMR offset of binary 11ØØ, a pel value of 111Ø, and the invert attribute selected. First the pel value is inverted:

```
     before invert: 111Ø
     after invert:  ØØØ1
```

Then CMR offset is applied:

```
     CMR offset:              1 1 Ø Ø (Ø)
     inverted pel value:        Ø Ø Ø  1
                                --------
     CMR register number:     1 1 Ø Ø  1
```

The displayed pel has the color in CMR19.

To illustrate the bits per pel inversion, suppose a two bit per pel fixed object in list mode 5 had a CMR offset of 11Ø, a pel value of Ø1, and the invert attribute selected.

```
before invert: Ø1
after invert:  1Ø
```

Then CMR offset is applied:

```
CMR offset:                (1) 1 1 Ø (Ø)
inverted pel value:                1  Ø
                               ---------
CMR register number:        Ø  1 1 1  Ø
```

The displayed pel has the color in CMRØE.

See Section 7.2.4.15 for more details about attribute interaction.

7.2.4.5 Underline

Underline applies to alphanumeric and some redefinable characters in list modes 4 and 5. It fills the tenth row from the top of the character (row 9) with color. The alphanumerics get their foreground color, and the redefinable characters get CMR address Ø1111, no matter how many bits per pel are selected. The CMR offset is applied to this, if there is one.

The invert and flash attributes work on the underline the same as on the character itself; underline is applied as described in Section 7.2.4.15 on attribute interaction.

If this attribute is selected when each character row is only 8 scan lines high, no underline is displayed.

7.2.4.6 Double High

Double high characters are stretched to twice their normal height, while their width remains the same. For example, a double high character in 1Ø lines per character row mode becomes twenty lines high. It occupies all of two contiguous character rows. The same character code and attributes must be written into both of the normal size character locations that the double high character occupies in the display list.

Care should be used when using double high characters and the vertical scroll feature. When the upper of the two character rows that the character occupies is off the screen, a special bit must be set. The Vertical Scrolling register contains the double high preset bit, which must be set when the top row of the screen is expected to display the bottom half of double high characters. The RMS only examines this bit during the four scan lines just before the start of vertical active display, so it may be updated at any other time without causing display problems. See Section 9.3.8.

When the RMS finds a character row containing the double high attribute, it displays the top half of the character unless one of two conditions is met: 1) this is the top visible character row and the double high preset bit is set, 2) the character row above this one displayed the top half of at least one character. In these two cases, the bottom half is displayed. This test is applied to all the characters that appear on the screen, plus the one character to the right of the displayed screen, if there is one. It is not applied to the full virtual screen. The character to the right of the displayed screen can be partially displayed if the horizontal scrolling and offset registers are set appropriately, so it must be included in the checking. Horizontal scrolling is discussed in Sections 6.4, 6.5, 9.3.9, 9.3.2.2, and 12.4. When a row is being displayed, no reference is made to the previous row, so if attributes and characters do not match, the display may be incomprehensible. In particular, trying to put the top half of a character on a row that the RMS treats as a bottom half row causes the bottom half to be displayed. In the next row, the desired bottom half displays as a top half. This can be very hard to decipher. See Figure 7-3 for the correct and incorrect uses of this attribute, and Figure 7-4 to see how Figure 7-3 would actually display.

The double high and double wide attributes may be used together to make a double size character.

7.2.4.7 Double Wide

The double wide attribute causes an 8-pel-wide character to be displayed with twice its normal width, but with normal height. The character is stretched to the right so that the character that normally follows it is covered by its extra width. The character code and attributes of the second character are ignored by the RMS.

Care should be used when using double wide characters and horizontal scrolling. When the screen has been scrolled so that only the right half of the double wide character is displayed at the screen's left edge, there can be a problem. There are two ways to deal with this.

The first is to use the double wide preset bit located in the same control register as the horizontal scrolling bits. Using this bit requires that the second character space, which is usually ignored, have the same character and attributes as the first space. The double wide preset bit is used to indicate that if a double wide character is found in the first character location, the right half of the character should be displayed. The disadvantage of this technique is that the MPU will have to update the double wide preset bit on each character row in real time if the first half of some characters are under the second half of others.

The second technique is to write the same character and attributes into the second character location, but with the double wide attribute off. As the screen is scrolled so that only the right half of the character should be displayed, the character will default to a normal width character. The information on the screen is preserved, although the format is not. This technique has the advantage that no real time software is required.

There are no restrictions on the spacing relationships between two or more double wide characters. All combinations work as long as the double wide preset bit is handled properly.

7.2.4.8 Color/Resolution

Some redefinable characters in list modes 3 and 4 allow trading some resolution for an increased color range. This applies only to 1 and 2 bits per pel resolution. When the bit is set, the RMS uses the image table pel data from two pels to get one color for both pels. For example, with the bit reset in 1 bit per pel resolution, each image table byte defines 8 pels in two colors. With the bit set, the first two bits now pick one of four colors for the first two pels, and the next two bits pick another of the four colors for the second pair of pels. The byte always provides color information for 8 pels. In 2 bit per pel resolution, each byte selects four colors for each of four pels unless the COL/RES flag is set. Then the first two pels are colored by the byte's first 4 bits, providing 16 color choices. The second pair of pels are similarly colored using the byte's second nibble.

For example, in list mode 4 at 2 bits per pel, a DRC's first line of video data in the image table is:

```
     first byte:    1 Ø 1 Ø 1 1 Ø 1
     second byte:   Ø Ø Ø 1 Ø 1 Ø 1
```

The colors displayed when the color/resolution attribute is not selected are:

```
     CMR:             2     2     3     1     Ø     1     1     1
                   ------------------------------------------------
     pel:         | 1 Ø | 1 Ø | 1 1 | Ø 1 | Ø Ø | Ø 1 | Ø 1 | Ø 1 |
                   ------------------------------------------------
```

With color/resolution, the colors are:

```
     CMR:                A           D           1           5
                   ------------------------------------------------
     pel:         |  1 Ø 1 Ø  |  1 1 Ø 1  |  Ø Ø Ø 1  |  Ø 1 Ø 1  |
                   ------------------------------------------------
```

In both cases, 8 pixels are displayed, but with this attribute selected, pixels are grouped into pairs to form the pels.

See Section 7.2.4.15 for attribute interaction.

7.2.4.9 Separation

The separation attribute is only used on mosaic characters. It changes the format of the block so that some background color shows through, even if all of the blocks are on. It is active low.

The separation attribute leaves a one pel border on the bottom and right side of each block. This area is always displayed in background color. See Figure 7-4.

7.2.4.1Ø Mosaic 4 and 6

The mosaic characters in list modes 4 and 5 can display 4 or 6 blocks as described in Section 7.2.2. Mosaic 4 is active high.

7.2.4.11 Priority

Priority is used to simulate a third dimension in the display. It is used to determine which item is in front (visible) when two objects occupy the same place on the screen. The higher the value of the priority attribute, the closer the character is considered to be to the front of the screen.

None of the ROM-based or redefinable characters described so far can share display space with any of the others, since the display list reserves a separate part of the screen for each one. Priority becomes useful only with true objects, which can be displayed anywhere on the screen independent of the display list. True objects are the subject of Chapter 8.

The fixed objects of list modes 1, 3, and 4 can have any one of 8 priorities. This is the primary difference between fixed objects and all the ROM-based and redefinable characters, which all have the lowest priority.

The fixed object priorities interleave with the 8 true object priorities. When a true object is placed at the same location as a fixed object with the same priority, the fixed object is visible. As a result, a true object with priority Ø will appear in front of all alphanumerics and DRCs but will be behind fixed objects with priority Ø and higher. See Section 8.6 for a table of priorities.

Actually, the attribute is more flexible than this. Each fixed object can have two priorities, as described for the color collision attribute, Section 7.2.4.13.

7.2.4.12 Collision Enable

Collision is another attribute of fixed objects to make them easier to use with true objects. Collision is a one-bit code. A true object with collision enabled can collide with fixed objects that also have their collision enable bit set. That is, when a true object is moved to a fixed object's XY location on the screen, a collision is reported only if they both have their collision enables set.

Collisions are reported to the MPU on a true object basis. Each true object has one bit to report collisions with any fixed objects, and other bits to report collsions with other true objects.

Collision and priority are entirely separate attributes. Collisions occur depending only on the overlap of the two objects in the screen and on both objects having collision enabled. Priorities are ignored for collision reporting.

7.2.4.13 Color Collision

Color collision is used with fixed objects to determine which part of the fixed object is allowed to cause a collision. It is a two-bit code. The value of the fixed object's pel data, before CMR offset is combined, must be greater than the color collision value for a collision to be detected. Some attributes can affect the pel value used for the color collision test. See Section 7.2.4.15 for a discussion of attribute interaction.

This same test is used to determine priority. If the pel data is greater than the color collision value, then that pel's priority is determined by the character's priority attribute. If it is less than or equal to the color collision code, then its priority is the same as alphanumerics.

Color collision example:

> A game using list mode 4 has a scene with blue sky, white clouds, and a green and brown mountain. True objects with priority Ø through 3 must move behind the mountain. The rest of the true objects must move in front of the mountain; true object 4 can hit the mountain and needs collision reporting. A fixed object that contains parts of all four elements (cloud, sky, 2 colors of mountain) needs to be created using 4 bits per pel.
>
> This attribute (and shading, in Section 7.2.4.14) depends on the order of the colors in the CMR. The sky and cloud have the lowest priority, since they are behind everything, and the mountain colors are the highest. Assign a pel value of Ø to blue, 1 to white, 2 to brown, and 4 to green, and build the fixed object in its image table. Now put the character number in the display list, along with its attributes. Set color collision to 2, priority to 3, and set the collision enable bit; clear the rest of the attributes, and put the colors in the CMR: blue in CMRØØ, white in CMRØ1, brown in CMRØ2, and green in CMRØ4 (CMRØ3 was skipped, to show that they do not have to be contiguous). White and blue values are less than the color collision value, so their priority is Ø, and all true objects can appear in front of the cloud and sky. The green and brown pel values are the same

as or greater than the color collision, so they take the priority attribute, which was set to 3. True objects Ø through 3 will appear "behind" the mountain, and objects 4 through 7 will appear in front of it. If true object 4 has its collision enabled, collisions will be reported between it and the brown and green colors, but not the blue and white; the color collision test prevents them.

If the CMR assignments cause interference with other colors, they can be changed in three ways: first, the CMR offset can be used to move into higher CMR. Setting the character's CMR offset attribute to 11ØØ (binary) and setting CMR18 to blue, CMR19 to white, CMR1A to brown, and CMR1C to green, will cause exactly the same screen performance, because it is the pel value, not the CMR address, that is checked for color collision. Second, the green and brown pel values could be set as high as $ØF, since their only restriction is to be at least as large as the color collision attributes value. Third, the color collision could be increased to 3, and the blue and white could be set to a pel value of Ø, 1, or 2. All three methods can be combined.

7.2.4.14 Shading

Shading is a fixed object attribute in list mdes 3 and 4. It has no effect on the appearance of the fixed object, but it changes the colors of the true objects that are put in front of it. When a pel of a true object occupies the same screen coordinates as pel of a shading fixed object, and when the color collision test (see the previous section) has been passed, the most significant bit of the CMR address is inverted for the true object's pel. This requires coordination of two color sets in the CMR: one set colors the true object when it is not shaded, and the other colors it when it is. True object color selection is discussed in Chapter 8, but briefly, a true object can use any (or all) of the 24 CMR locations whose last two binary bits of address are nonzero. For instance, CMRØØ and CMR1C cannot be used, but CMRØ1 and CMR1E can be. See Section 8.3. If a true object used CMRØ1, CMR12, and CMR1F when not shaded, the shaded CMR locations would be CMR11, CMRØ2, and CMRØF. There is, of course, no limit on the color choices for shade.

To continue the example from the previous section, the white clouds block light directly below them, so true objects moving in the blue sky below the clouds must be darkened in a pattern matching the cloud. All blue sky must be the same shade. The color collision test uses the fixed object's pel value to determine whether it is background and nonshading or a color with a priority and shading; if the sky is to be all one color, two CMR locations must contain the same blue value. One (in CMRØØ, Ø1, or Ø2, if color collision is set to 3) is the background blue, and the other is the foreground/shading blue, which can be in any CMR location at least as large as the color collision value (CMRØ3 and higher if color collision is 3). None of this depends at all on the true object's colors or CMR locations. The unshading area is always background, and all true objects appear in front of it, but the shading areas have priority, so they can be in front of some true objects. If this is not desired, set the priority to zero and do not use fixed object Ø in that part of the screen.

See Section 7.2.4.15 for attribute interaction.

7.2.4.15 Attribute Interaction

Some attributes are affected by the selected values for other attributes. The attributes are applied one at a time, so the order of application controls the interaction. The order is:

Ø) pel value (the input to the attribute logic)

1) Color/Resolution (redefinable characters only)

2) Underline (1Øth scanline of character row only)

3) Flash

4) Invert

TEST) Color Collision, Shading

5) CMR Offset, Foreground-Background

Every list mode pel goes through this sequence in this order. If the attribute is not available or not selected, the pel value is not affected by that step, so the next step gets the same pel value.

The zeroth step is to select the initial pel value. For alphanumerics and mosaics, this is a 1-bit value from internal ROM. For redefinable characters, it is fetched from an image table in DRAM, and can be 1, 2, or 4 bits long. The particular pel fetched depends not only on the position within the character, but also whether the double high and double wide attributes are selected, and also (if the character is a mosaic), whether the separation or mosaic 4 attributes are selected.

The first attribute step is color/resolution. If it is selected, the 1-bit or 2-bit pel values are combined to form 2-bit or 4-bit values. Color/resolution cannot be used with 4 bit per pel data. Following this step, the pel value is treated as a 4-bit value, with leading zeroes added to the 1 or 2-bit data.

The second step is underline. If this is the tenth scanline of the character row, and if underline is selected, the pel value is set to binary 1111.

The third step is flash. Flash causes the character to disappear and reappear. If flash is not available or selected, or if the character is to appear normally at this stage of the flash cycle, the pel value is unchanged. If this is the disappearing stage of the flash cycle, the pel value is set to binary ØØØØ.

The fourth step is invert. Invert does a 1's complement on the pel data, but only on the correct number of bits per pel. The bits per pel value in the Display Data Mode register is used to fetch image table data, but it may not be the value used with invert. When the pel data is from the internal ROM or an alphanumeric-type DRC, the invert bits per pel value is 1. If it is a graphics DRC or a fixed object, the bits per pel value is the value from the Display Data Mode register. In either case, if color/resolution is available and color is selected, the bits per pel value for invert is doubled.

If the pel data is for a fixed object, the value at this point is used for color collision and shading tests. This has no effect on the pel's appearance.

The last step in pel value alteration depends on the character type. If the character is a fixed object or a graphics-type DRC, the 4-bit pel data is inclusive OR'ed with the 5-bit CMR offset. If the character is an alphanumeric, mosaic, or alpha-type DRC, the least significant bit of the 4-bit pel data is used to select the foreground color (if a 1) or the background color (if a 0).

Example 1

A 4 bit per pel (bpp) fixed object with binary pel value 0001, invert selected, and a CMR offset of 10000:

pel value	0001	input value to attribute logic
Color/Res	0001	no change
Underline	0001	no change
Flash	0001	no change
Invert	1110	4-bit invert because 4 bpp data from image table
CMR offset	11110	10000 OR'ed with 1110
CMR register:	1E	this pel gets the color in CMR1E

Example 2

A 1 bpp DRC with pel values of 1 and 0 (the 2 adjacent pels), with color/resolution and invert selected, and a CMR offset of 01000:

pel value	1,0	two pels to be combined
Color/Res	0010	two pels combined, leading zeroes added
Underline	0010	no change
Flash	0010	no change
Invert	0001	2-bit invert (1 bpp input plus Color/Res doubling)
CMR offset	01001	01000 OR'ed with 0001
CMR register:	09	this pel gets the color in CMR09

Example 3

An alphanumeric background pel in the 10th scanline of its character row with underline, flash, and invert selected. It is in the disappearing part of the flash cycle:

pel value	0	input value to attribute logic
Color/Res	0000	no change, leading zeroes added
Underline	1111	converts to $F, regardless of input pel value
Flash	0000	converts to $0, regardless of input pel value
Invert	0001	1-bit invert because 1 bpp data from alphanumeric

The LSB of the final pel value is 1, so the pel is displayed in foreground color.

7.2.5 List Mode Displays

Six list modes are available in the RMS. The user selects the active mode in the Display Data Mode register (see Section 9.3.2). They differ in the number of bytes they use to represent one character, the types of characters they allow, and the types of attributes the characters may have. All bits are active high except mosaic separation.

7.2.5.1 List Mode 0

List mode 0 uses one byte per character. It is a simple list mode that offers alphanumerics, mosaics, and a few DRC's in the low horizontal resolution modes (HRES2-4). Very little is available in the way of attributes. This mode's advantage is that it uses a very small amount of memory to display a screen. The characters are:

Bit	Alphanumeric	DRC	Mosaic
7	Always 0	Always 0	Always 1
6	Char Code 6 (MSB)	Always 0	Char Code 6 (MSB)
5	Char Code 5	Always 0	Separation
4	Char Code 4	Char Code 4 (MSB)	Char Code 4
3	Char Code 3	Char Code 3	Char Code 3
2	Char Code 2	Char Code 2	Char Code 2
1	Char Code 1	Char Code 1	Char Code 1
0	Char Code 0 (LSB)	Char Code 0 (LSB)	Char Code 0 (LSB)

Table 7-5 List Mode 0 Characters

Note that some bits are always 1 or always 0. This is how the RMS distinguishes one character type from another.

There are 7 character code bits to define an alphanumeric character, but only the upper 96 of the 128 possible codes are valid. The RMS uses ASCII coding for alphanumerics. The 96 normal alphanumerics are displayed, and the 32 lowest ASCII codes, which are used as control characters, are not printable.

The 32 lowest character numbers are used for the Dynamically Redefinable Character Set in mode Ø. The image table must contain the number of bits per pel set in the Display Data Mode register, but only the LSB of each pel value is used. For all three character types, the foreground color is CMRØF, and the background color is CMRØØ.

The mosaic characters displayed in list mode Ø are always mosaic 6. The separation attribute is available for mosaics; see Section 7.2.4.9. For more information on mosaic characters see Section 7.2.2.

7.2.5.2 List Mode 1

List mode 1 is a two bytes per character mode that is oriented towards video games in the low horizontal resolution modes (HRES2-4). It does not offer as much performance as the other games modes, but it can be used in a system with less memory, and therefore lower cost.

Mode 1 allows alphanumeric, DRC, mosaic 6, and fixed objects as characters. Characters are coded as shown below.

	Bit	Alphanumeric	DRC	Mosaic
Byte 1	7	Always Ø	Always Ø	Always 1
	6	Char Code 6 (MSB)	Always Ø	Char Code 6 (MSB)
	5	Char Code 5	Always Ø	Separation
	4	Char Code 4	Char Code 4(MSB)	Char Code 4
	3	Char Code 3	Char Code 3	Char Code 3
	2	Char Code 2	Char Code 2	Char Code 2
	1	Char Code 1	Char Code 1	Char Code 1
	Ø	Char code Ø (LSB)	Char Code Ø (LSB)	Char Code Ø (LSB)
Byte 2	7	Always Ø	Always Ø	Always 1
	6	Flash 1	Flash 1	Flash 1
	5	Foregnd 2 (MSB)	Foregnd 2 (MSB)	Foregnd 2 (MSB)
	4	Foregnd 1	Foregnd 1	Foregnd 1
	3	Foregnd Ø (LSB)	Foregnd Ø (LSB)	Foregnd Ø (LSB)
	2	Backgnd 2 (MSB)	Backgnd 2 (MSB)	Backgnd 2 (MSB)
	1	Backgnd 1	Backgnd 1	Backgnd 1
	Ø	Backgnd Ø (LSB)	Backgnd Ø (LSB)	Backgnd Ø (LSB)

Table 7-6 List Mode 1 Alphanumerics, DRCS, and Mosaics

	Bit	Fixed A	Fixed B
	7	Always 1	Always Ø
	6	CMR Offset 3	CMR Offset 3
	5	CMR Offset 2	CMR Offset 2
Byte 1	4	CMR Offset 1	CMR Offset 1
	3	Color Collision Ø (LSB)	Color Collision Ø (LSB)
	2	Priority 1	Priority 1
	1	Priority Ø (LSB)	Priority Ø (LSB)
	Ø	Collision Enable	Collision Enable
	7	Always Ø	Always 1
	6	Flash 1	Flash 1
	5	Char Code 5	Char Code 5
Byte 2	4	Char Code 4	Char Code 4
	3	Char Code 3	Char Code 3
	2	Char Code 2	Char Code 2
	1	Char Code 1	Char Code 1
	Ø	Char Code Ø (LSB)	Char Code Ø (LSB)

Table 7-7 List Mode 1 Fixed Objects

The bits that are always Ø or 1 are used by RMC to determine what type of character is in use.

DRC characters are located in the 32 lowest character codes of alphanumeric characters. The DRCS image table must contain the number of bits per pel set in the Display Data Mode register, but only the LSB is used. It selects between foreground color (if a 1) and background color (if a Ø).

Foregnd stands for foreground color. Backgnd stands for background color.

Fixed A and B are both fixed objects. The difference between them is that the most significant bits of the two bytes are used to complete the specification of some of the attributes. Bit 7 of the first byte is the most significant CMR offset bit (CMR 4), color collision bit (Color Collision 1), and priority bit (Priority 2). Bit 7 of the second byte is the most signficant bit of the character code (Char code 6).

Fixed Object Type	A	B
Character code	Ø-63	64-127
CMR offset	8-F	Ø-7
Color collision	2-3	Ø-1
Priority	4-7	Ø-3

Table 7-8 Ranges of Values for Fixed Objects

Only 2 Hz flash is available.

Fixed A and B support 64 characters each. The total number of characters available in mode 1 is 128 fixed objects, 96 alphanumerics, 64 mosaics and 32 DRCs.

7.2.5.3 List Mode 2

List mode 2 is designed for use in simple word processing systems that need to use both redefinable and ASCII characters on the screen at once. It must be used with one of the high resolution modes (HRES modes 6 or 7). It displays either two 8-pel-wide ASCII characters, or one 16-pel-wide redefinable character during each memory cycle.

List mode 2 has very little in the way of attributes. It is designed for simple text displays and can also do simple graphics.

	Bit	Alphanumeric	Mosaic
Byte 1	7	Always Ø	Always 1
	6	Char Code 6-1 (MSB)	Char Code 6-1 (MSB)
	5	Char Code 5-1	Separation 1
	4	Char Code 4-1	Char Code 4-1
	3	Char Code 3-1	Char Code 3-1
	2	Char Code 2-1	Char Code 2-1
	1	Char Code 1-1	Char Code 1-1
	Ø	Char Code Ø-1 (LSB)	Char Code Ø-1 (LSB)
Byte 2	7	Always Ø	Always 1
	6	Char Code 6-2 (MSB)	Char Code 6-2 (MSB)
	5	Char Code 5-2	Separation 2
	4	Char Code 4-2	Char Code 4-2
	3	Char Code 3-2	Char Code 3-2
	2	Char Code 2-2	Char Code 2-2
	1	Char Code 1-2	Char Code 1-2
	Ø	Char Code Ø-2 (LSB)	Char Code Ø-2 (LSB)

Table 7-9 List Mode 2 Alphanumerics and Mosaics

	Bit	Fixed	DRC
Byte 1	7	Always 1	Always Ø
	6	Char Code 13 (MSB)	Char Code 13 (MSB)
	5	Char Code 12	Char Code 12
	4	Char Code 11	Char Code 11
	3	Char Code 1Ø	Char Code 1Ø
	2	Char Code 9	Char Code 9
	1	Char Code 8	Char Code 8
	Ø	Char Code 7	Char code 7
Byte 2	7	Always Ø	Always 1
	6	Char Code 6	Char Code 6
	5	Char Code 5	Char Code 5
	4	Char Code 4	Char Code 4
	3	Char Code 3	Char Code 3
	2	Char Code 2	Char Code 2
	1	Char Code 1	Char Code 1
	Ø	Char Code Ø (LSB)	Char Code Ø (LSB)

Table 7-1Ø List Mode 2 DRCS and Fixed Objects

The bits that are shown as always 1 or Ø are used by the RMC hardware to determine the type of character to display.

The screen is divided up into 16-pel-wide blocks. The RMS can display one 16-pel-wide DRC or fixed object, or two alphanumeric or mosaic characters in each of these spaces. It is not possible to put an alphanumeric and a mosaic in the same block.

The alphanumeric characters are the standard ASCII set available from the internal ROM. Small (8-wide) DRC's are not usable in this mode. Therefore there are 32 alphanumeric codes ($ØØ through $1F) that should not be used.

The character codes for both of these characters are 14 bits each. The bits are arranged in such a way as to make it convenient to implement Japanese Industrial Standard C 6226.

7.2.5.4 List Mode 3

List mode 3 is designed to display DRC's and fixed objects. It is a games-oriented mode that uses horizontal resolution modes 6 and 7.

	BIT	DRC	Fixed
	7	Always Ø	Always 1
	6	Char Code 13 (MSB)	Color Collision 1 (MSB)
	5	Char Code 12	Color Collision Ø (LSB)
Byte 1	4	Char Code 11	Priority 2 (MSB)
	3	Char Code 1Ø	Priority 1
	2	Char Code 9	Priority Ø (LSB)
	1	Char Code 8	Char Code 8 (MSB)
	Ø	Char Code 7	Char Code 7
	7	Always 1	Always Ø
	6	Char Code 6	Char Code 6
	5	Char Code 5	Char Code 5
Byte 2	4	Char Code 4	Char Code 4
	3	Char Code 3	Char Code 3
	2	Char Code 2	Char Code 2
	1	Char Code 1	Char Code 1
	Ø	Char Code Ø (LSB)	Char Code Ø (LSB)
	7	2xW	Collision Enable
	6	CMR Offset 4 (MSB)	CMR Offset 4 (MSB)
	5	CMR Offset 3	CMR Offset 3
Byte 3	4	CMR Offset 2	CMR Offset 2
	3	CMR Offset 1	CMR Offset 1
	2	2xH	Shading
	1	Flash 1 (MSB)	Flash 1
	Ø	Flash Ø (LSB)	Color/Resolution

Table 7-11 List Mode 3 Characters

Mode 3 supports 16,384 DRC's and 512 fixed objects.

7.2.5.5 List Mode 4

List mode 4 is a three byte per character mode that mixes several types of characters and provides all the RMS' attributes that apply to games. It uses the low horizontal resolution modes (HRES2-4).

	Bit	Alphanumeric	Mosaic
	7	Always Ø	Always 1
	6	Char Code 6 (MSB)	Char Code 6 (MSB)
	5	Char Code 5	Separation
Byte 1	4	Char Code 4	Char Code 4
	3	Char Code 3	Char Code 3
	2	Char Code 2	Char Code 2
	1	Char Code 1	Char Code 1
	Ø	Char Code Ø (LSB)	Char Code Ø (LSB)
	7	Always Ø	Always 1
	6	2xW	2xW
	5	Spare	Spare
Byte 2	4	Invert	Invert
	3	Foregnd 3 (MSB)	Foregnd 3 (MSB)
	2	Foregnd 2	Foregnd 2
	1	Foregnd 1	Foregnd 1
	Ø	Foregnd Ø (LSB)	Foregnd Ø (LSB)
	7	Underline	Mosaic 4
	6	Backgnd 3 (MSB)	Backgnd 3 (MSB)
	5	Backgnd 2	Backgnd 2
Byte 3	4	Backgnd 1	Backgnd 1
	3	Backgnd Ø (LSB)	Backgnd Ø (LSB)
	2	2xH	2xH
	1	Flash 1 (MSB)	Flash 1 (MSB)
	Ø	Flash Ø (LSB)	Flash Ø (LSB)

Table 7-12 List Mode 4 Alphanumerics and Mosaics

	Bit	DRC (Alphanumeric)	Fixed	DRC
	7	Always Ø	Always 1	Always Ø
	6	Always Ø	Color Collision 1 (MSB)	Color/Resolution
	5	Always Ø	Color Collision Ø (LSB)	Invert
Byte 1	4	Char Code 4 (MSB)	Priority 2 (MSB)	2xW
	3	Char Code 3	Priority 1	Spare
	2	Char Code 2	Priority Ø (LSB)	Spare
	1	Char Code 1	Char Code 8 (MSB)	Char Code 8 (MSB)
	Ø	Char Code Ø (LSB)	Char Code 7	Char Code 7
	7	Always Ø	Always Ø	Always 1
	6	2xW	Char Code 6	Char Code 6
	5	Spare	Char Code 5	Char Code 5
Byte 2	4	Invert	Char Code 4	Char Code 4
	3	Foregnd 3 (MSB)	Char Code 3	Char Code 3
	2	Foregnd 2	Char Code 2	Char Code 2
	1	Foregnd 1	Char Code 1	Char Code 1
	Ø	Foregnd Ø (LSB)	Char Code Ø (MSB)	Char Code Ø (LSB)
	7	Underline	Collision Enable	Underline
	6	Backgnd 3 (MSB)	CMR Offset 4 (MSB)	CMR Offset 4 (MSB)
	5	Backgnd 2	CMR Offset 3	CMR Offset 3
Byte 3	4	Backgnd 1	CMR Offset 2	CMR Offset 2
	3	Backgnd Ø (LSB)	CMR Offset 1	CMR Offset 1
	2	2xH	Shading	2xH
	1	Flash 1 (MSB)	Flash 1 (MSB)	Flash 1 (MSB)
	Ø	Flash Ø (LSB)	Flash Ø (LSB)	Flash Ø (LSB)

Table 7-13 List Mode 4 DRCS and Fixed Objects

The first 32 of the 512 DRC's that the DRCS start address register points to can be used either as the $ØØ-$1F alphanumerics or as the standard DRC's. When used as alphanumerics, the DRC's use only the LSB of the pel values in the image table to select between foreground (if a 1 after attributes are applied) and background (if a Ø after attributes). When used as graphics characters, the DRC's use the full image table pel values.

7.2.5.6 List Mode 5

List mode 5 is designed for high resolution text applications, primarily word processing. It also has sufficient graphics capability to be useful in other applications. It is a three byte per memory cycle mode that is designed to be used with HRES modes 6 and 7. The three bytes are used to generate 2 alphanumeric or mosaic characters or one DRC or fixed object.

Alphanumerics and mosaics are displayed in pairs. They cannot be mixed with each other. The first byte contains the character code for the left character, the second byte identifies the second character, and the third byte contains the attributes for both characters. The least significant 4 bits (bits Ø-3) of this third byte contain the attributes for the first character, and the most significant bits (bits 4-7) are for the second character.

If redefinable characters are used, they are 16 pels wide and occupy the same amount of space as two alphanumerics or mosaics.

	Bit	Alphanumeric	Mosaic
	7	Always Ø	Always 1
	6	Char Code 6-1 (MSB)	Char Code 6-1 (MSB)
	5	Char Code 5-1	Separation 1
Byte 1	4	Char Code 4-1	Char Code 4-1
	3	Char Code 3-1	Char Code 3-1
	2	Char Code 2-1	Char Code 2-1
	1	Char Code 1-1	Char Code 1-1
	Ø	Char Code Ø-1 (LSB)	Char Code Ø-1 (LSB)
	7	Always Ø	Always 1
	6	Char Code 6-2 (MSB)	Char Code 6-2 (MSB)
	5	Char Code 5-2	Separation 2
Byte 2	4	Char Code 4-2	Char Code 4-2
	3	Char Code 3-2	Char Code 3-2
	2	Char Code 2-2	Char Code 2-2
	1	Char Code 1-2	Char Code 1-2
	Ø	Char Code Ø-2 (LSB)	Char Code Ø-2 (LSB)
	7	Underline (Char 2)	Mosaic 4 (Char 2)
	6	Invert	Invert
	5	Flash 1	Flash 1
Byte 3	4	CMR Offset 4 (MSB)	CMR Offset 4 (MSB)
	3	Underline (Char 1)	Mosaic 4 (Char 1)
	2	Invert	Invert
	1	Flash 1	Flash 1
	Ø	CMR Offset 4 (MSB)	CMR Offset 4 (MSB)

Table 7-14 List Mode 5 Alphanumerics and Mosaics

	Bit	DRC	Fixed
Byte 1	7	Always Ø	Always 1
	6	Char Code 13 (MSB)	Char Code 13 (MSB)
	5	Char Code 12	Char Code 12
	4	Char Code 11	Char Code 11
	3	Char Code 1Ø	Char Code 1Ø
	2	Char Code 9	Char Code 9
	1	Char Code 8	Char Code 8
	Ø	Char Code 7	Char Code 7
Byte 2	7	Always 1	Always Ø
	6	Char Code 6	Char Code 6
	5	Char Code 5	Char Code 5
	4	Char Code 4	Char Code 4
	3	Char Code 3	Char Code 3
	2	Char Code 2	Char Code 2
	1	Char Code 1	Char Code 1
	Ø	Char Code Ø (LSB)	Char Code Ø (LSB)
Byte 3	7	Underline	Underline
	6	Invert	Invert
	5	CMR Offset 3	CMR Offset 3
	4	CMR Offset 2	CMR Offset 2
	3	CMR Offset 1	CMR Offset 1
	2	2xH	2xH
	1	Flash 1 (MSB)	Flash 1 (MSB)
	Ø	Flash Ø (LSB)	Flash Ø (LSB)

Table 7-15 List Mode 5 DRCS and Fixed Objects

7.2.5.7 List Mode Summary

The table shows which HRES modes may be used with each of the list modes.

List Mode	HRES Mode
Ø	2, 3, 4
1	2, 3, 4
2	6, 7
3	6, 7
4	2, 3, 4
5	6, 7

Table 7-16 List Modes and HRES Modes

Table 7-17 shows which characters are available in each list mode, and Table 7-18 shows what attributes are available for each character in each list mode.

				Alphas	Mosaics	DRC	DRC	Fixed	DRC	Fixed
List	HRES	Bytes	Chars _ Width (Pixels)	8	8		8	8	16	16
Mode	Modes	Cycle	Cycle Character	Wide	Wide	Alpha*	Wide	Wide	Wide	Wide
Ø	2, 3, 4	1	1-8	96	64	32				
1	2, 3, 4	2	1-8	96	64	32		128		
2	6, 7	2	2-8, 1-16	96	64				16K	16K
3	6, 7	3	1-16						16K	512
4	2, 3, 4	3	1-8	96	16, 64	32	512	512		
5	6, 7	3	2-8, 1-16	96	16, 64				16K	16K

Table 7-17 Character/Object Availability By List Mode

	Alphanumerics					Mosaics					DRC's							Fixed Objects				
List Mode:	Ø	1	2	4	5	Ø	1	2	4	5	Ø	1	2	3	4A	4	5	1	2	3	4	5
Attributes																						
Character Codes	96	96	96	96	96	64	64	64	16 64	16 64	32	32	16K	16K	32	512	16K	128	16K	512	512	16K
Flash Rates		1		3	1		1		3	1		1		3	3	3	3	1		1	3	3
Foregnd & Backgrnd Colors		8		16			8		16			8			16							
CMR Offset					2					2				16		16	8	16		16	16	8
Priority																		8		8	8	
Color Collision																		4		4	4	

Table 7-18a Attributes by Character and List Mode

	Alphanumerics					Mosaics					DRC's							Fixed Objects				
List Mode:	Ø	1	2	4	5	Ø	1	2	4	5	Ø	1	2	3	4A	4	5	1	2	3	4	5
Attributes																						
Invert				x	x				x	x					x	x	x					x
Underline				x	x										x	x	x					x
2xH				x					x					x	x	x	x					x
2xW				x					x					x	x	x						
Separation						x	x	x	x	x												
Color/Res																x				x		
Shading																				x	x	
Collision Enable																		x		x	x	
Mosaic 4/6									x	x												

Table 7-18b Attributes by Character and List Mode

7.2.5.8 Display List

The display list is defined by three registers. The Virtual Screen Start Address register (Section 9.3.2Ø) identifies the display list's beginning. The Virtual Screen Size register (Section 9.3.23) sets the number of bytes in the entire display list. The Virtual Screen Width register (Section 9.3.24) contains the number of bytes in one scanline (bit-plane) or character row (list mode).

The display list must be stored in DRAM by the MPU. It is retrieved from DRAM automatically by RMS, as it is needed. The MPU must store the data sequentially in DRAM. The first byte of the first character must be stored at the address contained in the Virtual Screen Start Address register. It must be followed by the second and third bytes of the first character (if applicable), and then the first byte of the second character. There cannot be any gaps in the display list.

The Virtual Screen Start Address, Horizontal Offset, and Vertical Offset registers combined must point to the first byte of a character, never the second or third byte. When the programmer is using scrolling, the offset registers must be incremented by a value that advances to the next character, which may or may not be the next byte. See Sections 6.5 and 12.4.

8.Ø TRUE OBJECTS

True objects are RMS hardware-intensive objects that are designed to move around on the screen and to run into other objects. Eight identical sets of registers allow simultaneous independent operations, but objects may be reused in real time, so more than 8 can appear on the screen at one time.

True objects are described in an image table similar to the list mode character tables, and each object is positioned on the screen using its own X and Y Coordinate registers.

Each true object also has a register to report to the MPU that it overlaps other true objects and fixed objects.

The true object hardware is independent of display mode, so true objects can be used in bit-plane mode and all 6 list modes. Other than the true object registers, the parts of the RMS that can affect the way true objects look are the CMR colors, the horizontal and vertical resolution, and some fixed object attributes (when the true object overlaps the fixed object on the screen).

Each true object can be individually enabled or disabled by a bit in its position registers.

8.1 Object Position

The position on the screen of the upper lefthand corner of the object is defined by X and Y coordinates. The XY grid is larger than the visible screen, so it is possible to position an object off the screen in all four directions. A discussion of onscreen positioning follows the X and Y discussions.

Since true objects are located using screen coordinates, scrolling the display data has no effect on true objects, but it is possible to move the true objects while scrolling, either to maintain the relationship between them or to create more complex movements.

The X coordinates are generated by a counter that runs at the picture element clock (PCLK) rate. This counter begins counting from Ø during horizontal blanking, counts through the left screen border, then the visible screen, then the right screen border. The value of X that represents the first pixel on the left side of the screen varies with HRES mode. These values are listed below, along with the value of the last pixel inside the border on the right edge of the screen, and the value of the last count used. The X counter is 1Ø bits long, which would allow it to reach a count of 1Ø23. In practice it does not reach this count, because the next line begins before it can get to this value. The last count listed below is the maximum count it reaches in a given mode. If the object's X coordinate is set larger than the maximum X, it makes the object inactive. The object is not displayed on the screen and cannot have a collision with any other part of the display.

HRES Mode	Left Edge	Right Edge	Last Count
Ø	53	3Ø8	367
1	53	3Ø8	367
2	53	3Ø8	367
3	13	268	295
4	45	364	423
5	Reserved		
6	29	54Ø	591
7	29	668	735

Table 8-1 X Coordinates and HRES Modes

The Y coordinate counter is the counter used for vertical timing signal generation. It is a ten-bit counter that counts at the horizontal line rate. It begins counting from Ø on the video line immediately following the trailing edge of vertical sync. The value of Y that corresponds to the first line inside the border varies with VRES mode and the choice of 525 or 625 line timing. The values of the last line inside the border and the last count also vary. The use of both interlace sync and data doubles the values. Y coordinates are shown for the available combinations in the following table.

Timing	Interlace Data	VRES Mode	First Line	Last Line	Max Count
525	NO	0	39	230	262
525	NO	1	39	230	262
525	NO	2	39	230	262
525	NO	3	35	234	262
525	NO	4	30	239	262
525	NO	5	Not Applicable		
525	NO	6	Not Applicable		
525	YES	0	78	460	525
525	YES	1	78	460	525
525	YES	2	78	460	525
525	YES	3	70	468	525
525	YES	4	60	478	525
525	YES	5	Not Applicable		
525	YES	6	Not Applicable		
625	NO	0	69	260	312
625	NO	1	69	260	312
625	NO	2	69	260	312
625	NO	3	65	264	312
625	NO	4	60	269	312
625	NO	5	45	284	312
625	NO	6	40	289	312
625	YES	0	138	520	625
625	YES	1	138	520	625
625	YES	2	138	520	625
625	YES	3	130	528	625
625	YES	4	120	538	625
625	YES	5	90	568	625
625	YES	6	80	578	625

Table 8-2 Y Coordinates and VRES Modes

The user has separate registers for the X and Y coordinates of each object. They are all available in the RMS memory map, and the user may update these registers at any time. If the RMS is displaying or preparing to display that object, the display in that video frame may be disturbed. See Sections 9.3.3 and 9.3.5 for ways to coordinate changes with the RMS' display process.

Active true objects can collide with each other even when they are partly or completely off screen, but fixed objects cannot cause collision reporting unless they are onscreen.

Since the XY position locates the upper left corner of the true object, the entire object is off screen if either the X or the Y coordinate is larger than the maximum on screen value, but it may be only partly off screen for X and Y values smaller than the on-screen values. It may not be possible to move long objects or zoomed objects (discussed later) completely offscreen at the top or left edge.

8.2 Names

The object logic knows what pattern to generate because the user programs it with an object name (see Section 9.3.27). The name is used in the same way as a character code. The name is 8 bits long, so the user can define 256 different object patterns. Each name is allocated 128 bytes of DRAM for pattern storage. The True Object Image Table Start Address register must be set to the address of the first byte of object $00 (see Section 9.3.11). The patterns for the other objects must follow in numerical order.

Since the user can change objects quickly by changing the name byte, and it is possible to have patterns defined for a large number of objects, it is easy to create an animation effect by having several different views of the same object and changing between them.

8.3 Pattern Data

Each true object can be described in one of two ways: bit-plane encoding or run-length encoding. Bit-plane encoding has 14 individually colored pels per video line, and run-length encoding has seven colors. However, each run-length color can be 1, 3, 5, or 7 pels long. Run-length has more flexibility in sizes; bit-plane has more flexibility in colors. In either case, the first 4-byte packet describes the top video line, and the second packet the line under it; similarly, the first pel description in a line describes the leftmost pel, and the last pel is the rightmost pel.

The choice between run-length and bit-plane is made separately for each object using a bit in the object's X Coordinate register (see Section 9.3.26).

Each 4-byte packet describes one video line. The more significant nibble of the first byte of each packet contains an active low last-line bit and a 3-bit CMR offset:

B7	B6	B5	B4	B3	B2	B1	B0
LL	CMR4	CMR3	CMR2	(Beginning of Data for Video Line)			

If the last-line bit is reset, the rest of the line is ignored (see Section 8.4). If it is set, the 3-bit offset is used for the most significant bits of the CMR address. The meaning of the following 3-1/2 bytes is different for the two types of encoding, but they both use 2 bits for each color selection, so they have 4 choices. A unique feature of true objects is that the color choice of 00 makes the pels transparent, so that whatever it passes over can show through. Transparent pels cannot collide with anything, so the user can define functional as well as visible shapes by putting transparent pels around a small visible object. See Section 8.7. The other three color choices (01, 10, 11) use the 3-bit CMR offset in the video line data's first nibble to get the color's CMR address. Each line can have 3 visible colors, and, because each line has its own CMR offset, each object can have 24 colors. The eight CMR addresses ending in ØØ are replaced by transparency, so they are not available for true objects.

For example, if the CMR offset bits were 1Ø1, a color of Ø1 selects CMR15 (1Ø1Ø1); if the offset were Ø1Ø, a color of 11 selects CMRØB (Ø1Ø11). If the CMR offset were 11Ø, that video line could display the colors in CMR19, CMR1A, and CMR1B.
It is possible to use less than 128 bytes to define the entire object. This does not save memory space; all 128 bytes are still allocated to the object. What it does is free the object hardware as soon as the visible part of the object is completed. Therefore, it can be put to use displaying a different object as soon as possible. See Section 8.4.

8.3.1 Run Length

In run-length encoding, each nibble describes a color segment. The most significant two bits select the length: a code of ØØ means 1 pel, Ø1 means 3 pels, 1Ø means 5 pels, and 11 means 7 pels. The least significant two bits of the nibble select the color as described in Section 8.3. A video line's 4-byte packet is run-length encoded as follows:

	B7	B6	B5	B4	B3	B2	B1	BØ
1st Byte	LL	CMR4	CMR3	CMR2	1R1	1RØ	1C1	1CØ
2nd Byte	2R1	2RØ	2C1	2CØ	3R1	3RØ	3C1	3CØ
3rd Byte	4R1	4RØ	4C1	4CØ	5R1	5RØ	5C1	5CØ
4th Byte	6R1	6RØ	6C1	6CØ	7R1	7RØ	7C1	7CØ

LL	means last line (active low)
CMRy	means the y bit of the CMR offset
xRy	means the y bit of the run length for pel x
xCy	means the y bit of the color for pel x

Table 8-3 Run Length Encoding

8.3.2 Bit-plane

In bit-plane encoding the 3-1/2 bytes of video data following the last-line bit and CMR offset are treated as 14 2-bit colors. Each 2-bit color describes one pel, so a bit-plane object is 14 pels wide. A bit-plane encoded video line packet is arranged as follows:

	B7	B6	B5	B4	B3	B2	B1	B0
1st BYTE	LL	CMR4	CMR3	CMR2	1P1	1P0	2P1	2P0
2nd BYTE	3P1	3P0	4P1	4P0	5P1	5P0	6P1	6P0
3rd BYTE	7P1	7P0	8P1	8P0	9P1	9P0	10P1	10P0
4th BYTE	11P1	11P0	12P1	12P0	13P1	13P0	14P1	14P0

LL means last line (active low)
CMRy means bit y of the CMR offset
xPy means the y bit of the color for pel x

Table 8-4 Bit-plane Encoding

8.4 Last-line Flag

Each entry in the true object image table must contain a 4-byte packet with the last-line flag reset. Systems using interlace data must have two such lines.

The RMS is designed so that, once the scan line is reached that matches the true object's Y coordinate, it starts fetching and displaying 4-byte packets until it finds a packet with its MSB clear or until vertical retrace. If interlaced data is being used, two independent versions of that process are performed. Each time a last-line packet is encountered, the object's object-available flag is set, so for interlaced data, it is set twice. If the last-line bit is set in all 32 of the 4-byte packets, the object is repeated, one version directly under the previous one, until vertical retrace.

It is possible to perform several real-time operations using the last-line flag: alternately setting and clearing this flag in the object's first line causes the object to flash. In interlaced data systems, the flashing could be made more complex by alternating between the even and the odd field data. Stepping the last-line flag down through the data causes the object to appear gradually.

It is also possible to use an object whose first line is transparent and whose second line's flag is reset as an invisible marker. This is a similar function to the real-time output flag; a true object could be used if RTO were already occupied. See Section 9.3.15 for a discussion of real-time output registers.

8.5 Zoom Factors

Each true object can be individually zoomed horizontally or vertically or both without changing DRAM. Two bits in each object's X position register and in its Y position register select the display size: 00 means normal size, 01 means 2X, 10 means 4X, and 11 means 8X. The origin for the expansion is the object's XY register value, which is the upper left corner of the object. As a result, horizontal expansion is to the right, and vertical expansion is down. The size change is accomplished by multiplying the size of each pel. See Section 9.3.26.

8.6 Priority

Priority refers to the apparent depth of one true object compared to another true object or a fixed object. When two objects occupy the same space on the screen, one of them is "in front of" the other and therefore visible. The second object disappears behind the first. The first object has a higher priority than the second.

The priority of true objects is defined in hardware and cannot be selected by the programmer. Object 7 has the highest priority, and object Ø has the lowest. A true object with a particular priority is hidden by all fixed objects with the same or higher priority. Alphanumerics and bit plane have a lower priority than the lowest priority object, and the border has the highest priority.

Priority is a separate attribute from collision enable. They do not affect each other in any way.

Highest Priority	Border
	Fixed object, priority 7
	True object number 7
	Fixed object, priority 6
	True object number 6
	Fixed object, priority 5
	True object number 5
	Fixed object, priority 4
	True object number 4
	Fixed object, priority 3
	True object number 3
	Fixed object, priority 2
	True object number 2
	Fixed object, priority 1
	True object number 1
	Fixed object, priority Ø
	True object number Ø
Lowest Priority	Alphanumerics, Mosaics, DRC's, and Bit-plane

Table 8-5 Display Priorities

8.7 Collision

The RMS control register map contains eight bytes of registers for collision reporting. There is one register for each true object.

The bits within a true object's collision register indicate which other true objects it has collided with since the last time the register was read. Bit 7 true would show that it had collided with true object number 7, bit 6 for true object 6, and so on.

The bit that indicates the register's own object (e.g., bit 5 of the register belonging to object 5) is used to indicate collisions with fixed objects.

Before an object can have a reportable collision, its collision enable bit and the collision enable bit of the true object or fixed object it collides with must both be set. Collisions are not reported if they involve objects that do not have their collision enable bits set.

If true objects 3 and 4 have their collision enable bits set and object 5's bit is reset, and if all three objects are moved so they overlap, the collision flags are set as follows. The collision-reporting register for object 3 will indicate a collision with object 4, but not with 5. The collision-reporting register of object 4 will indicate a collision with object 3, but not with 5. The collision-reporting register of object 5 will not show any collisions.

For a true object to have a collision with another true object, it must involve the solid part of the object. Transparent pels are not considered when checking for collisions.

For a true object to collide with a fixed object, the fixed object pel involved must pass the fixed object's color collision test. See Section 7.2.4.13.

Priority level is not involved in determining collisions. Two true objects, or a true object and fixed object, can collide regardless of their priority level, as long as they pass the collision enable, transparency, and color collision tests.

9.Ø CONTROL REGISTERS

This chapter discusses the two types of machine operation, the folded and unfolded memory maps, and the individual registers. An RMS register map is in Appendix A.

9.1 Machine Types

Machine 1 mode is the standard operating mode, and all RMS functions are available in it. The RMS is in Machine 1 mode after reset. Every feature described in Section 9.3 can be used by Machine 1. Machine 2 is backward-compatible with the MC6847-MC6883 combination, which is used with the MC6809E. Machine 2 has no additional features, but it has slightly different control. Machine 2 is discussed in Chapter 14.

9.2 Memory Maps

The RMS control registers may be configured as 192 contiguous bytes or as three 64-byte pages. The 192-byte unfolded option should be used with the MC68ØØØ family MPU's. The map must be folded for MC6847-MC6883 compatibility (Machine 2 operation). For more details see Chapter 1Ø.

9.3 Individual Registers

The following discussion of each register includes the unfolded and folded address and whether it is read/write, read only, or write only. Every register appears at one unfolded address and at one folded address/page except that the first 16 registers are mapped into the first and third pages, and 2 bits of those are mapped into the second page. All bits of all registers are active high.

In the following descriptions of the registers, their hexadecimal addresses are preceded by an x. The x is satisfied if the RMS chip select, located on RMI, is true (active low).

Unused bits are shown as blanks on the register diagrams; they read as Ø's. The state of all registers' bits is undefined after a reset except as noted in the following sections.

9.3.1 Memory Map

Unfolded Address $xFFEØØ, $xFFE8Ø (and $xFFE4Ø)

Folded Address $xFFF8Ø with Page ØØ, 1Ø, (and Ø1)

B7	B6	B5	B4	B3	B2	B1	BØ
MP1	MPØ	UF	M2				

The Memory Map register is multiply mapped. All of it is available at two locations, and bits 6 and 7 are also available at a third location in CMRØØ (see Section 9.3.25). It is a read/write register.

Bits 7 and 6 are MP1 and MPØ. They are used to select pages of the RMS register map when the folded map option is in use. They have no effect when the unfolded map option is in use. The page statement at the beginning of each subsection of Section 9.3 shows what combination of MP1 and MPØ is required to access the register in the folded map.

Bit 5 is UF. When it is set the control register map is being used in the unfolded mode. When it is reset, the control register map is folded. UF is set after a system reset.

Bit 4 changes the entire memory map to the Machine 2 memory map, which is backwards compatible with the MC6883 and MC6847. See Chapter 14 for more information.

After a reset, bits 4 through 7 are zeros.

9.3.2 Display Data Mode

Unfolded Address $xFFEØ1, or $xFFE81

Folded Address $xFFF81 with Page ØØ or 1Ø

B7	B6	B5	B4	B3	B2	B1	BØ
BP	LM2	LM1	LMØ	LPR1	LPRØ	BPP1	BPPØ

The Display Data Mode register is used to contol how the data from DRAM is converted into a pel pattern. It is a read/write register. It is available at two addresses.

Bit 7 is BP. When it is set, the RMS is operating in bit-plane display mode. When it is reset the RMS is in list mode.

Bits 6, 5, and 4 are used to choose among the list modes. If the BP bit is set, these bits have no effect. If BP is reset, the LM bits are used to select a particular list mode.

LM2	LM1	LMØ	List Mode
Ø	Ø	Ø	Ø
Ø	Ø	1	1
Ø	1	Ø	2
Ø	1	1	3
1	Ø	Ø	4
1	Ø	1	5
1	1	Ø	Reserved
1	1	1	Reserved

Table 9-1 List Mode Encoding

Bits 3 and 2 are LPR1 and LPRØ. When the RMS is being operated in a list mode, these bits determine how many video lines are used in each character row. They are coded as follows.

LPR1	LPRØ	Number of Lines
Ø	Ø	8
Ø	1	1Ø
1	Ø	12
1	1	16

Table 9-2 Lines per Row Encoding

The LPR bits have no effect on the display if bit-plane mode has been selected.

Bits 1 and Ø of the display mode register are BPP1 and BPPØ. They are used to select the number of bits per pel. These bits have meaning in both bit plane and list modes.

These two bits must be coded to select the number of data bits from DRAM to use to define each pel.

BPP1	BPPØ	Bits per Pel
Ø	Ø	1
Ø	1	2
1	Ø	4
1	1	Reserved

Table 9-3 Bits per Pel Encoding

This number is used for all pels in bit-plane mode. In the list modes it is used for the image tables of the DRCS and fixed objects.

When HRES mode 6 or 7 (512 or 64Ø pixels) is in use, 4 bits per pel cannot be selected, because it exceeds the data fetch rate of the RMS.

9.3.3 Interrupt Status

Unfolded Address $xFFEØ2, $xFFE82

Folded Address $xFFF82 with Page ØØ, 1Ø

B7	B6	B5	B4	B3	B2	B1	BØ
IPT			RTI	RTO	OFN	BLK	COL

The Interrupt Status register is a multiply mapped register. The entire register is available at two locations with the unfolded map.

Reading the register gets status: bits that are set indicate that those conditions have occurred since the last time the register was read. Writing to the register enables or disables interrupts; 1's allow the RMC to generate an interrupt when the condition occurs, and Ø's keep the RMC from doing that. The status bits get set when the conditions occur even if interrupts are disabled, and the status bits are cleared when the register is read.

Bit 7 is IPT, the general interrupt bit. If a Ø is written to it, all RMC-generated interrupts are disabled. If a 1 is written to it, the conditions in the rest of the register can generate interrupts if 1's are written to their registers, too. If a 1 is read in bit 7, it indicates that one of the register's conditions has occurred, whether or not the interrupt was enabled.

Bits 6 and 5 are reserved.

Bit 4 is RTI, the Real Time Interrupt bit. When RMC's RTI pin goes low, the current screen X and Y coordinates (discussed in Section 8.1) are stored in registers (see Section 9.3.16), and this bit and bit 7 are set. If a 1 was written to this bit and bit 7, then the RMC generates an interrupt.

Bit 3 is RTO, the Real Time Output bit. When the current screen X and Y coordinates (see Section 8.1) match the RTO registers (see Section 9.3.15), this bit is set. The interrupt handling is the same as for bit 4.

Bit 2 is OFN, the Object Finished bit. This bit is set each time the RMS finishes displaying a true object, so that the MPU may reprogram it as another object later in the field. Chapter 8 discusses true objects, and Section 9.3.5 discusses the Object Available register that lets the MPU tell which object is done. The interrupt handling is the same as for bit 4.

Bit 1 is BLK, the Blanking bit. It is set when the raster reaches the righthand border of the last active video line, so there is nothing but border to display for the rest of the video field. This has two primary functions. First, it provides a simple method of timing for the MPU to smoothly move objects at a speed related to the field rate. Second, it gives the user the maximum possible time to alter RMS operation before the next active video line. This allows changing everything but border color without glitches. Interrupt handling is as for bit 4.

Bit Ø is COL, the Collision Reporting bit. This bit is set when any true object collides with another true object or with a fixed object, if both objects have collisions enabled. See Chapter 8 and Sections 9.3.13 and 9.3.14 for more on collision status. Interrupt handling is the same as for bit 4.

All of the bits in this register are automatically cleared immediately after they are read by the MPU. If a condition is occuring at the same time the MPU is reading, the appropriate bit will be set after the current bits have been read and then reset.

It is also possible to read all of the bits even if they have not been enabled. For example, if a collision occurs, but the collision interrupt has not been set, no interrupt is generated, but the MPU will still find the collision bit set when it reads the byte.

Enabling a condition resets the bit so that old data is not reported.

9.3.4 Border Color

Unfolded Address $xFFEØ3 or $xFFE83

Folded Address $xFFF83 with Page ØØ or 1Ø

B7	B6	B5	B4	B3	B2	B1	BØ
WC	MAPA	DV	BC4	BC3	BC2	BC1	BCØ

The Border Color register is multiply mapped. It also contains bits that are not related to border color. It is a read/write register.

Bit 7 is WC, for Wrap Control. The RMS allows a virtual screen to be defined that is larger than the displayed screen, and it allows scrolling the displayed screen within the virtual screen. The WC bit controls what is displayed when the displayed screen is scrolled off the edges of the virtual screen: when it is reset, the RMS does barrel scrolling, which means that the virtual screen is treated as if it were wrapped around into a cylinder (or barrel), with the right and left edges brought together. The RMS also does barrel scrolling for the top and bottom edges, so with Wrap Control reset it is possible to put all four corners of the virtual screen in the center of the displayed screen and have four separate parts of the virtual screen displayed: the upper left quadrant would display the lower right corner of the virtual screen, the upper right quadrant would display the lower left corner of the virtual screen, the lower left quadrant would display the upper right corner, and the lower right quadrant would display the upper left corner.

Setting Wrap Control has no effect on list modes; they operate as just described for WC being reset. Setting Wrap Control causes bit-plane screens to scroll to a constant. When WC is set and the displayed screen is scrolled past the edge of the virtual screen, the area beyond the virtual screen is filled with the first 8 pixels in the virtual screen (for HRES modes 2, 3, and 4) or the first 16 pixels (HRES 6 and 7), repeated as needed. This can be used for special effects, or the first pixels at the virtual screen start address can be set to the border color. This makes the border seem to follow the edges of the display as it is moved.

Bit 6 selects between two different memory maps. These maps relate to the S bus output from RMI, which is used to select devices outside of the RMS, such as ROM and I/O. There are two options for the memory map, one designed for ROM-intensive, the other for DRAM-intensive applications. Chapter 1Ø has a detailed discussion of the memory map.

Bit 5 is DV (display video). It is used as a general video enable. It is particularly useful at power-up before the proper data has been placed in DRAM. When the DV bit is reset, the border color is displayed over the entire screen. The DV bit is reset after system reset.

Bits 4 through Ø are BC4 through BCØ; they are used to define the border color. BC4 is the most significant bit. These five bits are used as an address to the CMR during border time, or at all times if DV is reset.

9.3.5 Object Available

Unfolded Address $xFFEØ4 or $xFFE84

Folded Address $xFFF84 with Page ØØ or 1Ø

B7	B6	B5	B4	B3	B2	B1	BØ
O7A	O6A	O5A	O4A	O3A	O2A	O1A	OØA

The Object Available register is used to report which true objects are not currently in use. It is designed for use with the interrupt status register in order to reuse objects.

The Object Available register is a read only register that is multiply mapped. It contains one bit for each of the true objects. If the bit is a 1, then the object is available for use. If it is a 0, then the object is already in use and has not been completely displayed in this field.

An object's bit is set either when its last-line flag is found (see Section 8.7) or at vertical sync. It is reset when the MPU writes to its X Coordinate register (see Section 9.3.26); the new X coordinate does not have to be different from the old one to reset the bit. It is also reset by the RMS if it detects a match between the current scan line count and the object's Y coordinate.

If an object's X coordinate register were updated after vertical sync, that object's bit would be reset (indicating that the object is busy) from the time the X coordinate is written until the RMS is finished displaying it.

However, if the X coordinate register were updated before vertical sync, or if it were not updated at all, the Object Available register would show that object as being busy only during the video lines containing the true object's data.

9.3.6 Paging

Unfolded Address $xFFE05 or $xFFE85

Folded Address $xFFF85 with Page 00 or 10

B7	B6	B5	B4	B3	B2	B1	B0
			SWAP	PG3	PG2	PG1	PG0

The Paging register is a multiply mapped read/write register. It is used by the MC6809E MPU to manipulate up to 1 Mbyte of memory.

The MC6809E only supplies 16 address bits on the X bus, but a total of 20 are required to work with 1 Mbyte of memory. The additional bits are supplied by PG0-PG3 in the Paging register. They are used to control which part of the 1 Mbyte the MC6809E can access.

Once a 64 Kbyte block has been selected by PG0 to PG3, SWAP may be used to invert MPU address bit 15. This serves to swap the top and bottom halves of the 64 Kbyte block. This feature is required so that the MPU can get to DRAM locations that are hidden underneath the upper and lower Page Independent Blocks, the I/O, and the RMS control registers (see Section 9.3.7).

PG0 to PG3 are set and SWAP is reset following system reset.

This register has no effect on MC68000 family MPU operation. MC6809E applications require careful coordination of Paging and Page Independent Block registers (see the next section) to ensure that the RMS' registers do not disappear from the memory map.

9.3.7 Page Independent Blocks

Unfolded Address $xFFE06 or $xFFE86

Folded Address $xFFF86 with Page 00 or 10

B7	B6	B5	B4	B3	B2	B1	B0
UEN	UPI2	UPI1	UPI0	LEN	LPI2	LPI1	LPI0/VEC

The Page Independent Block register is a multiply mapped read/write register. It is used by the MC6809E as a memory management aid and by the MC68000 family to help select S-bus outputs. Since the MC6809E cannot directly address more than 64 Kbytes of memory, it uses the page independent blocks to help steer it around the 1 Mbyte address range of RMS. The page independent blocks (PIB's) are used with the Paging register (Section 9.3.6).

The PIB's are used to hold two blocks of the RMS memory map fixed at the top and bottom of the MPU's memory map. They can be used to hold some scratchpad RAM, the I/O, and the RMS register sections of the memory map at fixed addresses while the paging register is used to change the center of the memory map.

There is an upper page independent block (UPI bits) and a lower PIB (LPI bits). Each has an associated enable bit (UEN and LEN). If an enable bit is set, then its PIB is in use.

The value of the UPI or LPI bits determines the size of the PIB as shown below.

Bit 2	Bit 1	Bit 0	PIB Size
0	0	0	256 bytes
0	0	1	512 bytes
0	1	0	1K bytes
0	1	1	2K bytes
1	0	0	4K bytes
1	0	1	8K bytes
1	1	0	16K bytes
1	1	1	32K bytes

Table 9-4 PIB Size Encoding

The least significant bit of this register is used by the MC68000 family processors to select DRAM (B0 set) or ROM (B0 reset) for the exception vectors. See Section 10.1.

Refer to Section 10.2 for more information on PIB's. Following system reset, both PIB's are disabled.

9.3.8 Vertical Scroll

Unfolded Address $xFFE07 or $xFFE87

Folded Address $xFFF87 with Page 00 or 10

B7	B6	B5	B4	B3	B2	B1	B0
			DHP	VSC3	VSC2	VSC1	VSC0

The Vertical Scroll register is a read/write, multiply mapped register that is involved in performing smooth vertical scrolling while in a list mode. It has no effect in bit plane mode.

In list mode, bits VSC 0 to 3 are used to determine which scan line of the character row at the top of the displayed screen is to be used as the first scan line of the displayed screen. Subsequent rows begin with row 0, as usual.

The programmer should avoid using numbers that are too large, as the results are undefined. For example, if 12-scan-line-high characters are being used, no number larger than 11 should be used in the vertical scroll register. Eleven works because the scan lines within a character row are numbered from 0, the top line, down to N-1, the bottom line, where N is the number of scan lines in a character row. Bit 4 (Double High Preset) assists vertical scrolling when characters using the double high attribute (see Section 7.2.4.6) are displayed. Scrolling is discussed in Chapter 12.

The Vertical Scroll register is used with the Vertical Offset register. See Sections 9.3.21, 6.4, and 6.5.

9.3.9 Horizontal Scroll

Unfolded Address $xFFE08 or $xFFE88

Folded Address $xFFF88 with Page 00 or 10

B7	B6	B5	B4	B3	B2	B1	B0
			DWP	HSC3	HSC2	HSC1	HSC0

The Horizontal Scroll register is similar to the Vertical Scroll register. It is also multiply mapped and read/write.

The HSC bits are used for pixel by pixel (smooth) scrolling in both bit-plane and list modes. The number of HSC bits used depends on the mode in use.

Horizontal resolution modes 6 and 7 use all four bits, since they both generate 16 pixels during each memory cycle. The other HRES modes only generate 8 pixels per memory cycle. Therefore they do not use HSC3, and it should always be a Ø for these modes.

Bit 4 (Double Wide Preset) assists horizontal scrolling when characters using the double wide attribute (see Section 7.2.4.7) are displayed. Scrolling is discussed in detail in Sections 6.5 and 12.4.

A character row full of double wide characters displays only every other character position, so getting DWP out of synchronization in a row could cause problems. In order to have smooth scrolling with a constant format, the hidden character positions must contain characters identical (including all attributes) to the characters that hide them, and the DWP must be cleared when the hiding characters are in the second displayed column, and set when the hidden characters are first.

The Horizontal Scroll register is used with the Horizontal Offset register. See Sections 9.3.22, 6.4, and 6.5.

9.3.1Ø DRCS Image Table Start Address

Unfolded Address $xFFEØA or $xFFE8A

Folded Address $xFFF8A with Page ØØ or 1Ø

B7	B6	B5	B4	B3	B2	B1	BØ
				DS19	DS18	DS27	DS16

Unfolded Address $xFFEØB or $xFFE8B

Folded Address $xFFF8B with Page ØØ or 1Ø

B7	B6	B5	B4	B3	B2	B1	BØ
DS15	DS14	DS13	DS12	DS11	DS1Ø	Ø	Ø

The DRCS Image Table Start Address register is used to define the area in DRAM that is reserved for storing the patterns of the Dynamically Redefinable Character Set. It is a multiply mapped read/write register.

The RMS treats these two bytes as if they were followed by a byte of Ø's, so the address in DRAM has ten least significant zeros. This allows the user to select any of the 1Ø24 1K boundaries in the 1 Mbyte address space as the start of the DRCS image table.

9.3.11 True Object Image Table Start Address

Unfolded Address $xFFEØC or $xFFE8C

Folded Address $xFFF8C with Page ØØ or 1Ø

B7	B6	B5	B4	B3	B2	B1	BØ
				TS19	TS18	TS17	TS16

Unfolded Address $xFFEØD or $xFFE8D

Folded Address $xFFF8D with Page ØØ or 1Ø

B7	B6	B5	B4	B3	B2	B1	BØ
TS15	TS14	TS13	TS12	TS11	TS1Ø	Ø	Ø

The True Object Image Table Start Address register is a multiply mapped read/write register. Its function is similar to the DRCS Image Table Start Address register, except it defines the start of pattern data for true objects.

It also has ten least significant Ø's, which allow the user to select one of 1Ø24 different 1 Kbyte boundries to start the true object image table.

9.3.12 Fixed Object Image Table Start Address

Unfolded Address $xFFEØE or $xFFE8E

Folded Address $xFFF8E with Page ØØ or 1Ø

B7	B6	B5	B4	B3	B2	B1	BØ
				FS19	FS18	FS17	FS16

Unfolded Address $xFFEØF or $xFFE8F

Folded Address $xFFF8F with Page ØØ or 1Ø

B7	B6	B5	B4	B3	B2	B1	BØ
FS15	FS14	FS13	FS12	FS11	FS1Ø	Ø	Ø

The Fixed Object Image Table Start Address register is a multiply mapped read/write register. Its function is the same as the DRCS Image Table Start Address register, except that it is used to locate the patterns of fixed objects.

It also selects one of 1Ø24 different 1 Kbyte boundries.

9.3.13 Collision Status

There are 8 read only registers used for reporting collisions. Each collision status register is used for a different true object. The general form of the register is shown below for true object number N. Bit Ø is used to report collisions between true object Ø and true object N. Bit 1 reports collisions between true objects 1 and N. The other bits are similar.

B7	B6	B5	B4	B3	B2	B1	BØ
NC7	NC6	NC5	NC4	NC3	NC2	NC1	NCØ

The Nth bit is used to report collisions between object N and fixed objects, since it is not needed to report collisions with itself. The addresses of the collision registers are shown below.

Unfolded Address	Folded Address/Page	For True Object
$xFFE1Ø	$FFF9Ø/ØØ	Ø
$xFFE11	$FFF91/ØØ	1
$xFFE12	$FFF92/ØØ	2
$xFFE13	$FFF93/ØØ	3
$xFFE14	$FFF94/ØØ	4
$xFFE15	$FFF95/ØØ	5
$xFFE16	$FFF96/ØØ	6
$xFFE17	$FFF97/ØØ	7

Table 9-5 Collision Register Addresses

In order for a collision to be reported, the following conditions must all be true.

1. The collision enable bits of both (all) objects involved must all be set.

2. At least one nontransparent pel of both (all) objects involved must occupy the same pel position.

3. The transparent pels of true objects, and the pels of fixed objects that don't pass the color collision test, are ignored.

Collisions between two true objects set two bits. For example, a collision between objects 3 and 7 would set bit 3 of object seven's register and bit 7 of object three's register.

The Collision registers are automatically cleared following an MPU read. No other condition clears them. The MPU may read the registers at any time. Collisions occuring at the time the register is being read are saved and put into the register after it has been automatically cleared.

Collisions flags are reset only in the Collision register read by the MPU. In the example above, reading object three's register would reset all bits in its register but not bit 3 of object seven's register.

9.3.14 Collision Enable

Unfolded Address $xFFE18

Folded Address $xFFF98 with Page ØØ

B7	B6	B5	B4	B3	B2	B1	BØ
CEN7	CEN6	CEN5	CEN4	CEN3	CEN2	CEN1	CENØ

The Collision Enable register is a read/write register used to activate or deactivate the true objects' collision reporting.

True objects collide with other true objects and with fixed objects by overlapping them on the screen, as described in Section 8.6. These collisions can be reported to the MPU using flags or an interrupt, as described in Sections 9.3.3 and 9.3.13, or the reporting can be disabled. This register allows each true object's collision-reporting to be individually selected or turned off. When CEN7 is set, for instance, true object 7 can report collisions. When CEN7 is reset, object 7 cannot report collisions, and other collision-enabled objects that overlap it cannot report collisions with it, either.

9.3.15 Real Time Output

Unfolded Address $xFFE1C

Folded Address $xFFF9C with Page 00

B7	B6	B5	B4	B3	B2	B1	B0
						OX9	OX8

Unfolded Address $xFFE1D

Folded Address $xFFF9D with Page 00

B7	B6	B5	B4	B3	B2	B1	B0
OX7	OX6	OX5	OX4	OX3	OX2	OX1	OX0

Unfolded Address $xFFE1E

Folded Address $xFFF9E with Page 00

B7	B6	B5	B4	B3	B2	B1	B0
						OY9	OY8

Unfolded Address $xFFE1F

Folded Address $xFFF9F with Page 00

B7	B6	B5	B4	B3	B2	B1	B0
OY7	OY6	OY5	OY4	OY3	OY2	OY1	OY0

The Real Time Output (RTO) registers are read/write registers. They are used to program the real time output logic so that it generates an MPU interrupt at the correct time.

The RTO interrupt bit is located in the Interrupt Status register; see Section 9.3.3. If it is enabled it generates an interrupt when the CRT's electron beam reaches the specified X coordinate on the scanline following the specified Y coordinate. The RTO registers use the same coordinate system as is used for true objects. If the RTO registers are set to coordinates that the beam does not reach (such as 1023, 1023 in all modes, or 400, 400 in HRES 4 mode), no RTO event can be triggered. For more information see Section 8.1. If the RTO function is needed at more than one screen position, but there is not time to change the RTO registers between occurrences, a true object can be used to perform a similar function. See Section 8.7.

9.3.16 Real Time Input

Unfolded Address $xFFE20

Folded Address $xFFFA0 with Page 00

B7	B6	B5	B4	B3	B2	B1	B0
						IX9	IX8

Unfolded Address $xFFE21

Folded Address $xFFFA1 with Page 00

B7	B6	B5	B4	B3	B2	B1	B0
IX7	IX6	IX5	IX4	IX3	IX2	IX1	IX0

Unfolded Address $xFFE22

Folded Address $xFFFA2 with Page 00

B7	B6	B5	B4	B3	B2	B1	B0
						IY9	IY8

Unfolded Address $xFFE23

Folded Address $xFFFA3 with Page 00

B7	B6	B5	B4	B3	B2	B1	B0
IY7	IY6	IY5	IY4	IY3	IY2	IY1	IY0

The Real Time Input (RTI) registers are read only registers. They are used with the RMC's RTI input pin and the RTI interrupt (see Section 9.3.3). One use for the RTI is to interface to a light pen, although any other event that satisfies the trigger requirements can be used.

A falling edge on the RTI pin causes the RTI registers to be loaded with the CRT beam's current X and Y coordinates. The XY coordinate system is the same as that for true objects; it is described in Section 8.1.

The MPU should read the RTI as soon after the interrupt as possible, or at some other time when it can be certain that the RTI input is not seeing a falling edge. The RTI data is maintained indefinitely between RTI inputs, but the data is not guaranteed to be correct if the MPU is reading the register at the same time a falling edge occurs on the RTI pin.

9.3.17 Memory Organization

Unfolded Address $xFFE24

Folded Address $xFFFA4 with Page ØØ

B7	B6	B5	B4	B3	B2	B1	BØ
MTP3	MTP2	MTP1	MTPØ	DB1	DBØ		

The Memory Organization register is a read/write register. It is used by the MPU to inform the RMS of the type of DRAM's in use and how the memory is organized.

The MTP bits are used to define the type and width of DRAM and should be programmed as follows.

MTP3	MTP2	MTP1	MTPØ	DRAM Type
Ø	Ø	Ø	Ø	16Kx1, 8 Bits Wide
Ø	Ø	Ø	1	16Kx4, 8 Bits Wide
Ø	Ø	1	Ø	64Kx1, 8 Bits Wide
Ø	Ø	1	1	Reserved
Ø	1	Ø	Ø	256Kx1, 8 Bits Wide
Ø	1	Ø	1	Reserved
Ø	1	1	Ø	Reserved
Ø	1	1	1	Reserved
1	Ø	Ø	Ø	16Kx1, 16 Bits Wide
1	Ø	Ø	1	16Kx4, 16 Bits Wide
1	Ø	1	Ø	64Kx1, 16 Bits Wide
1	Ø	1	1	Reserved
1	1	Ø	Ø	256Kx1, 16 Bits Wide
1	1	Ø	1	Reserved
1	1	1	Ø	Reserved
1	1	1	1	Reserved

Table 9-6 Memory Type Encoding

The MTP3 bit selects between the 8-bit-wide and 16-bit-wide data paths, so it must be set for the MC68000 and reset for the MC6809E and MC68008. Since banks are always 8 bits wide, MTP3 being set means there must be either 2 or 4 banks of DRAM in the system.

DB1 and DB0 must be set to indicate how many banks of DRAM are connected to the system. They are coded as follows.

DB1	DB0	Number of Banks
0	0	1
0	1	2
1	0	Reserved
1	1	4

Table 9-7 DRAM Bank Encoding

The DB bits can both be zero only with 8-bit MPU's, since a 16-bit data path requires 2 or 4 banks.

9.3.18 Video Operation

Unfolded Address $xFFE25

Folded Address $xFFFA5 with Page 00

B7	B6	B5	B4	B3	B2	B1	B0
IS	ID	VRES2	VRES1	VRES0	HRES2	HRES1	HRES0

The Video Operation register is a read/write register used by the MPU to control the basic screen format.

Bit 7 is Interlace Sync. When it is set, the display uses interlace sync, and when it is reset the sync is noninterlace.

Bit 6 is Interlace Data. When it is set the display uses interlace data. When it is reset the display uses noninterlace data. Interlace sync may be used with interlace data or noninterlace data (see Section 9.3.2.). Noninterlace data must be used if noninterlace sync is selected.

The VRES bits determine the vertical resolution of the active display area in scan lines. They are coded as follows.

VRES2	VRES1	VRES0	Lines per Field	VRES Mode
0	1	0	192	2
0	1	1	200	3
1	0	0	210	4
1	0	1	240	5
1	1	0	250	6
1	1	1	Reserved	Reserved

Table 9-8 VRES Encoding

If bits 6 and 7 are both set, then the vertical resolution is double what is shown here. See Section 9.3.2.

The 240 and 250 line resolutions work only with 625 line timing.

The HRES mode bits define the horizontal resolution of a scan line. They are coded as follows. After system reset, the RMS is in HRES 4.

HRES2	HRES1	HRES0	Pixels per Line	HRES Mode
0	1	0	256 (Narrow)	2
0	1	1	256 (Wide)	3
1	0	0	320	4
1	0	1	Reserved	Reserved
1	1	0	512	6
1	1	1	640	7

Table 9-9 HRES Encoding

For more information on HRES and VRES modes see sections 6.2.1 and 6.2.2.

9.3.19 Sync Mode

Unfolded Address $xFFE26

Folded Address $xFFFA6 with Page 00

B7	B6	B5	B4	B3	B2	B1	B0
				VIS2	VIS1	VIS0	GS

The Sync Mode register is a read/write register. It controls the function of the RMC's SYNC pin and G pin.

The VIS bits determine how the RMC's SYNC pin is used.

VIS2	VIS1	VISØ	Signal Direction	Signal
Ø	Ø	Ø	Output	Vertical Sync
Ø	Ø	1	Output	Horizontal Sync
Ø	1	x	Output	Composite Sync
1	x	x	Input	Field Sync

Table 9-1Ø Sync Encoding

When the GS bit is set, it selects combined video and sync output for the RMC's G output. When the GS bit is reset, the G pin has only the video output.

9.3.2Ø Virtual Screen Start Address

Unfolded Address $xFFE28

Folded Address $xFFFA8   with Page ØØ

B7	B6	B5	B4	B3	B2	B1	BØ

Unfolded Address $xFFE29

Folded Address $xFFFA9   with Page ØØ

B7	B6	B5	B4	B3	B2	B1	BØ
				VS19	VS18	VS17	VS16

Unfolded Address $xFFE2A

Folded Address $xFFFAA   with Page ØØ

B7	B6	B5	B4	B3	B2	B1	BØ
VS15	VS14	VS13	VS12	VS11	VS1Ø	Ø	Ø

Unfolded Address $xFFE2B

Folded Address $xFFFAB   with Page ØØ

B7	B6	B5	B4	B3	B2	B1	BØ
Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø

The Virtual Screen Start Address register is a 4-byte read/write register that holds the beginning address of the virtual screen memory. Its 4-byte size makes it the length of a long word in MC68ØØØ family MPU's.

The virtual screen is a block of display memory whose size and location are defined by the user. It must be at least as large as the displayed screen and can be as big as 512 Kbytes. The user can shift the displayed screen's position within the virtual screen. The virtual screen is defined by 3 registers: this one, Virtual Screen Size (9.3.23), and Virtual Screen Width (9.3.24).

9.3.21 Vertical Offset

Unfolded Address $xFFE2C

Folded Address $xFFFAC with Page ØØ

B7	B6	B5	B4	B3	B2	B1	BØ

Unfolded Address $xFFE2D

Folded Address $xFFFAD with Page ØØ

B7	B6	B5	B4	B3	B2	B1	BØ
(SIGN)	Ø	Ø	Ø	Ø	Y18	Y17	Y16

Unfolded Address $xFFE2E

Folded Address $xFFFAE with Page ØØ

B7	B6	B5	B4	B3	B2	B1	BØ
Y15	Y14	Y13	Y12	Y11	Y1Ø	Y9	Y8

Unfolded Address $xFFE2F

Folded Address $xFFFAF with Page ØØ

B7	B6	B5	B4	B3	B2	B1	BØ
Y7	Y6	Y5	Y4	Y3	Y2	Ø	Ø

The Vertical Offset register is a 4-byte read/write register that (along with the Horizontal Offset register) places the displayed screen within the virtual screen. It is a 32-bit two's complement number whose units are bytes. It is 4 bytes long to match the size of the MC68ØØØ family MPU's long word. The first byte is blank and always reads as zero, and bits B6-B3 of the next byte also read as zero.

In this section, "line" means a scanline in bit-plane and a character row in the list modes.

The Vertical Offset register must point to the beginning of a line in the virtual screen (the Horizontal Offset locates the screen within the line), and so it must contain the number of bytes per line in the virtual screen multiplied by the virtual screen's line number. The Virtual Screen Width register (Section 9.3.24) contains the number of bytes per line.

The two least significant bits are zero, so the register has the same 4-byte resolution as the Width register. Since it has the same range as the Virtual Screen Size register (Section 9.3.23), any line or character row in the virtual screen can be used as the top line of the displayed screen.

The Vertical Offset register's upper limit is one virtual screen width less than the virtual screen size, which puts the last scan line (bit-plane) or character row (list mode) at the top of the displayed screen. When the Wrap Control bit is set, the register's lower limit is the negative of its upper limit; when WC is reset, the lower limit is zero. Wrap Control is discussed in Section 9.3.4, but briefly, WC = Ø causes barrel scrolling and WC = 1 causes scrolling to a constant. Vertical barrel scrolling can be done repeatedly by adding the Virtual Screen Width register to the Vertical Offset register until its upper limit is reached, then using zero as the next offset. Scrolling to a constant allows the RMS to display beyond the edge of the virtual screen in bit-plane mode. The first 8 pixels (low-resolution) or 16 pixels (high resolution) are repeated to fill in this extra area.

In the list modes, each line is one character high, which means that each line is from 8 to 16 scan lines high. For smooth vertical scrolling in the list modes, changes must be coordinated between the Vertical Offset register and the Vertical Scroll register. Within a character row, the Scroll register must be changed; at a row boundary, the Scroll register and the Vertical Offset register must both be changed. See Sections 6.5, 9.3.8, and 12.4.

In the bit-plane mode, each line is one pel high, so the Vertical Scroll register is not used, and the Vertical Offset register is used for all vertical scrolling.

9.3.22 Horizontal Offset

Unfolded Address $xFFE30

Folded Address $xFFFB0 with Page 00

B7	B6	B5	B4	B3	B2	B1	B0

Unfolded Address $xFFE31

Folded Address $xFFFB1 with Page 00

B7	B6	B5	B4	B3	B2	B1	B0

Unfolded Address $xFFE32

Folded Address $xFFFB2 with Page 00

B7	B6	B5	B4	B3	B2	B1	B0
(SIGN)	0	0	0	0	X10	X9	X8

Unfolded Address $xFFE33

Folded Address $xFFFB3 with Page 00

B7	B6	B5	B4	B3	B2	B1	B0
X7	X6	X5	X4	X3	X2	X1	X0

The Horizontal Offset register is a 4-byte read/write register used to position the left edge of the displayed screen within the virtual screen. Horizontal Offset is a 32-bit two's complement number whose units are bytes. This register is 4 bytes long to match the MC68000 family MPU's long word. The first two bytes are blank and always read as zero, as do bits B6-B3 of the third byte.

The Horizontal Offset register must point to the beginning of a memory cycle, which is 8 pixels long in HRES modes 0-4 and 16 pixels long in HRES modes 6 and 7. Locating the displayed screen within the memory cycle is done for list mode and bit-plane mode with the Horizontal

Scroll register, discussed in Section 9.3.9. The number of bytes used to define a memory cycle's worth of display is related to the HRES mode and bits per pel in the bit-plane mode, and to the selected version of the list mode.

		HRESØ through HRES4	HRES6 and HRES7
Bits	1	1	2
per	2	2	4
Pel	4	4	NA

Table 9-11 Bit-plane Bytes per Memory Cycle

List Mode	Ø	1	2	3	4	5
Bytes/Cycle	1	2	2	3	3	3

Table 9-12 List Mode Bytes per Memory Cycle

The Horizontal Offset register has 1-byte resolution. The Horizontal Offset's upper limit is one memory cycle's worth of bytes less than the contents of the Virtual Screen Width register (Section 9.3.24). This allows the displayed screen's left edge to begin with the last 8 or 16 pixels of the virtual screen. If the Wrap Control bit is set, the lower limit is the negative of its upper limit; if WC is reset, the lower limit is zero. Wrap Control is discussed in Section 9.3.4. The WC bit's effect on the use of the Horizontal Offset register matches its effect on the Vertical Offset register (Section 9.3.21).

To scroll horizontally on a pixel-by-pixel basis, the Horizontal Offset register and the Horizontal Scroll register must be used together. See Sections 6.4, 6.5, and 12.4.

9.3.23 Virtual Screen Size

Unfolded Address $xFFE34

Folded Address $xFFFB4 with Page ØØ

B7	B6	B5	B4	B3	B2	B1	BØ

Unfolded Address $xFFE35

Folded Address $xFFFB5 with Page ØØ

B7	B6	B5	B4	B3	B2	B1	BØ
					V18	V17	V16

Unfolded Address $xFFE36

Folded Address $xFFFB6 with Page ØØ

B7	B6	B5	B4	B3	B2	B1	BØ
V15	V14	V13	V12	V11	V1Ø	V9	V8

Unfolded Address $xFFE37

Folded Address $xFFFB7 with Page ØØ

B7	B6	B5	B4	B3	B2	B1	BØ
V7	V6	V5	V4	V3	V2	Ø	Ø

The Virtual Screen Size register is a 4-byte read/write register that sets the size of the virtual screen in bytes. It is calculated by multiplying the number of lines in the virtual screen times the number of bytes per line. The virtual screen must have an integer number of lines; that is, every line must be the same length as the others. There must be at least enough lines to fill the displayed screen. The number of bytes per line is computed in Section 9.3.24. Adding the Virtual Screen Size register's contents to the address in the Virtual Screen Start Address register gives the address of the first byte following the virtual screen memory.

The least significant 2 bits are always zero, giving 4-byte resolution. Since this resolution matches the Virtual Screen Width's resolution, there is no resolution-imposed limit on the number of lines in the virtual screen. The maximum virtual screen size is 512 Kbytes. The largest possible screen can range from 256 lines, each line 2 Kbytes long, to 16,384 lines, each line 32 bytes long. These lines are scan lines (bit-plane) or character row (list mode).

9.3.24 Virtual Screen Width

Unfolded Address $xFFE38

Folded Address $xFFFB8 with Page ØØ

B7	B6	B5	B4	B3	B2	B1	BØ

Unfolded Address $xFFE39

Folded Address $xFFFB9 with Page ØØ

B7	B6	B5	B4	B3	B2	B1	BØ

Unfolded Address $xFFE3A

Folded Address $xFFFBA with Page ØØ

B7	B6	B5	B4	B3	B2	B1	BØ
					W1Ø	W9	W8

Unfolded Address $xFFE3B

Folded Address $xFFFBB with Page ØØ

B7	B6	B5	B4	B3	B2	B1	BØ
W7	W6	W5	W4	W3	W2	Ø	Ø

The Virtual Screen Width register is a 32-bit read/write register that gives the virtual screen's line width in bytes. It has a resolution of 4 bytes and a range of 2 Kbytes. In addition to being an integer multiple of 4 bytes, the lines must be an integer number of memory cycles wide, and they must be at least as wide as the visible screen. Tables in Section 9.3.22 on the Horizontal Offset register give the number of bytes per memory cycle for all display modes. The line length is in pels for bit-plane and in characters or pels for the list modes. Several ways of computing the line length in bytes follow. The number of pels per line must be a multiple of 8 for HRES modes Ø-4, and it must be a multiple of 16 for HRES modes 6 and 7.

Bit Plane:

```
LL  =  (PELS/LINE)   *  (BITS/PEL)   *  (1 BYTE/8 BITS)
       User-selected    1, 2, or 4
```

List Mode:

```
LL  =  (PELS/LINE)   *  (BYTES/CHARACTER)  /  (PELS/CHARACTER)
       User-selected     List mode-dependent   List mode-dependent

LL  =  (CHARACTERS/LINE)  *  (BYTES/CHARACTER)
       User-selected         List mode-dependent
```

List Mode	Ø	1	2		3	4	5	
BYTES/CHARACTER	1	2	1	2	3	3	1.5	3
PELS/CHARACTER	8	8	8	16	16	8	8	16

Table 9-13 List Mode Bytes and Pels per Character

Modes 2 and 5 must have an even number of 8-pel wide characters, since 2 are displayed in each memory cycle.

The maximum virtual screen widths are:

List Mode	Ø	1	2	3	4	5

Chars/Line (Pel Width)	2K(8)	1K(8)	2K(8)
			1K(16)
				68Ø(16)	68Ø(8)	136Ø(8)
						68Ø(16)

Table 9-14 List Mode Maximum Screen Widths

This is independent of the horizontal resolution and the bits per pel of the redefinable characters.

The maximum line width in pels in the bit-plane mode depends only on the bits per pel (except, of course, that HRES 6 and 7 cannot use 4 bits per pel): 1 bit per pel means 16K pels maximum; 2 bits per pel means 8K pels, and 4 bits per pel means 4K pels maximum.

9.3.25 Color Mapping RAM

There are 32 different Color Mapping RAM (CMR) registers. Their format and function are identical, except for CMRØØ, which has two additional bits. All of the CMR registers are read/write.

Each CMR register occupies two bytes and looks like this:

Lower Address

B7	B6	B5	B4	B3	B2	B1	BØ
			VEN	R3	R2	R1	RØ

Upper Address

B7	B6	B5	B4	B3	B2	B1	BØ
G3	G2	G1	GØ	B3	B2	B1	BØ

The VEN bit controls the RMC's video enable (VEN) output pin when this particular CMR location is accessed by the video data. The VEN output is designed to allow the user to implement a transparency function. See Section 4.7.4.

The R, G, and B bits are used as inputs to D/A converters that drive the R, G, and B output pins. R3 is the most significant bit of red and RØ is the least significant bit.

Each of the CMR locations has its own register to provide the MPU with the means to define its color.

CMR Location	Folded Map, Page Ø1	Unfolded Map
ØØ	$xFFF8Ø,1	$xFFE4Ø,1
Ø1	$xFFF82,3	$xFFE42,3
Ø2	$xFFF84,5	$xFFE44,5
Ø3	$xFFF86,7	$xFFE46,7
Ø4	$xFFF88,9	$xFFE48,9
Ø5	$xFFF8A,B	$xFFE4A,B
Ø6	$xFFF8C,D	$xFFE4C,D
Ø7	$xFFF8E,F	$xFFE4E,F
Ø8	$xFFF9Ø,1	$xFFE5Ø,1
Ø9	$xFFF92,3	$xFFE52,3
ØA	$xFFF94,5	$xFFE54,5
ØB	$xFFF96,7	$xFFE56,7
ØC	$xFFF98,9	$xFFE58,9
ØD	$xFFF9A,B	$xFFE5A,B
ØE	$xFFF9C,D	$xFFE5C,D
ØF	$xFFF9E,F	$xFFE5E,F
1Ø	$xFFFAØ,1	$xFFE6Ø,1
11	$xFFFA2,3	$xFFE62,3
12	$xFFFA4,5	$xFFE64,5
13	$xFFFA6,7	$xFFE66,7
14	$xFFFA8,9	$xFFE68,9
15	$xFFFAA,B	$xFFE6A,B
16	$xFFFAC,D	$xFFE6C,D
17	$xFFFAE,F	$xFFE6E,F
18	$xFFFBØ,1	$xFFE7Ø,1
19	$xFFFB2,3	$xFFE72,3
1A	$xFFFB4,5	$xFFE74,5
1B	$xFFFB6,7	$xFFE76,7
1C	$xFFFB8,9	$xFFE78,9
1D	$xFFFBA,B	$xFFE7A,B
1E	$xFFFBC,D	$xFFE7C,D
1F	$xFFFBE,F	$xFFE7E,F

Table 9-15 CMR Addresses

The CMR registers are available for use by the MPU at any time, but the RMC cannot access a particular CMR address in the same memory cycle that the MPU reads or writes to it. This can cause several pixels to be undefined in that scan line. The display is unaffected by MPU accesses to CMR registers not being used to display video at the time of the MPU access. See Section 12.2 for more real-time CMR changing details.

CMRØØ includes two extra bits.

CMR Register $ØØ

B7	B6	B5	B4	B3	B2	B1	BØ
MP1	MPØ		VEN	R3	R2	R1	RØ

Bits Ø to 4 of this byte of CMRØØ are identical to the other CMR registers. Bits 7 and 6 are remapped from the Paging register (see Section 9.3.1) so the MPU can change pages in the RMS' folded map. Because these bits have no effect in the unfolded map, their state does not need to be preserved in unfolded map writes to this register.

9.3.26 True Object Position

The True Object Position registers for each of the true objects are read/write registers. There are 4 bytes of register for each object. Two bytes each are used for the X and the Y coordinates.

X Coordinate

Lower Address

B7	B6	B5	B4	B3	B2	B1	BØ
BR	XZ1	XZØ				X9	X8

Upper Address

B7	B6	B5	B4	B3	B2	B1	BØ
X7	X6	X5	X4	X3	X2	X1	XØ

Y Coordinate

Lower Address

B7	B6	B5	B4	B3	B2	B1	BØ
OEN	YZ1	YZØ				Y9	Y8

Upper Address

B7	B6	B5	B4	B3	B2	B1	BØ
Y7	Y6	Y5	Y4	Y3	Y2	Y1	YØ

The locations of the registers for each object in the unfolded memory map are as follows.

Object Number	X Coordinate	Y Coordinate
Ø	$xFFE9Ø,1	$xFFE92,3
1	$xFFE94,5	$xFFE96,7
2	$xFFE98,9	$xFFE9A,B
3	$xFFE9C,D	$xFFE9E,F
4	$xFFEAØ,1	$xFFEA2,3
5	$xFFEA4,5	$xFFEA6,7
6	$xFFEA8,9	$xFFEAA,B
7	$xFFEAC,D	$xFFEAE,F

Table 9-16 Object Position Registers (Unfolded Map)

In the folded map, the paging bits must be set to 1Ø. Then the registers are located as follows.

Object Number	X Coordinate	Y Coordinate
Ø	$xFFF9Ø,1	$xFFF92,3
1	$xFFF94,5	$xFFF96,7
2	$xFFF98,9	$xFFF9A,B
3	$xFFF9C,D	$xFFF9E,F
4	$xFFFAØ,1	$xFFFA2,3
5	$xFFFA4,5	$xFFFA6,7
6	$xFFFA8,9	$xFFFAA,B
7	$xFFFAC,D	$xFFFAE,F

Table 9-17 Object Position Registers (Folded Map)

The X and the Y coordinates occupy only the ten least significant bits of their registers. In each coordinate, the most significant 3 bits are used for object control.

The X coordinate's most significant bit is B/R. It selects between the bit-plane and run-length display formats described in Section 8.3. Clearing the bit selects run-length format, and setting it selects bit-plane format.

The X coordinate's next 2 bits are XZ1 and XZØ, respectively. Together, they set the horizontal zoom factor:

XZ1	XZØ	Horizontal Zoom Factor
Ø	Ø	1x (Normal)
Ø	1	2x
1	Ø	4x
1	1	8x

Table 9-18 Horizontal Zoom Encoding

The left edge of the true object remains stationary during horizontal zoom, so the object expands to the right.

The Y coordinate's most significant bit OEN, or Object Enable. It must be set for the object to appear and have collisions. The Y coordinate's next 2 bits are YZ1 and YZØ, and they control the vertical zoom factor:

YZ1	YZØ	Vertical Zoom Factor
Ø	Ø	1x (Normal)
Ø	1	2x
1	Ø	4x
1	1	8x

Table 9-19 Vertical Zoom Encoding

The top edge of the true object remains stationary during vertical zoom, so the object expands downward.

9.3.27 True Object Names

The True Object Name registers are read/write registers. There is one for each of the eight true objects. The user should place the name (character code) of the desired image table entry in this register. This name is used by the RMS to calculate the DRAM address to find the pattern data for the object. The name is a byte with this format:

B7	B6	B5	B4	B3	B2	B1	B0
N7	N6	N5	N4	N3	N2	N1	N0

The Name registers for each of the objects are located at these addresses.

Object Number	Unfolded Address	Folded Address, Page 10
0	$xFFEB0	$xFFFB0
1	$xFFEB1	$xFFFB1
2	$xFFEB2	$xFFFB2
3	$xFFEB3	$xFFFB3
4	$xFFEB4	$xFFFB4
5	$xFFEB5	$xFFFB5
6	$xFFEB6	$xFFFB6
7	$xFFEB7	$xFFFB7

Table 9-20 Object Name Register Addresses

The relationship between a true object's name and the address of the entry in the true object image table is:

```
IMAGE TABLE ENTRY ADDRESS
     = TRUE OBJECT IMAGE TABLE START ADDRESS + (128 * OBJECT NAME)
```

1Ø.Ø SYSTEM MEMORY MAP

When the RMS is in Machine 1 mode, it provides four system memory maps for each microprocessor. Machine 2 mode, which is backwards compatible with the MC6883 and MC6847, is discussed in Chapter 14.

The size and location of ROM, DRAM, and I/O within the RMS' 1 Mbyte address space is controlled by the RMI's 3-bit S bus (see Section 4.4). The S bus is designed to drive a 74ALS138 or equivalent, providing 7 chip selects for ROM and I/O. This chapter discusses the types of S bus outputs and the RMS registers that control them.

The S bus output is independent of the MPU's R/W line, so the "ROM" label is arbitrary; for instance, the "ROM" area could be static RAM (with battery backup, if desired).

Addresses in this section are shown as 6 digit hexadecimal numbers. The most significant digit is an "x"; it stands for the RMS chip select, which must be satisfied before any address decoding can occur.

If less than 1 Mbyte of DRAM is used, it maps to more than one address. This is discussed in Section 1Ø.3.

1Ø.1 M68ØØØ Family MPU's

The MC68ØØØ family MPU's can use a map with 6Ø Kbytes of ROM or one with 252 Kbytes of ROM. With each of these, the exception vectors can be in ROM or DRAM. See Table 1Ø-1.

The amount of ROM in the map is selected by the MAPA bit in the Border Color register (see Section 9.3.4). Resetting this bit selects the larger amount of ROM. It is reset at system reset.

After system reset, the exception vectors are decoded in ROM, and the bottom 1 Kbyte block of DRAM is not accessible to the RMS. Setting the VEC bit in the Page Independent Block register (see Section 9.3.7) selects DRAM exception vectors, and the ROM vectors are no longer accessible. When VEC is reset the S bus generates a Ø for the exception vector addresses; this can be used to select a separate ROM. When VEC is set, these addresses generate an S bus value of 7, the same DRAM/RMS code as the memory directly above the exception vectors. The ROM/DRAM transition may require transferring the ROM vectors to DRAM; if it does, either the MPU must load a vector from ROM, toggle the VEC bit, and store it to DRAM, repeating this until all vectors are set, or copy the vectors to a block of DRAM that is always accessible, set the VEC bit, and copy them back. This assumes that the ROM and DRAM are not multiply-mapped. The ROM can be multiply-mapped by adding logic to the S decoding. The DRAM is multiply-mapped if any of its most significant address lines are unused, as they are if 512 Kbytes or less is installed. If they are multiply-mapped, of course, they are both available to the processor with either VEC setting, so the DRAM can be set while the ROM is selected, then the VEC bit set.

The RMS provides DTACK to the MPU for all decodes except I/O blocks D and E. The user must supply either DTACK or the MC68ØØ peripheral interface signals for these blocks.

Chip Select	MAPA	VEC	Start	End	S Bus	Name	Size
Ø	Ø	Ø	ØØØØØ	ØØ3FF	Ø	Excp Vector	1K
			ØØ4ØØ	BFFFF	7	DRAM	767K
			CØØØØ	CFFFF	1	ROM 9	64K
			DØØØØ	DFFFF	2	ROM A	64K
			EØØØØ	EFFFF	3	ROM B	64K
			FØØØØ	FEFFF	4	ROM C	6ØK
			FFØØØ	FF7FF	5	I/O D	2K
			FF8ØØ	FFBFF	6	I/O E	1K
			FFCØØ	FFDFF	7	Reserved	256
			FFEØØ	FFEBF	7	RMS Registers	192
			FFECØ	FFFFF	7	Reserved	576
Ø	Ø	1	ØØØØØ	BFFFF	7	DRAM	768K
			CØØØØ	CFFFF	1	ROM 9	64K
			DØØØØ	DFFFF	2	ROM A	64K
			EØØØØ	EFFFF	3	ROM B	64K
			FØØØØ	FEFFF	4	ROM C	6ØK
			FFØØØ	FF7FF	5	I/O D	2K
			FF8ØØ	FFBFF	6	I/O E	1K
			FFCØØ	FFDFF	7	Reserved	256
			FFEØØ	FFEBF	7	RMS Registers	192
			FFECØ	FFFFF	7	Reserved	576
Ø	1	Ø	ØØØØØ	ØØ3FF	Ø	Excp Vectors	1K
			ØØ4ØØ	EFFFF	7	DRAM	959K
			FØØØØ	FEFFF	4	ROM C	6ØK
			FFØØØ	FF7FF	5	I/O D	2K
			FF8ØØ	FFBFF	6	I/O E	1K
			FFCØØ	FFDFF	7	Reserved	256
			FFEØØ	FFEBF	7	RMS Registers	192
			FFECØ	FFFFF	7	Reserved	576
Ø	1	1	ØØØØØ	EFFFF	7	DRAM	96ØK
			FØØØØ	FEFFF	4	ROM C	6ØK
			FFØØØ	FF7FF	5	I/O D	2K
			FF8ØØ	FFBFF	6	I/O E	1K
			FFCØØ	FFDFF	7	Reserved	256
			FFEØØ	FFEBF	7	RMS Registers	192
			FFECØ	FFFFF	7	Reserved	576
1	X	X	RMS Not Selected			S Bus = 7	

Table 1Ø-1 MC68ØØX Memory Maps

1Ø.2 The MC68Ø9E MPU

The MC68Ø9E can make use of the Machine 1 memory map with its 1 Mbyte of addressing capability, but to do so it must use the PIB and Paging control registers. The MC68Ø9E can only address 64 Kbytes of the memory map at one time. By using the PIB's and Paging register it can control which parts of the total 1 Mbyte make up the accessible 64 Kbytes.

The lower PIB is used to select a block of DRAM that permanently resides at the bottom of the MPU's 64 Kbyte address space. The MPU determines the size of this block by programming the lower PIB bits in the PIB control register. These bits allow the user to select a lower PIB size of Ø to 32K bytes. See Section 9.3.7.

The user does not have control over which addresses in the 1 Mbyte of memory are used for the lower PIB. The locations used for the lower PIB always begin at physical address $xØØØØØ and extend up as far as they have to in order to supply the amount of DRAM requested by the user.

The upper PIB is very similar to the lower PIB. It has the same type of function, and the same size options are available. The difference is that the locations that make up the upper PIB are taken from the top of the physical memory and reside in the top of the MPU's 64K address space.

The programming bits for the upper PIB are located in the same control register byte as the lower PIB. Both PIB's have enable bits that must be set before the PIB can be used. The PIB registers are programmed as shown in Table 1Ø-2.

Bit 2	Bit 1	Bit Ø	PIB Size
Ø	Ø	Ø	256 bytes
Ø	Ø	1	512 bytes
Ø	1	Ø	1K bytes
Ø	1	1	2K bytes
1	Ø	Ø	4K bytes
1	Ø	1	8K bytes
1	1	Ø	16K bytes
1	1	1	32K bytes

Table 1Ø-2 PIB Size Encoding

The lower PIB may be used to keep a small amount of DRAM permanently located at the bottom of the memory map, where it can be used for scratch RAM.

The upper PIB is designed to keep the RMS control registers, some I/O, and possibly some program ROM, in the memory map at all times.

Following reset with a MC68Ø9E MPU, the control registers are in their unfolded mode, the PIB's are not enabled, and the paging register bits are set so that PGØ to PG3 are 1's and SWAP is Ø. The user should make certain that at least the upper PIB is set up before changing the

					S		
MAPA	Unfold	Start		End	Bus	Name	Size
Ø	Ø	ØØØØØ	-	F7FFF	7	DRAM	992K
		F8ØØØ	-	F9FFF	1	ROM Ø	8K
		FAØØØ	-	FBFFF	2	ROM 1	8K
		FCØØØ	-	FFEFF	3	ROM 2	16K - 256
		FFFØØ	-	FFF1F	4	I/O Ø	32
		FFF2Ø	-	FFF3F	5	I/O 1	32
		FFF4Ø	-	FFF5F	6	I/O 2	32
		FFF6Ø	-	FFF7F	7	Reserved	32
		FFF8Ø	-	FFFBF	7	RMS Registers	64
		FFFCØ	-	FFFDF	7	Reserved	32
		FFFEØ	-	FFFFF	2	ROM 1 (Vectors)	32
Ø	1	ØØØØØ	-	F7FFF	7	DRAM	992K
		F8ØØØ	-	F9FFF	1	ROM Ø	8K
		FAØØØ	-	FBFFF	2	ROM 1	8K
		FCØØØ	-	FFDFF	3	ROM 2	16K - 512
		FFEØØ	-	FFEBF	7	RMS Registers	192
		FFECØ	-	FFEFF	7	Reserved	64
		FFFØØ	-	FFF1F	4	I/O Ø	32
		FFF2Ø	-	FFF3F	5	I/O 1	32
		FFF4Ø	-	FFF5F	6	I/O 2	32
		FFF6Ø	-	FFFDF	7	Reserved	128
		FFFEØ	-	FFFFF	2	ROM 1 (Vectors)	32
1	Ø	ØØØØØ	-	FFEFF	7	DRAM	1M - 256
		FFFØØ	-	FFF1F	4	I/O Ø	32
		FFF2Ø	-	FFF3F	5	I/O 1	32
		FFF4Ø	-	FFF5F	6	I/O 2	32
		FFF6Ø	-	FFF7F	7	Reserved	32
		FFF8Ø	-	FFFBF	7	RMS Registers	64
		FFFCØ	-	FFFDF	7	Reserved	32
		FFFEØ	-	FFFFF	2	ROM 1 (Vectors)	32
1	1	ØØØØØ	-	FFDFF	7	DRAM	1M - 512
		FFEØØ	-	FFEBF	7	RMS Registers	192
		FFECØ	-	FFEFF	7	Reserved	64
		FFFØØ	-	FFF1F	4	I/O Ø	32
		FFF2Ø	-	FFF3F	5	I/O 1	32
		FFF4Ø	-	FFF5F	6	I/O 2	32
		FFF6Ø	-	FFFDF	7	Reserved	128
		FFFEØ	-	FFFFF	2	ROM 1 (Vectors)	32

Table 1Ø-3 MC68Ø9E Memory Maps

paging register. If the paging register is changed before the upper PIB is set up, the control registers may disappear from the memory map and it will be impossible to get them back.

Following reset, the MAPA control bit is reset and the second memory map in Table 1Ø-2 is available to the user. Folding the RMS registers gives the first memory map in Table 1Ø-3. See Section 9.3.4 for the MAPA bit's location.
If the user sets the MAPA bit to select the alternative memory map, the third memory map is available. Folding the RMS registers gives the fourth memory map in Table 1Ø-3.

ROM 1 appears in two places when MAPA is reset. The last 32 locations of ROM 1 are found in two places in the memory map, so they can be used as reset and interrupt vectors.

1Ø.3 Multiple DRAM Mapping

If less than the full 1 Mbyte of DRAM is used with the RMS, the DRAM maps to more than one address, since from 1 to 6 of the most significant address bits are not connected. Table 1Ø-4 shows the number of banks and the DRAM types in all combinations. The basic address range starts at address $ØØØØØ where all unused address lines are zero; the first remap starts directly above the basic address range, where the least significant unused address line is a one and the others are zero.

Number of Banks	DRAM Type	Total DRAM (Bytes)	Last Address of Basic Address Range
1	16K x 1	16K	Ø3FFF
2		32K	Ø7FFF
4		64K	ØFFFF
1	16K x 4	16K	Ø3FFF
2		32K	Ø7FFF
4		64K	ØFFFF
1	64K x 4	64K	ØFFFF
2		128K	1FFFF
4		256K	3FFFF
1	256K x 1	256K	3FFFF
2		512K	7FFFF
4		1M	FFFFF

Note: 1-bank memory applies only to MC68Ø9E and MC68ØØ8.

Table 1Ø-4 Multiple Mapping of DRAM

One-bank maps apply only to the MC68ØØ8 and MC68Ø9E. The number of remaps of DRAM depends both on the total DRAM amount and on the selected memory map. For example, an MC68ØØ8 with one 16K bank can access 48 copies if MAPA is Ø and 6Ø copies if MAPA is 1. The MC68Ø9E situation is more complex because of the many combinations of PIB sizes.

Because the DRAM is contiguously remapped, the MPU can treat the DRAM as starting at any address and ending at that address plus the DRAM amount, as long as it does not overlap I/O and ROM. Using this technique, an MC68ØØØ family MPU with ROM exception vectors can use all of a 256 Kbyte system's DRAM by ignoring its basic address range and addressing its second map at $4ØØØØ-$7FFFF; it could also use all of a 512 Kbyte system's DRAM by addressing it at $ØØ4ØØ-$8Ø3FF, which a 1 Kbyte shift to get at the 1 Kbyte of DRAM blocked by the vector ROM.

One danger is shared by MC68ØØØ family MPU's using DRAM exception vectors and MC68Ø9E's using the lower PIB: writing off the upper end of DRAM begins overwriting the contents of these special areas. The symptoms of this error vary from the subtle to the spetacular, and they are data-dependent.

11.0 VIDEO OVERLAY

It is possible to synchronize the video output of the RMS with another source of video. This external video source is considered to be the master, and the RMS is the slave which changes its video timing to match the master timing.

11.1 Vertical Synchronization

Synchronizing the vertical sync pulses of the RMS to those of the external video signal is the first step in achieving synchronization. This is accomplished by setting up the RMS' Sync Mode register so that the RMC's SYNC pin can act as an input for the external signal's field sync. See Section 9.3.19. The leading edge of this field sync signal should be a rising edge.

11.2 Horizontal Synchronization

The second step in synchronization is to match horizontal sync from the two sources. This is done by means of a PLL circuit that compares the trailing edge of the external video's horizontal sync to the trailing edge of HSYNC. HSYNC is generated by the RMC and available at a pin. The output of the PLL is used to control the 36 MHz master oscillator of the RMS located at the RMI. See Figure 4-3.

This oscillator can still be crystal controlled in this application. The PLL circuit will be able to change its frequency enough to pull the horizontal sync pulses into sync quickly. It is quite easy to change the frequency of the master oscillator by a minimum of 1500 Hz. This much change means the horizontal syncs can be brought together in a few seconds, even in the worst case.

Achieving a match on horizontal and vertical sync is all that is required if the video signals are RGB. If they are composite video, one more stage is required to match the color subcarriers. This final stage does not affect the RGB application, so it is optional there.

11.3 Color Subcarrier

If the user desires a composite video output signal, the RGB outputs of the RMS must be converted to composite video. The color burst of the new composite video must agree in frequency and phase with the external video for proper results.

The modulator used to change the RGB to composite video needs a source of color subcarrier. In most applications it is easiest to use the color subcarrier output provided on the RMI (CSC). In order for this to work without color errors, the master oscillator of the RMI should be injection locked to the burst signal on the external source of composite video.

This achieves final synchronization for all applications.

11.4 Video Switching

If overlay is in use, then each of the video sources should be providing part of the final video signal. The share that comes from each source would be controlled by time division multiplexing.

The VIDEN signal that comes from the RMC is user programmable and appropriate for use as the select signal for a high speed video multiplexer. See Section 9.3.25 on Color Mapping RAM for control of the VIDEN output.

12.Ø REAL TIME SOFTWARE

This chapter highlights some of the display changes that the user can make by modifying the contents of the RMS control registers during each video field. It also points out some of the pitfalls the user should avoid.

12.1 Reusing True Objects

Assume that the user has programmed true object 1 to display a figure near the top of the video screen. After the object hardware has finished displaying this figure, it goes idle until the next field. There will be several milliseconds of active display time during which the object logic is idle. The user can reprogram the object logic to display another figure farther down the screen. The restriction is that the two figures may not have any video data displayed on the same scan line.

The Object Available register and object finished interrupt may be useful to the user to determine when to reprogram the object logic. The object available flag for each object is reset when the user writes to that object's X Coordinate register. It is set when the hardware has completed displaying the object. The rising edge of any object available bit may cause an object finished interrupt, signifying that some object hardware has finished its task for this video field.

It would also be possible for the user to make use of the RTO interrupt to signal that objects are available.

Any or all of the object registers (position, name, attributes) may be changed when the object is reused.

12.2 CMR Registers

The user may want to change the data in some CMR registers, either during each field, or between fields. This technique can be used to allow more than 32 colors to be displayed on a single screen, or to cause some screen areas to change color.

The RMS cannot access any individual CMR register in the same memory cycle that the MPU is accessing it, so if the MPU is reading or writing to a CMR register when that color is displayed, several pels can be undefined. There is no difficulty with MPU access to CMR registers not being read by the RMS. The following are examples of MPU accesses to the CMR that do not cause video errors:

1) Accessing CMRØØ, at any point in a scanline that does not use CMRØØ.
2) Accessing CMR1F, at any point in a scanline after its last display of CMR1F's color.
3) Accessing CMRØ9, before the scanline gets to the display of CMRØ9's color (but see below).

4) Accessing any CMR register except the one containing border color, when border color is displayed.
5) Accessing any CMR register including the one containing border color, during horizontal or vertical blanking.

Users of method 3 above must be aware of two points if they try to make CMR changes close to the CMR's use: there is pipeline delay, so the CMR data is fetched ahead of display time; and there are RMS cycles that steal MPU cycles during the border display, so cycle counting can fail as a way to calculate when it is safe to write to a register before it is used. The number of stolen cycles is higher for 1-bank memory systems than for 2-bank or 4-bank memory systems. The timing of the cycle stealing is HRES mode dependent.

12.3 Displayed Screen Address

The user can change the displayed screen in real time by changing the Virtual Screen Start Address, Size, and Width registers, by changing the offset and scrolling registers, or both. These registers are discussed in Sections 9.3.8, 9.3.9, 9.3.2Ø, 9.3.21, and 9.3.22. These registers can be written to at any time, but if they are being read by the RMS at the same time one field may be affected.

The Virtual Screen Start Address, Size, and Width registers are read by the RMS once per scanline (bit-plane) or character row (list mode); the offset and scrolling registers are read once during each memory cycle that prepares data to display. These registers should be changed early in the blanking interval; because of the pipeline delay, the RMS must have the values for the next field before it starts. The blanking interrupt gives enough time to change these registers if they are changed quickly. The time available varies with display mode and horizontal resolution.

12.4 Scrolling

The RMS is designed to allow efficient smooth scrolling of the displayed screen anywhere within the virtual screen in real time. The RMS does this using the Horizontal and Vertical Offset registers and the Horizontal and Vertical Scroll registers. The Wrap Control bit in the Border Color register (see Section 9.3.4) affects scrolling but is not intended for modification in real time.

For clean scrolling, the registers should be changed after the last time the RMS reads them for the current field and before it reads them for the next field. The blanking interrupt provides a convenient indication of this point. Because of the pipeline delay in list modes, the registers should be changed soon after the blanking interrupt. Bit-plane timing is less critical.

12.4.1 Vertical Scrolling

Vertical scrolling a scanline at a time in bit-plane or a character row at a time in list mode can be done using only the Vertical Offset register. When it is zero, the top scanline/character row on the displayed screen is the first line/row in the virtual screen. Adding the contents of the Virtual Screen Width register to the Vertical Offset register moves the displayed screen's top line/row down to the virtual screen's second line/row. Since the displayed screen is fixed with respect to the viewer, this appears as if the virtual screen moved up one line/row. This register can be set to any integer multiple of the virtual screen width, subject to the limitations described in Section 9.3.21.

Vertical scrolling within a character row in the list modes can be done using just the Vertical Scroll register. When the register is zero, the top scanline of the displayed screen is the first scanline from the character row pointed to by the Vertical Offset register. When 1 is added to the scroll register, the top scanline of the displayed screen becomes the second scanline of that character row. Character rows can be 8, 1Ø, 12, or 16 scanlines high; the Vertical Scroll register must not exceed 7, 9, 11, or 15, respectively.

Vertical scrolling across character row boundaries requires simultaneous changes to both registers. If the scroll register contains its maximum value (1 less than the number of scanlines in a character row), the top scanline of the displayed screen will be the bottom scanline of the character row pointed to by the offset register. To move the top of the displayed screen down to the top of the next character row, the scroll register must be set to zero, and the offset register must be increased by the number of bytes in the size register. Similarly, if the scroll register is zero, the top scanline of the displayed screen is the first scanline in the character row pointed to by the offset register; to move the displayed screen up one scanline, to the bottom of the character row above the current top displayed row, the scroll register must be set to its maximum value (one less than the number of scanlines in a character row) and the number of bytes in the size register must be subtracted from the offset register.

12.4.2 Horizontal Scrolling

Horizontal scrolling a memory cycle at a time in bit-plane or list mode can be done with just the Horizontal Offset register. When it is zero, the first pixel of active video on each line of the displayed screen comes from the first memory cycle of the virtual screen; adding the number of bytes in a memory cycle (see Section 9.3.22) moves the displayed screen's leftmost pixels to the second memory cycle of the virtual screen. This appears to the viewer to be a shift to the left by the virtual screen.

Horizontal scrolling one pixel at a time within a memory cycle is done using the Horizontal Scroll register. When it is zero, the leftmost pixel in the displayed screen is the first pixel of the memory cycle pointed to by the Horizontal Offset register. Adding 1 to the scroll register shifts the displayed screen to the right by one pixel, which appears to be a one-pixel left shift by the virtual screen. The number of pixels in a memory cycle depends only on the HRES mode: there are 8 pixels for HRESØ-4 and 16 pixels for HRES6 and 7. Exceeding 7 in the low resolution modes and 15 in the high resolution modes causes an undefined display.

For horizontal single pixel scrolling from one memory cycle to the next, both horizontal registers must change. If the displayed screen is being moved to the left within the virtual screen (which appears to the viewer as the data moving to the right), then the Horizontal Scroll register must be decremented. When it is zero, the next shift will require that the Horizontal Offset register have the number of bytes in one memory cycle subtracted from it, and the Horizontal Scroll register must be set to one less than the number of pels in a memory cycle: set it to 15 for HRES modes 6 and 7, and to 7 for modes Ø-4. Similarly, horizontal scrolling the visible screen to the right one pel at a time requires setting the scroll register to zero and increasing the offset register by the number of bytes in a memory cycle when the scroll register reaches 15 (HRES 6, 7) or 7 (HRES Ø-4). The method is identical for bit-plane and list modes.

12.4.3 Scrolling off the Edge

This discussion has assumed so far that the offset register was not moving outside the virtual screen. If wraparound is desired, when the offset is already at one of its limits and it needs to move a step beyond that, it should be set to the other limit. If the Wrap Control bit is reset, that means setting the register to zero if it is already at its maximum value, or setting it to its maximum value if it was already zero. If the Wrap Control bit is set (valid only for bit-plane), the limits are plus and minus the upper limit.

12.4.4 Vertical Scrolling with Double High Characters

The Double High Preset bit in the Vertical Scroll register allows vertical scrolling when characters using the double high attribute (Section 7.2.4.6) are displayed. To display a double high character, the RMS requires that two character rows contain identical characters with the double high attribute selected in both, one character below the other on the displayed screen. The top character is used to display the top half of the double high image, and the bottom character is used for the image's bottom half. The display of each character row is an independent event to the RMS, so it must have a way to decide whether to show the top or the bottom half of a double high character when it finds one in the display list. An internal flag is used for this; when it is reset, the RMS displays the top half, and when it is set, it displays the bottom half. At the beginning of each field, the flag is set to the value of the DHP bit.

For each following character row, the flag is toggled, unless the row just displayed had no double high characters. In that case, the flag is reset. This allows the next row with a double high character to always display its top half no matter how many rows later it is.

For example, if the DHP bit were reset, the first character row would display the top halves, and the flag would toggle to a 1 at the end of the row. The second row would display the bottom halves, then the flag would toggle back to a Ø. If there were no double high characters in the third row, the flag would be reset, and it would be reset in each subsequent row until another double high character were found. It would then be set for the row following that one.

The displayed screen starts at the Vertical Offset Register; the Vertical Scroll register indicates the scan line within the row to use as the first scan line of the displayed screen. Even if the Scroll register is at its maximum value, so that only one scanline of the character row is displayed, that row is the first row on the screen as far as the internal double high flag is concerned.

The sequence of steps to scroll the displayed screen down through a double high character is as follows. In this example, there are double high characters in the top character row, and the Vertical Scroll register is initially zero. The scroll register being zero means that the DHP bit is zero and that the top scanline of the character row is also the top scanline of the displayed screen. The first row will display the top halves, and the bottom row will display the bottom halves. At each blanking interval (or each Nth blanking interval, if slower scrolling is wanted), add 1 to the scroll register. This causes the the displayed screen to start one scanline lower in the top character row, which has the appearance to the viewer of the displayed data moving upward on the CRT. After 7, 9, 11, or 15 increments, depending on the number of scanlines per character row, only the bottom scanline of the top character row is displayed. All through these increments, DHP has remained reset and the Vertical Offset register has continued to point to the character row containing the top halves of the characters, so the display has been correct. At this point, three actions must take place during the next blanking interval: the Vertical Offset register must be increased by the contents of the Virtual Screen Width register (which causes it to point at the character row containing the bottom halves of the double high characters), the DHP bit must be set (to cause those bottom halves, now in the top row, to display as bottom halves), and the scrolling bits in the Vertical Scroll register must be cleared (to start the displayed screen at the top scanline of the new character row).

Throughout this process the internal flag was toggled after the character row with the bottom halves, so the RMS was ready to start new double high characters in the third row or any following row.

The step size in the above example is not the only choice; instead of stepping one scanline at a time, the display could move one character row at a time by changing the offset register, setting the DHP bit as needed, and leaving the scrolling bits in the scroll register alone. Any other combination of changes is also valid.

Also important is the issue of what part of the display list is searched to determine if a character row has double high characters: in each row, the search starts with the first displayed character (even if it has been horizontally scrolled so that only its rightmost column of pels is visible) and goes through the last displayed character. If the sides of the displayed screen are on character boundaries, the character to the right of the displayed screen is searched, if there is one.

12.4.5 Horizontal Scrolling with Double Wide Characters

The Double Wide Preset bit in the Horizontal Scroll register allows horizontal scrolling when characters using the double wide attribute (Section 7.2.4.7) are displayed. When DWP is reset, double wide characters in the first displayed character position of all character rows are displayed so that the first position contains the left half of the character and the second character position contains the right half. The character underneath the right half is not displayed. If DWP is set, the first position contains the right half of the double wide character, and the second position contains the second character displayed normally.

13.Ø PIN DRIVE AND LOADING

Each pin of the RMI and RMC is examined here. The drive capability of output pins is specified, and the loading of input pins is also described. Input/output pins are listed in both categories.

These ratings apply over a temperature range of Ø to 7Ø°C with a power supply voltage of 5.ØV plus or minus 5%. The specified values are true under the worst-case conditions.

These values are believed to be accurate, but the data sheet takes precedence.

13.1 RMI Inputs

The XØ-X9, HSYNC, R/W, LDS(A5), UDS(A7), AS(A6) and CS inputs of the RMI all meet the following specification:

The minimum voltage recognized as a high level is 2.Ø volts. The maximum level recognized as a low is Ø.8 volts.

The input current with an input voltage of 2.7 volts will not exceed 2Ø microamps.

The input current with an input voltage of Ø.4 volts will not be less than -Ø.2 milliamps.

The capacitance of an input pin will not exceed 1Ø picofarads.

13.2 RMI Outputs

The RMI outputs, SØ-S2, ADEN, CSC, DBEN, VTCLK, PCLK, and MTCLK all meet the following specification:

The high level output voltage will be at least 2.7 volts while sourcing 4ØØ microamps of current.

The low level output voltage will be no more than Ø.5 volts while sinking 8 milliamps of current.

The ZØ-Z8, CASØ-CAS3, WE and RAS outputs have more drive capability and meet the following specification:

The high level output voltage will be at least 2.4 volts while sourcing up to 3.Ø milliamps.

The low level output voltage will not exceed Ø.5 volts while sinking 24 milliamps.

The CLK(E) and DTACK(Q) outputs have the following specification:

The high level CLK(E) output voltage will be at least Vcc-.75 volts while sourcing Ø.2 milliamps.

The high level DTACK(Q) output voltage will be at least 2.7 volts while sourcing 4ØØ microamps of current.

The low level CLK(E) and DTACK(Q) output levels will not exceed Ø.5 volts while sinking 4.Ø milliamps.

13.3 RMC Inputs

The RMC inputs XØ-X9, AØ-A7, and BØ-B7 all meet the following specification. They are all bidirectional pins.

As inputs their current requirements will not exceed 2Ø microamps.

The voltage level required to be recognized as a high will not exceed 2.Ø volts.

The voltage level for a low to be recognized will not be less than Ø.8 volts.

The ADSEL, VTCLK, MTCLK, PCLK, R/W, RTI, CASØ-CAS3, and DBEN inputs meet the above voltage levels, but their current requirements will not exceed 2.5 microamps.

The pin capacitance of any input will not exceed 1Ø picofarads.

13.4 RMC Outputs

As outputs, XØ-X9, AØ-A7, BØ-B7, and INT meet the following specification:

With a high level out, the voltage will be at least 2.4 volts while sourcing a current of 4ØØ microamps.

With a low level out, the voltage will be no more than Ø.5 volts while sinking 5.3 milliamps.

The outputs HSYNC, REN, VIDEN, and SYNC will also provide 2.4 volts at 4ØØ microamps for a high level out, but will only sink 1.6 milliamps at Ø.5 volts for a low level.

The R, G and B video outputs provide special voltage levels out. They are designed to drive a DC load of 1Ø Kohms connected to ground, at voltage levels of 1.5 volts to 2.5 volts, depending on the value of the 4-bit number provided to the D/A converter that drives these pins. The nominal voltage level for each of the possible 4-bit digital words is listed below.

D/A Converter Input	Voltage Out
$F	2.5
$E	2.4375
$D	2.375
$C	2.3125
$B	2.25
$A	2.1875
$9	2.125
$8	2.Ø625
$7	2.Ø
$6	1.9375
$5	1.875
$4	1.8125
$3	1.75
$2	1.6875
$1	1.625
Blanking	1.5
Sync Tip	1.Ø7

Table 13-1 D/A Converter Output Voltages

All outputs can reach blanking level. Only the G output can reach sync tip.

14.0 MACHINE 2

Machine 2 allows software compatibility with the MC6883-MC6847 combination when the RMS is used with an MC6809E MPU. The RMS requires the hardware design discussed in Section 2.1, Chapters 3 through 5, and parts of Chapter 6, and it needs initialization software different than the MC6883-MC6847. If this is done, existing software will run using the RMS. This hardware arrangement also supports the full power of Machine 1 operation, limited only by the amount of DRAM being used.

Once the RMS registers are initialized, MC6883-type control bits and a PIA-simulating register are used to control the RMS.

The Machine 2 mode is invoked by setting the M2 bit in the Memory Map register. The RMS looks like two machines at once while it is in Machine 2 mode. In addition to the normal control register map, other control bits appear in the memory map. These bits do not add new capabilities to the system. Instead, they offer a way of controlling some of the RMS' functions that is backwards compatible with the MC6883 and MC6847.

The MC6883 control bits work differently than the normal RMS controls. Each control bit occupies two bytes in the memory map. Writing to the odd byte sets the control bit, and writing to the even byte resets the control bit. Reading these locations has no effect on the control registers and does not return control register information.

The control bits and their read and write addresses follow.

Bit	Set Address	Reset Address
TY	$FFDF	$FFDE
M1	NOT USED BY RMS	
M0	NOT USED BY RMS	
R0	NOT USED BY RMS	
R1	NOT USED BY RMS	
P1	$FFD5	$FFD4
F6	$FFD3	$FFD2
F5	$FFD1	$FFD0
F4	$FFCF	$FFCE
F3	$FFCD	$FFCC
F2	$FFCB	$FFCA
F1	$FFC9	$FFC8
F0	$FFC7	$FFC6
V2	$FFC5	$FFC4
V1	$FFC3	$FFC2
V0	$FFC1	$FFC0

Table 14-1 MC6883 Control Bits

The TY control bit selects between two system memory maps. These two system memory maps are decoded by the RMS and are available on the S bus. See Tables 14-5 and 14-6.

The M1 and MØ bits were originally used to define the size of the RAM devices that would be used in the system. Since the user has already had to program the RMS' memory organization control registers, the M1 and MØ bits are redundant and have no effect on the system. The M1 and MØ bits allowed choices between 4K, 16K and 64K by 1 DRAM's. The user is not restricted to these choices. The normal choices explained in Chapter 3 and Section 9.3.17 apply. Since these choices allow more DRAM than the MC6883 allowed, this is not a restriction on Machine 2 operation.

The R1 and RØ bits originally gave the user control over MPU execution speed; the MPU could be placed in a high speed mode. These bits have no effect in the RMS.

The P1 bit is a paging bit for use with memory map Ø. It is used in place of A15 for addresses from $ØØØØ through $7FFF. This allows the ROM/RAM map to contain two 32K pages of RAM in the lower half of the address range. See Table 14-5.

The FØ through F6 bits allow the display to be shifted from its base location at $ØØØØ. The address of the top left corner of the display is:

DISPLAY START = $ØØØØ + (512 * F)

where "F" is F6 (MSB) through FØ (LSB).

The V bits are part of the bits used to determine the display mode of the RMS. They are discussed in Section 14.3.

The other control bits are the MC6847 control bits. They are located in the system memory map as if they were being controlled by a PIA. They are located at $FF22 in the system memory map as follows.

Machine 2 Register

B7	B6	B5	B4	B3	B2	B1	BØ
G/A		GM1	GMØ	CSS			

G/A is equivalent to the MC6847's Graphics/Alpha bit.

The Graphics Mode bits GM1 and GMØ provide a superset of the MC6883-MC6847 display modes. As discussed in Section 14.3, they resemble GM1, GMØ, and E/I on the MC6847. CSS is the same as the MC6847's Color Set Select bit.

14.1 Machine 2 Initialization

At power-up or reset, the RMS goes into the unfolded map, Machine 1 state. The RMS registers should be initialized as follows. See Chapter 9 for register addresses. Register page 00:

Registers	Value	Comments
Memory Map	$10	Machine 2, Folded Map Page 00
Display Data Mode	$00	
Interrupt Status	$00	
Border Color	$10	CMR10, Video Disabled
Object Available	N/A	Read Only
Paging	$0F	Page F
Page Independent Blocks	$00	
Vertical Scroll	$00	
Horizontal Scroll	$00	
DRC Start Address	$0000	
True Object Start Address	$0000	
Fixed Object Start Address	$0000	
Collision Status	N/A	Read Only
Collision Enable	$00	
Real Time Output	$00000000	
Real Time Input	$00000000	
Memory Organization	As Described in Section 9.3.17	
Video Operation	$12	256 x 192
Sync Mode	$04	Composite Sync Output
Virtual Screen Start Address	$00000000	
Vertical Offset	$00000000	
Horizontal Offset	$00000000	
Virtual Screen Size	$00001800	6 Kbytes
Virtual Screen Width	$00000020	32 Bytes

Table 14-2 Machine 2 Initialization Page 00

This completes the initialization of RMS register page 00. To get to RMS register page 01, write $50 to the Memory Map register. To get to page 10, write $90.

Start Address	End Address	Name	Size
$xFØØØØ	$xFFFØØ	RAM	64K-256
$xFFFØØ	$xFFF1F	I/O$_{Ø}$	32
$xFFF2Ø	$xFFF22	Machine 2 Register	3
$xFFF23	$xFFF3F	I/O$_{1}$	29
$xFFF4Ø	$xFFF5F	I/O$_{2}$	32
$xFFF6Ø	$xFFF7F	Reserved	32
$xFFF8Ø	$xFFFBF	RMS Registers	64
$xFFFCØ	$xFFFDF	Mach 2 Ctrl Bits	32
$xFFFEØ	$xFFFFF	ROM$_{1}$	32

Table 14-6 Machine 2 Memory Map (TY = 1)

The RMI S bus may be decoded in the same fashion as the MC6883 S bus, in order to provide chip selects for the entire system.

14.3 Machine 2 Display Modes

The RMS offers a variety of display modes, as shown in the following table:

V2	V1	VØ	GM1	GMØ	G/A	Display Mode
Ø	Ø	Ø	Ø	Ø	Ø	Alpha-1/Semigraphics
Ø	Ø	Ø	Ø	1	Ø	Alpha-2/Semigraphics
Ø	Ø	Ø	1	Ø	Ø	Alpha-3/Semigraphics
Ø	Ø	Ø	1	1	Ø	Alpha-4/Semigraphics
Ø	Ø	1	x	Ø	1	Color Graphics One
Ø	Ø	1	x	1	1	Resolution Graphics One
Ø	1	Ø	x	Ø	1	Color Graphics Two
Ø	1	1	x	1	1	Resolution Graphics Two
1	Ø	Ø	x	Ø	1	Color Graphics Three
1	Ø	1	x	1	1	Resolution Graphics Three
1	1	Ø	x	Ø	1	Color Graphics Six
1	1	Ø	x	1	1	Resolution Graphics Six

Table 14-7 Display Mode Selection

The Color Set Select bit can be used with every display mode to choose between two sets of colors, as shown in the display tables in the following sections.

14.3.1 Alpha/Semigraphics

The 4 types of alpha/semigraphics are supersets of the MC6847's Alphanumeric Internal and Semigraphics 4 display modes. Each can display 16 rows of characters, with 32 characters per row.

14.3.1.1 Semigraphics

Semigraphics are selected by resetting the three V bits and setting the display data byte's most significant bit. The 7 least significant bits are used exactly as in the MC6847. All semigraphics patterns and colors can be used in any combination with the selected alphanumeric characters. Figure 14-1 shows the sixteen possible patterns, with the foreground color shown black and the background color shown white. The display data byte is:

Semigraphic Block

B7	B6	B5	B4	B3	B2	B1	BØ
1	C2	C1	CØ	L3	L2	L1	LØ

Lx	C2	C1	CØ	Color	CMR Address
Ø	x	x	x	Black	Ø8
1	Ø	Ø	Ø	Green	ØØ
1	Ø	Ø	1	Yellow	Ø1
1	Ø	1	Ø	Blue	Ø2
1	Ø	1	1	Red	Ø3
1	1	Ø	Ø	Buff	Ø4
1	1	Ø	1	Cyan	Ø5
1	1	1	Ø	Magenta	Ø6
1	1	1	1	Orange	Ø7

Table 14-8 Semigraphics Colors

14.3.1.2 Alphanumerics

The four alphanumerics sets allow all uppercase, upper and lowercase, and inverted versions of these. As shown at the beginning of this section, they are selected using GM1 and GMØ in the Machine 2 Register. The Color Set Select bit can be used to select the foreground and background colors, in conjunction with the GM1 bit, which serves as a screen invert when it is set.

CSS	GM1	Foreground Color	Background Color
Ø	Ø	Black (CMRØ8)	Green (CMRØØ)
Ø	1	Green (CMRØØ)	Black (CMRØ8)
1	Ø	Black (CMRØ8)	Orange (CMRØ7)
1	1	Orange (CMRØ7)	Black (CMRØ8)

Table 14-9 Alphanumerics Colors

The GMØ bit selects uppercase-only when reset, or upper and lowercase when set. The fonts are identical to Machine 1 as shown in Figure 7-1, and the placement of the 5x7 character in the 8x12 space is also the same: one blank line above and 4 below; one blank line to the right and 2 to the left. The two character sets are shown in Figure 14-2. Setting GM1 reverses these characters' foreground and background.

Alpha	GM1	GMØ	B7	B6	B5	Characters
One	Ø	Ø	Ø	Ø	Ø	Noninverted Uppercase Alphabet
			Ø	Ø	1	Noninverted Punctuation & Numbers
			Ø	1	Ø	Inverted Uppercase Alphabet
			Ø	1	1	Inverted Punctuation & Numbers
Two	Ø	1	Ø	Ø	Ø	Inverted Lowercase Alphabet
			Ø	Ø	1	Noninverted Punctuation & Numbers
			Ø	1	Ø	Inverted Uppercase Alphabet
			Ø	1	1	Inverted Punctuation & Numbers
Three	1	Ø	Ø	Ø	Ø	Inverted Uppercase Alphabet
			Ø	Ø	1	Inverted Punctuation & Numbers
			Ø	1	Ø	Noninverted Uppercase Alphabet
			Ø	1	1	Noninverted Punctuation & Numbers
Four	1	1	Ø	Ø	Ø	Noninverted Lowercase Alphabet
			Ø	Ø	1	Inverted Punctuation & Numbers
			Ø	1	Ø	Noninverted Uppercase Alphabet
			Ø	1	1	Noninverted Punctuation & Numbers

Table 14-1Ø Character Selection

Alpha Three is the invert of Alpha One, and Alpha Four is the invert of Alpha Four. In Alpha One and Three, the characters can be inverted on a character-by-character or a screen basis, but lowercase characters are not available. In Alpha Two and Four, the characters can be inverted only on a screen basis, but lowercase characters are available.

Alphanumeric Character

B7	B6	B5	B4	B3	B2	B1	BØ
Ø	CC6	CC5	CC4	CC3	CC2	CC1	CCØ

The display data requires 512 bytes.

14.3.2 The Graphics Modes

The 8 graphics modes are similar to Machine 1's bit-plane mode. They offer a variety of resolutions and color ranges and use from 1Ø24 to 6144 bytes to hold the display data. There are 4 modes that emphasize color and 4 that emphasize resolution. The color modes offer these color choices:

CSS	C1	CØ	Color	Border
Ø	Ø	Ø	Green (CMRØØ)	Green (CMR12)
	Ø	1	Yellow (CMRØ1)	
	1	Ø	Blue (CMRØ2)	
	1	1	Red (CMRØ3)	
1	Ø	Ø	Buff (CMRØ4)	Buff (CMR13)
	Ø	1	Cyan (CMRØ5)	
	1	Ø	Magenta (CMRØ6)	
	1	1	Orange (CMRØ7)	

Table 14-11 Color-Mode Colors

The resolution modes offer these color choices:

CSS	Lx	Color	Border
Ø	Ø	Black (CMRØ8)	Green (CMR12)
	1	Green (CMRØØ)	
1	Ø	Black (CMRØ8)	Buff (CMR13)
	1	Buff (CMRØ4)	

Table 14-12 Resolution-Mode Colors

Each byte of display data describes 4 picture elements in the color modes:

Color Mode Display Data

B7	B6	B5	B4	B3	B2	B1	BØ
3C1	3CØ	2C1	2CØ	1C1	1CØ	ØC1	ØCØ

Color Mode Display

E3	E2	E1	EØ

In the resolution modes, each byte of display data describes 8 picture elements:

Resolution Mode Display Data

B7	B6	B5	B4	B3	B2	B1	BØ
L7	L6	L5	L4	L3	L2	L1	LØ

Resolution Mode Display

L7	L6	L5	L4	L3	L2	L1	LØ

The modes differ from each other in the size of each picture element, and therefore in the number of elements on a screen, and therefore in the amount of required memory.

	C1G	C1R	C2G	C2R	C3G	C3R	C6G	C6R
Picture Element Height	3	3	3	2	2	1	1	1
Picture Element Width	4	2	2	2	2	2	2	1
Elements per Row	64	128	128	128	128	128	128	256
Number of Rows	64	64	64	96	96	192	192	192
Memory, Bytes	1Ø24	1Ø24	2Ø48	1536	3Ø72	3Ø72	6144	6144

Table 14-13 Display Memory Requirements

APPENDIX A

RMS REGISTER MAP

The following register map shows all 192 bytes in the RMS register space, including the unused ones. The least significant four hexadecimal digits of the addresses are shown: unfolded address to the left of each byte, and folded address/page number to the right. The full folded and unfolded addresses for each register are shown in Section 9.3.

```
        *********************************************************************************
        |  CTL REG PAGE     | UNFOLD  | MACHINE |         |         |         |         |
        .................................................................................
$FFEØØ  | MP1     | MPØ     | UF      | M2      |         |         |         |         |  $FFF8Ø/ØØ
        *********************************************************************************
        *********************************************************************************
        | BIT/LIST|      LIST MODE SELECT       | LINES PER ROW     | BITS PER PEL      |
        .................................................................................
$FFEØ1  | BP      | LM2     | LM1     | LMØ     | LPR1    | LPRØ    | BPP1    | BPPØ    |  $FFF81/ØØ
        *********************************************************************************
        *********************************************************************************
        | INTRPT  |         INTERRUPT BITS  (WRITE ENABLES AND READ STATUS)             |
        .................................................................................
$FFEØ2  | IPT     |         |         | RTI     | RTO     | OFN     | BLK     | COL     |  $FFF82/ØØ
        *********************************************************************************
        *********************************************************************************
        | WRAP    | MEM MAP | VID ENBL|                 BORDER COLOR                      |
        .................................................................................
$FFEØ3  | WC      | MAPA    | DV      | BC4     | BC3     | BC2     | BC1     | BCØ     |  $FFF83/ØØ
        *********************************************************************************
        *********************************************************************************
        |                           OBJECT AVAILABLE                                    |
        .................................................................................
$FFEØ4  | O7A     | O6A     | O5A     | O4A     | O3A     | O2A     | O1A     | OØA     |  $FFF84/ØØ
        *********************************************************************************
        *********************************************************************************
        |         |         |         | SWAP    |                PAGING                 |
        .................................................................................
$FFEØ5  |         |         |         | SWAP    | PG3     | PG2     | PG1     | PGØ     |  $FFF85/ØØ
        *********************************************************************************
        *********************************************************************************
        | UPPER PAGE INDEPENDENT BLOCK          | LOWER PAGE INDEPENDENT BLOCK          |
        .................................................................................
$FFEØ6  | UEN     | UPI2    | UPI1    | UPIØ    | LEN     | LPI2    | LPI1    |LPIØ/VEC|  $FFF86/ØØ
        *********************************************************************************
        *********************************************************************************
        |         |         |         |2H PRESET|     VERTICAL SCROLLING                |
        .................................................................................
$FFEØ7  |         |         |         | DHP     | VSC3    | VSC2    | VSC1    | VSCØ    |  $FFF87/ØØ
        *********************************************************************************
        *********************************************************************************
        |         |         |         |2W PRESET|     HORIZONTAL SCROLLING              |
        .................................................................................
$FFEØ8  |         |         |         | DWP     | HSC3    | HSC2    | HSC1    | HSCØ    |  $FFF88/ØØ
        *********************************************************************************
        *********************************************************************************
        |         |         |         |         |         |         |         |         |
        .................................................................................
$FFEØ9  |         |         |         |         |         |         |         |         |  $FFF89/ØØ
        *********************************************************************************
        *********************************************************************************
        |         DYNAMICALLY REDEFINABLE CHARACTER IMAGE TABLE START ADDRESS           |
        .................................................................................
$FFEØA  |         |         |         |         | DS19    | DS18    | DS17    | DS16    |  $FFF8A/ØØ
        *********************************************************************************
        |         DYNAMICALLY REDEFINABLE CHARACTER IMAGE TABLE START ADDRESS           |
        .................................................................................
$FFEØB  | DS15    | DS14    | DS13    | DS12    | DS11    | DS1Ø    | Ø       | Ø       |  $FFF8B/ØØ
        *********************************************************************************
        *********************************************************************************
        |         TRUE OBJECT IMAGE TABLE START ADDRESS                                 |
        .................................................................................
$FFEØC  |         |         |         |         | TS19    | TS18    | TS17    | TS16    |  $FFF8C/ØØ
        *********************************************************************************
        |         TRUE OBJECT IMAGE TABLE START ADDRESS                                 |
        .................................................................................
$FFEØD  | TS15    | TS14    | TS13    | TS12    | TS11    | TS1Ø    | Ø       | Ø       |  $FFF8D/ØØ
        *********************************************************************************
        *********************************************************************************
        |         FIXED OBJECT IMAGE TABLE START ADDRESS                                |
        .................................................................................
$FFEØE  |         |         |         |         | FS19    | FS18    | FS17    | FS16    |  $FFF8E/ØØ
        *********************************************************************************
        |         FIXED OBJECT IMAGE TABLE START ADDRESS                                |
        .................................................................................
$FFEØF  | FS15    | FS14    | FS13    | FS12    | FS11    | FS1Ø    | Ø       | Ø       |  $FFF8F/ØØ
        *********************************************************************************
        *********************************************************************************
        |                    OBJECT Ø COLLISION STATUS                                  |
        |...............................................................................|
$FFE1Ø  | ØC7     | ØC6     | ØC5     | ØC4     | ØC3     | ØC2     | ØC1     | ØCF     |  $FFF9Ø/ØØ
        *********************************************************************************
        *********************************************************************************
        |                    OBJECT 1 COLLISION STATUS                                  |
        |...............................................................................|
$FFE11  | 1C7     | 1C6     | 1C5     | 1C4     | 1C3     | 1C2     | 1CF     | 1CØ     |  $FFF91/ØØ
        *********************************************************************************
        *********************************************************************************
        |                    OBJECT 2 COLLISION STATUS                                  |
        |...............................................................................|
$FFE12  | 2C7     | 2C6     | 2C5     | 2C4     | 2C3     | 2CF     | 2C1     | 2CØ     |  $FFF92/ØØ
        *********************************************************************************
        *********************************************************************************
        |                    OBJECT 3 COLLISION STATUS                                  |
        |...............................................................................|
$FFE13  | 3C7     | 3C6     | 3C5     | 3C4     | 3CF     | 3C2     | 3C1     | 3CØ     |  $FFF93/ØØ
        *********************************************************************************
        *********************************************************************************
        |                    OBJECT 4 COLLISION STATUS                                  |
        |...............................................................................|
$FFE14  | 4C7     | 4C6     | 4C5     | 4CF     | 4C3     | 4C2     | 4C1     | 4CØ     |  $FFF94/ØØ
        *********************************************************************************
        *********************************************************************************
        |                    OBJECT 5 COLLISION STATUS                                  |
        |...............................................................................|
$FFE15  | 5C7     | 5C6     | 5CF     | 5C4     | 5C3     | 5C2     | 5C1     | 5CØ     |  $FFF95/ØØ
        *********************************************************************************
        *********************************************************************************
        |                    OBJECT 6 COLLISION STATUS                                  |
        |...............................................................................|
$FFE16  | 6C7     | 6CF     | 6C5     | 6C4     | 6C3     | 6C2     | 6C1     | 6CØ     |  $FFF96/ØØ
        *********************************************************************************
        *********************************************************************************
        |                    OBJECT 7 COLLISION STATUS                                  |
        |...............................................................................|
$FFE17  | 7CF     | 7C6     | 7C5     | 7C4     | 7C3     | 7C2     | 7C1     | 7CØ     |  $FFF97/ØØ
        *********************************************************************************
        *********************************************************************************
        |                    COLLISION ENABLES                                          |
        |...............................................................................|
$FFE18  | CEN7    | CEN6    | CEN5    | CEN4    | CEN3    | CEN2    | CEN1    | CENØ    |  $FFF98/ØØ
        *********************************************************************************
        *********************************************************************************
        |.........|.........|.........|.........|.........|.........|.........|.........|
$FFE19  |         |         |         |         |         |         |         |         |  $FFF99/ØØ
        *********************************************************************************
        *********************************************************************************
        |.........|.........|.........|.........|.........|.........|.........|.........|
$FFE1A  |         |         |         |         |         |         |         |         |  $FFF9A/ØØ
        *********************************************************************************
        *********************************************************************************
        |.........|.........|.........|.........|.........|.........|.........|.........|
$FFE1B  |         |         |         |         |         |         |         |         |  $FFF9B/ØØ
        *********************************************************************************
        *********************************************************************************
        |          REAL TIME OUTPUT  X COORDINATE                                       |
        |...............................................................................|
$FFE1C  |         |         |         |         |         |         | OX9     | OX8     |  $FFF9C/ØØ
        *********************************************************************************
        |          REAL TIME OUTPUT  X COORDINATE                                       |
        |...............................................................................|
$FFE1D  | OX7     | OX6     | OX5     | OX4     | OX3     | OX2     | OX1     | OXØ     |  $FFF9D/ØØ
        *********************************************************************************
        *********************************************************************************
        |          REAL TIME OUTPUT  Y COORDINATE                                       |
        |...............................................................................|
$FFE1E  |         |         |         |         |         |         | OY9     | OY8     |  $FFF9E/ØØ
        *********************************************************************************
        |          REAL TIME OUTPUT  Y COORDINATE                                       |
        |...............................................................................|
$FFE1F  | OY7     | OY6     | OY5     | OY4     | OY3     | OY2     | OY1     | OYØ     |  $FFF9F/ØØ
        *********************************************************************************
        *********************************************************************************
        |         REAL TIME INPUT  X COORDINATE                                         |
        .................................................................................
$FFE20  |         |         |         |         |         |         | IX9     | IX8     |   $FFFAØ/ØØ
        *********************************************************************************
        |         REAL TIME INPUT  X COORDINATE                                         |
        .................................................................................
$FFE21  | IX7     | IX6     | IX5     | IX4     | IX3     | IX2     | IX1     | IXØ     |   $FFFA1/ØØ
        *********************************************************************************
        *********************************************************************************
        |         REAL TIME INPUT  Y COORDINATE                                         |
        .................................................................................
$FFE22  |         |         |         |         |         |         | IY9     | IY8     |   $FFFA2/ØØ
        *********************************************************************************
        |         REAL TIME INPUT  Y COORDINATE                                         |
        .................................................................................
$FFE23  | IY7     | IY6     | IY5     | IY4     | IY3     | IY2     | IY1     | IYØ     |   $FFFA3/ØØ
        *********************************************************************************
        *********************************************************************************
        |         MEMORY PART TYPE              | NUMBER OF BANKS   |         |         |
        .................................................................................
$FFE24  | MTP3    | MTP2    | MTP1    | MTPØ    | DB1     | DBØ     |         |         |   $FFFA4/ØØ
        *********************************************************************************
        *********************************************************************************
        | ITL SYN | ITL DAT | VERTICAL RESOLUTION         | HORIZONTAL RESOLUTION       |
        .................................................................................
$FFE25  | IS      | ID      | VRES2   | VRES1   | VRESØ   | HRES2   | HRES1   | HRESØ   |   $FFFA5/ØØ
        *********************************************************************************
        *********************************************************************************
        |         |         |         |         | SYNC SIGNAL MODE            | GRN SYN |
        .................................................................................
$FFE26  |         |         |         |         | VIS2    | VIS1    | VISØ    | GS      |   $FFFA6/ØØ
        *********************************************************************************
        *********************************************************************************
        |         |         |         |         |         |         |         |         |
        .................................................................................
$FFE27  |         |         |         |         |         |         |         |         |   $FFFA7/ØØ
        *********************************************************************************
        *********************************************************************************
        |                   VIRTUAL SCREEN START ADDRESS                                |
        .................................................................................
$FFE28  |         |         |         |         |         |         |         |         |   $FFFA8/ØØ
        *********************************************************************************
        |                   VIRTUAL SCREEN START ADDRESS                                |
        .................................................................................
$FFE29  |         |         |         |         | VS19    | VS18    | VS17    | VS16    |   $FFFA9/ØØ
        *********************************************************************************
        |                   VIRTUAL SCREEN START ADDRESS                                |
        .................................................................................
$FFE2A  | VS15    | VS14    | VS13    | VS12    | VS11    | VS1Ø    | Ø       | Ø       |   $FFFAA/ØØ
        *********************************************************************************
        |                   VIRTUAL SCREEN START ADDRESS                                |
        .................................................................................
$FFE2B  | Ø       | Ø       | Ø       | Ø       | Ø       | Ø       | Ø       | Ø       |   $FFFAB/ØØ
        *********************************************************************************
        *********************************************************************************
        |                   VERTICAL OFFSET REGISTER                                    |
        .................................................................................
$FFE2C  |         |         |         |         |         |         |         |         |   $FFFAC/ØØ
        *********************************************************************************
        |                   VERTICAL OFFSET REGISTER                                    |
        .................................................................................
$FFE2D  | SIGN    |    Ø    |    Ø    |    Ø    |    Ø    | Y18     | Y17     | Y16     |   $FFFAD/ØØ
        *********************************************************************************
        |                   VERTICAL OFFSET REGISTER                                    |
        .................................................................................
$FFE2E  | Y15     | Y14     | Y13     | Y12     | Y11     | Y1Ø     | Y9      | Y8      |   $FFFAE/ØØ
        *********************************************************************************
        |                   VERTICAL OFFSET REGISTER                                    |
        .................................................................................
$FFE2F  | Y7      | Y6      | Y5      | Y4      | Y3      | Y2      | Ø       | Ø       |   $FFFAF/ØØ
        *********************************************************************************
        *****************************************************************************
        |                   HORIZONTAL OFFSET REGISTER                              |
        |...........................................................................|
$FFE30  |        |        |        |        |        |        |        |         |   $FFFBØ/ØØ
        *****************************************************************************
        |                   HORIZONTAL OFFSET REGISTER                              |
        |...........................................................................|
$FFE31  |        |        |        |        |        |        |        |         |   $FFFB1/ØØ
        *****************************************************************************
        |                   HORIZONTAL OFFSET REGISTER                              |
        |...........................................................................|
$FFE32  | SIGN   |   Ø    |   Ø    |   Ø    |   Ø    | X1Ø    | X9     | X8      |   $FFFB2/ØØ
        *****************************************************************************
        |                   HORIZONTAL OFFSET REGISTER                              |
        |...........................................................................|
$FFE33  | X7     | X6     | X5     | X4     | X3     | X2     | X1     | XØ      |   $FFFB3/ØØ
        *****************************************************************************
        *****************************************************************************
        |                   VIRTUAL SCREEN SIZE REGISTER                            |
        |...........................................................................|
$FFE34  |        |        |        |        |        |        |        |         |   $FFFB4/ØØ
        *****************************************************************************
        |                   VIRTUAL SCREEN SIZE REGISTER                            |
        |...........................................................................|
$FFE35  |        |        |        |        |        | V18    | V17    | V16     |   $FFFB5/ØØ
        *****************************************************************************
        |                   VIRTUAL SCREEN SIZE REGISTER                            |
        |...........................................................................|
$FFE36  | V15    | V14    | V13    | V12    | V11    | V1Ø    | V9     | V8      |   $FFFB6/ØØ
        *****************************************************************************
        |                   VIRTUAL SCREEN SIZE REGISTER                            |
        |...........................................................................|
$FFE37  | V7     | V6     | V5     | V4     | V3     | V2     | Ø      | Ø       |   $FFFB7/ØØ
        *****************************************************************************
        *****************************************************************************
        |                   VIRTUAL SCREEN WIDTH REGISTER                           |
        |...........................................................................|
$FFE38  |        |        |        |        |        |        |        |         |   $FFFB8/ØØ
        *****************************************************************************
        |                   VIRTUAL SCREEN WIDTH REGISTER                           |
        |...........................................................................|
$FFE39  |        |        |        |        |        |        |        |         |   $FFFB9/ØØ
        *****************************************************************************
        |                   VIRTUAL SCREEN WIDTH REGISTER                           |
        |...........................................................................|
$FFE3A  |        |        |        |        |        | W1Ø    | W9     | W8      |   $FFFBA/ØØ
        *****************************************************************************
        |                   VIRTUAL SCREEN WIDTH REGISTER                           |
        |...........................................................................|
$FFE3B  | W7     | W6     | W5     | W4     | W3     | W2     | Ø      | Ø       |   $FFFBB/ØØ
        *****************************************************************************
        *****************************************************************************
        |........|........|........|........|........|........|........|.........|
$FFE3C  |        |        |        |        |        |        |        |         |   $FFFBC/ØØ
        *****************************************************************************
        *****************************************************************************
        |........|........|.........|       |        |     ....|........|........|
$FFE3D  |        |        |        |        |        |        |        |         |   $FFFBD/ØØ
        *****************************************************************************
        *****************************************************************************
        |........|........|........|........|........|........|........|.........|
$FFE3E  |        |        |        |        |        |        |        |         |   $FFFBE/ØØ
        *****************************************************************************
        *****************************************************************************
        |........|........|........|........|........|........|........|.........|
$FFE3F  |        |        |        |        |        |        |        |         |   $FFFBF/ØØ
        *****************************************************************************
```

```
         *********************************************************************************
         |                   COLOR MAPPING RAM  LOCATION $ØØ                             |
         .................................................................................
$FFE4Ø   |  MP1    |  MPØ    |         |  VEN    |  R3     |  R2     |  R1     |  RØ     |  $FFF8Ø/Ø1
         *********************************************************************************
         |                   COLOR MAPPING RAM  LOCATION $ØØ                             |
         .................................................................................
$FFE41   |  G3     |  G2     |  G1     |  GØ     |  B3     |  B2     |  B1     |  BØ     |  $FFF81/Ø1
         *********************************************************************************
         *********************************************************************************
         |                   COLOR MAPPING RAM  LOCATION $Ø1                             |
         .................................................................................
$FFE42   |         |         |         |  VEN    |  R3     |  R2     |  R1     |  RØ     |  $FFF82/Ø1
         *********************************************************************************
         |                   COLOR MAPPING RAM  LOCATION $Ø1                             |
         .................................................................................
$FFE43   |  G3     |  G2     |  G1     |  GØ     |  B3     |  B2     |  B1     |  BØ     |  $FFF83/Ø1
         *********************************************************************************
         *********************************************************************************
         |                   COLOR MAPPING RAM  LOCATION $Ø2                             |
         .................................................................................
$FFE44   |         |         |         |  VEN    |  R3     |  R2     |  R1     |  RØ     |  $FFF84/Ø1
         *********************************************************************************
         |                   COLOR MAPPING RAM  LOCATION $Ø2                             |
         .................................................................................
$FFE45   |  G3     |  G2     |  G1     |  GØ     |  B3     |  B2     |  B1     |  BØ     |  $FFF85/Ø1
         *********************************************************************************
         *********************************************************************************
         |                   COLOR MAPPING RAM  LOCATION $Ø3                             |
         .................................................................................
$FFE46   |         |         |         |  VEN    |  R3     |  R2     |  R1     |  RØ     |  $FFF86/Ø1
         *********************************************************************************
         |                   COLOR MAPPING RAM  LOCATION $Ø3                             |
         .................................................................................
$FFE47   |  G3     |  G2     |  G1     |  GØ     |  B3     |  B2     |  B1     |  BØ     |  $FFF87/Ø1
         *********************************************************************************
         *********************************************************************************
         |                   COLOR MAPPING RAM  LOCATION $Ø4                             |
         .................................................................................
$FFE48   |         |         |         |  VEN    |  R3     |  R2     |  R1     |  RØ     |  $FFF88/Ø1
         *********************************************************************************
         |                   COLOR MAPPING RAM  LOCATION $Ø4                             |
         .................................................................................
$FFE49   |  G3     |  G2     |  G1     |  GØ     |  B3     |  B2     |  B1     |  BØ     |  $FFF89/Ø1
         *********************************************************************************
         *********************************************************************************
         |                   COLOR MAPPING RAM  LOCATION $Ø5                             |
         .................................................................................
$FFE4A   |         |         |         |  VEN    |  R3     |  R2     |  R1     |  RØ     |  $FFF8A/Ø1
         *********************************************************************************
         |                   COLOR MAPPING RAM  LOCATION $Ø5                             |
         .................................................................................
$FFE4B   |  G3     |  G2     |  G1     |  GØ     |  B3     |  B2     |  B1     |  BØ     |  $FFF8B/Ø1
         *********************************************************************************
         *********************************************************************************
         |                   COLOR MAPPING RAM  LOCATION $Ø6                             |
         .................................................................................
$FFE4C   |         |         |         |  VEN    |  R3     |  R2     |  R1     |  RØ     |  $FFF8C/Ø1
         *********************************************************************************
         |                   COLOR MAPPING RAM  LOCATION $Ø6                             |
         .................................................................................
$FFE4D   |  G3     |  G2     |  G1     |  GØ     |  B3     |  B2     |  B1     |  BØ     |  $FFF8D/Ø1
         *********************************************************************************
         *********************************************************************************
         |                   COLOR MAPPING RAM  LOCATION $Ø7                             |
         .................................................................................
$FFE4E   |         |         |         |  VEN    |  R3     |  R2     |  R1     |  RØ     |  $FFF8E/Ø1
         *********************************************************************************
         |                   COLOR MAPPING RAM  LOCATION $Ø7                             |
         .................................................................................
$FFE4F   |  G3     |  G2     |  G1     |  GØ     |  B3     |  B2     |  B1     |  BØ     |  $FFF8F/Ø1
         *********************************************************************************
```

```
          ****************************************************************************
          |                   COLOR MAPPING RAM  LOCATION $Ø8                         |
          .............................................................................
$FFE5Ø    |         |         |         | VEN     | R3      | R2      | R1      | RØ      |  $FFF9Ø/Ø1
          ****************************************************************************
          |                   COLOR MAPPING RAM  LOCATION $Ø8                         |
          .............................................................................
$FFE51    | G3      | G2      | G1      | GØ      | B3      | B2      | B1      | BØ      |  $FFF91/Ø1
          ****************************************************************************
          ****************************************************************************
          |                   COLOR MAPPING RAM  LOCATION $Ø9                         |
          .............................................................................
$FFE52    |         |         |         | VEN     | R3      | R2      | R1      | RØ      |  $FFF92/Ø1
          ****************************************************************************
          |                   COLOR MAPPING RAM  LOCATION $Ø9                         |
          .............................................................................
$FFE53    | G3      | G2      | G1      | GØ      | B3      | B2      | B1      | BØ      |  $FFF93/Ø1
          ****************************************************************************
          ****************************************************************************
          |                   COLOR MAPPING RAM  LOCATION $ØA                         |
          .............................................................................
$FFE54    |         |         |         | VEN     | R3      | R2      | R1      | RØ      |  $FFF94/Ø1
          ****************************************************************************
          |                   COLOR MAPPING RAM  LOCATION $ØA                         |
          .............................................................................
$FFE55    | G3      | G2      | G1      | GØ      | B3      | B2      | B1      | BØ      |  $FFF95/Ø1
          ****************************************************************************
          ****************************************************************************
          |                   COLOR MAPPING RAM  LOCATION $ØB                         |
          .............................................................................
$FFE56    |         |         |         | VEN     | R3      | R2      | R1      | RØ      |  $FFF96/Ø1
          ****************************************************************************
          |                   COLOR MAPPING RAM  LOCATION $ØB                         |
          .............................................................................
$FFE57    | G3      | G2      | G1      | GØ      | B3      | B2      | B1      | BØ      |  $FFF97/Ø1
          ****************************************************************************
          ****************************************************************************
          |                   COLOR MAPPING RAM  LOCATION $ØC                         |
          .............................................................................
$FFE58    |         |         |         | VEN     | R3      | R2      | R1      | RØ      |  $FFF98/Ø1
          ****************************************************************************
          |                   COLOR MAPPING RAM  LOCATION $ØC                         |
          .............................................................................
$FFE59    | G3      | G2      | G1      | GØ      | B3      | B2      | B1      | BØ      |  $FFF99/Ø1
          ****************************************************************************
          ****************************************************************************
          |                   COLOR MAPPING RAM  LOCATION $ØD                         |
          .............................................................................
$FFE5A    |         |         |         | VEN     | R3      | R2      | R1      | RØ      |  $FFF9A/Ø1
          ****************************************************************************
          |                   COLOR MAPPING RAM  LOCATION $ØD                         |
          .............................................................................
$FFE5B    | G3      | G2      | G1      | GØ      | B3      | B2      | B1      | BØ      |  $FFF9B/Ø1
          ****************************************************************************
          ****************************************************************************
          |                   COLOR MAPPING RAM  LOCATION $ØE                         |
          .............................................................................
$FFE5C    |         |         |         | VEN     | R3      | R2      | R1      | RØ      |  $FFF9C/Ø1
          ****************************************************************************
          |                   COLOR MAPPING RAM  LOCATION $ØE                         |
          .............................................................................
$FFE5D    | G3      | G2      | G1      | GØ      | B3      | B2      | B1      | BØ      |  $FFF9D/Ø1
          ****************************************************************************
          ****************************************************************************
          |                   COLOR MAPPING RAM  LOCATION $ØF                         |
          .............................................................................
$FFE5E    |         |         |         | VEN     | R3      | R2      | R1      | RØ      |  $FFF9E/Ø1
          ****************************************************************************
          |                   COLOR MAPPING RAM  LOCATION $ØF                         |
          .............................................................................
$FFE5F    | G3      | G2      | G1      | GØ      | B3      | B2      | B1      | BØ      |  $FFF9F/Ø1
          ****************************************************************************
```

```
        *********************************************************************************
        |                    COLOR MAPPING RAM  LOCATION $1Ø                            |
        .................................................................................
$FFE6Ø  |         |         |         | VEN     | R3      | R2      | R1      | RØ      |  $FFFAØ/Ø1
        *********************************************************************************
        |                    COLOR MAPPING RAM  LOCATION $1Ø                            |
        .................................................................................
$FFE61  | G3      | G2      | G1      | GØ      | B3      | B2      | B1      | BØ      |  $FFFA1/Ø1
        *********************************************************************************
        *********************************************************************************
        |                    COLOR MAPPING RAM  LOCATION $11                            |
        .................................................................................
$FFE62  |         |         |         | VEN     | R3      | R2      | R1      | RØ      |  $FFFA2/Ø1
        *********************************************************************************
        |                    COLOR MAPPING RAM  LOCATION $11                            |
        .................................................................................
$FFE63  | G3      | G2      | G1      | GØ      | B3      | B2      | B1      | BØ      |  $FFFA3/Ø1
        *********************************************************************************
        *********************************************************************************
        |                    COLOR MAPPING RAM  LOCATION $12                            |
        .................................................................................
$FFE64  |         |         |         | VEN     | R3      | R2      | R1      | RØ      |  $FFFA4/Ø1
        *********************************************************************************
        |                    COLOR MAPPING RAM  LOCATION $12                            |
        .................................................................................
$FFE65  | G3      | G2      | G1      | GØ      | B3      | B2      | B1      | BØ      |  $FFFA5/Ø1
        *********************************************************************************
        *********************************************************************************
        |                    COLOR MAPPING RAM  LOCATION $13                            |
        .................................................................................
$FFE66  |         |         |         | VEN     | R3      | R2      | R1      | RØ      |  $FFFA6/Ø1
        *********************************************************************************
        |                    COLOR MAPPING RAM  LOCATION $13                            |
        .................................................................................
$FFE67  | G3      | G2      | G1      | GØ      | B3      | B2      | B1      | BØ      |  $FFFA7/Ø1
        *********************************************************************************
        *********************************************************************************
        |                    COLOR MAPPING RAM  LOCATION $14                            |
        .................................................................................
$FFE68  |         |         |         | VEN     | R3      | R2      | R1      | RØ      |  $FFFA8/Ø1
        *********************************************************************************
        |                    COLOR MAPPING RAM  LOCATION $14                            |
        .................................................................................
$FFE69  | G3      | G2      | G1      | GØ      | B3      | B2      | B1      | BØ      |  $FFFA9/Ø1
        *********************************************************************************
        *********************************************************************************
        |                    COLOR MAPPING RAM  LOCATION $15                            |
        .................................................................................
$FFE6A  |         |         |         | VEN     | R3      | R2      | R1      | RØ      |  $FFFAA/Ø1
        *********************************************************************************
        |                    COLOR MAPPING RAM  LOCATION $15                            |
        .................................................................................
$FFE6B  | G3      | G2      | G1      | GØ      | B3      | B2      | B1      | BØ      |  $FFFAB/Ø1
        *********************************************************************************
        *********************************************************************************
        |                    COLOR MAPPING RAM  LOCATION $16                            |
        .................................................................................
$FFE6C  |         |         |         | VEN     | R3      | R2      | R1      | RØ      |  $FFFAC/Ø1
        *********************************************************************************
        |                    COLOR MAPPING RAM  LOCATION $16                            |
        .................................................................................
$FFE6D  | G3      | G2      | G1      | GØ      | B3      | B2      | B1      | BØ      |  $FFFAD/Ø1
        *********************************************************************************
        *********************************************************************************
        |                    COLOR MAPPING RAM  LOCATION $17                            |
        .................................................................................
$FFE6E  |         |         |         | VEN     | R3      | R2      | R1      | RØ      |  $FFFAE/Ø1
        *********************************************************************************
        |                    COLOR MAPPING RAM  LOCATION $17                            |
        .................................................................................
$FFE6F  | G3      | G2      | G1      | GØ      | B3      | B2      | B1      | BØ      |  $FFFAF/Ø1
        *********************************************************************************
        *********************************************************************************
        |                   COLOR MAPPING RAM  LOCATION $18                             |
        .................................................................................
$FFE7Ø  |         |         |         | VEN     | R3      | R2      | R1      | RØ      |  $FFFBØ/Ø1
        *********************************************************************************
        |                   COLOR MAPPING RAM  LOCATION $18                             |
        .................................................................................
$FFE71  | G3      | G2      | G1      | GØ      | B3      | B2      | B1      | BØ      |  $FFFB1/Ø1
        *********************************************************************************
        *********************************************************************************
        |                   COLOR MAPPING RAM  LOCATION $19                             |
        .................................................................................
$FFE72  |         |         |         | VEN     | R3      | R2      | R1      | RØ      |  $FFFB2/Ø1
        *********************************************************************************
        |                   COLOR MAPPING RAM  LOCATION $19                             |
        .................................................................................
$FFE73  | G3      | G2      | G1      | GØ      | B3      | B2      | B1      | BØ      |  $FFFB3/Ø1
        *********************************************************************************
        *********************************************************************************
        |                   COLOR MAPPING RAM  LOCATION $1A                             |
        .................................................................................
$FFE74  |         |         |         | VEN     | R3      | R2      | R1      | RØ      |  $FFFB4/Ø1
        *********************************************************************************
        |                   COLOR MAPPING RAM  LOCATION $1A                             |
        .................................................................................
$FFE75  | G3      | G2      | G1      | GØ      | B3      | B2      | B1      | BØ      |  $FFFB5/Ø1
        *********************************************************************************
        *********************************************************************************
        |                   COLOR MAPPING RAM  LOCATION $1B                             |
        .................................................................................
$FFE76  |         |         |         | VEN     | R3      | R2      | R1      | RØ      |  $FFFB6/Ø1
        *********************************************************************************
        |                   COLOR MAPPING RAM  LOCATION $1B                             |
        .................................................................................
$FFE77  | G3      | G2      | G1      | GØ      | B3      | B2      | B1      | BØ      |  $FFFB7/Ø1
        *********************************************************************************
        *********************************************************************************
        |                   COLOR MAPPING RAM  LOCATION $1C                             |
        .................................................................................
$FFE78  |         |         |         | VEN     | R3      | R2      | R1      | RØ      |  $FFFB8/Ø1
        *********************************************************************************
        |                   COLOR MAPPING RAM  LOCATION $1C                             |
        .................................................................................
$FFE79  | G3      | G2      | G1      | GØ      | B3      | B2      | B1      | BØ      |  $FFFB9/Ø1
        *********************************************************************************
        *********************************************************************************
        |                   COLOR MAPPING RAM  LOCATION $1D                             |
        .................................................................................
$FFE7A  |         |         |         | VEN     | R3      | R2      | R1      | RØ      |  $FFFBA/Ø1
        *********************************************************************************
        |                   COLOR MAPPING RAM  LOCATION $1D                             |
        .................................................................................
$FFE7B  | G3      | G2      | G1      | GØ      | B3      | B2      | B1      | BØ      |  $FFFBB/Ø1
        *********************************************************************************
        *********************************************************************************
        |                   COLOR MAPPING RAM  LOCATION $1E                             |
        .................................................................................
$FFE7C  |         |         |         | VEN     | R3      | R2      | R1      | RØ      |  $FFFBC/Ø1
        *********************************************************************************
        |                   COLOR MAPPING RAM  LOCATION $1E                             |
        .................................................................................
$FFE7D  | G3      | G2      | G1      | GØ      | B3      | B2      | B1      | BØ      |  $FFFBD/Ø1
        *********************************************************************************
        *********************************************************************************
        |                   COLOR MAPPING RAM  LOCATION $1F                             |
        .................................................................................
$FFE7E  |         |         |         | VEN     | R3      | R2      | R1      | RØ      |  $FFFBE/Ø1
        *********************************************************************************
        |                   COLOR MAPPING RAM  LOCATION $1F                             |
        .................................................................................
$FFE7F  | G3      | G2      | G1      | GØ      | B3      | B2      | B1      | BØ      |  $FFFBF/Ø1
        *********************************************************************************
```

```
         *********************************************************************************
         | CTL REG PAGE      | UNFOLD  | MACHINE |         |         |         |         |
         |...................|.........|.........|.........|.........|.........|.........|
$FFE8Ø   | MP1     | MPØ'    | UF      | M2      |         |         |         |         |  $FFF8Ø/Ø2
         *********************************************************************************
         *********************************************************************************
         | BIT/LIST|      LIST MODE SELECT       | LINES PER ROW     | BITS PER PEL      |
         |.........|.............................|...................|...................|
$FFE81   | BP      | LM2     | LM1     | LMØ     | LPR1    | LPRØ    | BPP1    | BPPØ    |  $FFF81/Ø2
         *********************************************************************************
         *********************************************************************************
         | INTRPT  |            INTERRUPT BITS (WRITE ENABLES AND READ FLAGS)            |
         |.........|.....................................................................|
$FFE82   | IPT     |         |         | RTI     | RTO     | OFN     | BLK     | COL     |  $FFF82/Ø2
         *********************************************************************************
         *********************************************************************************
         | WRAP    | MEM MAP | VID ENBL|                  BORDER COLOR                   |
         |.........|.........|.........|.................................................|
$FFE83   | WC      | MAPA    | DV      | BC4     | BC3     | BC2     | BC1     | BCØ     |  $FFF83/Ø2
         *********************************************************************************
         *********************************************************************************
         |                              OBJECT AVAILABLE                                 |
         |...............................................................................|
$FFE84   | O7A     | O6A     | O5A     | O4A     | O3A     | O2A     | O1A     | OØA     |  $FFF84/Ø2
         *********************************************************************************
         *********************************************************************************
         |         |         |         | SWAP    |                PAGING                 |
         |.........|.........|.........|.........|.......................................|
$FFE85   |         |         |         | SWAP    | PG3     | PG2     | PG1     | PGØ     |  $FFF85/Ø2
         *********************************************************************************
         *********************************************************************************
         | UPPER PAGE INDEPENDENT BLOCK          | LOWER PAGE INDEPENDENT BLOCK          |
         |.......................................|.......................................|
$FFE86   | UEN     | UPI2    | UPI1    | UPIØ    | LEN     | LPI2    | LPI1    | LPIØ/VEC|  $FFF86/Ø2
         *********************************************************************************
         *********************************************************************************
         |         |         |         |2H PRESET|     VERTICAL SCROLLING                |
         |.........|.........|.........|.........|.......................................|
$FFE87   |         |         |         | DHP     | VSC3    | VSC2    | VSC1    | VSCØ    |  $FFF87/Ø2
         *********************************************************************************
         *********************************************************************************
         |         |         |         |2W PRESET|     HORIZONTAL SCROLLING              |
         |.........|.........|.........|.........|.......................................|
$FFE88   |         |         |         | DWP     | HSC3    | HSC2    | HSC1    | HSCØ    |  $FFF88/Ø2
         *********************************************************************************
         *********************************************************************************
         |         |         |         |         |         |         |         |         |
         |.........|.........|.........|.........|.........|.........|.........|.........|
$FFE89   |         |         |         |         |         |         |         |         |  $FFF89/Ø2
         *********************************************************************************
         *********************************************************************************
         |           DYNAMICALLY REDEFINABLE CHARACTER IMAGE TABLE START ADDRESS         |
         |...............................................................................|
$FFE8A   |         |         |         |         | DS19    | DS18    | DS17    | DS16    |  $FFF8A/Ø2
         *********************************************************************************
         |          DYNAMICALLY REDEFINABLE CHARACTER IMAGE TABLE START ADDRESS|         |
         |.....................................................................|.........|
$FFE8B   | DS15    | DS14    | DS13    | DS12    | DS11    | DS1Ø    | Ø       | Ø       |  $FFF8B/Ø2
         *********************************************************************************
         *********************************************************************************
         |           TRUE OBJECT IMAGE TABLE START ADDRESS                               |
         |...............................................................................|
$FFE8C   |         |         |         |         | TS19    | TS18    | TS17    | TS16    |  $FFF8C/Ø2
         *********************************************************************************
         |           TRUE OBJECT IMAGE TABLE START ADDRESS                               |
         |...............................................................................|
$FFE8D   | TS15    | TS14    | TS13    | TS12    | TS11    | TS1Ø    | Ø       | Ø       |  $FFF8D/Ø2
         *********************************************************************************
         *********************************************************************************
         |           FIXED OBJECT IMAGE TABLE START ADDRESS                              |
         |...............................................................................|
$FFE8E   |         |         |         |         | FS19    | FS18    | FS17    | FS16    |  $FFF8E/Ø2
         *********************************************************************************
         |           FIXED OBJECT IMAGE TABLE START ADDRESS                              |
         |...............................................................................|
$FFE8F   | FS15    | FS14    | FS13    | FS12    | FS11    | FS1Ø    | Ø       | Ø       |  $FFF8F/Ø2
         *********************************************************************************
```

```
        ********************************************************************************
        |                    TRUE OBJECT Ø  X POSITION REGISTER                        |
        :..............................................................................:
$FFE9Ø  | B/R    | XZ1    | XZØ    |        |        |        | X9     | X8     |    $FFF9Ø/Ø2
        ********************************************************************************
        |                    TRUE OBJECT Ø  X POSITION REGISTER                        |
        :..............................................................................:
$FFE91  | X7     | X6     | X5     | X4     | X3     | X2     | X1     | XØ     |    $FFF91/Ø2
        ********************************************************************************
        ********************************************************************************
        |                    TRUE OBJECT Ø  Y POSITION REGISTER                        |
        :..............................................................................:
$FFE92  | OEN    | YZ1    | YZØ    |        |        |        | Y9     | Y8     |    $FFF92/Ø2
        ********************************************************************************
        |                    TRUE OBJECT Ø  Y POSITION REGISTER                        |
        :..............................................................................:
$FFE93  | Y7     | Y6     | Y5     | Y4     | Y3     | Y2     | Y1     | YØ     |    $FFF93/Ø2
        ********************************************************************************
        ********************************************************************************
        |                    TRUE OBJECT 1  X POSITION REGISTER                        |
        :..............................................................................:
$FFE94  | B/R    | XZ1    | XZØ    |        |        |        | X9     | X8     |    $FFF94/Ø2
        ********************************************************************************
        |                    TRUE OBJECT 1  X POSITION REGISTER                        |
        :..............................................................................:
$FFE95  | X7     | X6     | X5     | X4     | X3     | X2     | X1     | XØ     |    $FFF95/Ø2
        ********************************************************************************
        ********************************************************************************
        |                    TRUE OBJECT 1  Y POSITION REGISTER                        |
        :..............................................................................:
$FFE96  | OEN    | YZ1    | YZØ    |        |        |        | Y9     | Y8     |    $FFF96/Ø2
        ********************************************************************************
        |                    TRUE OBJECT 1  Y POSITION REGISTER                        |
        :..............................................................................:
$FFE97  | Y7     | Y6     | Y5     | Y4     | Y3     | Y2     | Y1     | YØ     |    $FFF97/Ø2
        ********************************************************************************
        ********************************************************************************
        |                    TRUE OBJECT 2  X POSITION REGISTER                        |
        :..............................................................................:
$FFE98  | B/R    | XZ1    | XZØ    |        |        |        | X9     | X8     |    $FFF98/Ø2
        ********************************************************************************
        |                    TRUE OBJECT 2  X POSITION REGISTER                        |
        :..............................................................................:
$FFE99  | X7     | X6     | X5     | X4     | X3     | X2     | X1     | XØ     |    $FFF99/Ø2
        ********************************************************************************
        ********************************************************************************
        |                    TRUE OBJECT 2  Y POSITION REGISTER                        |
        :..............................................................................:
$FFE9A  | OEN    | YZ1    | YZØ    |        |        |        | Y9     | Y8     |    $FFF9A/Ø2
        ********************************************************************************
        |                    TRUE OBJECT 2  Y POSITION REGISTER                        |
        :..............................................................................:
$FFE9B  | Y7     | Y6     | Y5     | Y4     | Y3     | Y2     | Y1     | YØ     |    $FFF9B/Ø2
        ********************************************************************************
        ********************************************************************************
        |                    TRUE OBJECT 3  X POSITION REGISTER                        |
        :..............................................................................:
$FFE9C  | B/R    | XZ1    | XZØ    |        |        |        | X9     | X8     |    $FFF9C/Ø2
        ********************************************************************************
        |                    TRUE OBJECT 3  X POSITION REGISTER                        |
        :..............................................................................:
$FFE9D  | X7     | X6     | X5     | X4     | X3     | X2     | X1     | XØ     |    $FFF9D/Ø2
        ********************************************************************************
        ********************************************************************************
        |                    TRUE OBJECT 3  Y POSITION REGISTER                        |
        :..............................................................................:
$FFE9E  | OEN    | YZ1    | YZØ    |        |        |        | Y9     | Y8     |    $FFF9E/Ø2
        ********************************************************************************
        |                    TRUE OBJECT 3  Y POSITION REGISTER                        |
        :..............................................................................:
$FFE9F  | Y7     | Y6     | Y5     | Y4     | Y3     | Y2     | Y1     | YØ     |    $FFF9F/Ø2
        ********************************************************************************
```

```
         ****************************************************************************
         |                   OBJECT 4  X POSITION REGISTER                           |
         ............................................................................
$FFEAØ   | B/R     | XZ1     | XZØ     |         |         |         | X9      | X8       |   $FFFAØ/Ø2
         ****************************************************************************
         |                   OBJECT 4  X POSITION REGISTER                           |
         ............................................................................
$FFEA1   | X7      | X6      | X5      | X4      | X3      | X2      | X1      | XØ       |   $FFFA1/Ø2
         ****************************************************************************
         ****************************************************************************
         |                   OBJECT 4  Y POSITION REGISTER                           |
         ............................................................................
$FFEA2   | OEN     | YZ1     | YZØ     |         |         |         | Y9      | Y8       |   $FFFA2/Ø2
         ****************************************************************************
         |                   OBJECT 4  Y POSITION REGISTER                           |
         ............................................................................
$FFEA3   | Y7      | Y6      | Y5      | Y4      | Y3      | Y2      | Y1      | YØ       |   $FFFA3/Ø2
         ****************************************************************************
         ****************************************************************************
         |                   OBJECT 5  X POSITION REGISTER                           |
         ............................................................................
$FFEA4   | B/R     | XZ1     | XZØ     |         |         |         | X9      | X8       |   $FFFA4/Ø2
         ****************************************************************************
         |                   OBJECT 5  X POSITION REGISTER                           |
         ............................................................................
$FFEA5   | X7      | X6      | X5      | X4      | X3      | X2      | X1      | XØ       |   $FFFA5/Ø2
         ****************************************************************************
         ****************************************************************************
         |                   OBJECT 5  Y POSITION REGISTER                           |
         ............................................................................
$FFEA6   | OEN     | YZ1     | YZØ     |         |         |         | Y9      | Y8       |   $FFFA6/Ø2
         ****************************************************************************
         |                   OBJECT 5  Y POSITION REGISTER                           |
         ............................................................................
$FFEA7   | Y7      | Y6      | Y5      | Y4      | Y3      | Y2      | Y1      | YØ       |   $FFFA7/Ø2
         ****************************************************************************
         ****************************************************************************
         |                   OBJECT 6  X POSITION REGISTER                           |
         ............................................................................
$FFEA8   | B/R     | XZ1     | XZØ     |         |         |         | X9      | X8       |   $FFFA8/Ø2
         ****************************************************************************
         |                   OBJECT 6  X POSITION REGISTER                           |
         ............................................................................
$FFEA9   | X7      | X6      | X5      | X4      | X3      | X2      | X1      | XØ       |   $FFFA9/Ø2
         ****************************************************************************
         ****************************************************************************
         |                   OBJECT 6  Y POSITION REGISTER                           |
         ............................................................................
$FFEAA   | OEN     | YZ1     | YZØ     |         |         |         | Y9      | Y8       |   $FFFAA/Ø2
         ****************************************************************************
         |                   OBJECT 6  Y POSITION REGISTER                           |
         ............................................................................
$FFEAB   | Y7      | Y6      | Y5      | Y4      | Y3      | Y2      | Y1      | YØ       |   $FFFAB/Ø2
         ****************************************************************************
         ****************************************************************************
         |                   OBJECT 7  X POSITION REGISTER                           |
         ............................................................................
$FFEAC   | B/R     | XZ1     | XZØ     |         |         |         | X9      | X8       |   $FFFAC/Ø2
         ****************************************************************************
         |                   OBJECT 7  X POSITION REGISTER                           |
         ............................................................................
$FFEAD   | X7      | X6      | X5      | X4      | X3      | X2      | X1      | XØ       |   $FFFAD/Ø2
         ****************************************************************************
         ****************************************************************************
         |                   OBJECT 7  Y POSITION REGISTER                           |
         ............................................................................
$FFEAE   | OEN     | YZ1     | YZØ     |         |         |         | Y9      | Y8       |   $FFFAE/Ø2
         ****************************************************************************
         |                   OBJECT 7  Y POSITION REGISTER                           |
         ............................................................................
$FFEAF   | Y7      | Y6      | Y5      | Y4      | Y3      | Y2      | Y1      | YØ       |   $FFFAF/Ø2
         ****************************************************************************
```

	OBJECT Ø NAME REGISTER								
$FFEBØ	N7	N6	N5	N4	N3	N2	N1	NØ	$FFFBØ/Ø2
	OBJECT 1 NAME REGISTER								
$FFEB1	N7	N6	N5	N4	N3	N2	N1	NØ	$FFFB1/Ø2
	OBJECT 2 NAME REGISTER								
$FFEB2	N7	N6	N5	N4	N3	N2	N1	NØ	$FFFB2/Ø2
	OBJECT 3 NAME REGISTER								
$FFEB3	N7	N6	N5	N4	N3	N2	N1	NØ	$FFFB3/Ø2
	OBJECT 4 NAME REGISTER								
$FFEB4	N7	N6	N5	N4	N3	N2	N1	NØ	$FFFB4/Ø2
	OBJECT 5 NAME REGISTER								
$FFEB5	N7	N6	N5	N4	N3	N2	N1	NØ	$FFFB5/Ø2
	OBJECT 6 NAME REGISTER								
$FFEB6	N7	N6	N5	N4	N3	N2	N1	NØ	$FFFB6/Ø2
	OBJECT 7 NAME REGISTER								
$FFEB7	N7	N6	N5	N4	N3	N2	N1	NØ	$FFFB7/Ø2
$FFEB8									$FFFB8/Ø2
$FFEB9									$FFFB9/Ø2
$FFEBA									$FFFBA/Ø2
$FFEBB									$FFFBB/Ø2
$FFEBC									$FFFBC/Ø2
$FFEBD									$FFFBD/Ø2
$FFEBE									$FFFBE/Ø2
$FFEBF									$FFFBF/Ø2

APPENDIX B

FIGURES

The figures are numbered according to their use in each chapter.

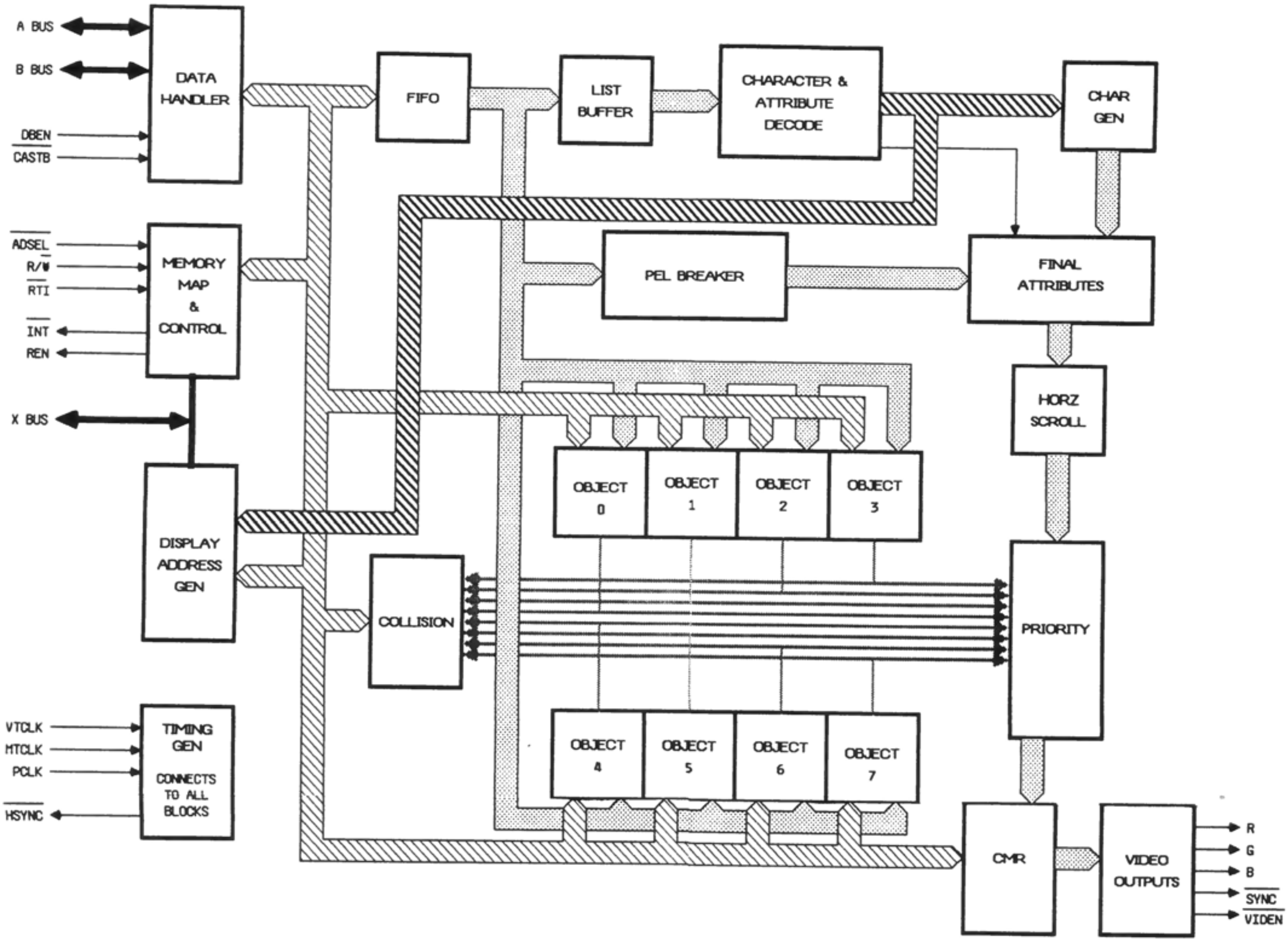


FIGURE 1-1 RASTER MEMORY CONTROLLER BLOCK DIAGRAM

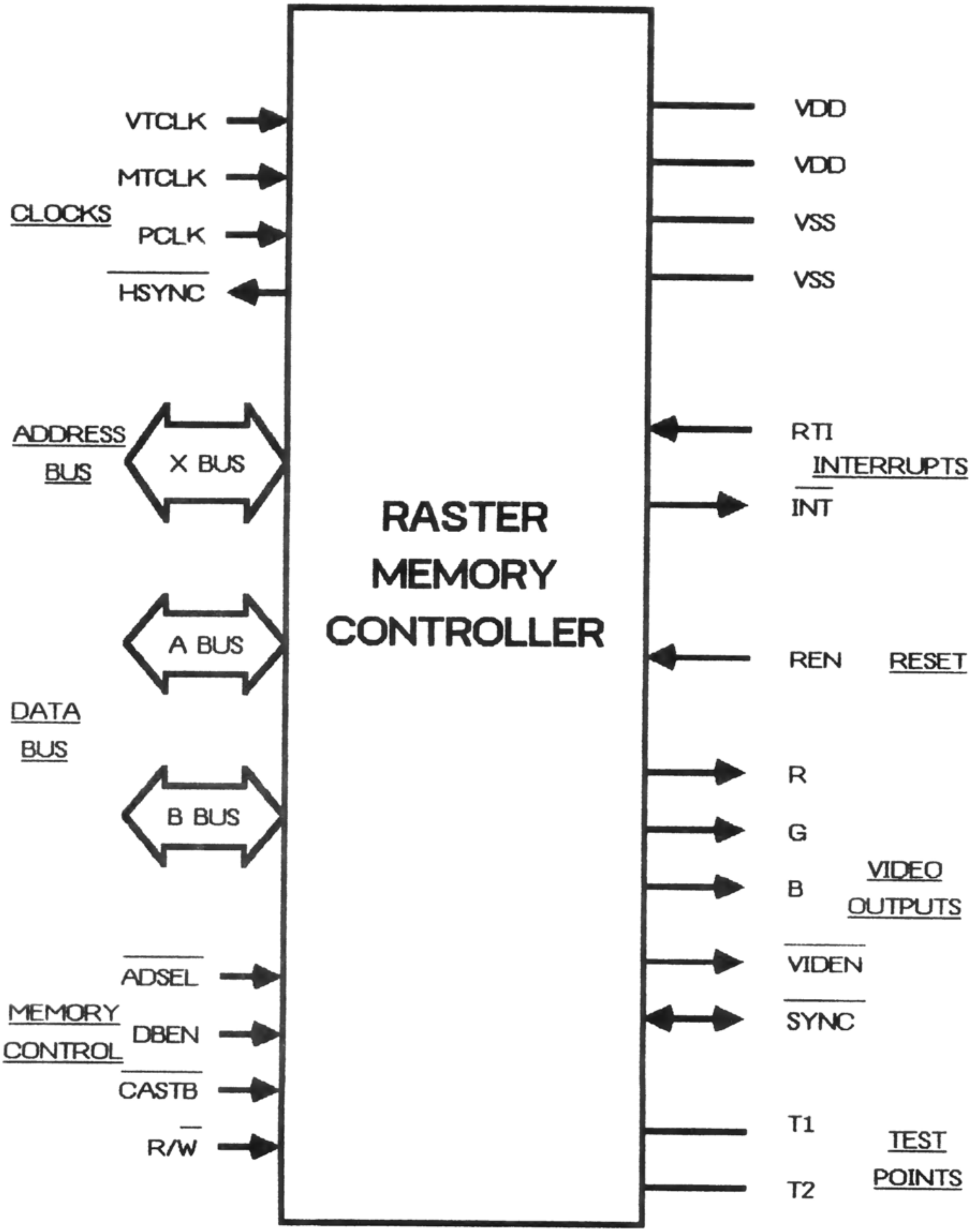


FIGURE 1-2 RMC FUNCTIONAL PINOUT

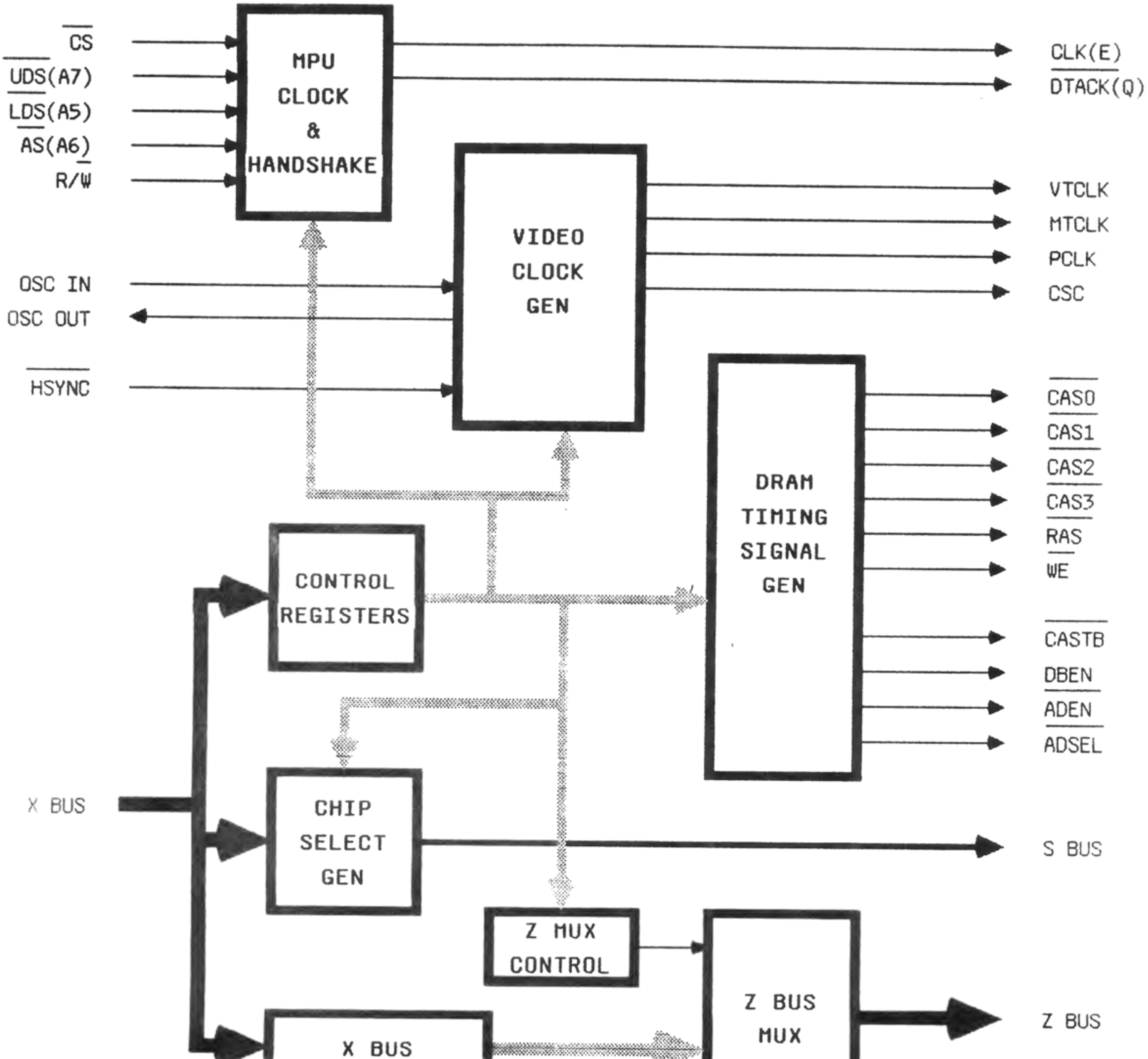


FIGURE 1-3

RASTER MEMORY INTERFACE BLOCK DIAGRAM

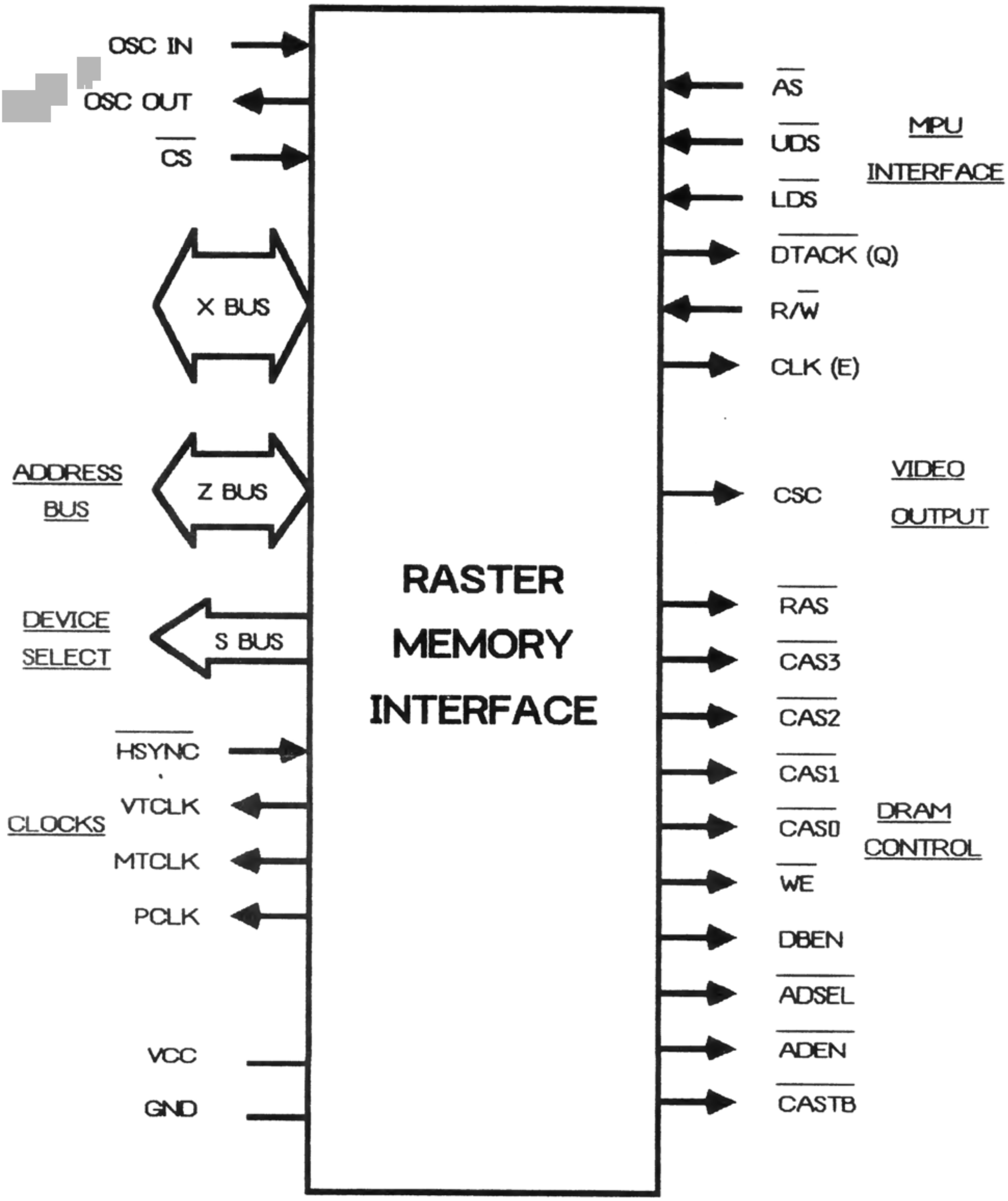


FIGURE 1-4 RMI FUNCTIONAL PINOUT

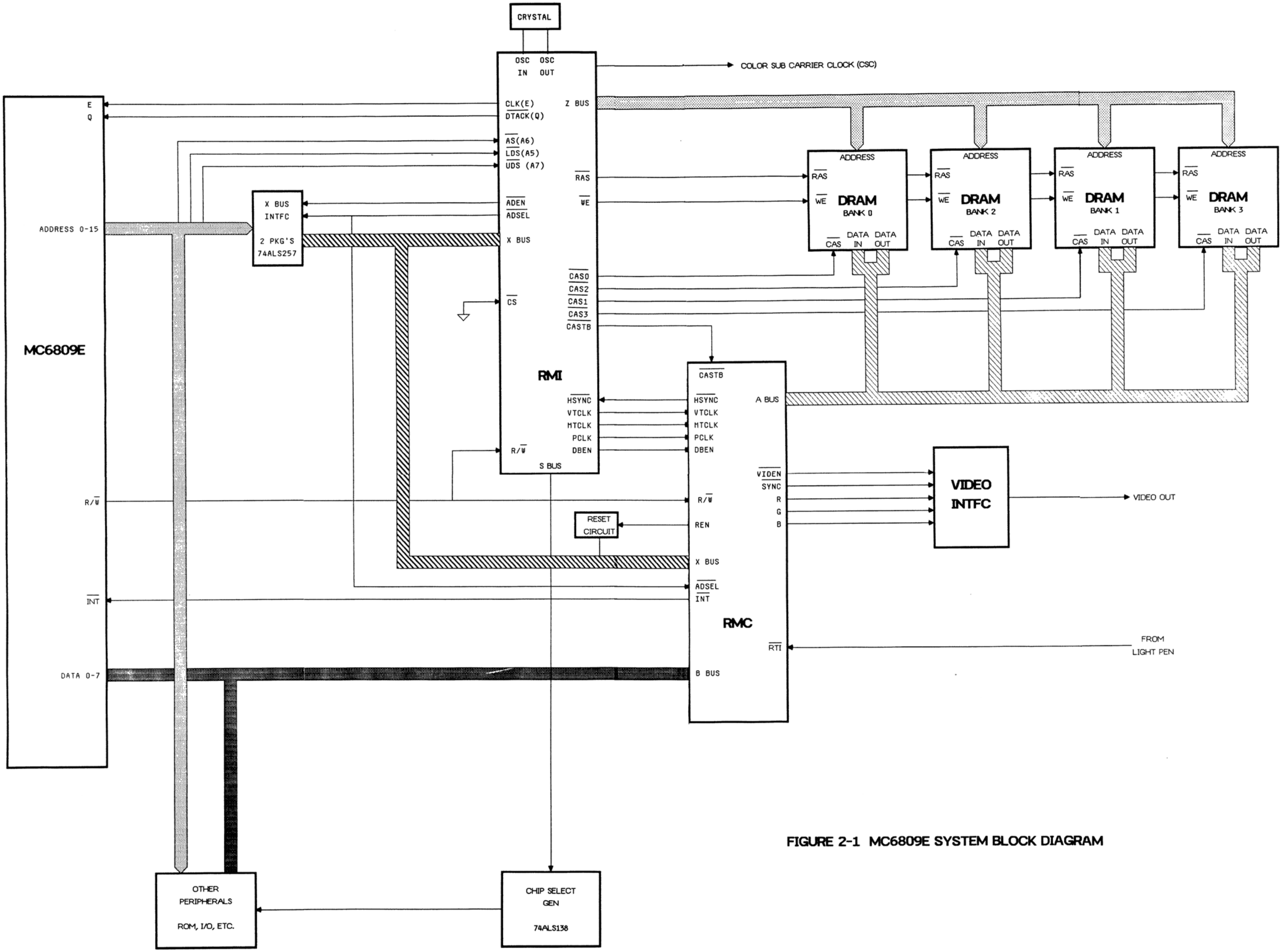


FIGURE 2-1 MC6809E SYSTEM BLOCK DIAGRAM

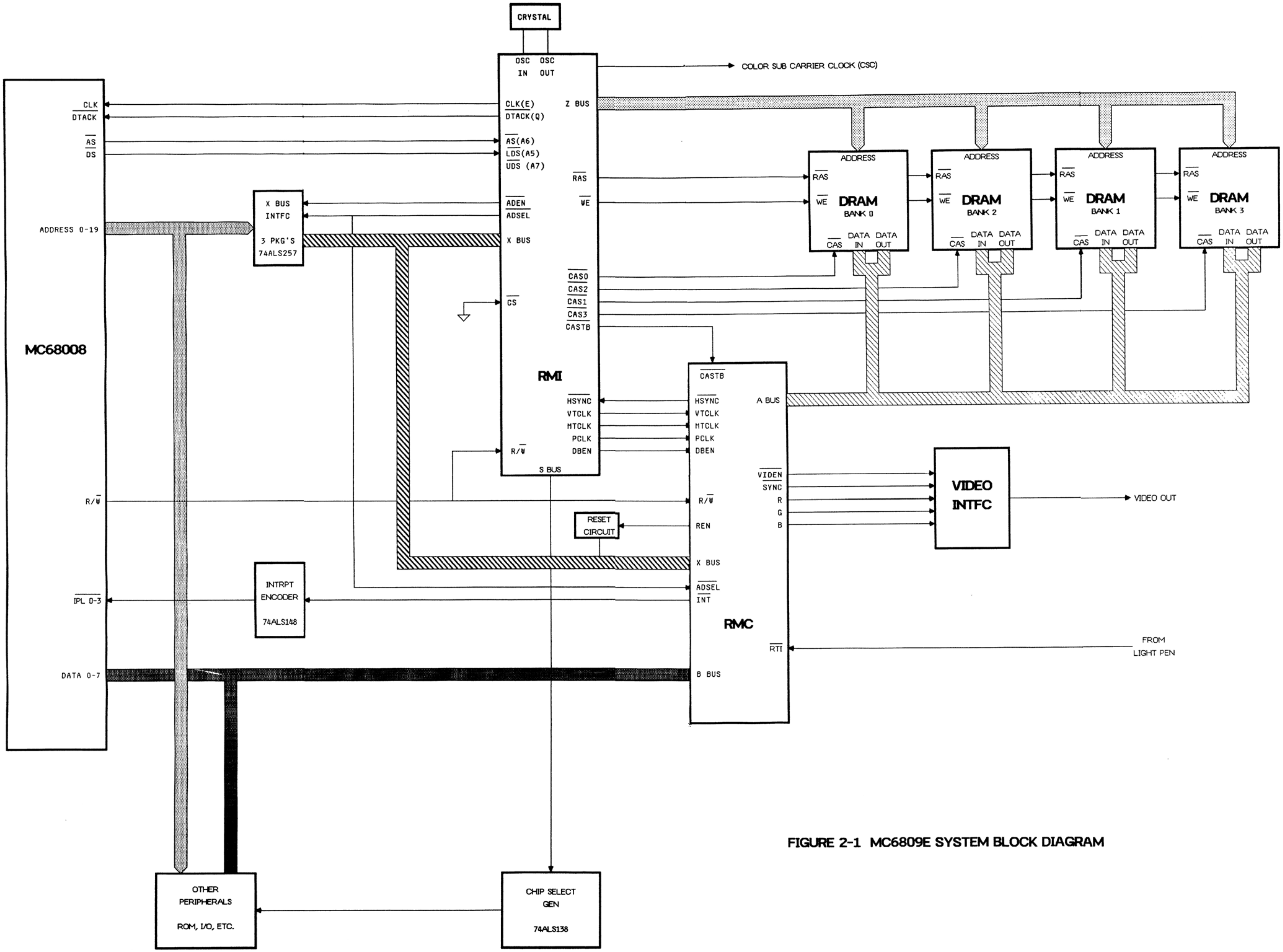


FIGURE 2-1 MC6809E SYSTEM BLOCK DIAGRAM

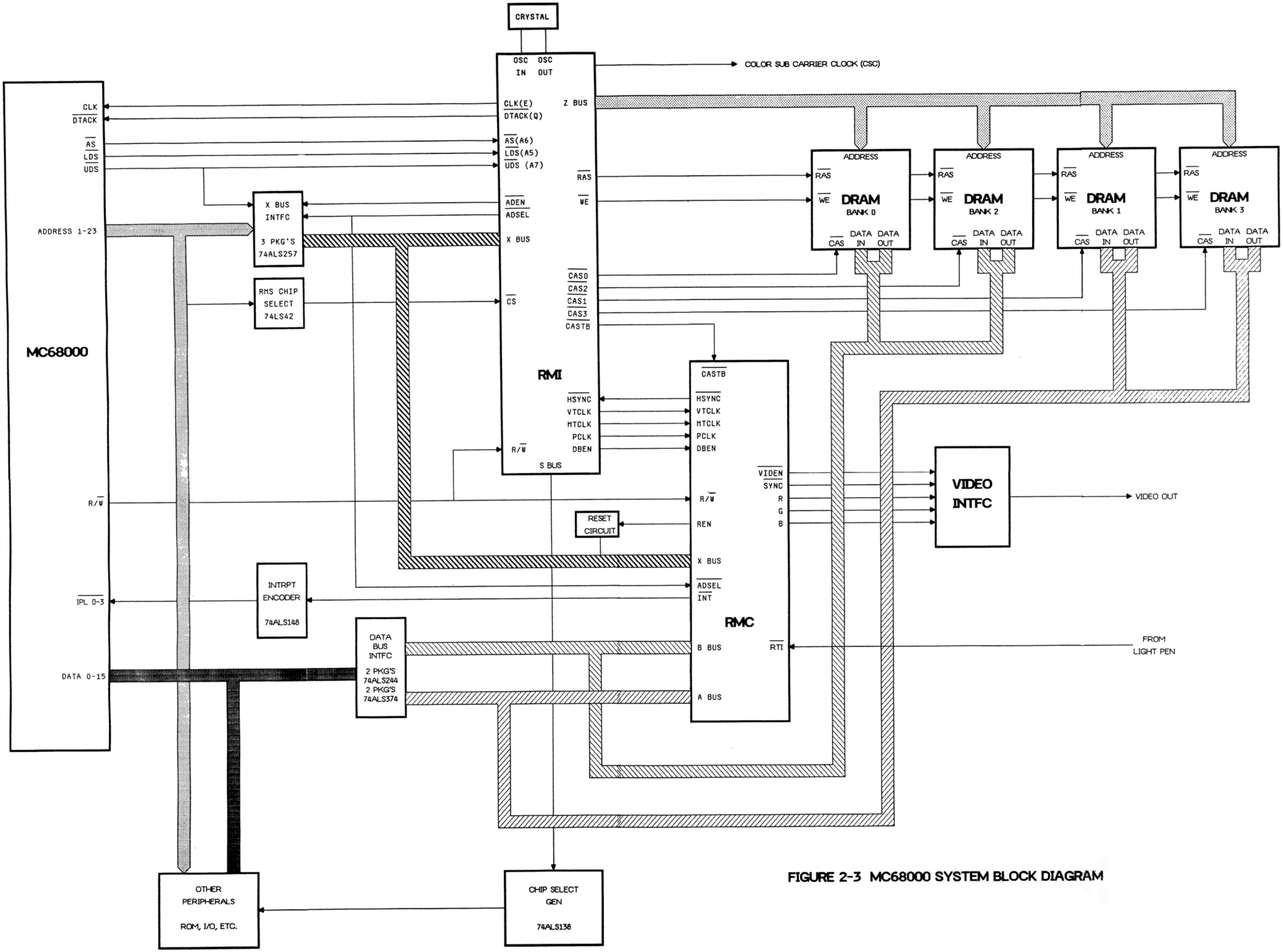


FIGURE 2-3 MC68000 SYSTEM BLOCK DIAGRAM

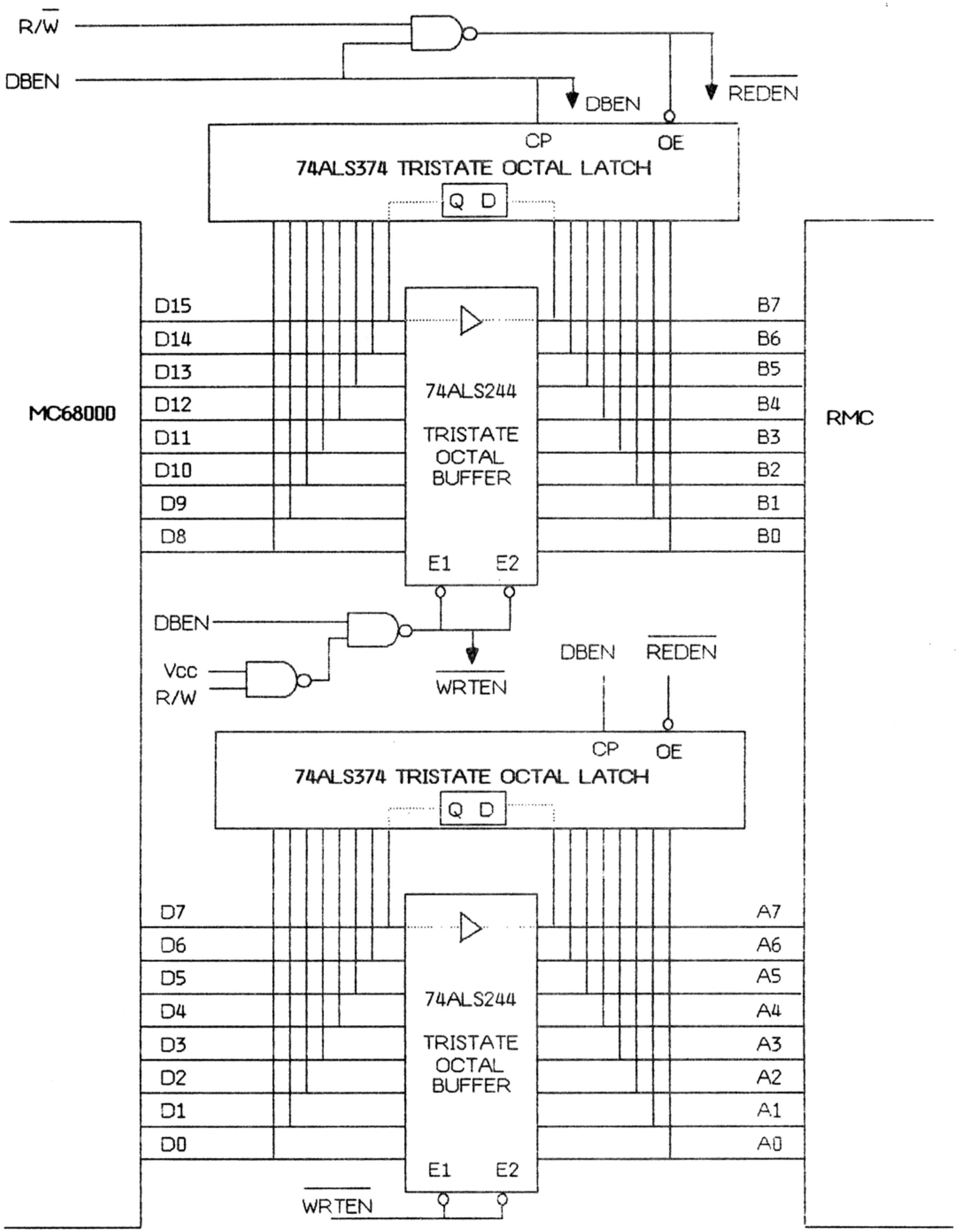
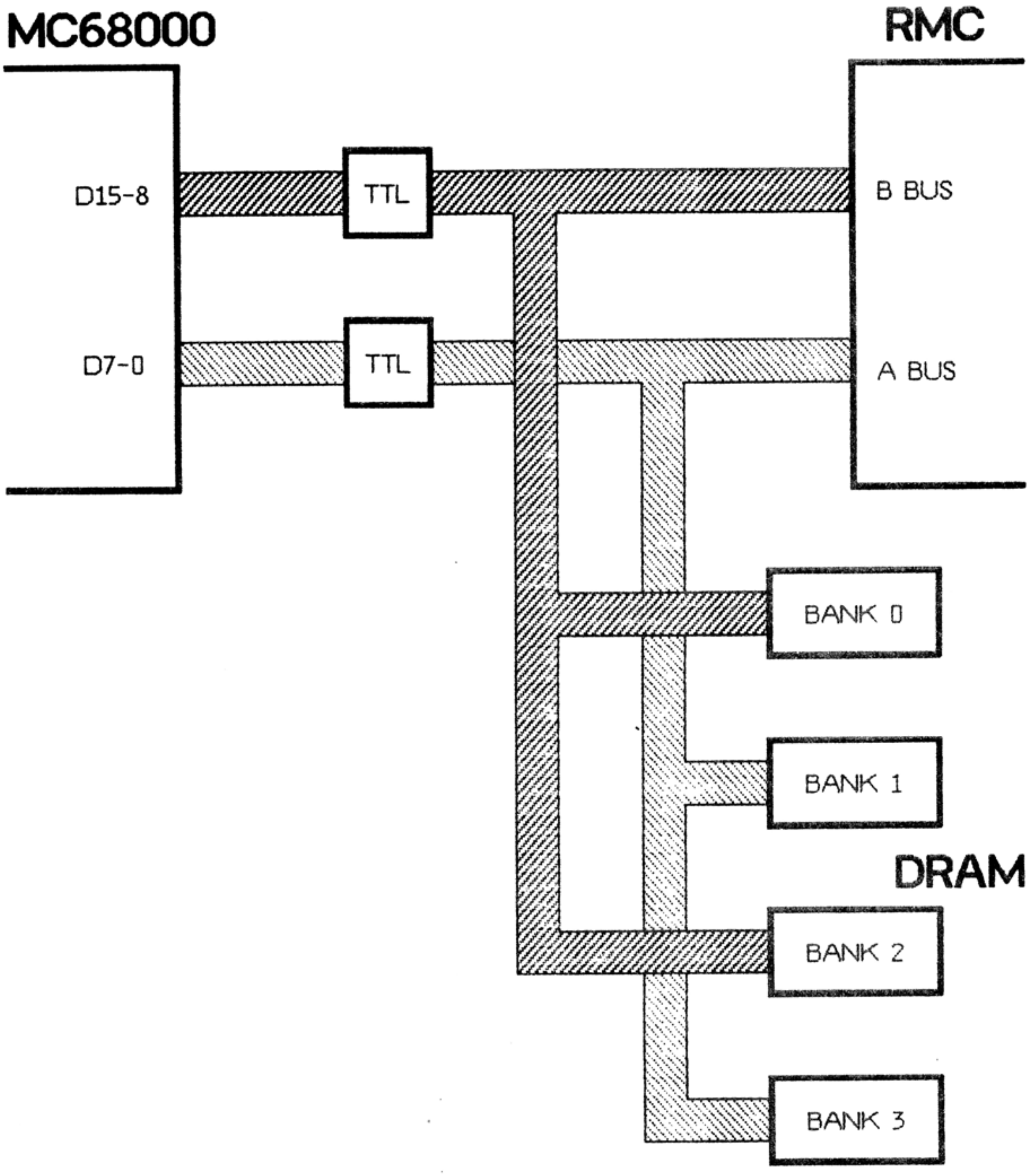
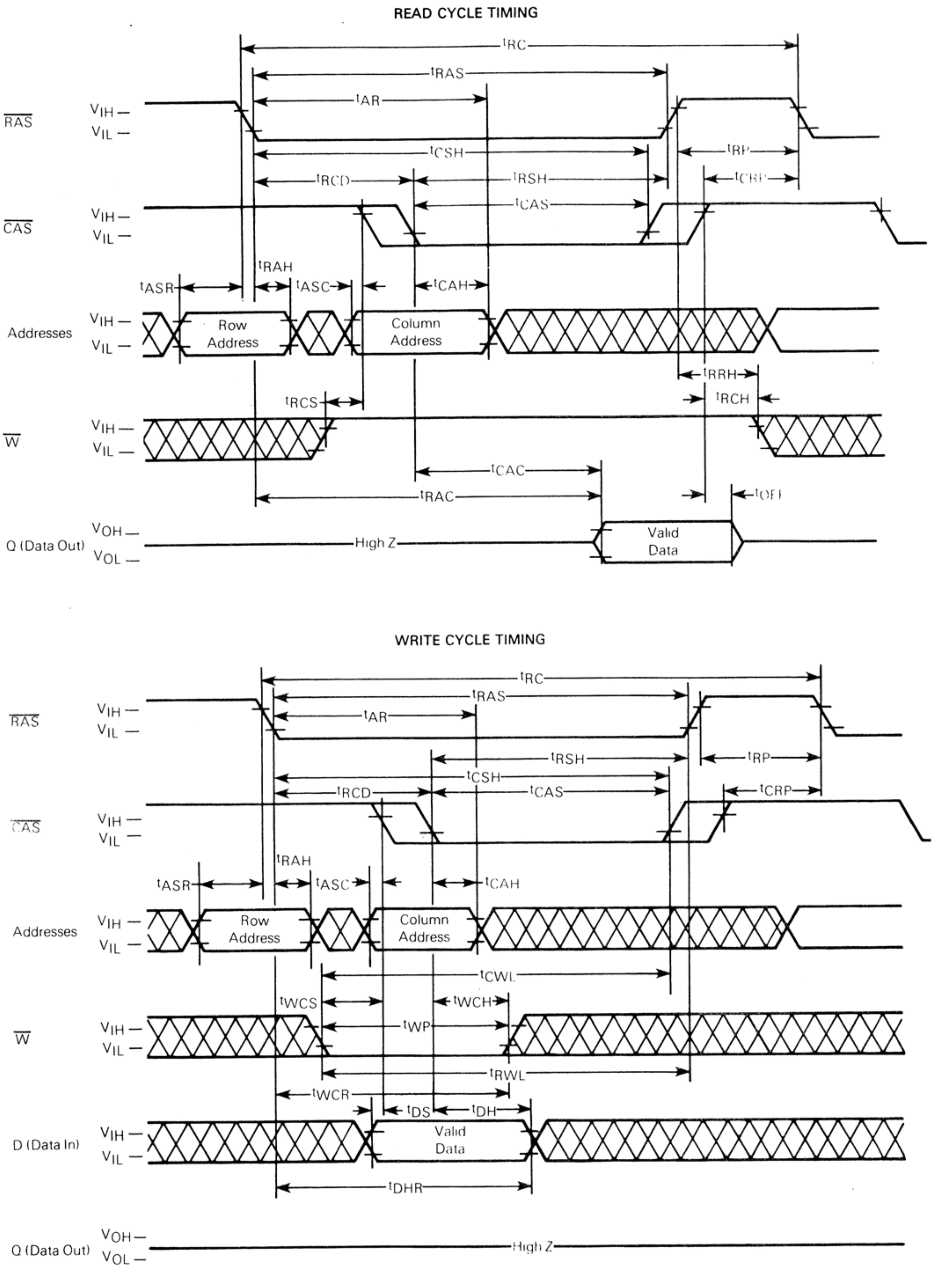


FIGURE 2-4 MC68000 DATA BUS LOGIC



	$\overline{\text{UDS}}$ ACTIVE	$\overline{\text{LDS}}$ ACTIVE
	D15	D0
ADDRESS MSB = 0	MSB BANK 0 LSB	MSB BANK 1 LSB
ADDRESS MSB = 1	MSB BANK 2 LSB	MSB BANK 3 LSB

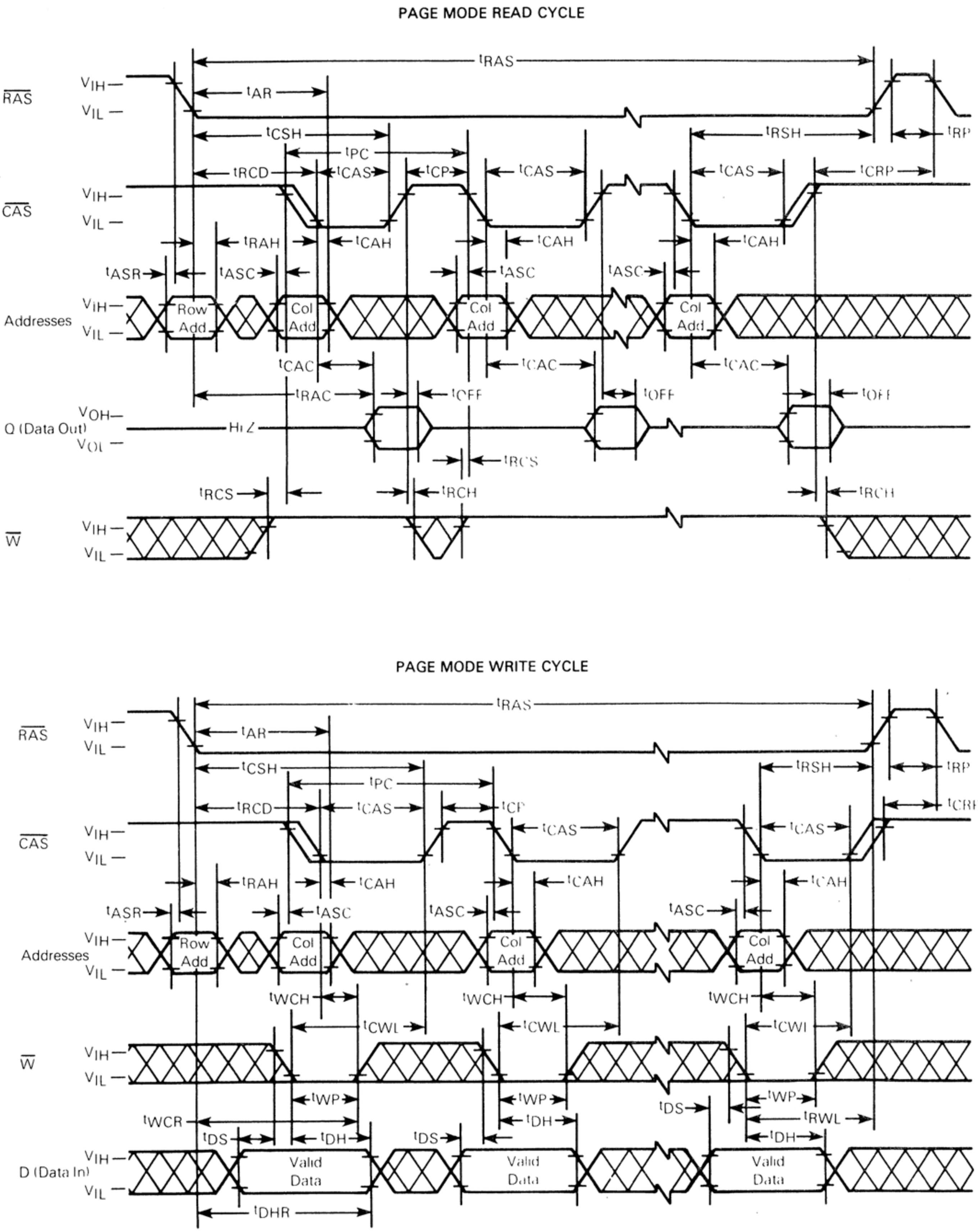
FIGURE 2-5 MC68000 DRAM BANK ORGANIZATION



READ CYCLE TIMING

WRITE CYCLE TIMING

FIGURE 3-1 READ AND WRITE CYCLES


PAGE MODE READ CYCLE

PAGE MODE WRITE CYCLE

FIGURE 3-2 PAGE MODE

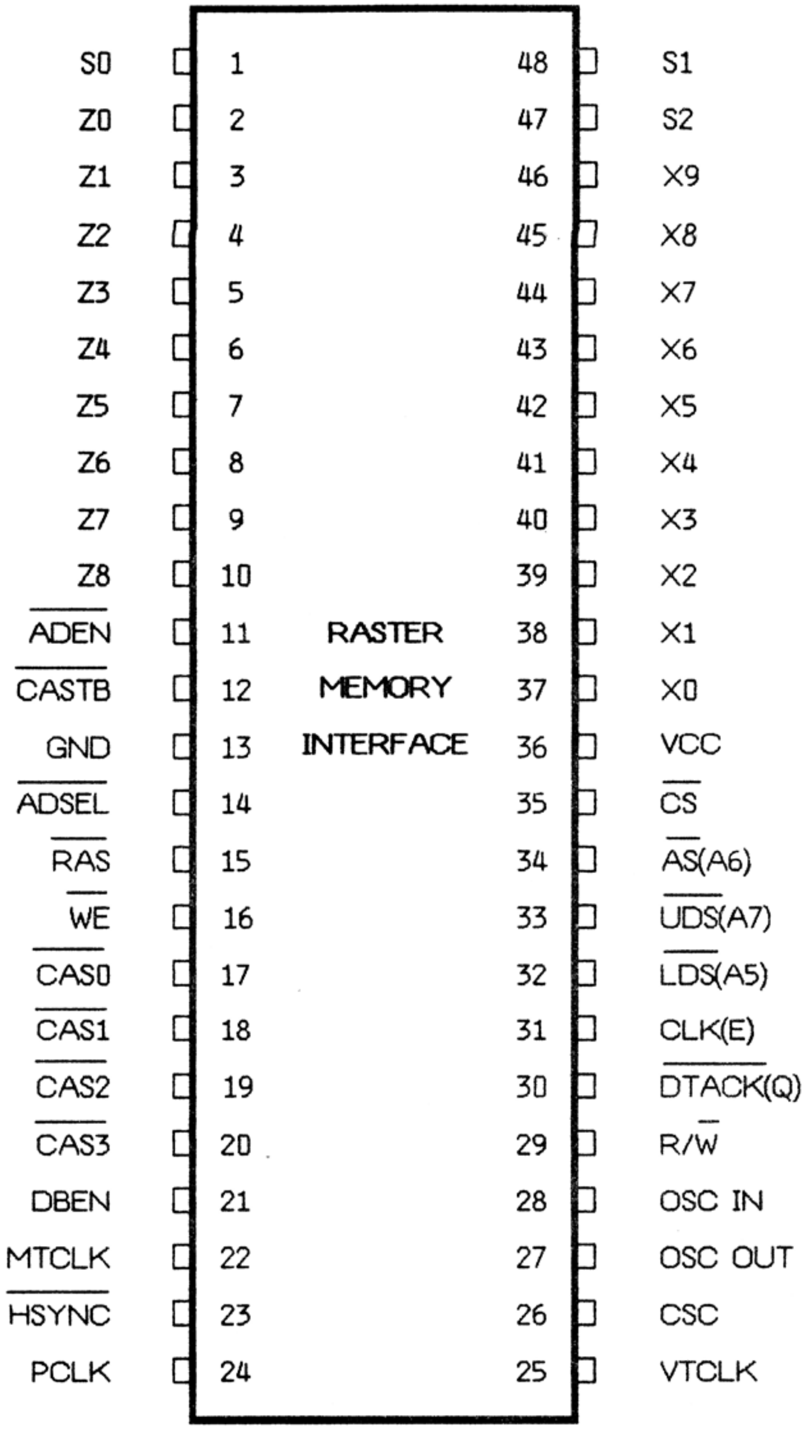


FIGURE 4-1 RMI PINOUT

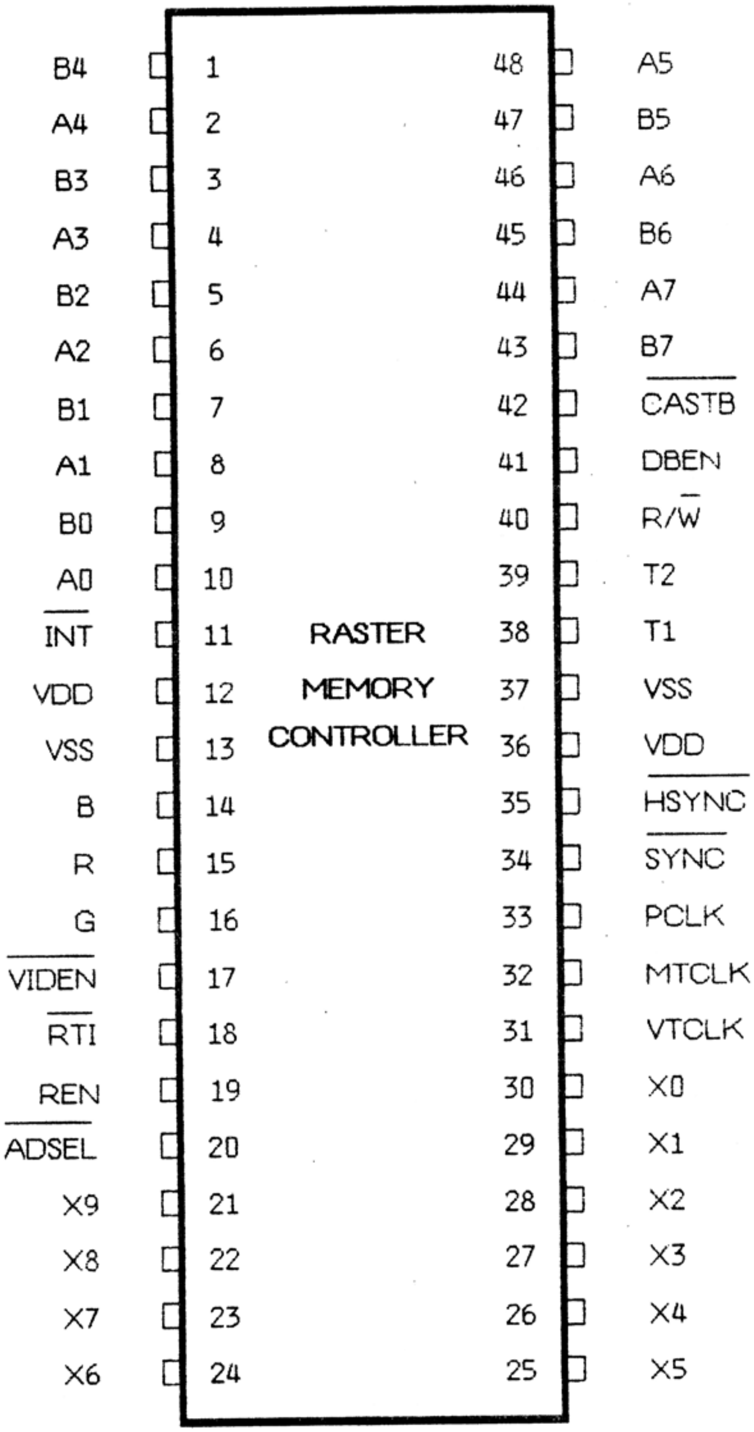
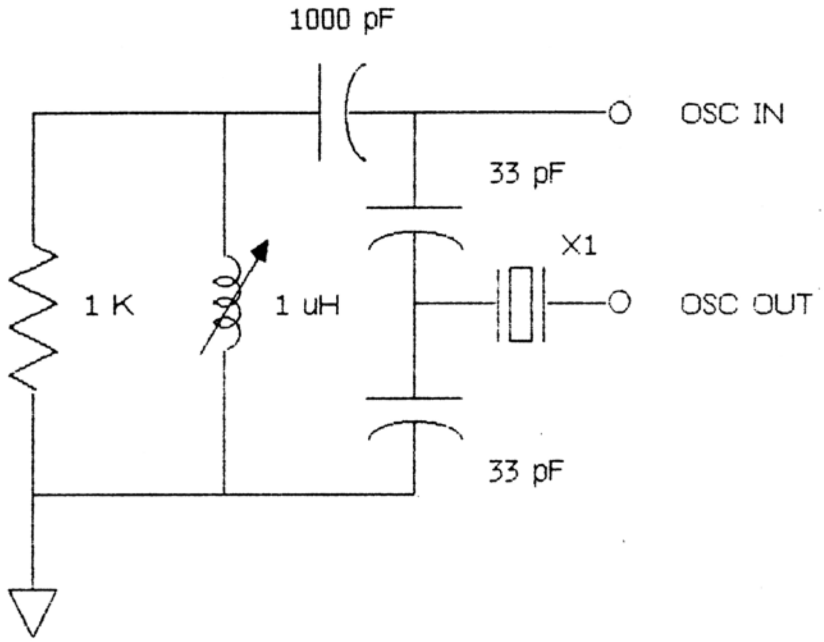


FIGURE 4-2 RMC PINOUT



X1 FREQUENCY

NTSC: 35.79545 MHz
PAL: 35.46895 MHz

FIGURE 4-3 CRYSTAL CIRCUIT

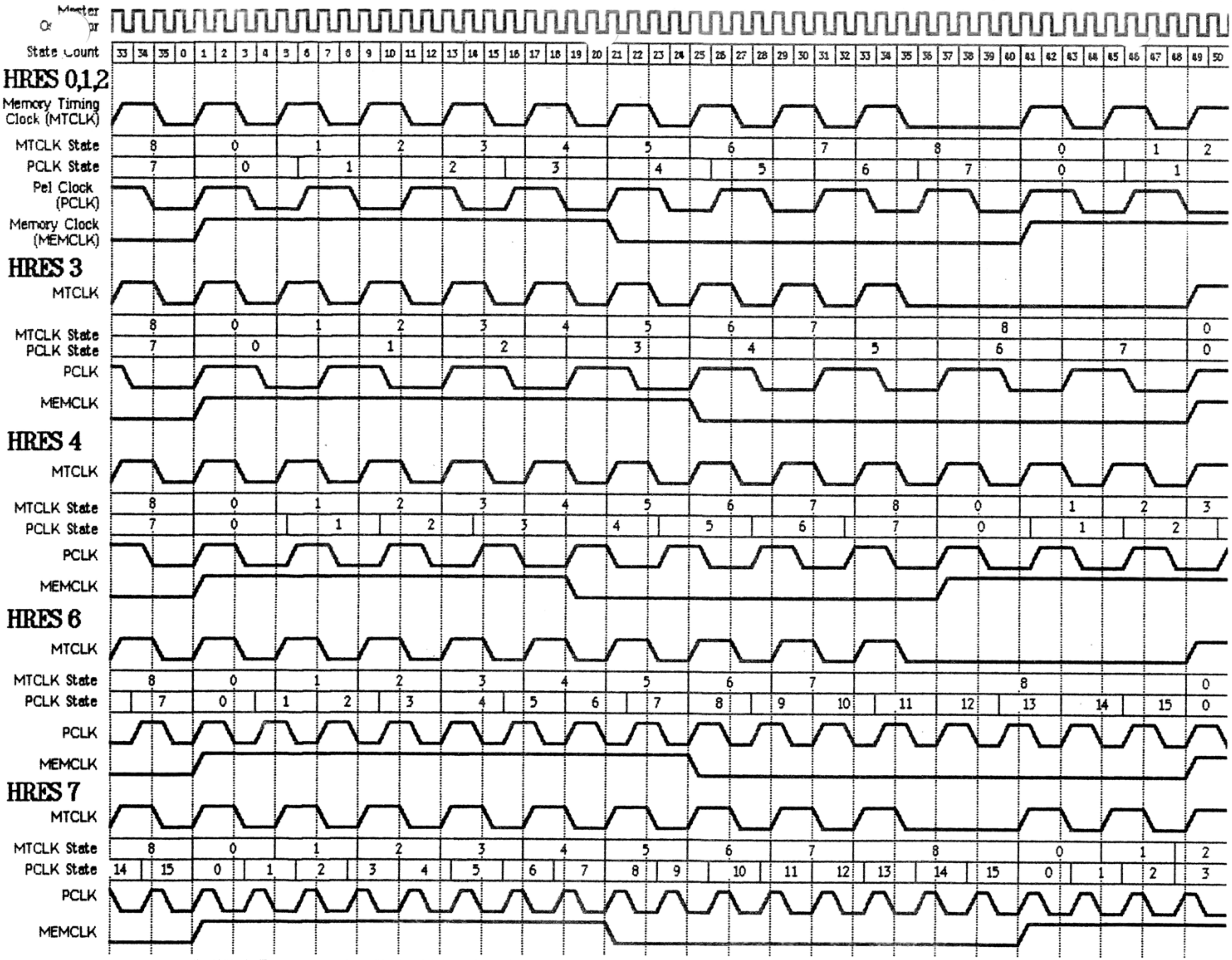


FIGURE 4-4 MTCLK AND PCLK vs HRES MODES

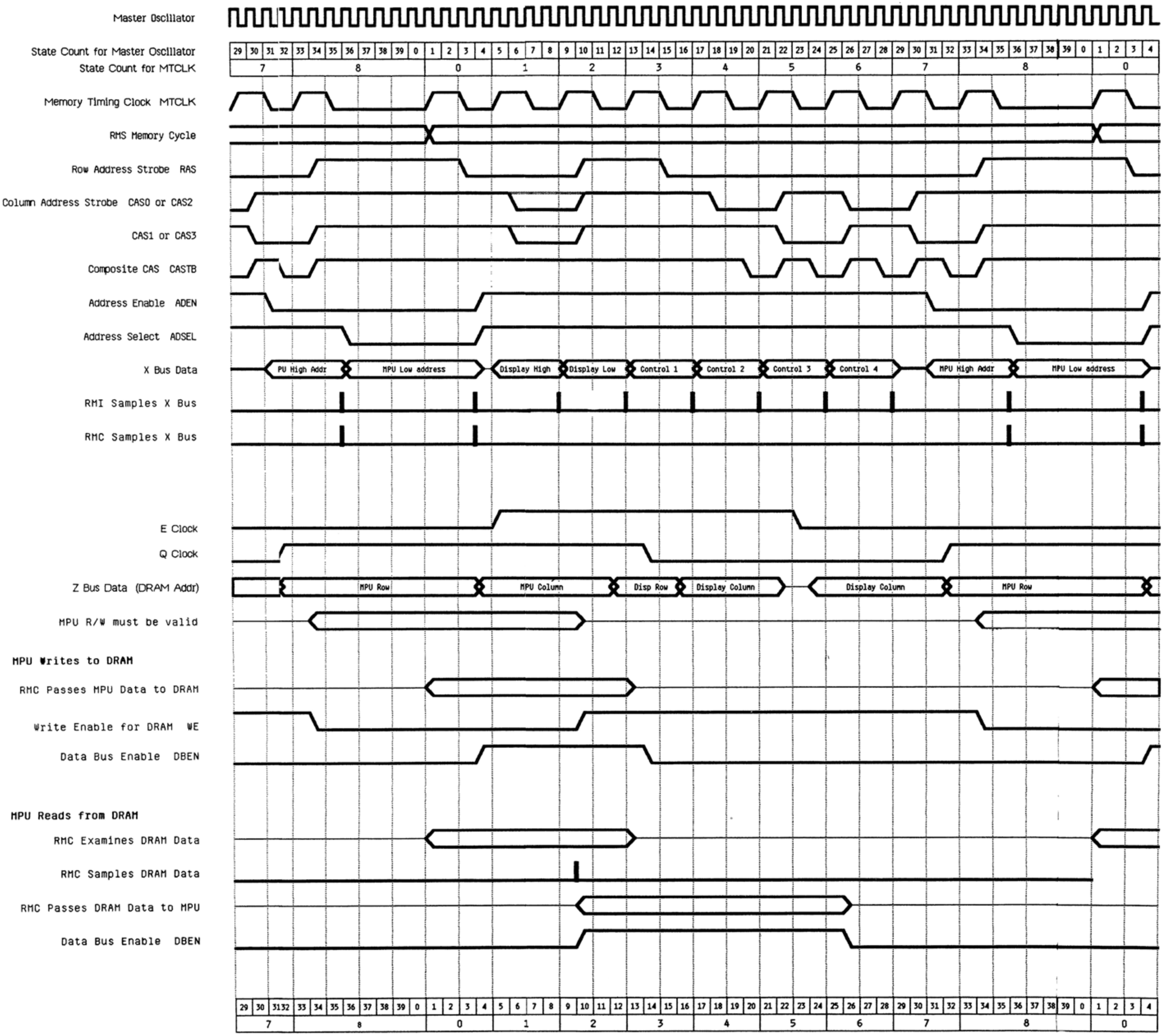


FIGURE 4-5a MC6809E X BUS TIMING FOR HRES 0,1,2,7

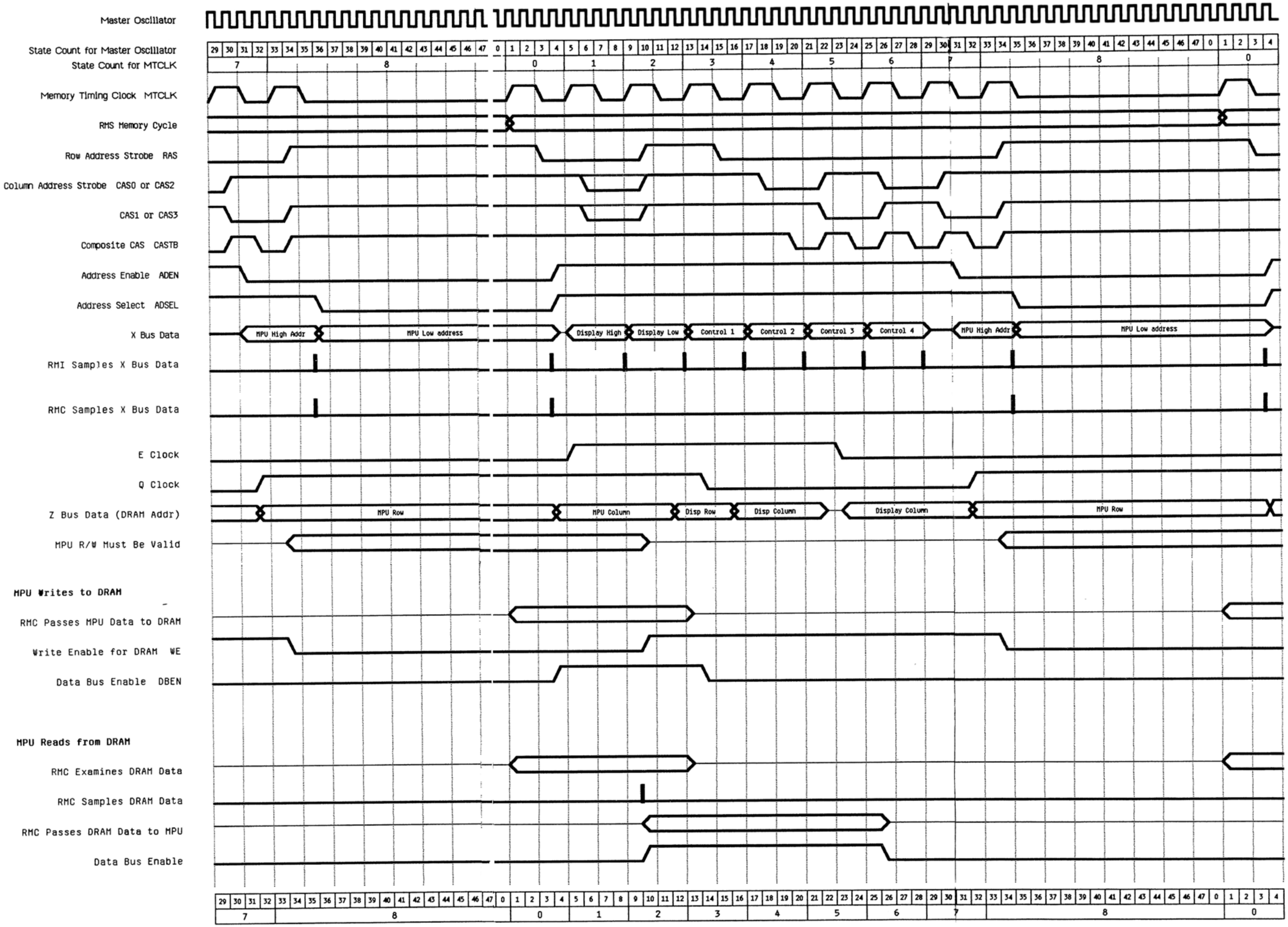


FIGURE 4-5b MC6809E X BUS TIMING FOR HRES 3,6

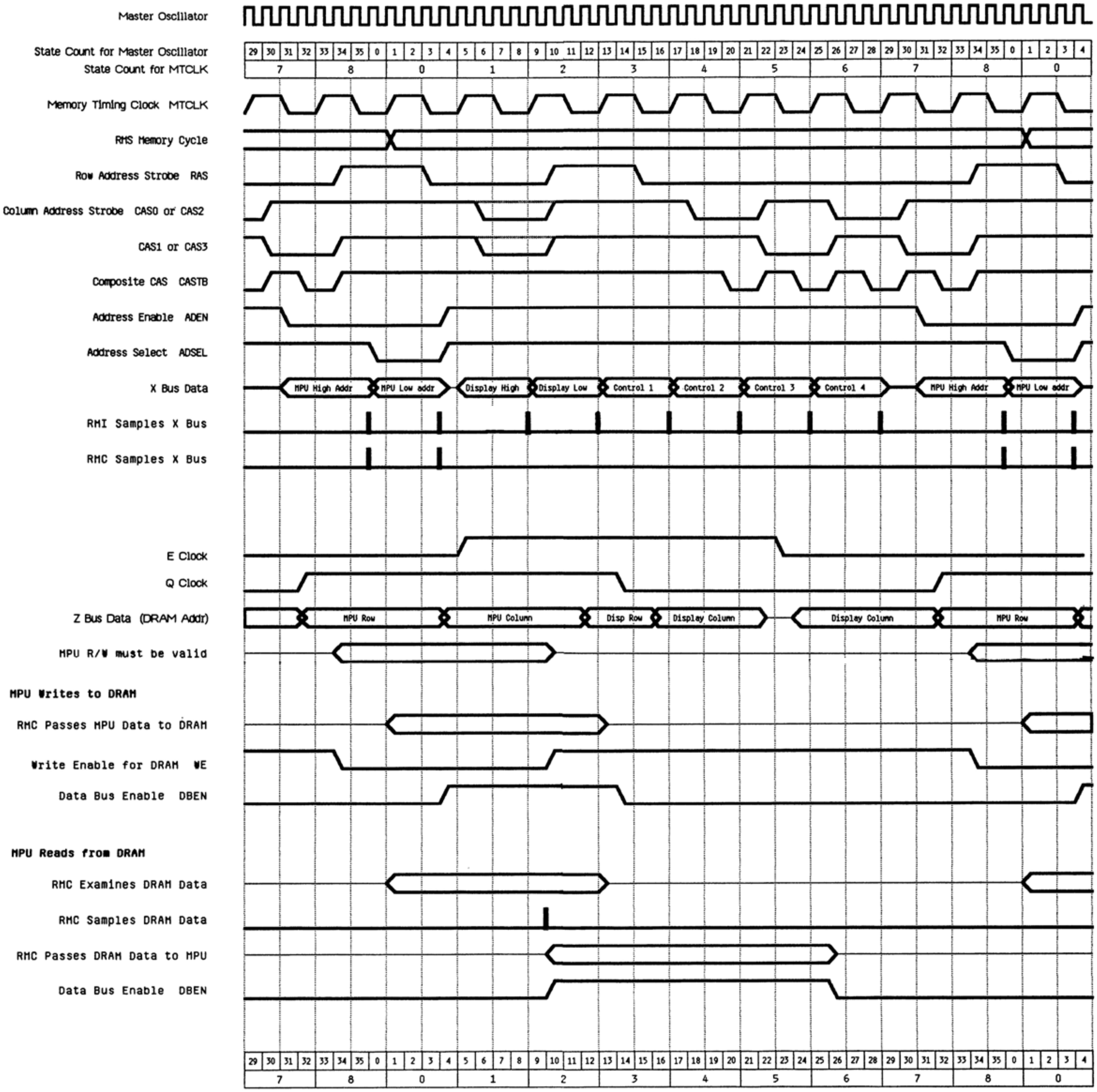


FIGURE 4-5c MC6809E X BUS TIMING FOR HRES 4

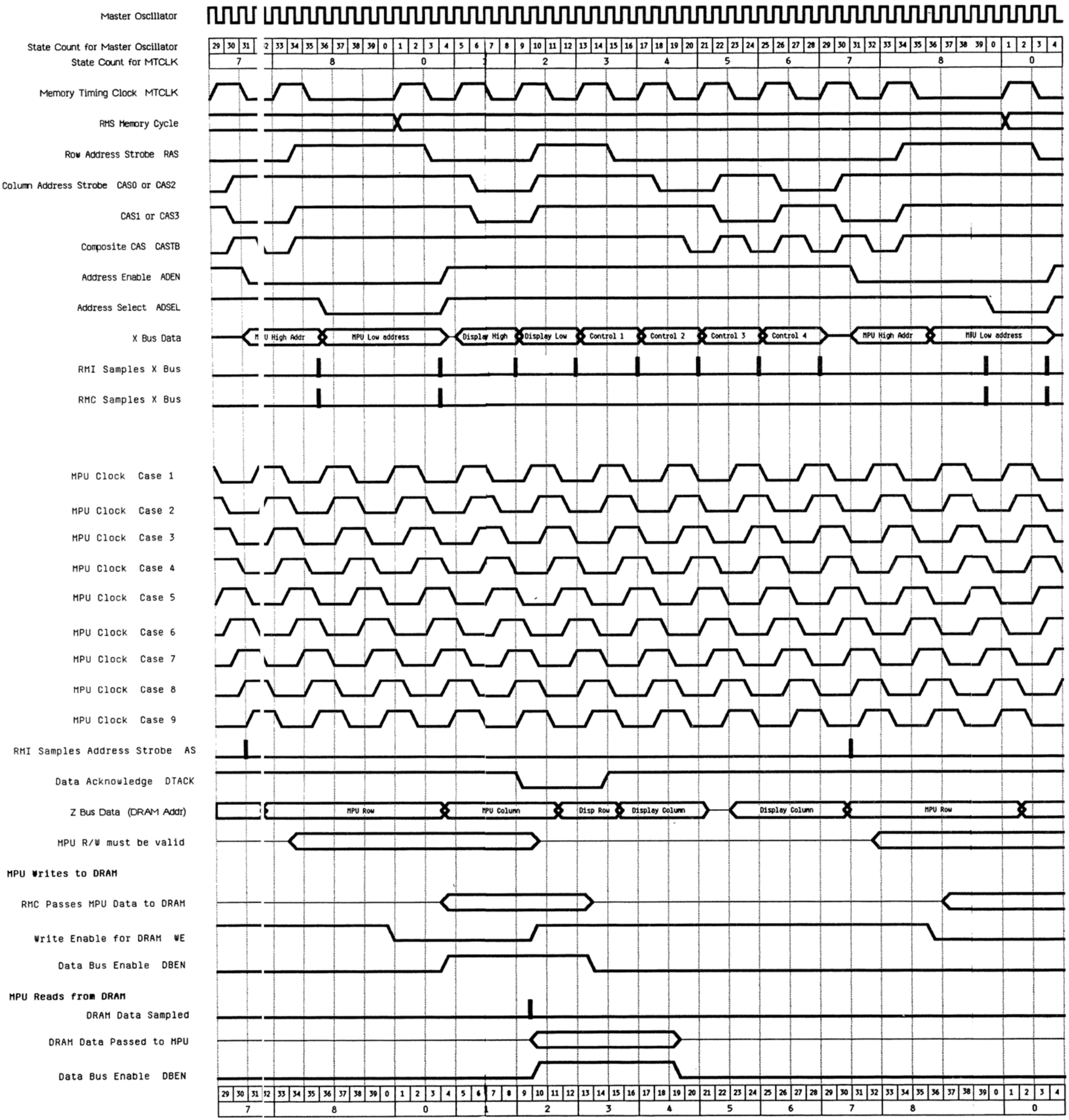


FIGURE 4-6a MC68000/8 X BUS TIMING FOR HRES 0, 1, 2, 7

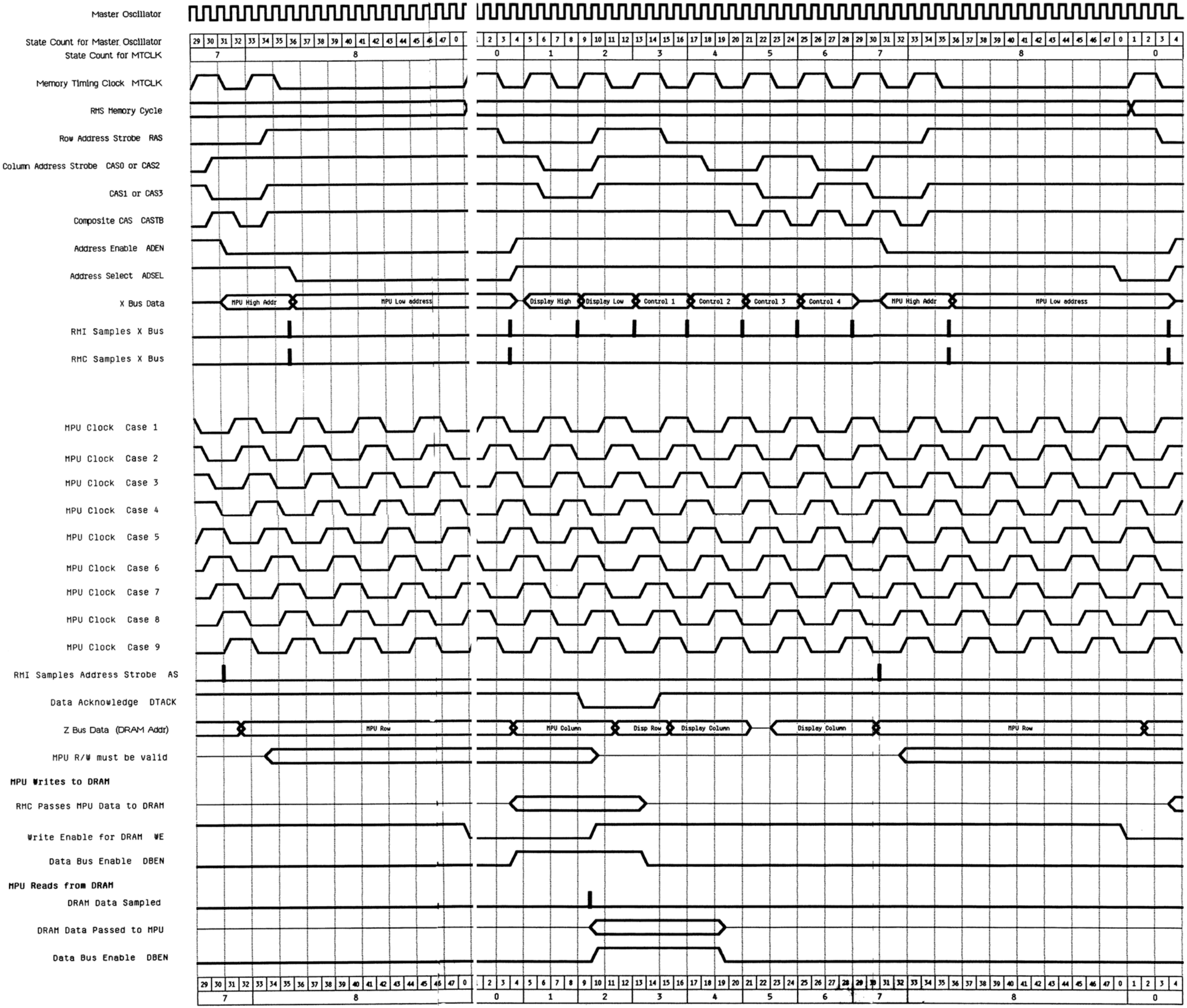


FIGURE 4-6b MC68000/8 X BUS TIMING FOR HRES 3, 6

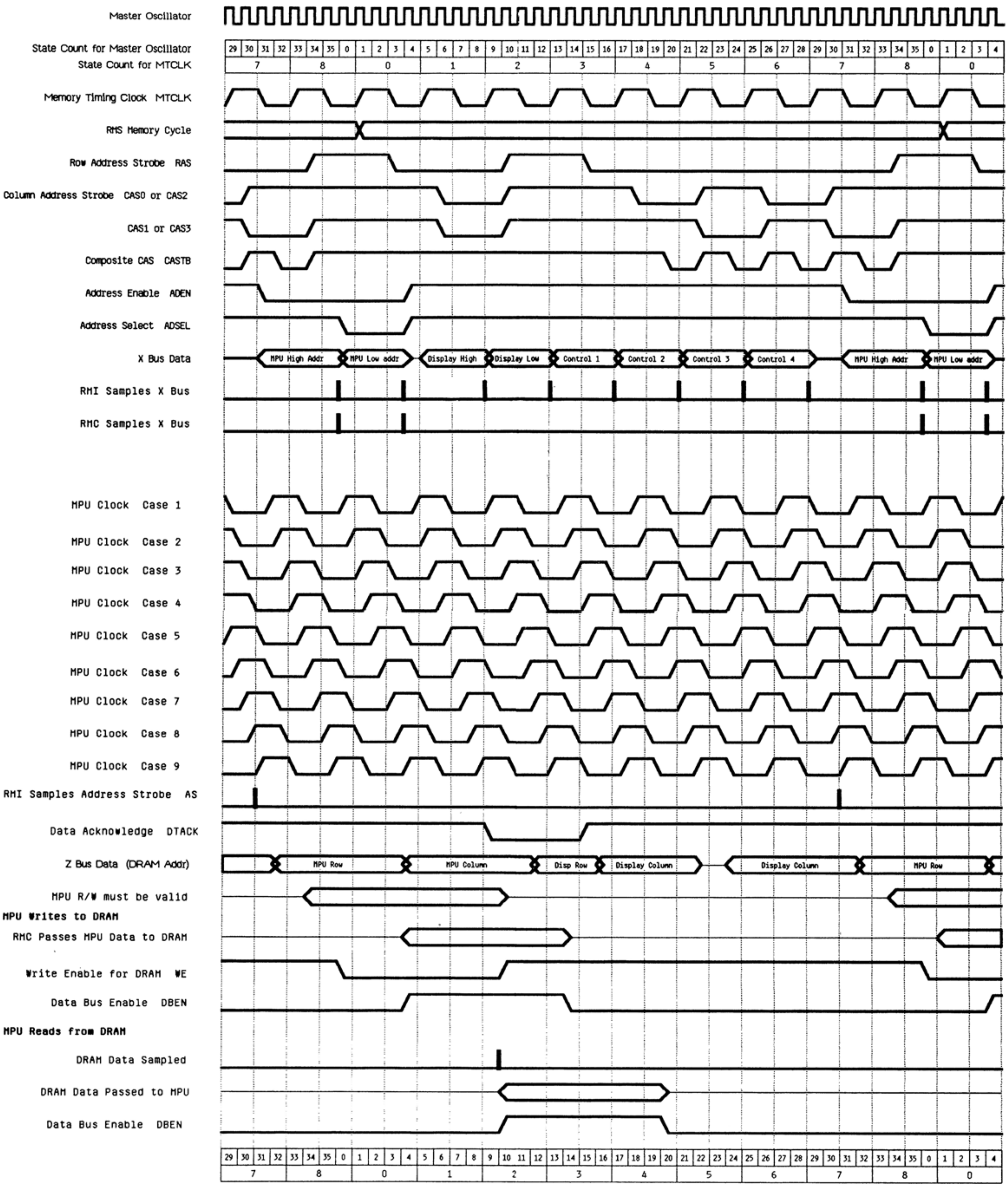
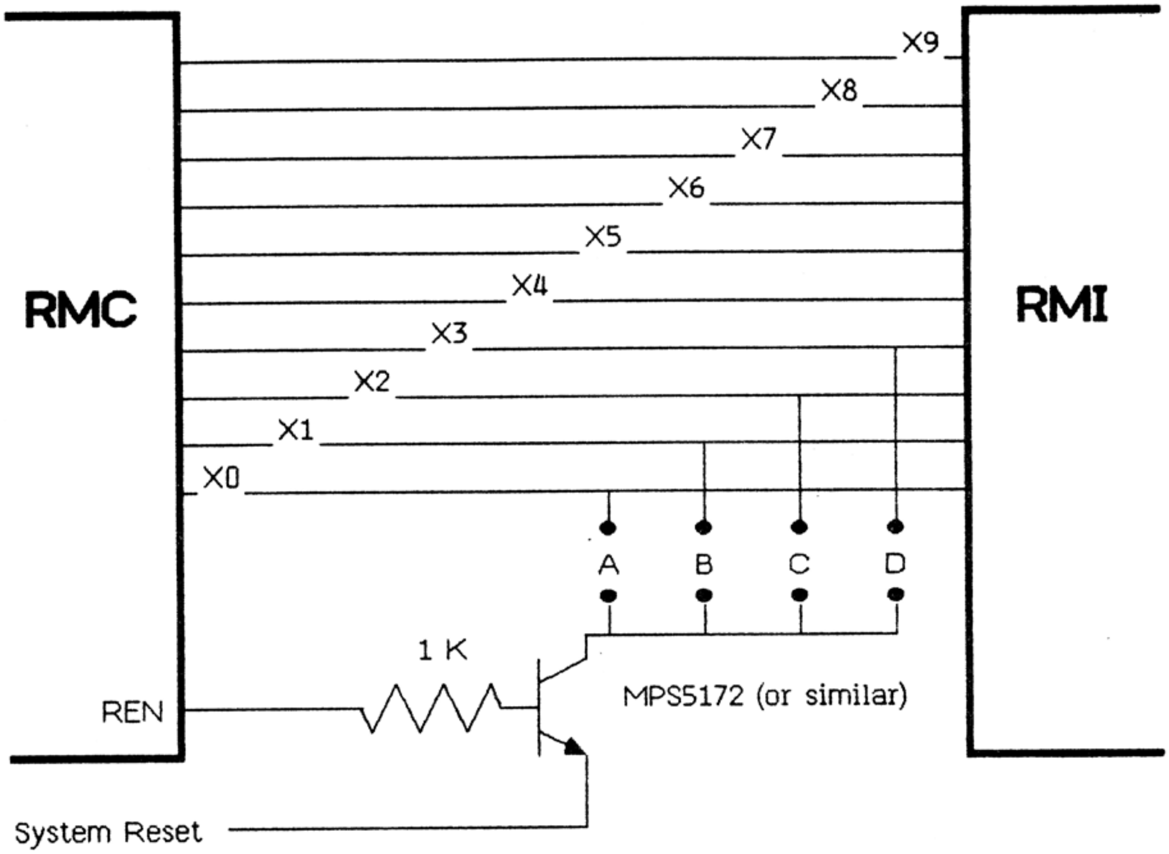


FIGURE 4-6c MC68000/8 X BUS TIMING FOR HRES 4



USE JUMPER A FOR MC6809E AND 625 LINE DISPLAY
USE JUMPER B FOR MC6809E AND 525 LINE DISPLAY
USE JUMPER C FOR MC68008 OR MC68000 AND 625 LINE DISPLAY
USE JUMPER D FOR MC68008 OR MC68000 AND 525 LINE DISPLAY

FIGURE 5-1 X BUS RESET

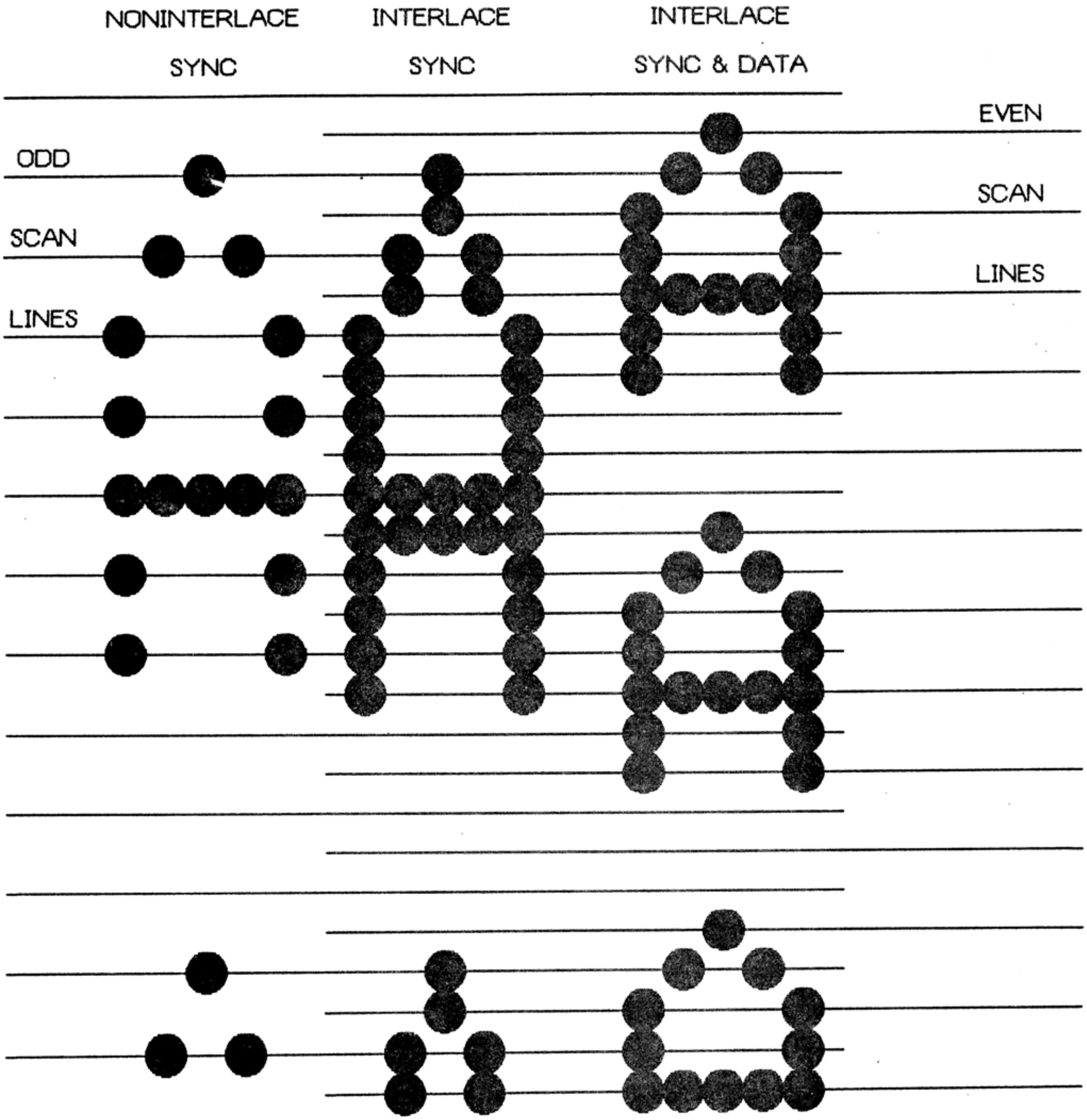


FIGURE 6-1 INTERLACE AND CHARACTER SIZE

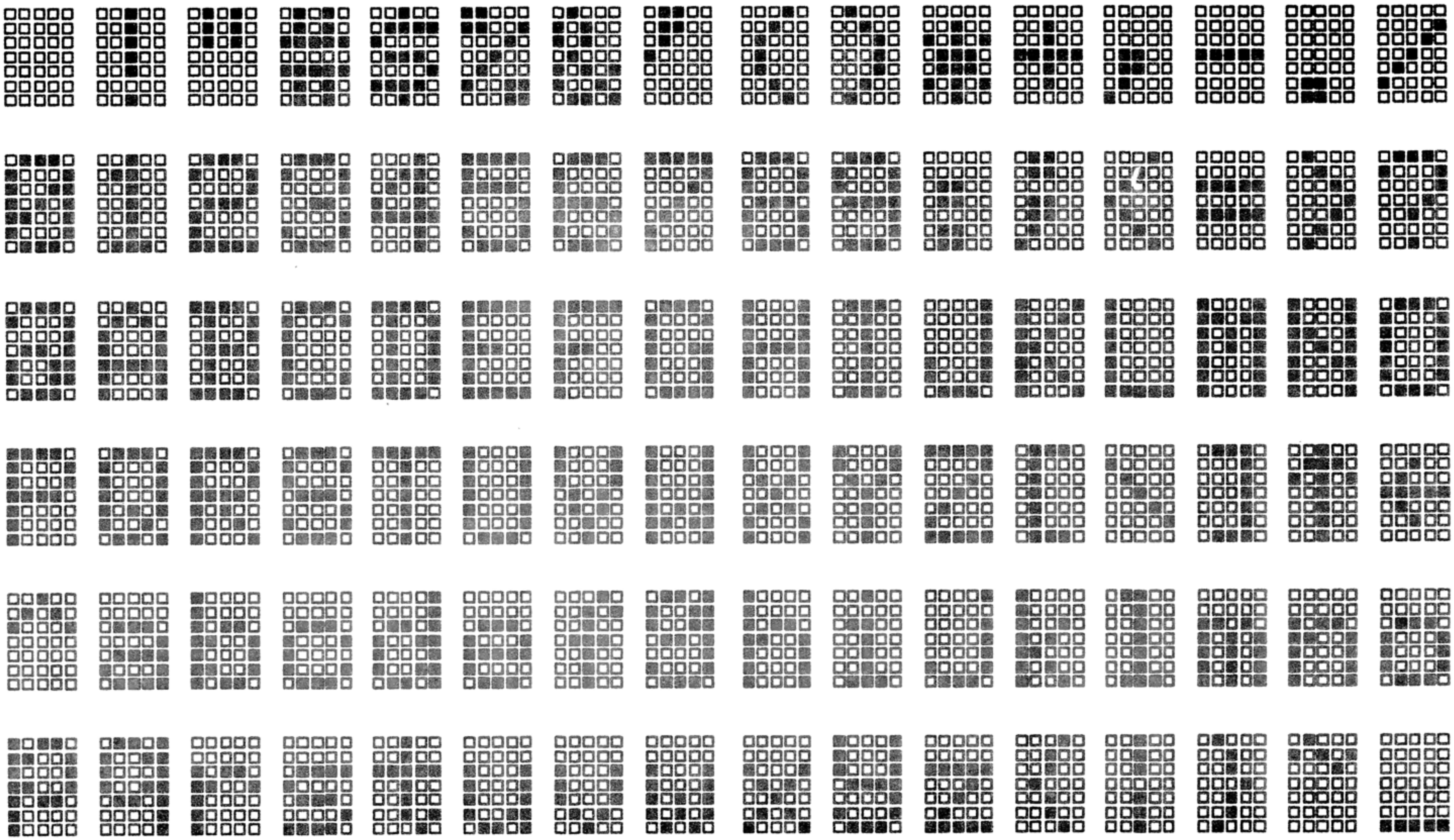
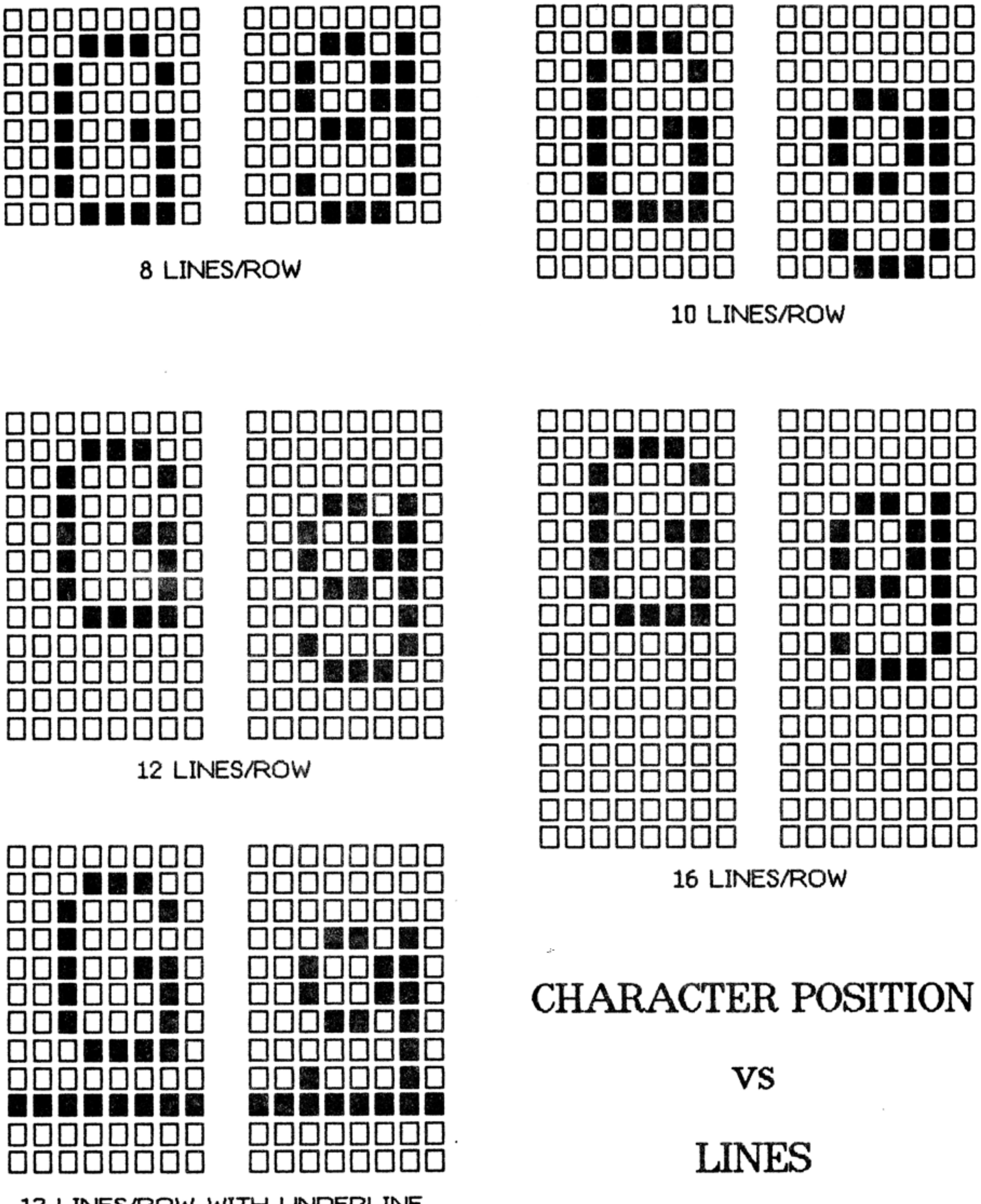


FIGURE 7-1 ASCII CHARACTER FORMAT



8 LINES/ROW

10 LINES/ROW

12 LINES/ROW

16 LINES/ROW

12 LINES/ROW WITH UNDERLINE

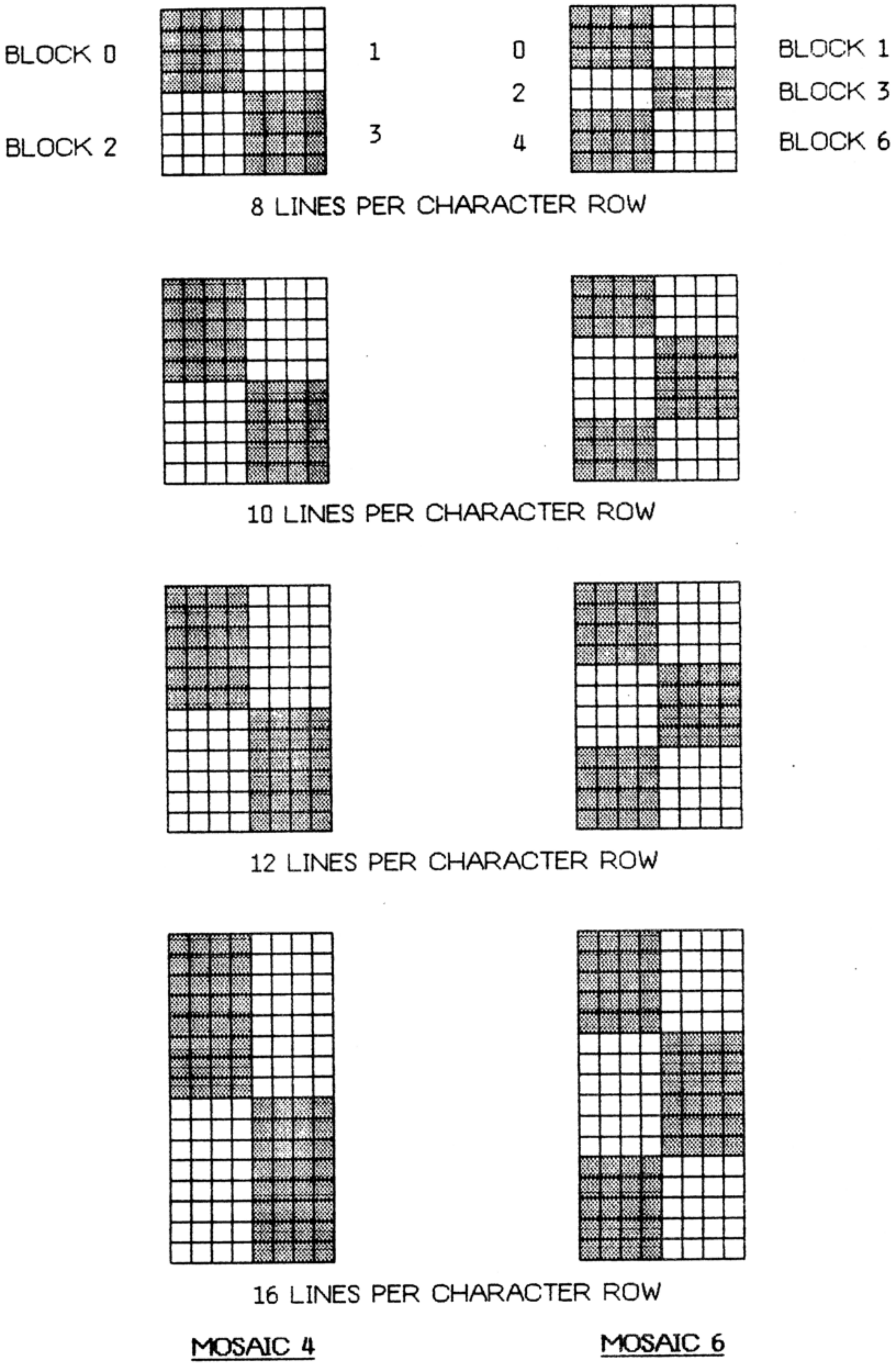
CHARACTER POSITION

VS

LINES

PER CHARACTER ROW

FIGURE 7-2



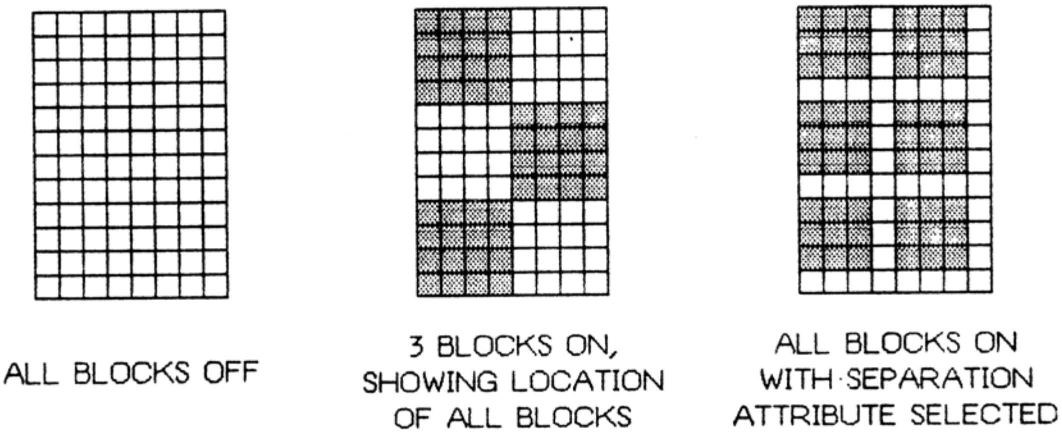
8 LINES PER CHARACTER ROW

10 LINES PER CHARACTER ROW

12 LINES PER CHARACTER ROW

16 LINES PER CHARACTER ROW

FIGURE 7–3 MOSAICS 4 AND 6



ALL BLOCKS OFF

3 BLOCKS ON,
SHOWING LOCATION
OF ALL BLOCKS

ALL BLOCKS ON
WITH SEPARATION
ATTRIBUTE SELECTED

FIGURE 7-4 MOSAIC SEPARATION

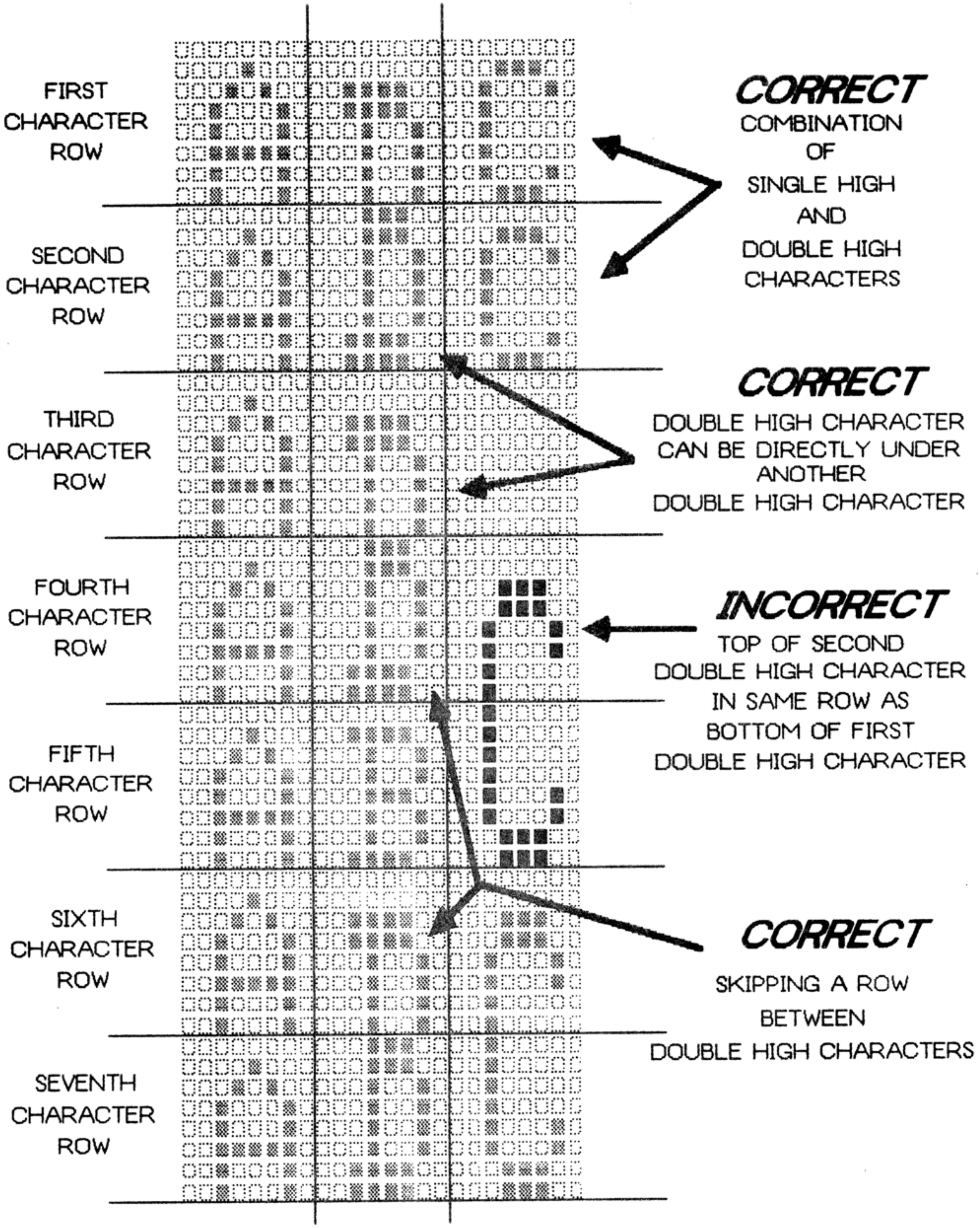
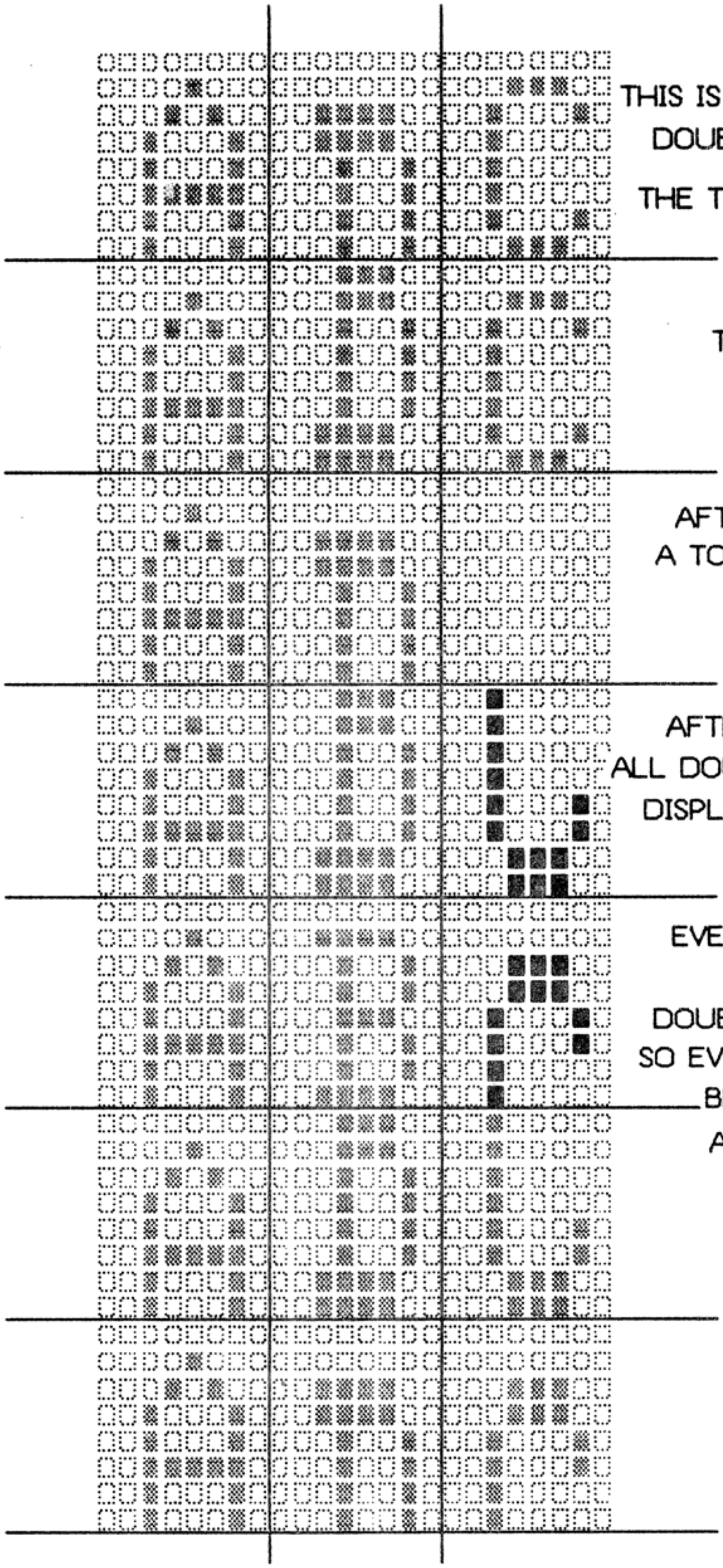

FIRST CHARACTER ROW
SECOND CHARACTER ROW
THIRD CHARACTER ROW
FOURTH CHARACTER ROW
FIFTH CHARACTER ROW
SIXTH CHARACTER ROW
SEVENTH CHARACTER ROW
CORRECT
COMBINATION OF SINGLE HIGH AND DOUBLE HIGH CHARACTERS
CORRECT
DOUBLE HIGH CHARACTER CAN BE DIRECTLY UNDER ANOTHER DOUBLE HIGH CHARACTER
INCORRECT
TOP OF SECOND DOUBLE HIGH CHARACTER IN SAME ROW AS BOTTOM OF FIRST DOUBLE HIGH CHARACTER
CORRECT
SKIPPING A ROW BETWEEN DOUBLE HIGH CHARACTERS

FIGURE 7-5 DOUBLE HIGH CHARACTERS



THIS IS THE FIRST ROW WITH A DOUBLE HIGH CHARACTER

THE TOP HALF IS DISPLAYED

THE BOTTOM HALF IS DISPLAYED

AFTER A BOTTOM HALF A TOP HALF IS DISPLAYED

AFTER A TOP-HALF ROW ALL DOUBLE HIGH CHARACTERS DISPLAY THE BOTTOM HALF

EVERY FOLLOWING ROW CONTAINS A DOUBLE HIGH CHARACTER SO EVERY ROW ALTERNATES BETWEEN TOP HALF AND BOTTOM HALF

FIGURE 7-6 DOUBLE HIGH CHARACTER ERROR

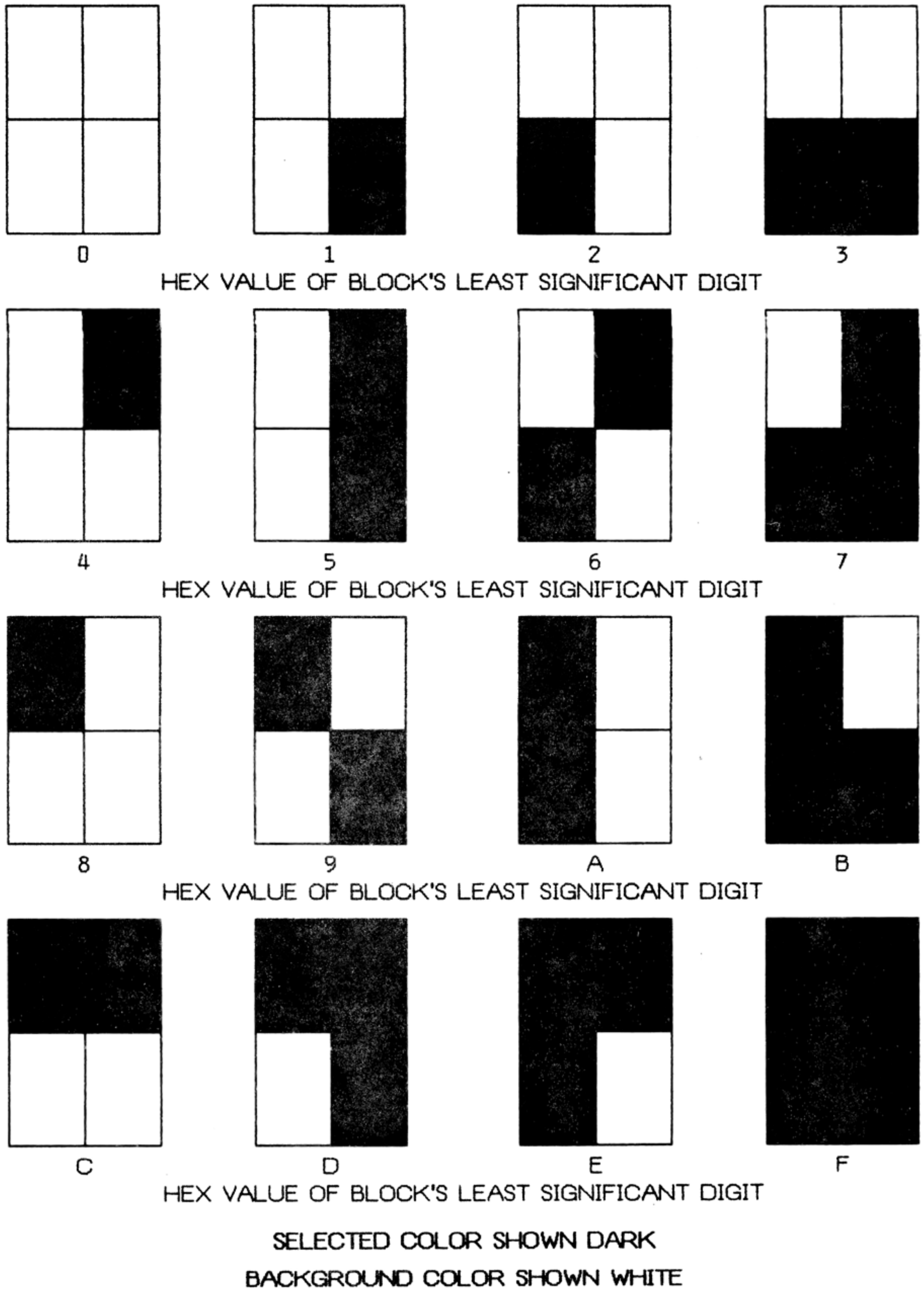
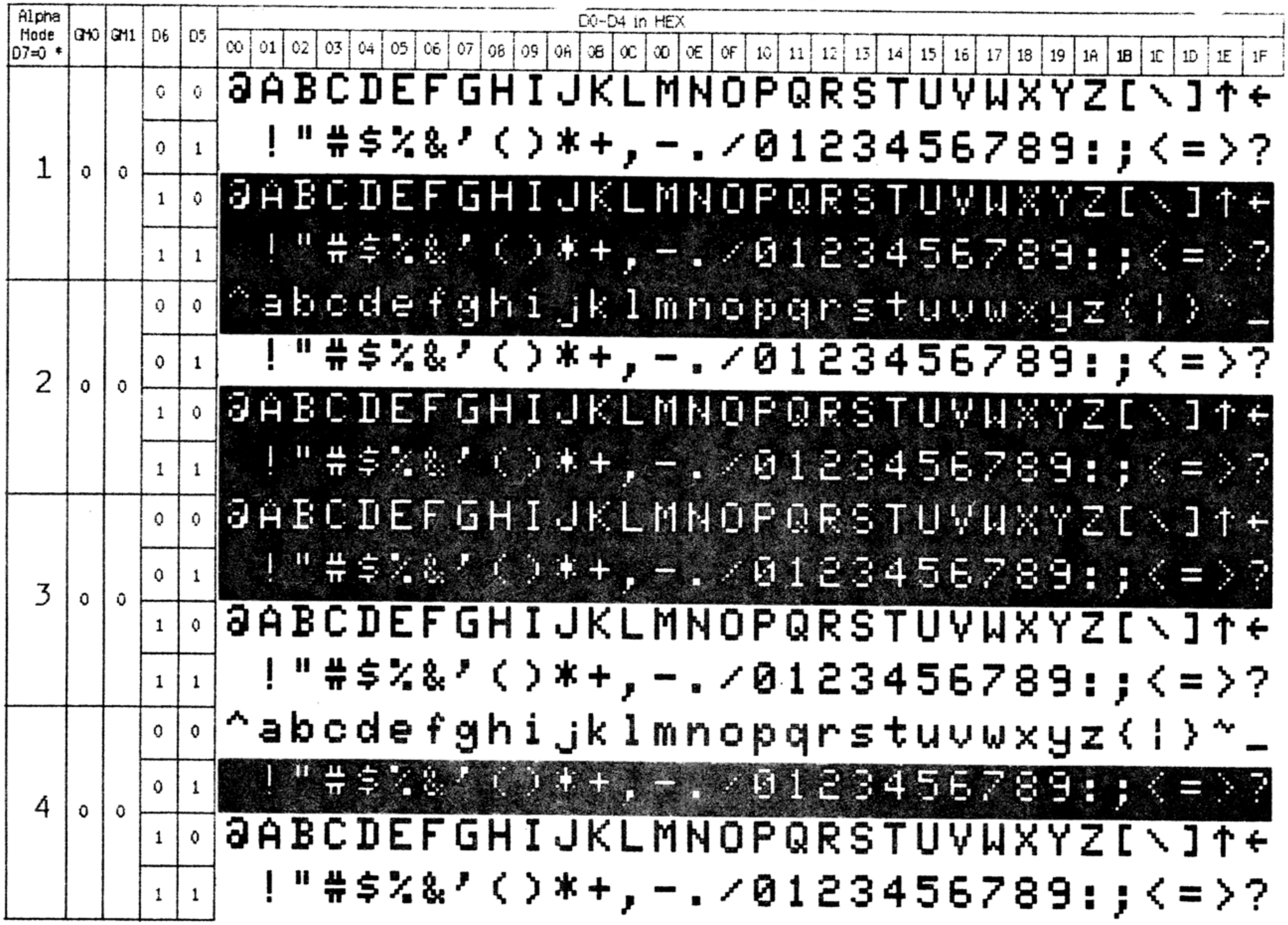

0
1
2
3
HEX VALUE OF BLOCK'S LEAST SIGNIFICANT DIGIT
4
5
6
7
HEX VALUE OF BLOCK'S LEAST SIGNIFICANT DIGIT
8
9
A
B
HEX VALUE OF BLOCK'S LEAST SIGNIFICANT DIGIT
C
D
E
F
HEX VALUE OF BLOCK'S LEAST SIGNIFICANT DIGIT
SELECTED COLOR SHOWN DARK
BACKGROUND COLOR SHOWN WHITE

FIGURE 14-1 SEMIGRAPHIC BLOCKS



Alpha Mode D7=0 *	GM0	GM1	D6	D5	D0-D4 in HEX																															
					00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
1	0	0	0	0	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	↑	←
			0	1		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
			1	0	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	↑	←
			1	1		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
2	0	0	0	0	^	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	_
			0	1		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
			1	0	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	↑	←
			1	1		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
3	0	0	0	0	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	↑	←
			0	1		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
			1	0	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	↑	←
			1	1		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	0	0	0	0	^	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	_
			0	1		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
			1	0	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	↑	←
			1	1		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?

NOTE : BLACK REPRESENTS THE FOREGROUND COLOR
WHITE REPRESENTS THE BACKGROUND COLOR

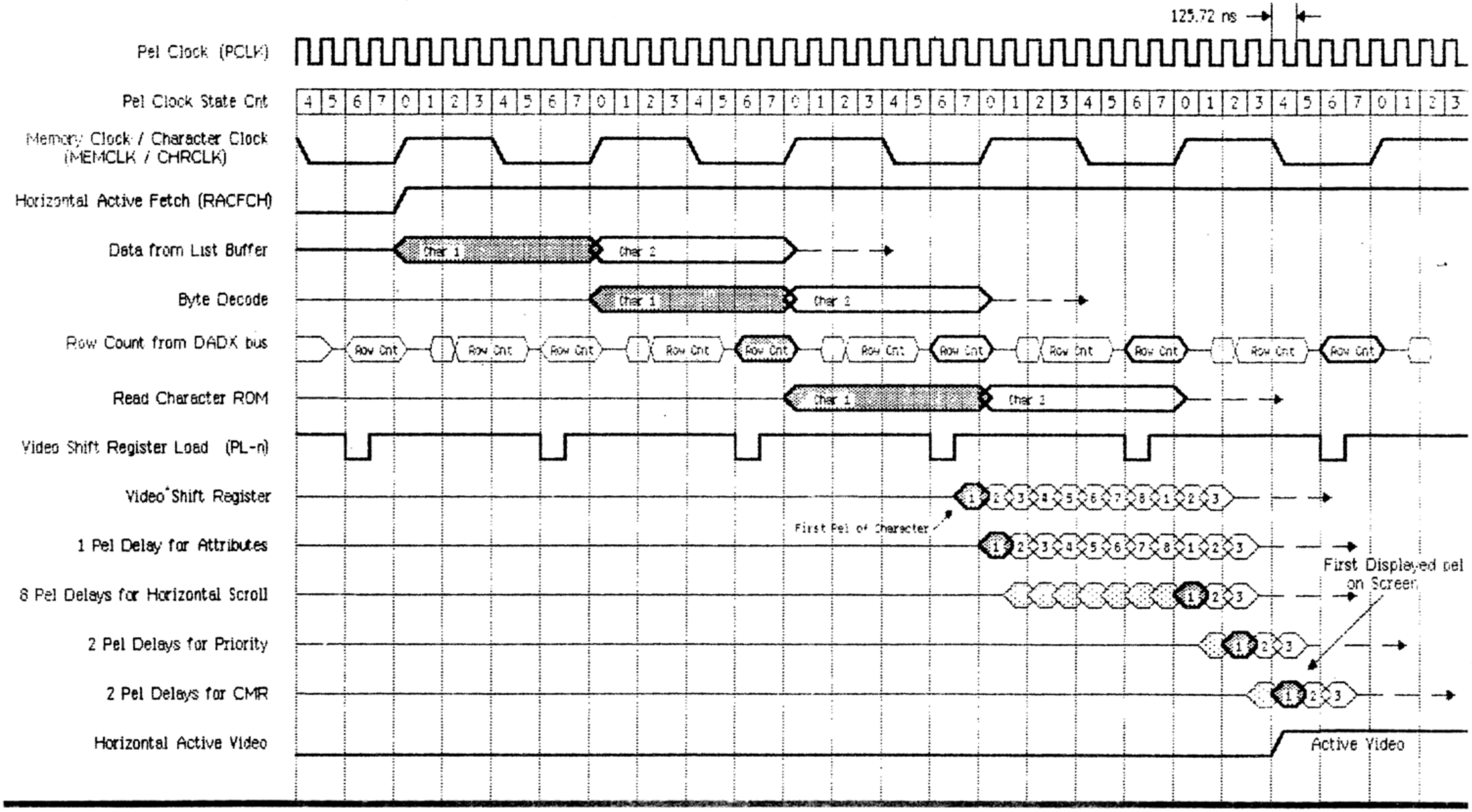
* D7 IS INTERNALLY CONNECTED TO $A/\overline{S}$. IF D7=1 THEN MOSAIC 4 CHARACTERS ARE SELECTED

FIGURE 14-2 MACHINE 2 ALPHANUMERIC FORMAT

Raster Memory System Timing Diagram

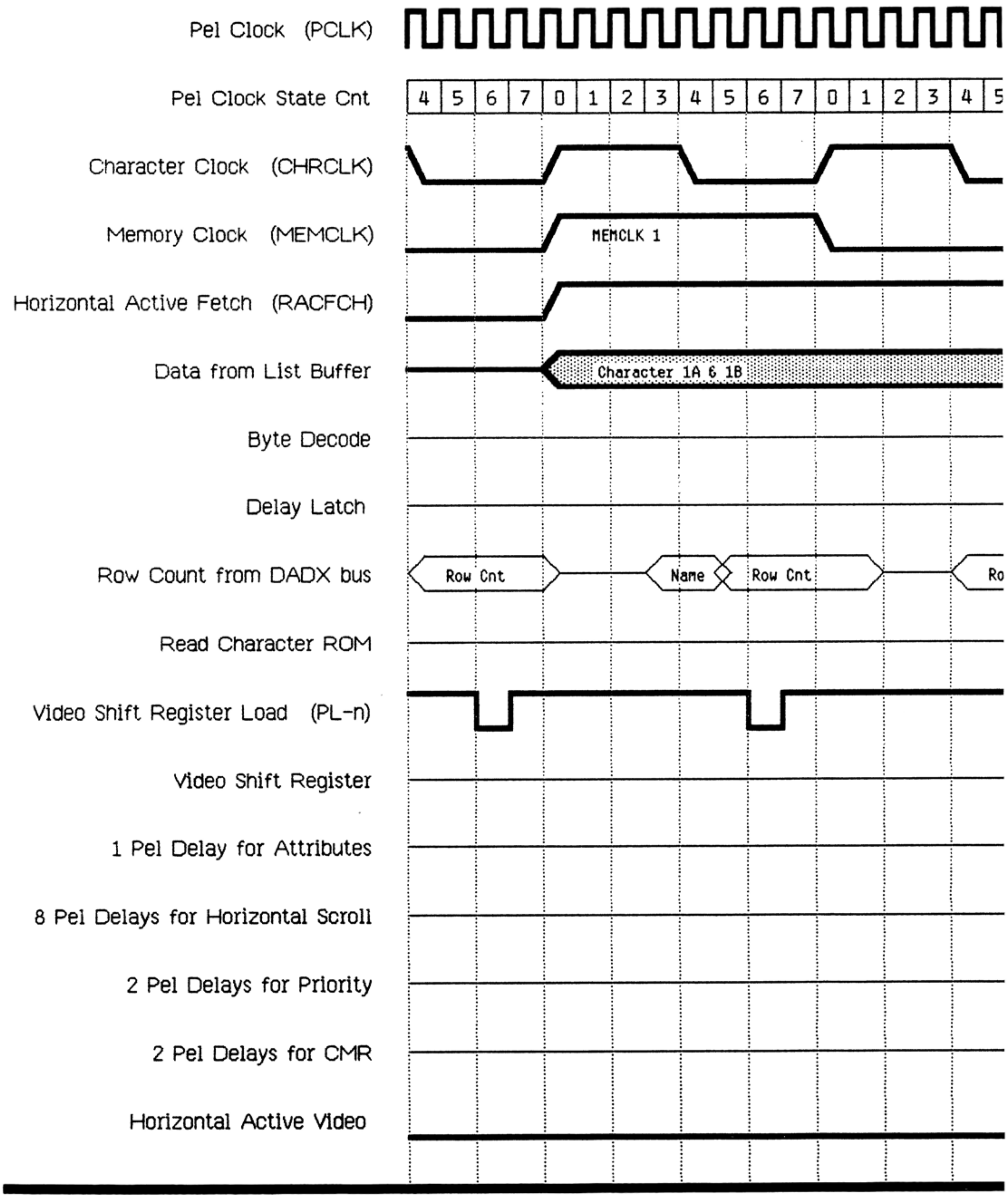
Video Pipe Line Delay for Internal Characters

Low Resolution (HRES 4)



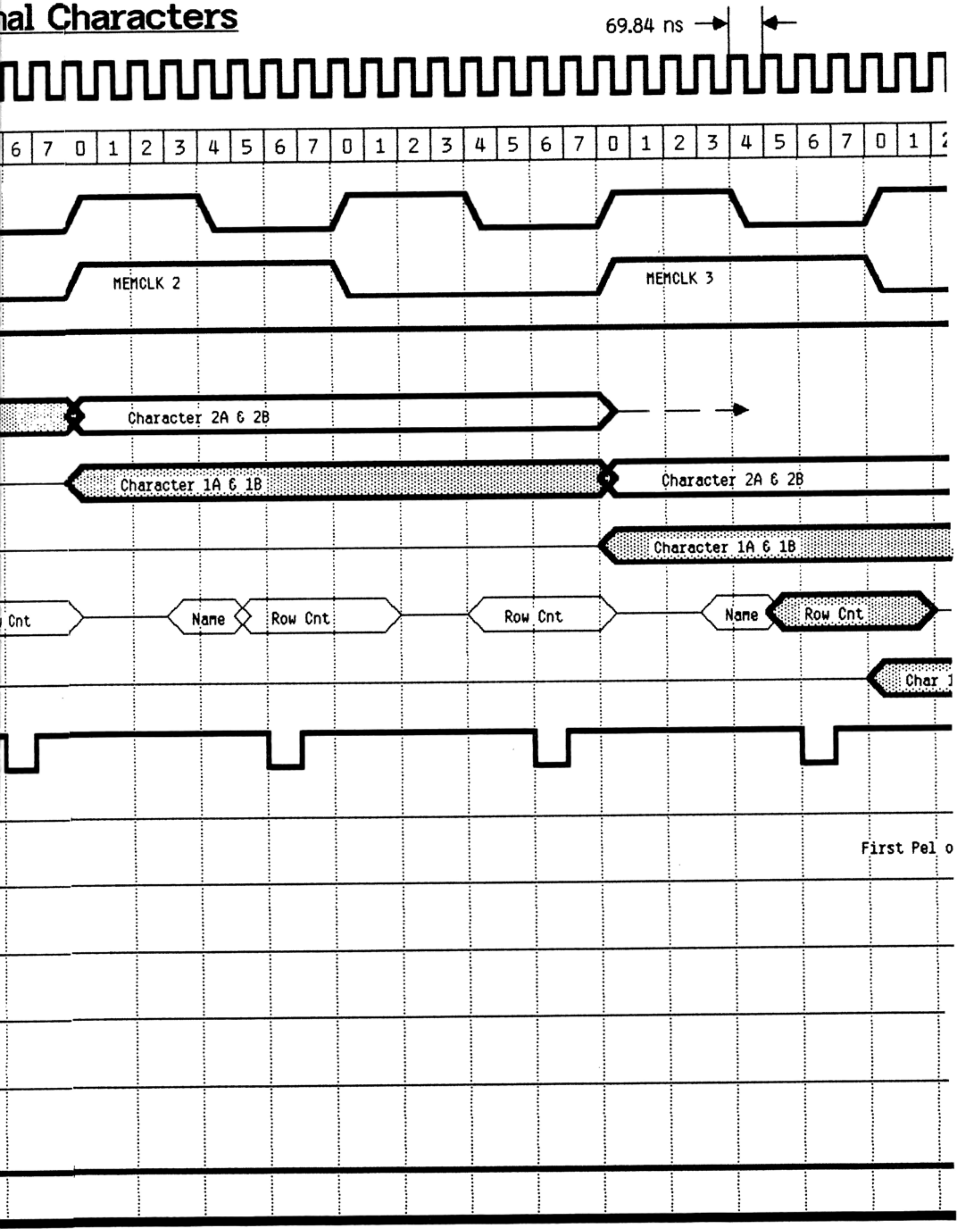
Raster Memory System Timing Diagram

Video Pipe Line Delay for Inter



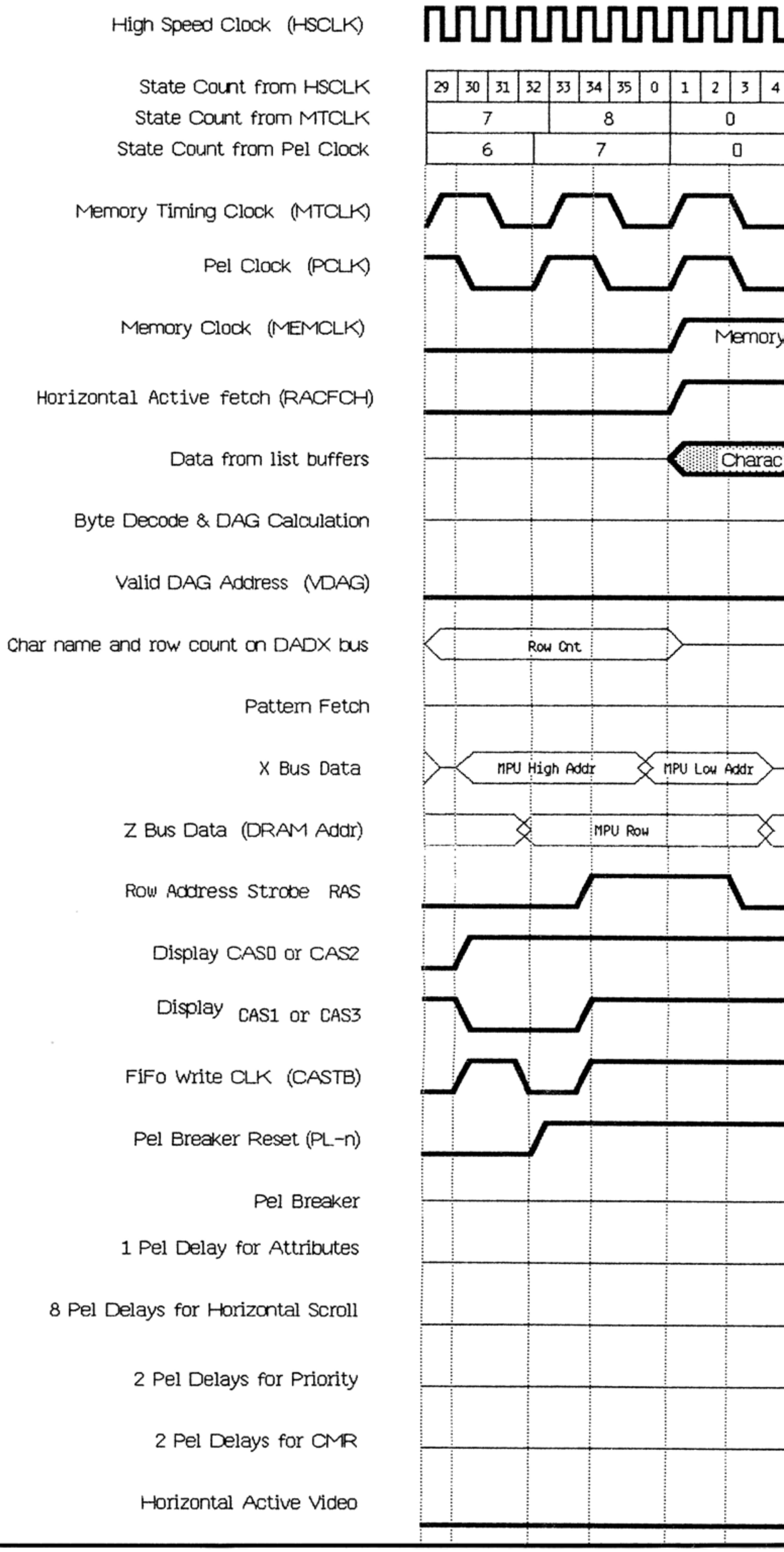
High Resolution (HRES 7)

nal Characters



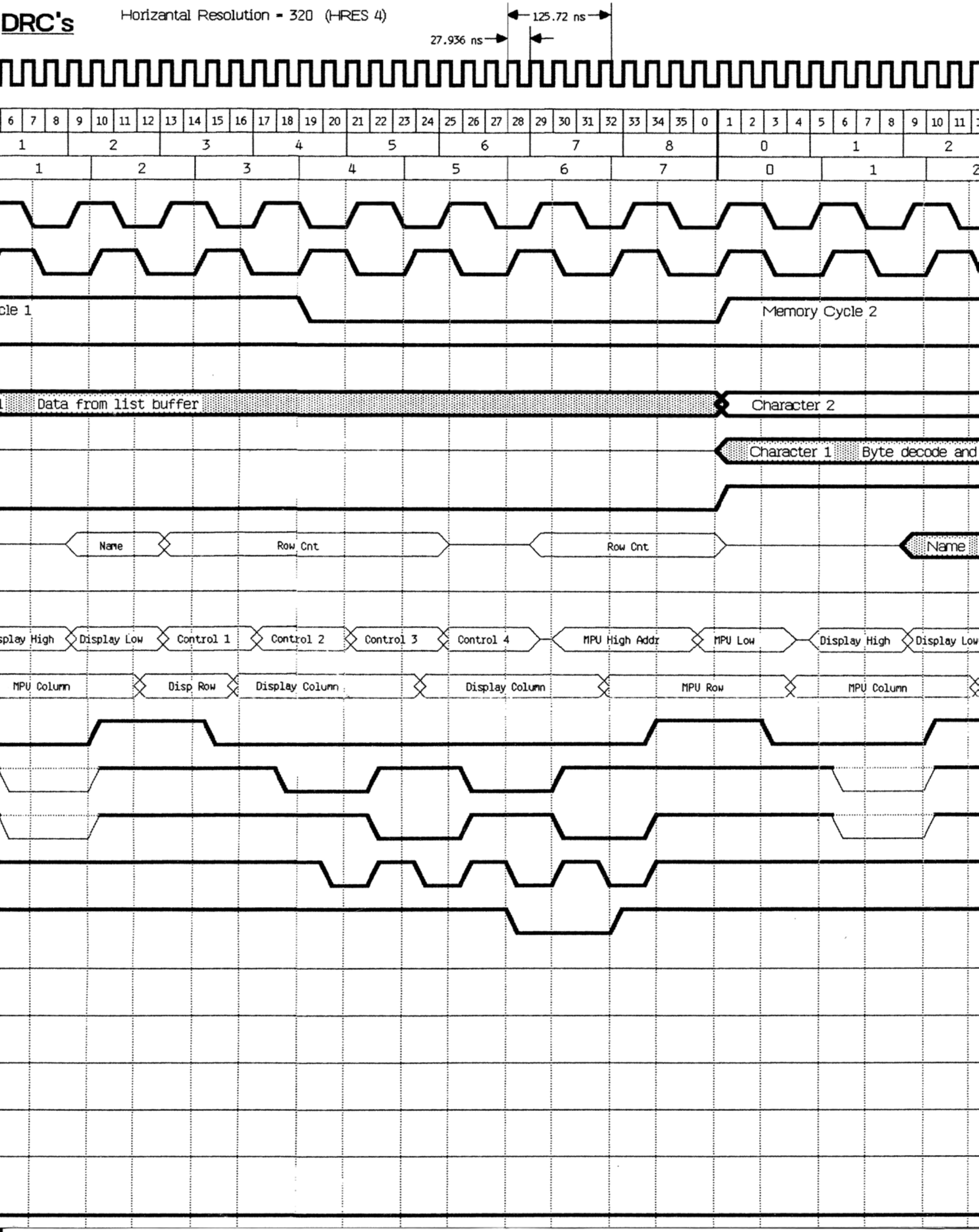
Raster Memory System Timing

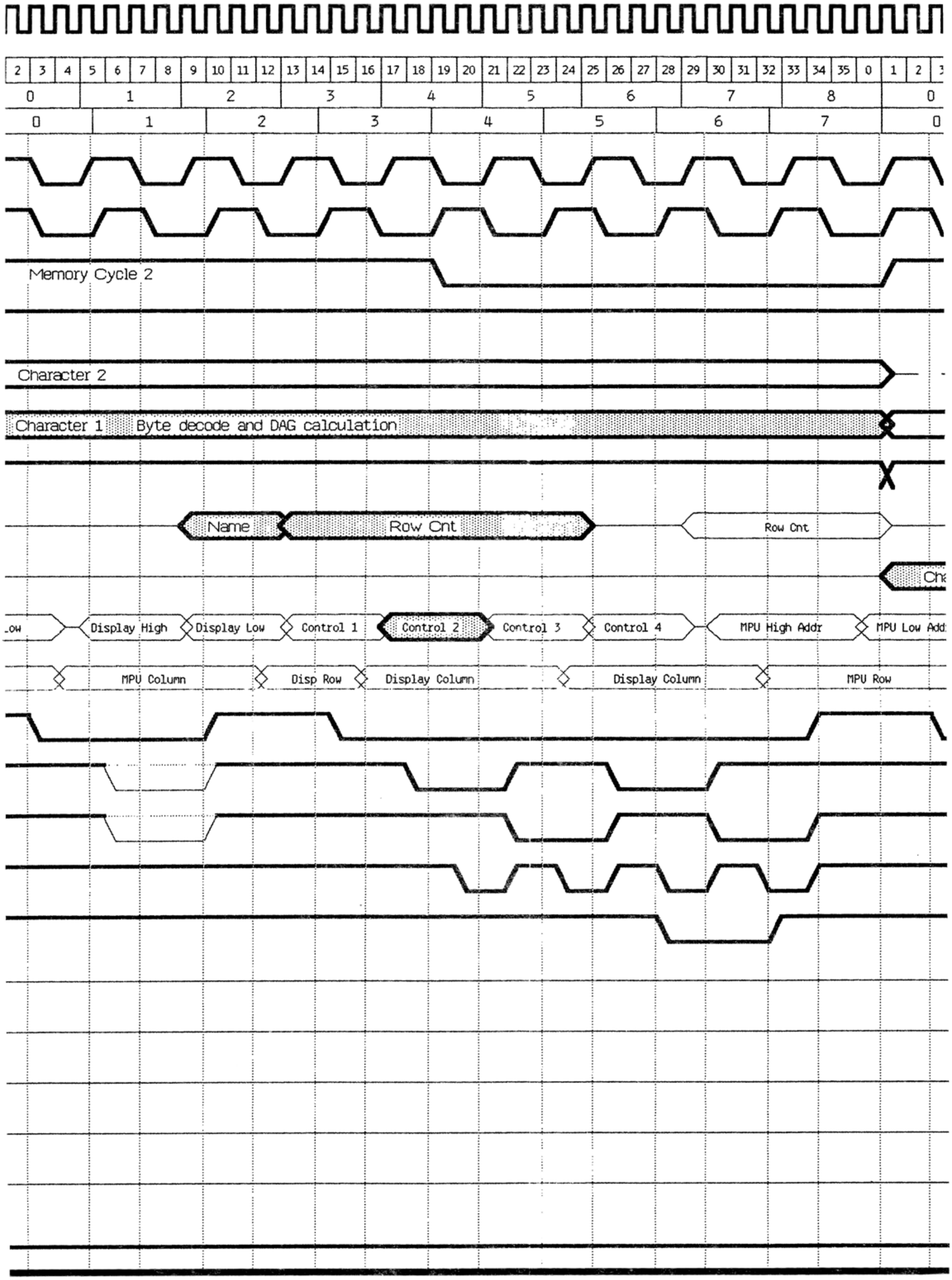
Video Pipe line Delay fo

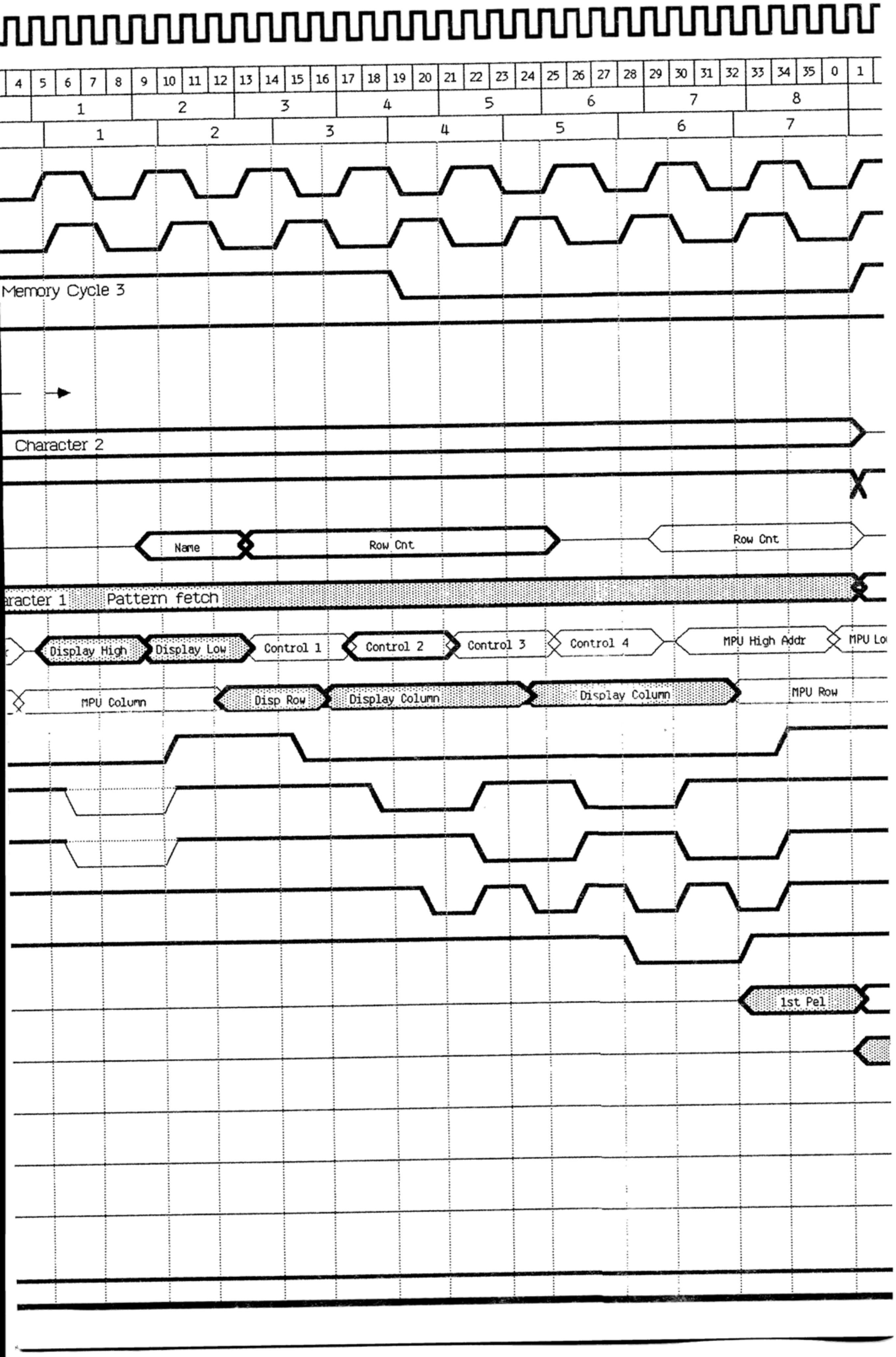


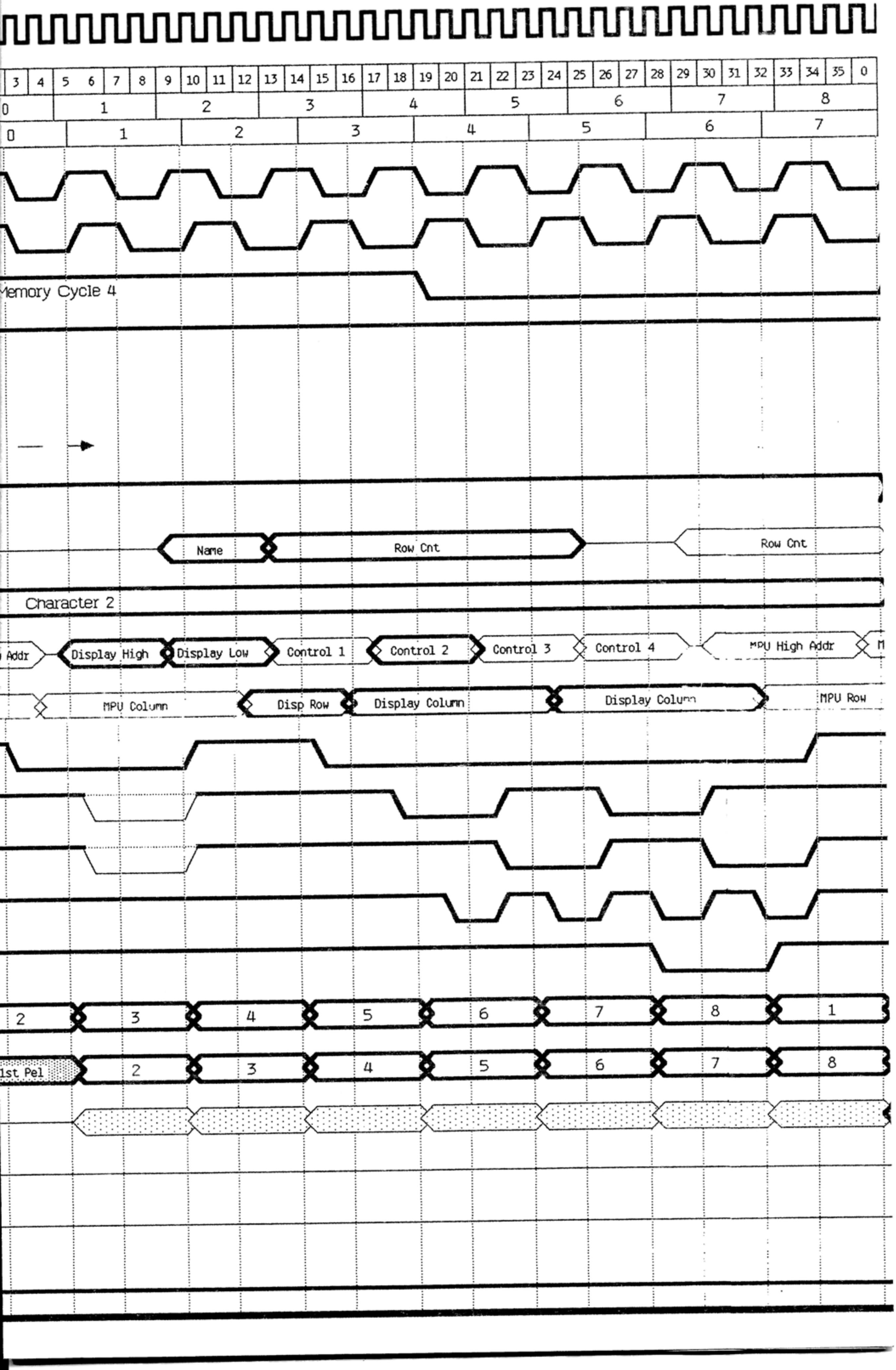
agram
DRC's

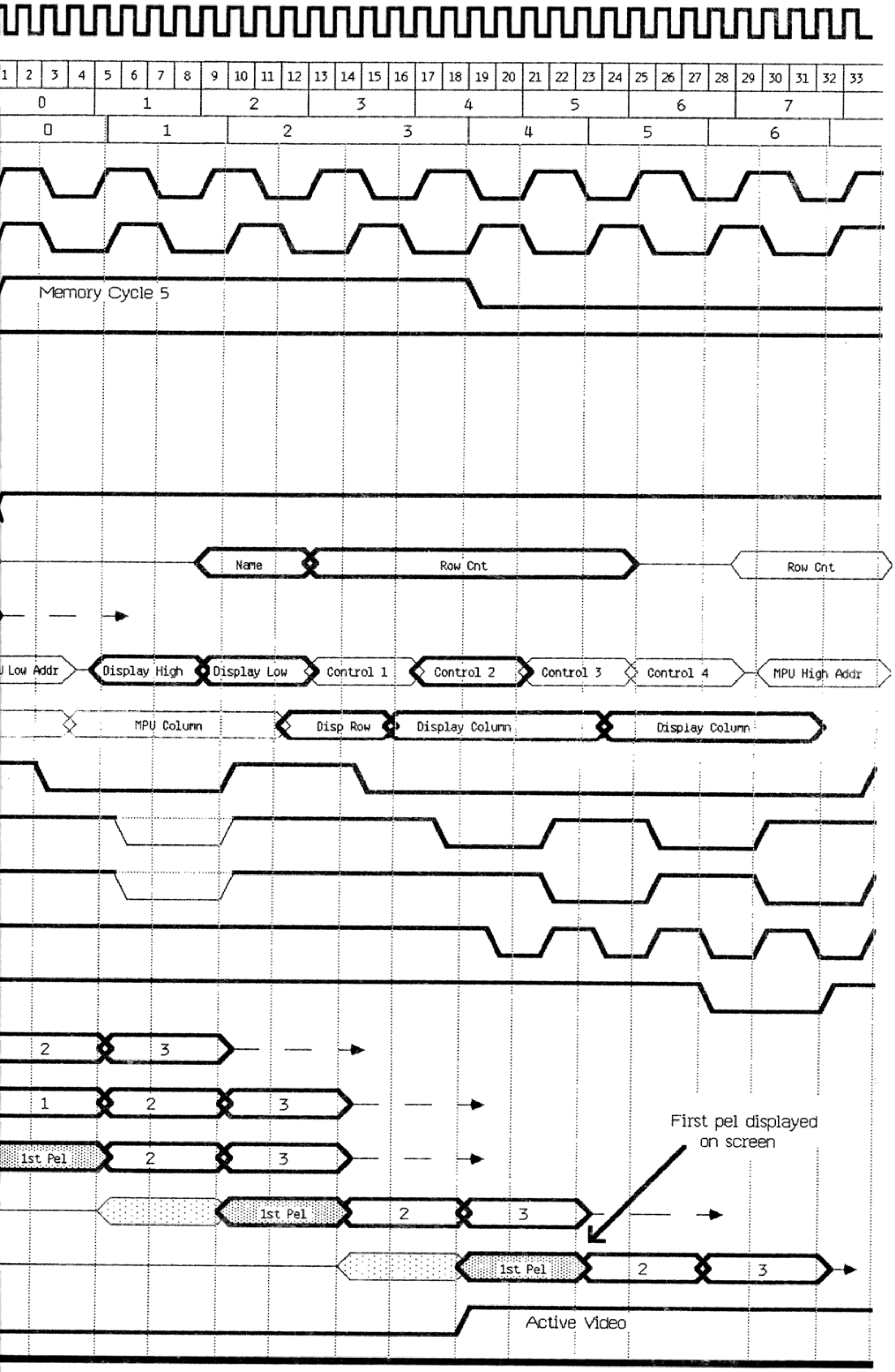
Horizantal Resolution - 320 (HRES 4)




2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 0 1 2 3
0 1 2 3 4 5 6 7 8 0
0 1 2 3 4 5 6 7 0
Memory Cycle 2
Character 2
Character 1
Byte decode and DAG calculation
Name
Row Cnt
Row Cnt
Cha
Low
Display High
Display Low
Control 1
Control 2
Control 3
Control 4
MPU High Addr
MPU Low Add
MPU Column
Disp Row
Display Column
Display Column
MPU Row

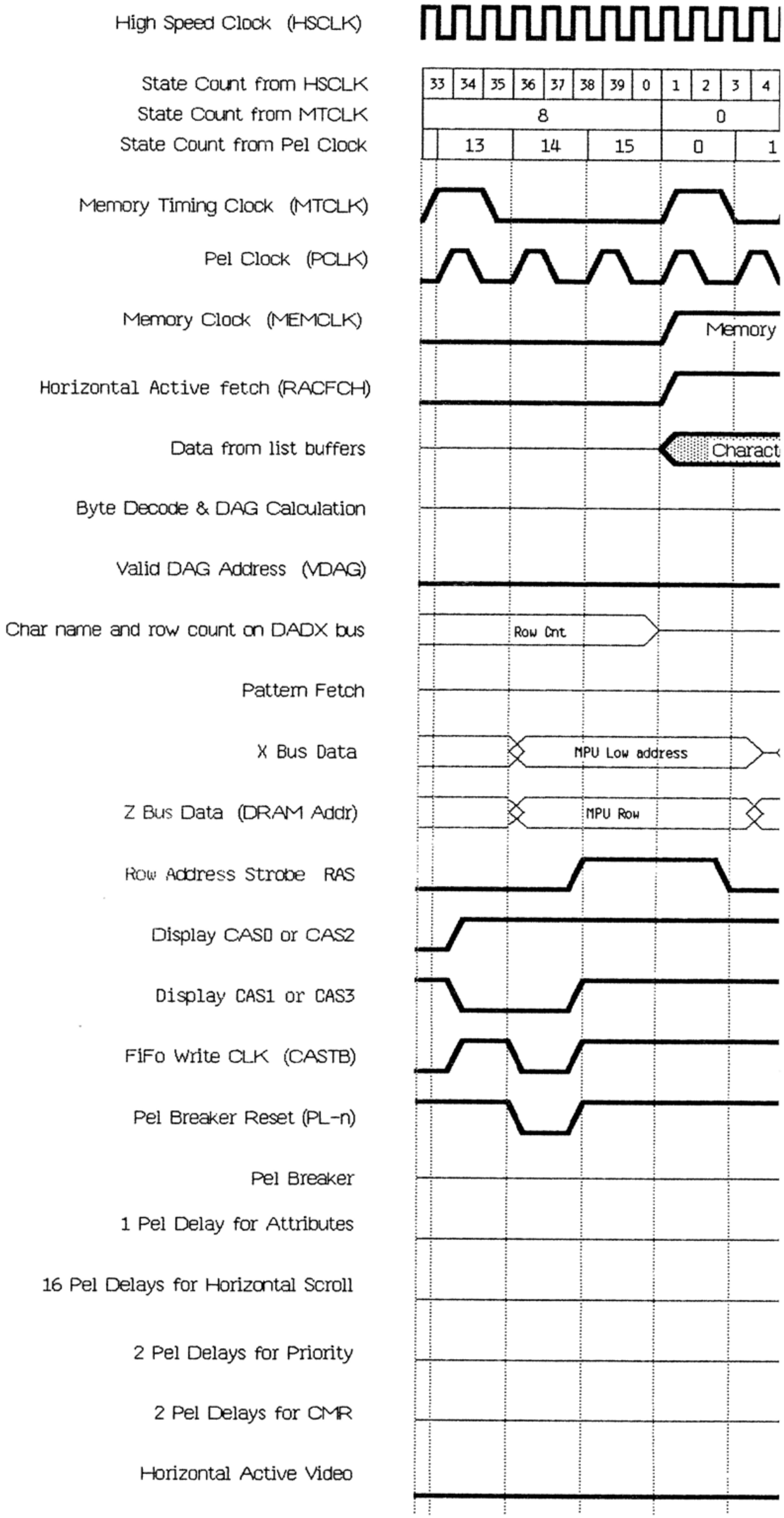

4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 0 1
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7
Memory Cycle 3
Character 2
Name
Row Cnt
Row Cnt
aracter 1
Pattern fetch
Display High
Display Low
Control 1
Control 2
Control 3
Control 4
MPU High Addr
MPU Lo
MPU Column
Disp Row
Display Column
Display Column
MPU Row
1st Pel


3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 0
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7
Memory Cycle 4
Name
Row Cnt
Row Cnt
Character 2
Addr
Display High
Display Low
Control 1
Control 2
Control 3
Control 4
MPU High Addr
M
MPU Column
Disp Row
Display Column
Display Column
MPU Row
2 3 4 5 6 7 8 1
1st Pel
2 3 4 5 6 7 8



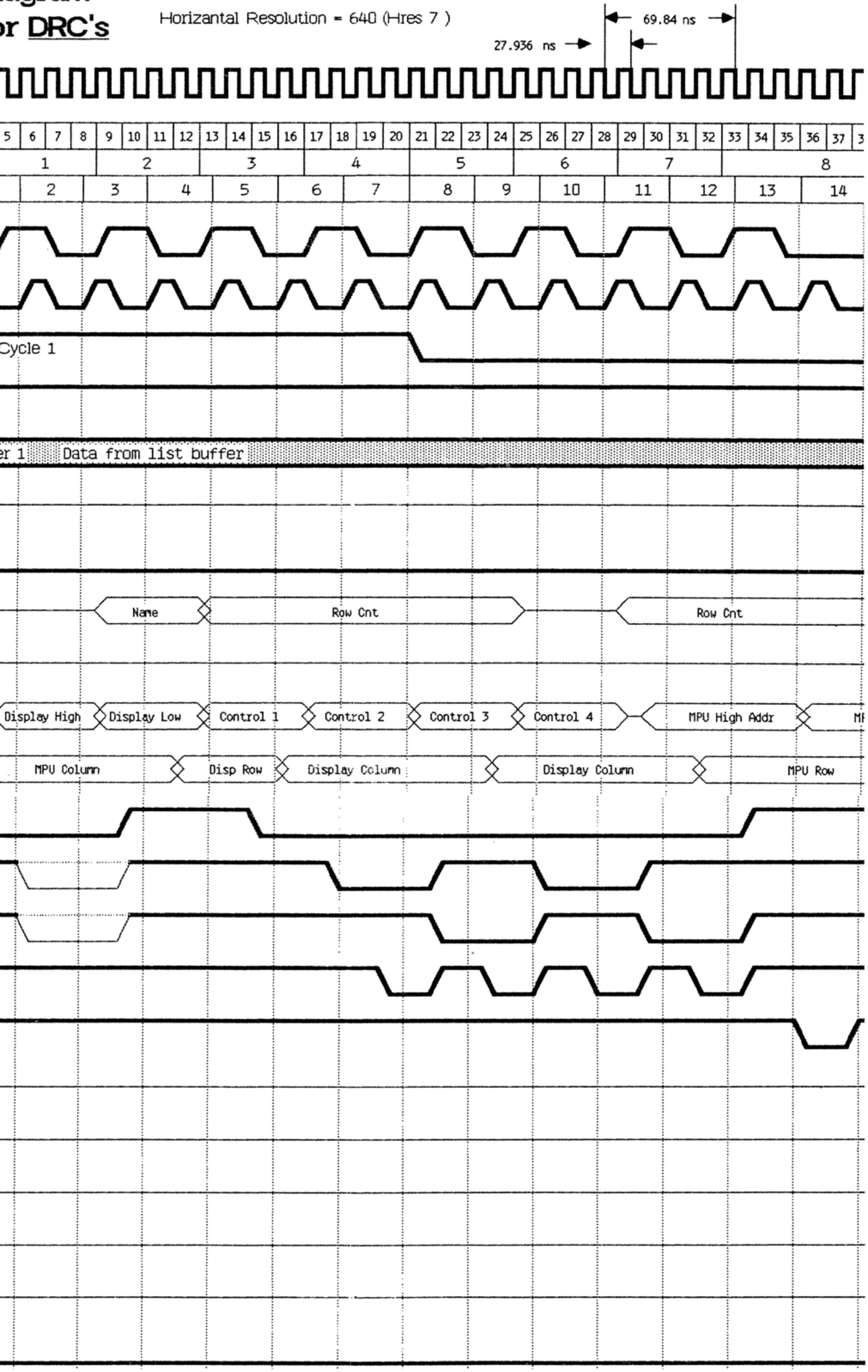
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6
Memory Cycle 5
Name
Row Cnt
Row Cnt
Low Addr
Display High
Display Low
Control 1
Control 2
Control 3
Control 4
MPU High Addr
MPU Column
Disp Row
Display Column
Display Column
2
3
1
2
3
1st Pel
2
3
1st Pel
2
3
First pel displayed on screen
1st Pel
2
3
Active Video

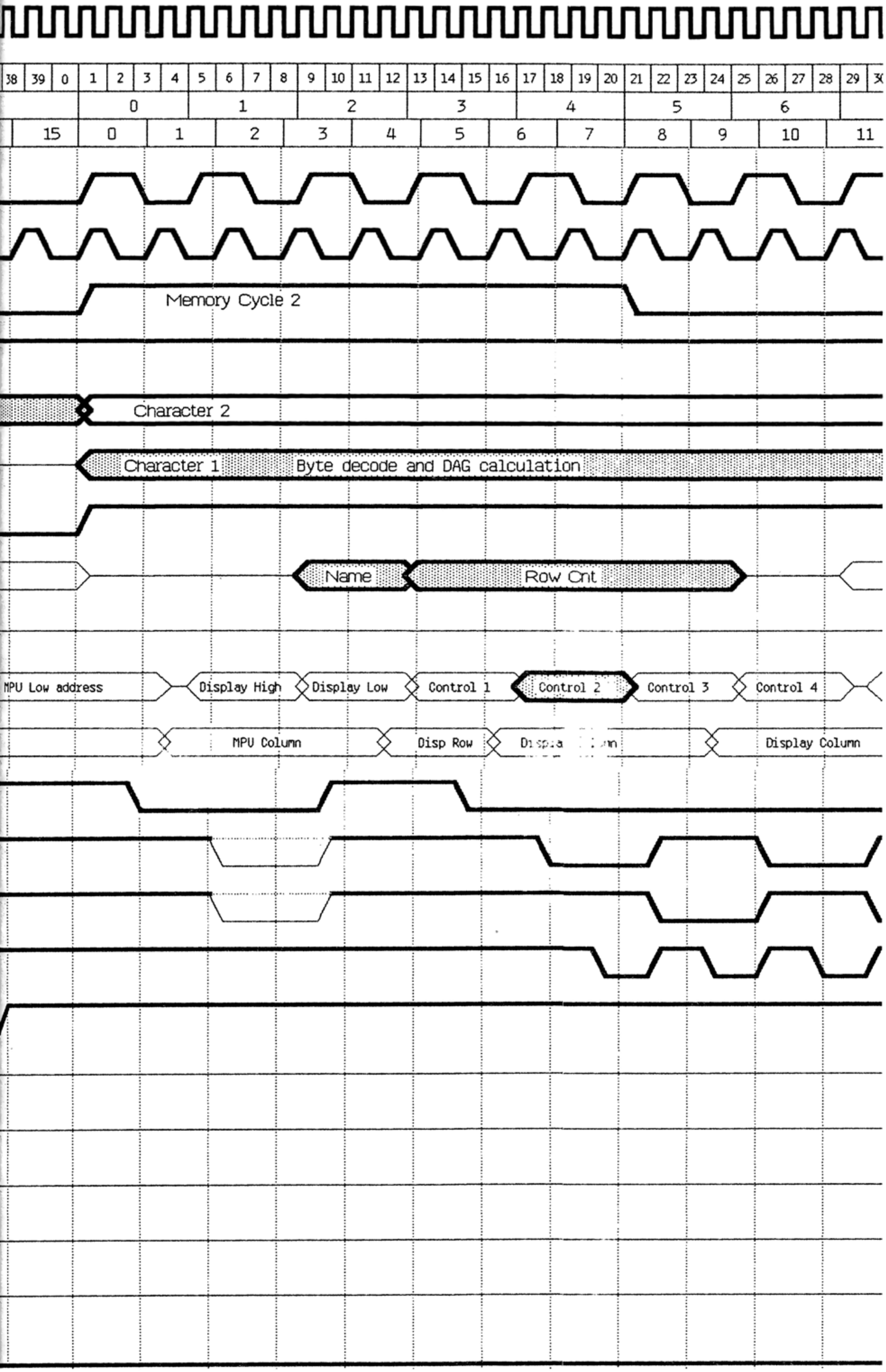
Raster Memory System Timing D
Video Pipe line Delay fo

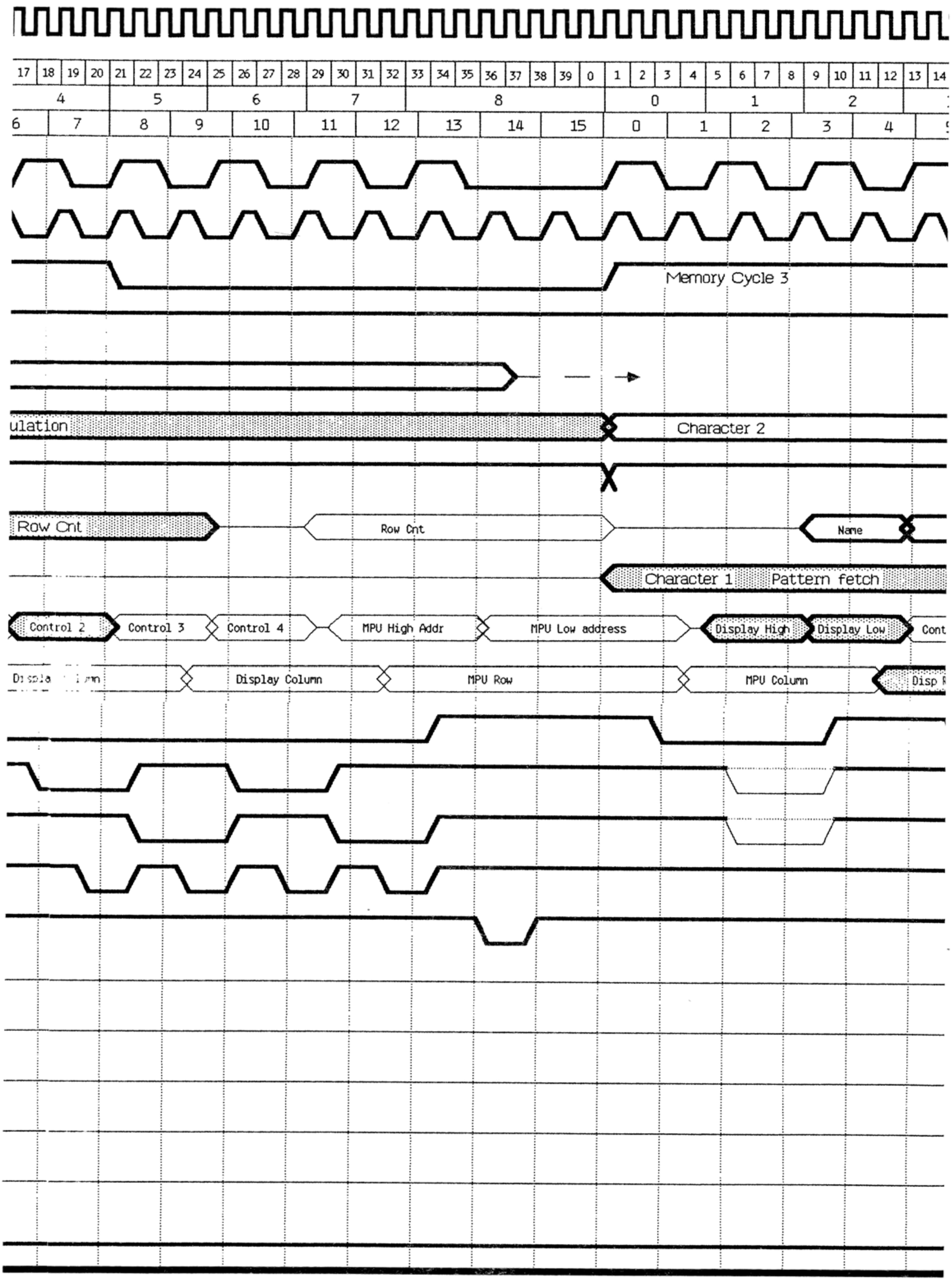


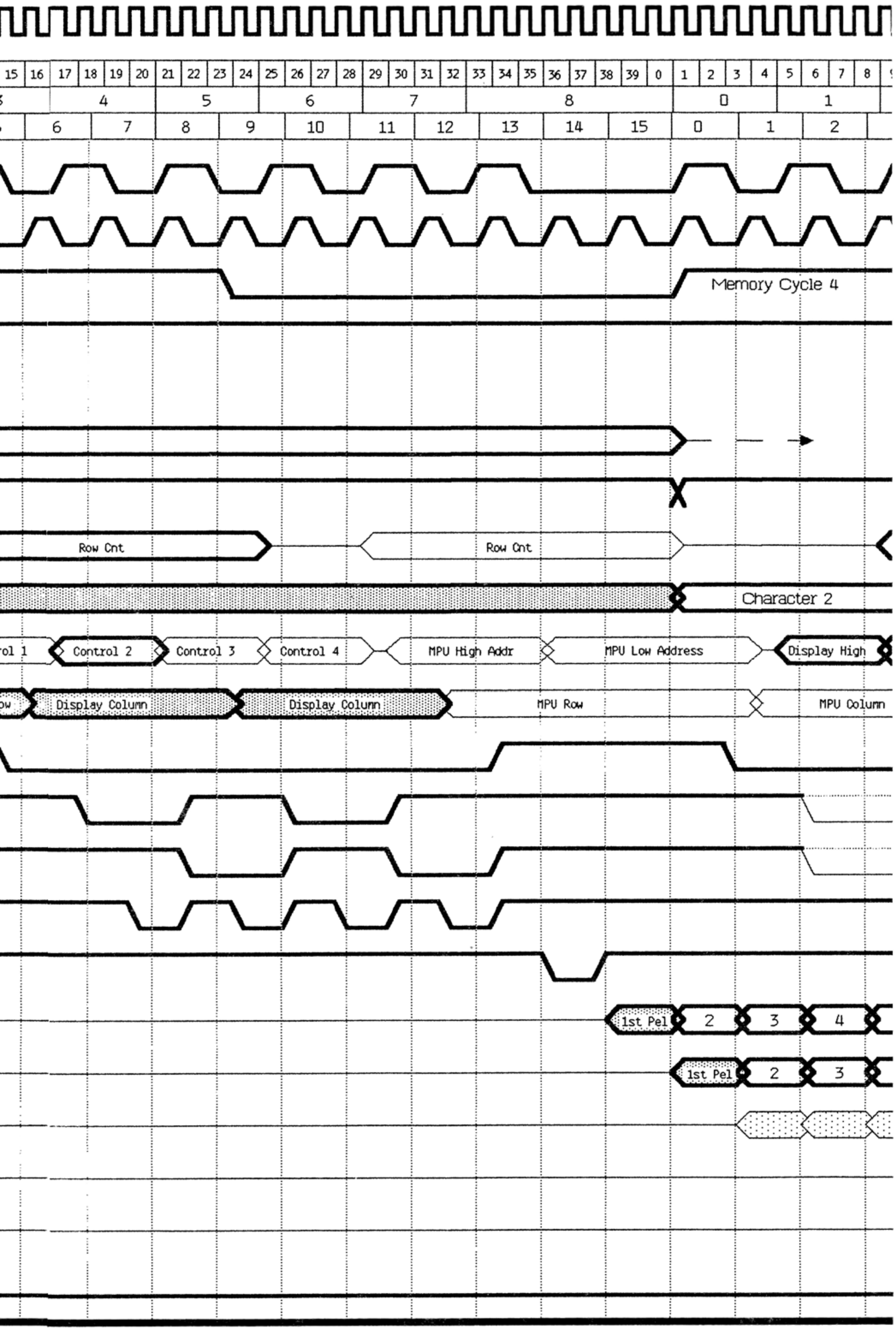
iagram
or DRC's

Horizantal Resolution = 640 (Hres 7)

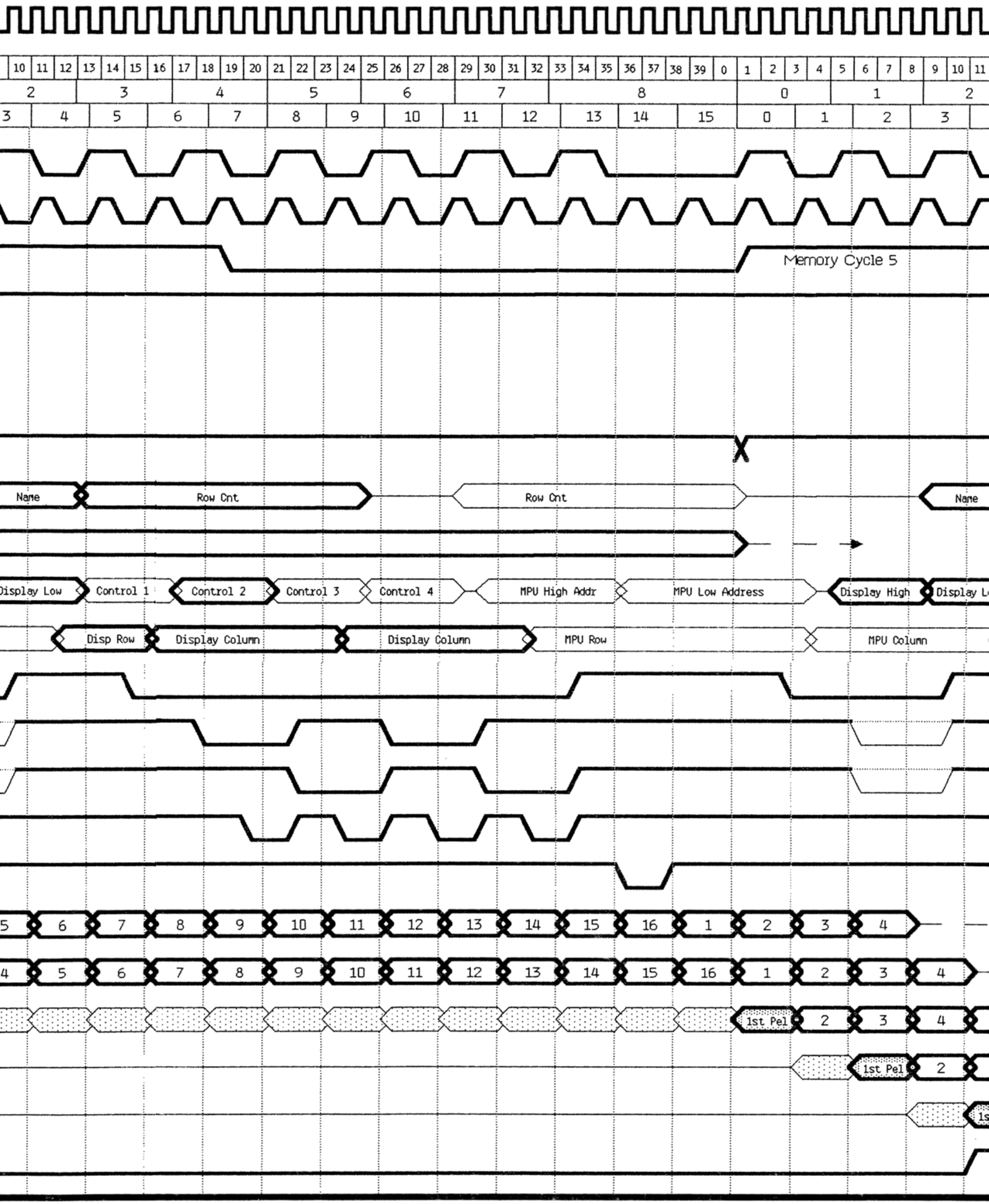


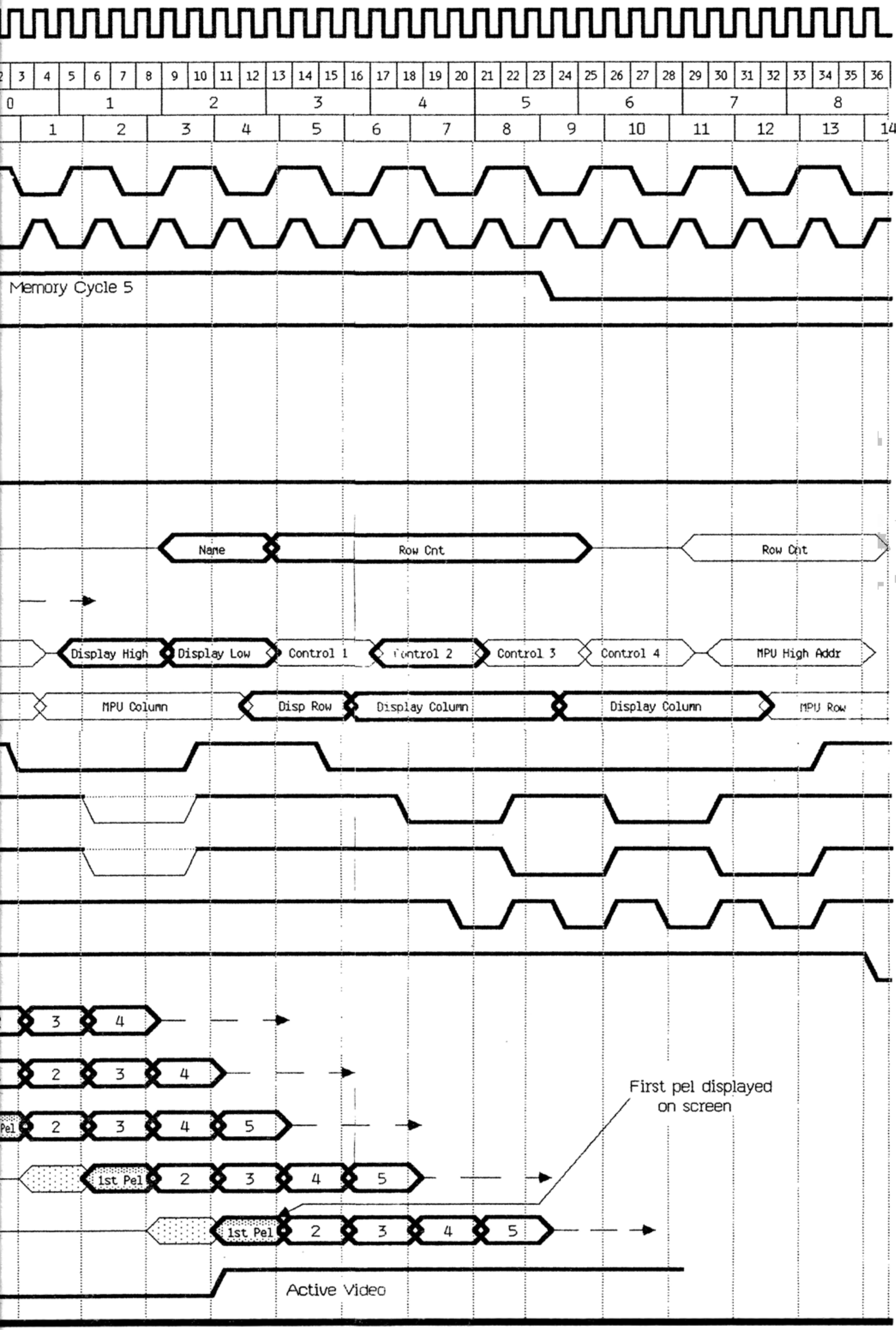

38 39 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
0 1 2 3 4 5 6
15 0 1 2 3 4 5 6 7 8 9 10 11
Memory Cycle 2
Character 2
Character 1
Byte decode and DAG calculation
Name
Row Cnt
MPU Low address
Display High
Display Low
Control 1
Control 2
Control 3
Control 4
MPU Column
Disp Row
Display Column
Display Column


17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
4 5 6 7 8 0 1 2
6 7 8 9 10 11 12 13 14 15 0 1 2 3 4
Memory Cycle 3
ulation
Character 2
Row Cnt
Row Cnt
Name
Character 1 Pattern fetch
Control 2
Control 3
Control 4
MPU High Addr
MPU Low address
Display High
Display Low
Cont
Displa lumn
Display Column
MPU Row
MPU Column
Disp



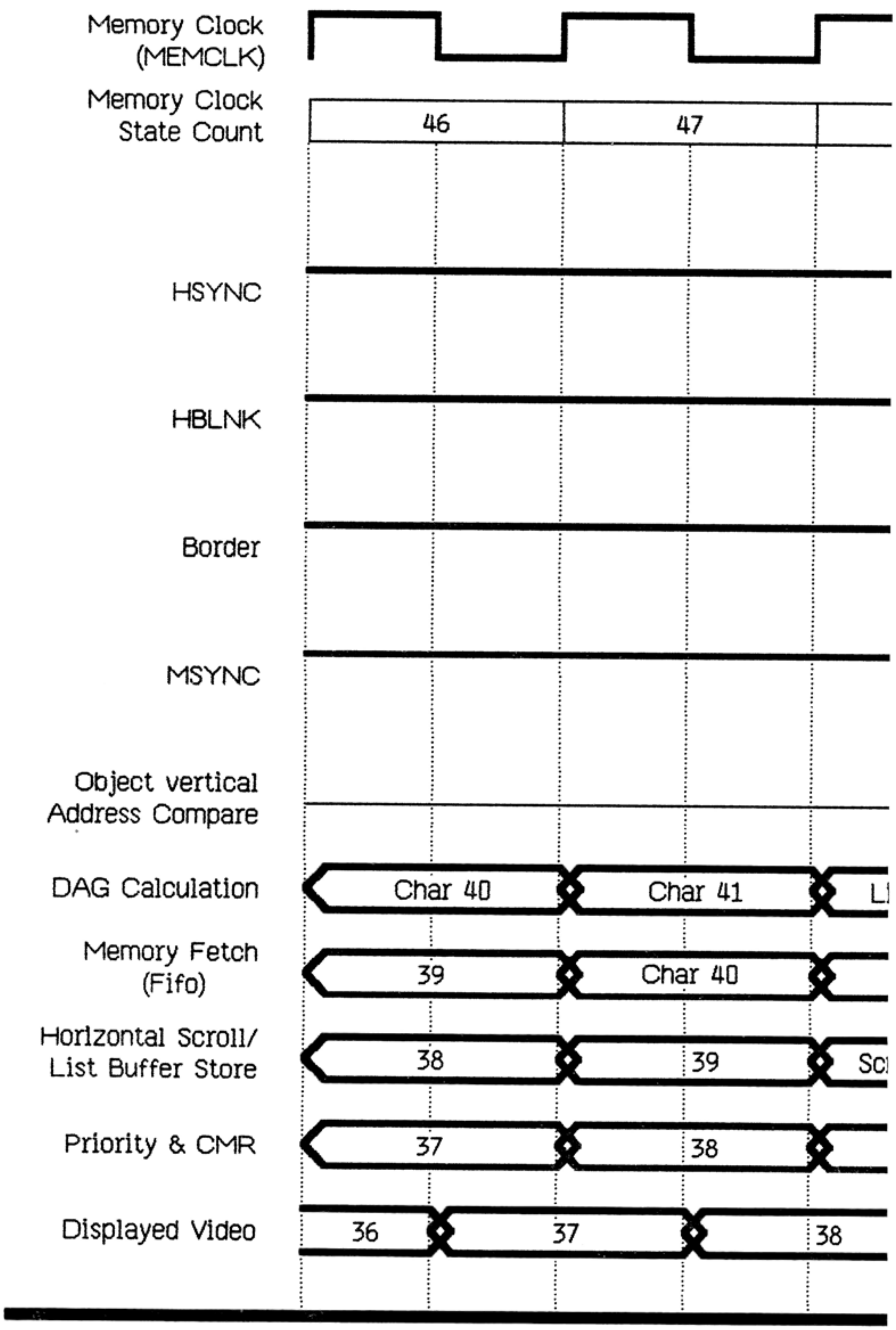
15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 0 1 2 3 4 5 6 7 8
4 5 6 7 8 0 1
6 7 8 9 10 11 12 13 14 15 0 1 2
Memory Cycle 4
Row Cnt
Row Cnt
Character 2
rol 1
Control 2
Control 3
Control 4
MPU High Addr
MPU Low Address
Display High
ow
Display Column
Display Column
MPU Row
MPU Column
1st Pel
2
3
4
1st Pel
2
3


10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 0 1 2 3 4 5 6 7 8 9 10 11
2 3 4 5 6 7 8 0 1 2
3 4 5 6 7 8 9 10 11 12 13 14 15 0 1 2 3
Memory Cycle 5
Name
Row Cnt
Row Cnt
Name
Display Low
Control 1
Control 2
Control 3
Control 4
MPU High Addr
MPU Low Address
Display High
Display L
Disp Row
Display Column
Display Column
MPU Row
MPU Column
5 6 7 8 9 10 11 12 13 14 15 16 1 2 3 4
4 5 6 7 8 9 10 11 12 13 14 15 16 1 2 3 4
1st Pel 2 3 4
1st Pel 2
1s


2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
0 1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8 9 10 11 12 13 14
Memory Cycle 5
Name
Row Cnt
Row Cnt
Display High
Display Low
Control 1
Control 2
Control 3
Control 4
MPU High Addr
MPU Column
Disp Row
Display Column
Display Column
MPU Row
3
4
2
3
4
Pel
2
3
4
5
1st Pel
2
3
4
5
1st Pel
2
3
4
5
First pel displayed
on screen
Active Video

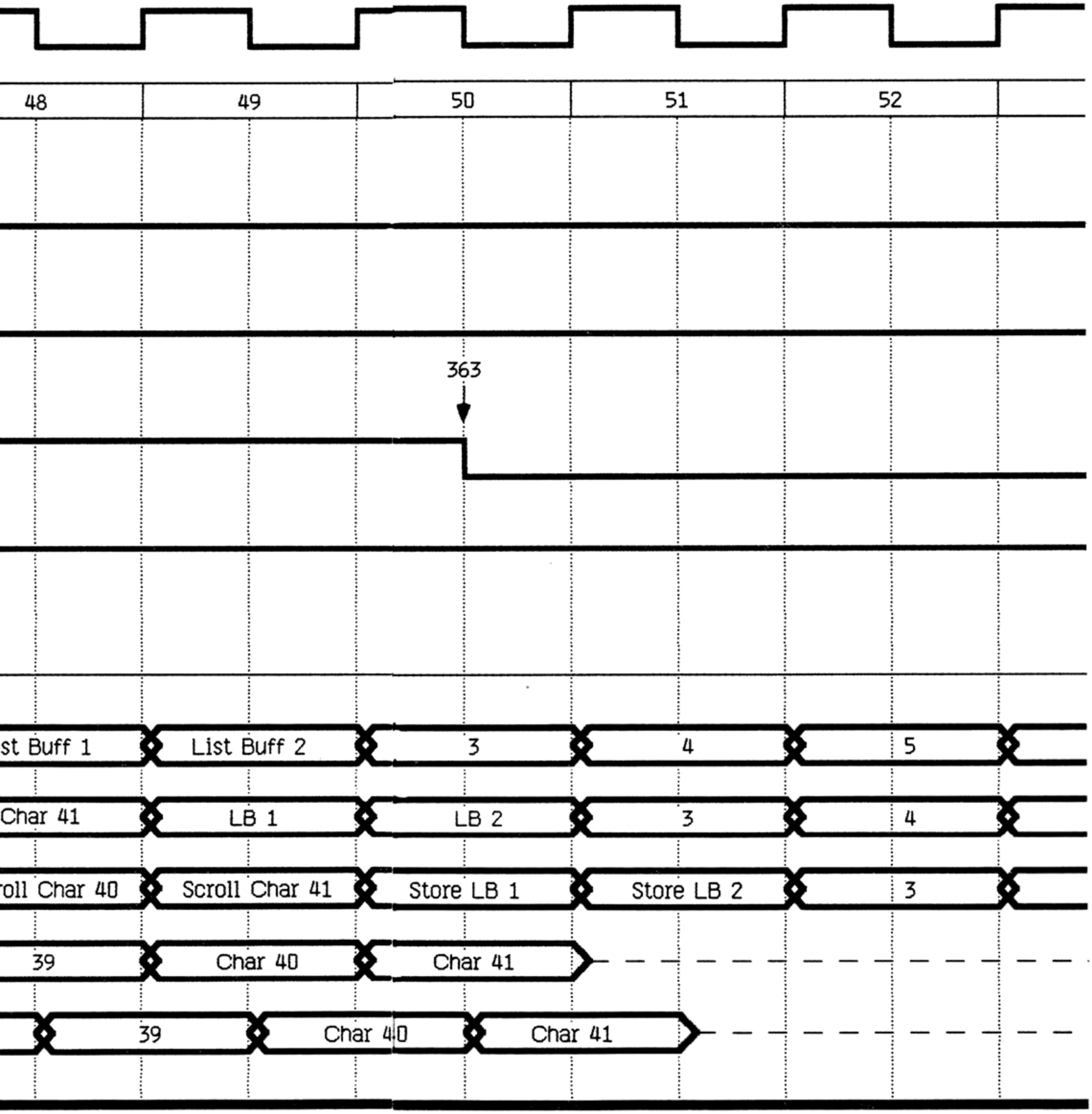
Raster Memory System Timing Dia

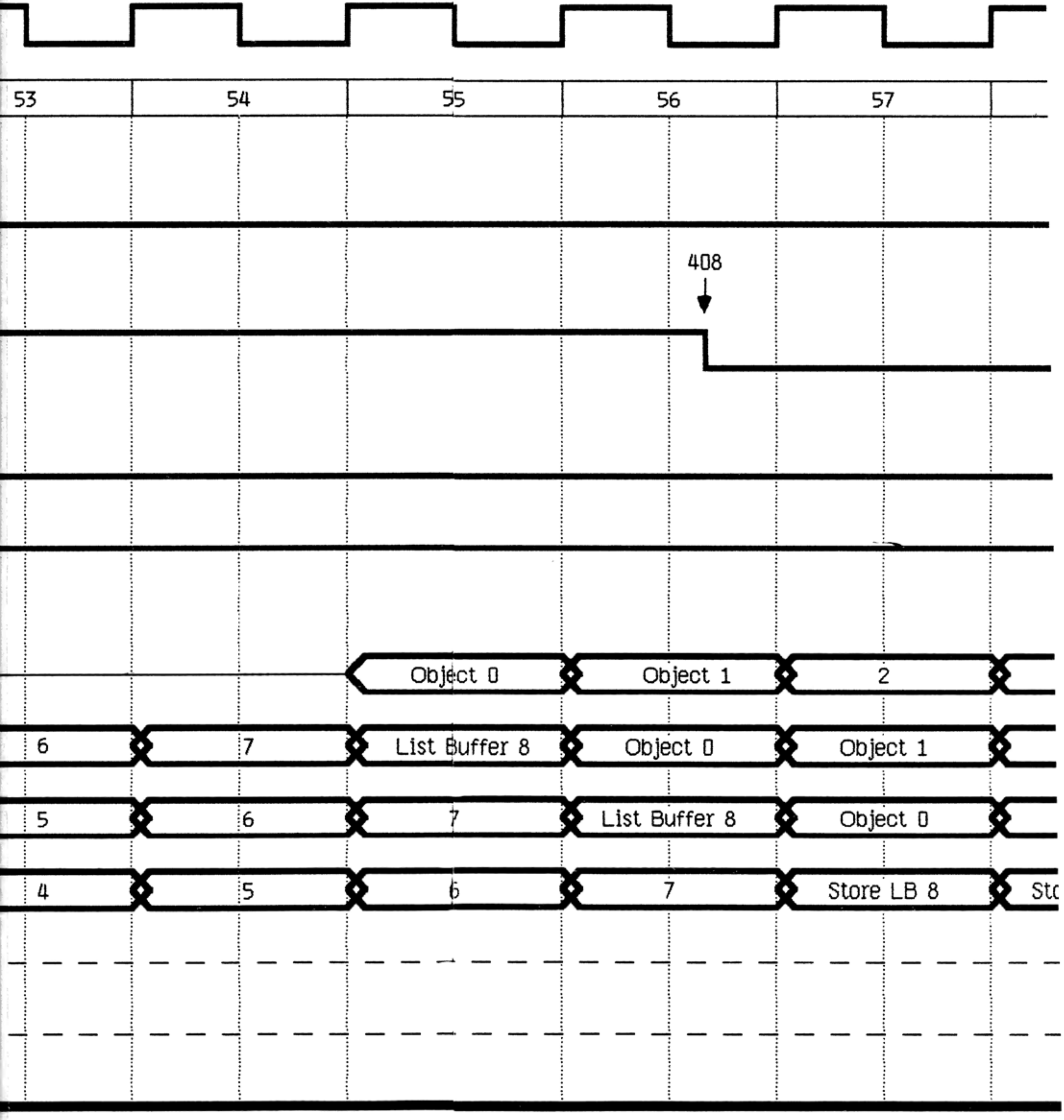
Horizontal Timing Events

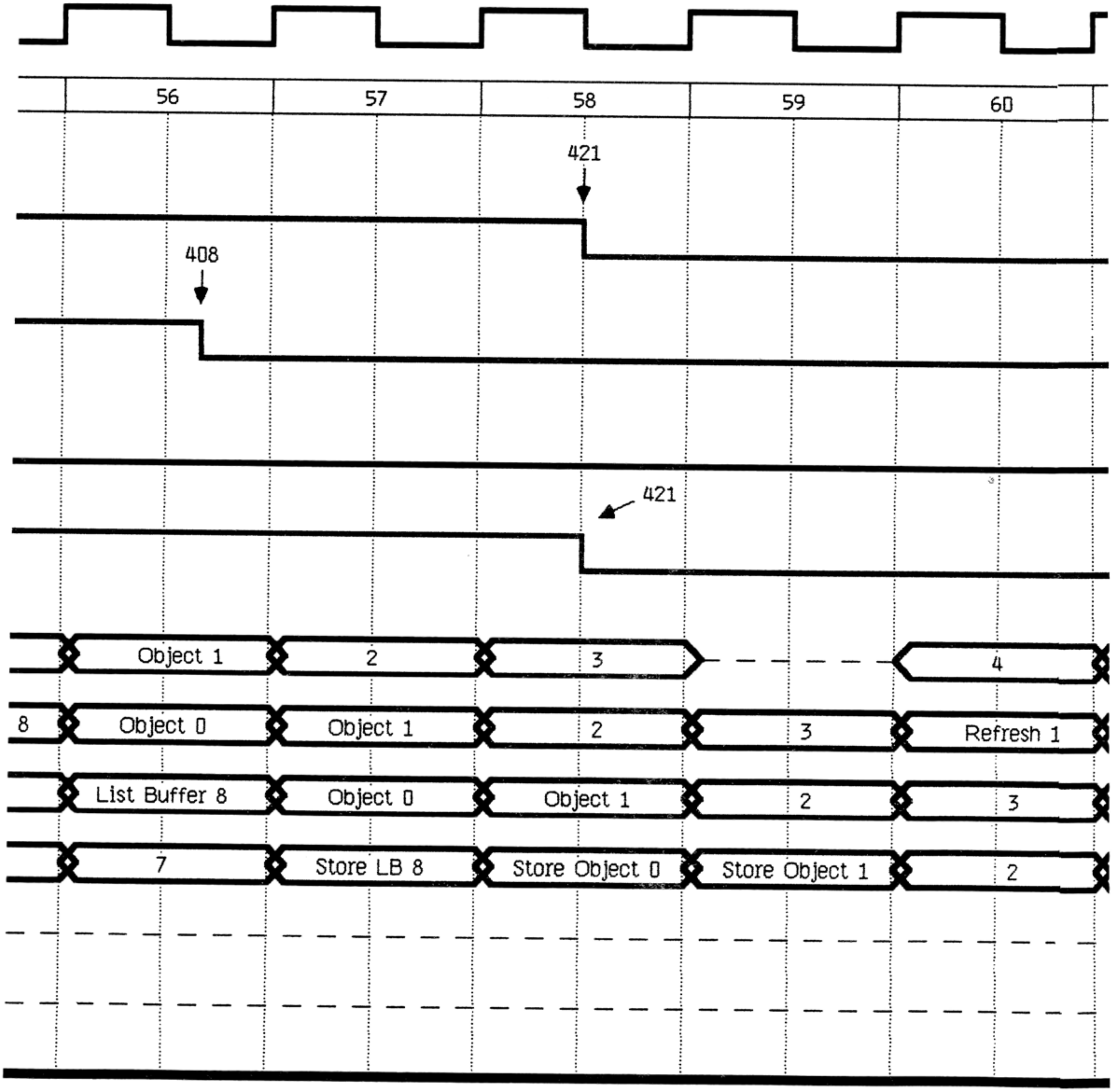


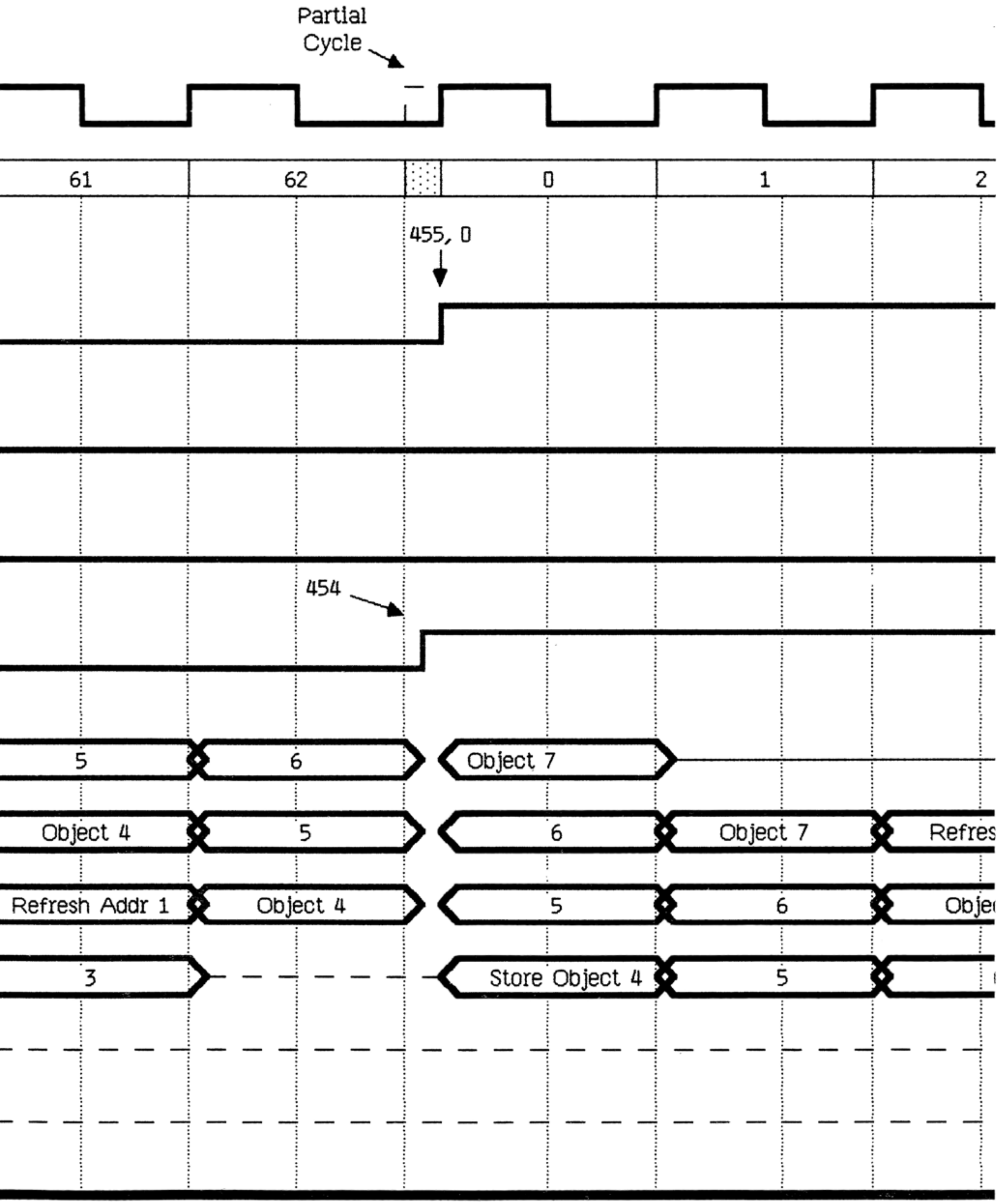
gram

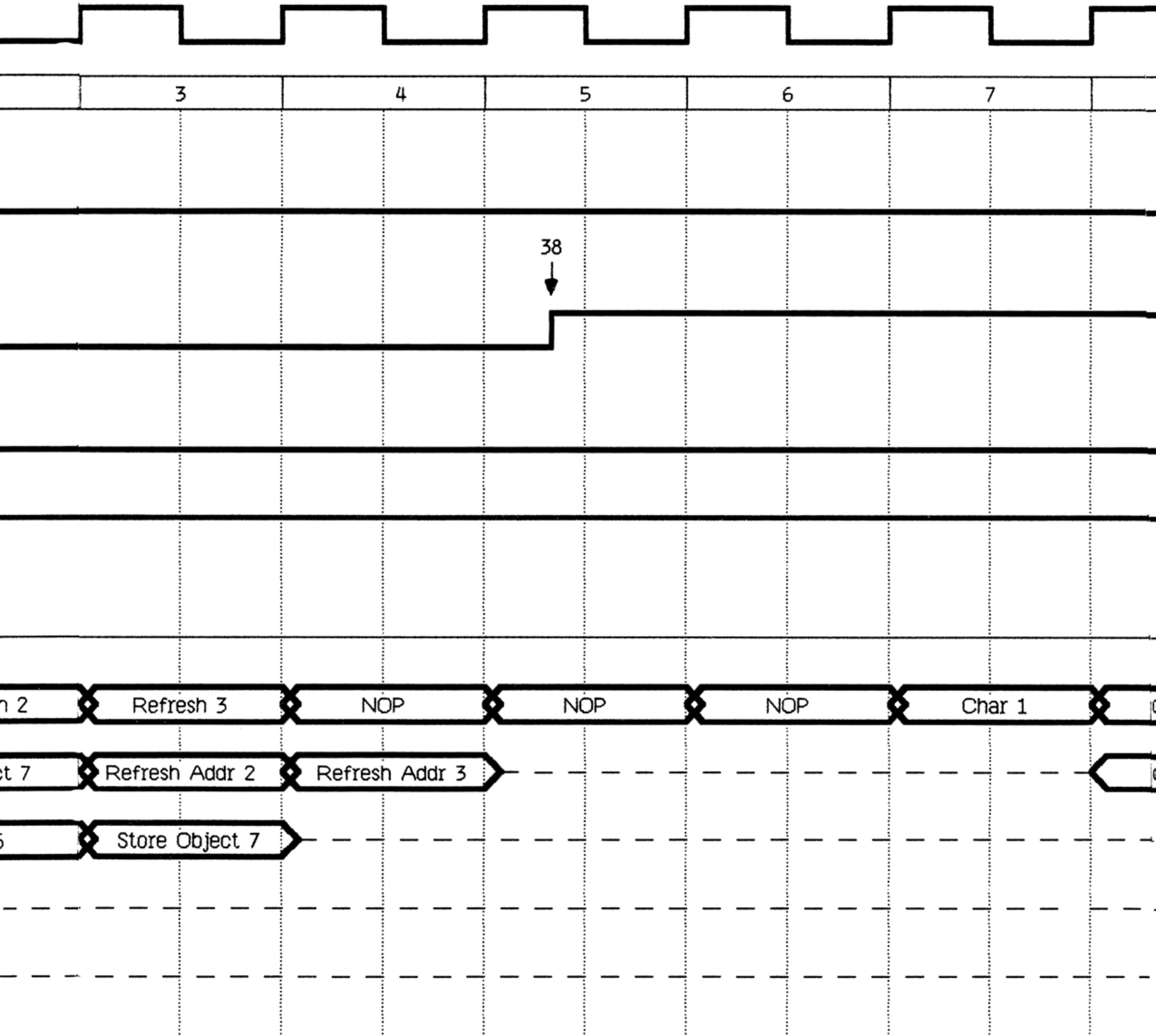
Horizontal Resolution = 320 (HRES 4)

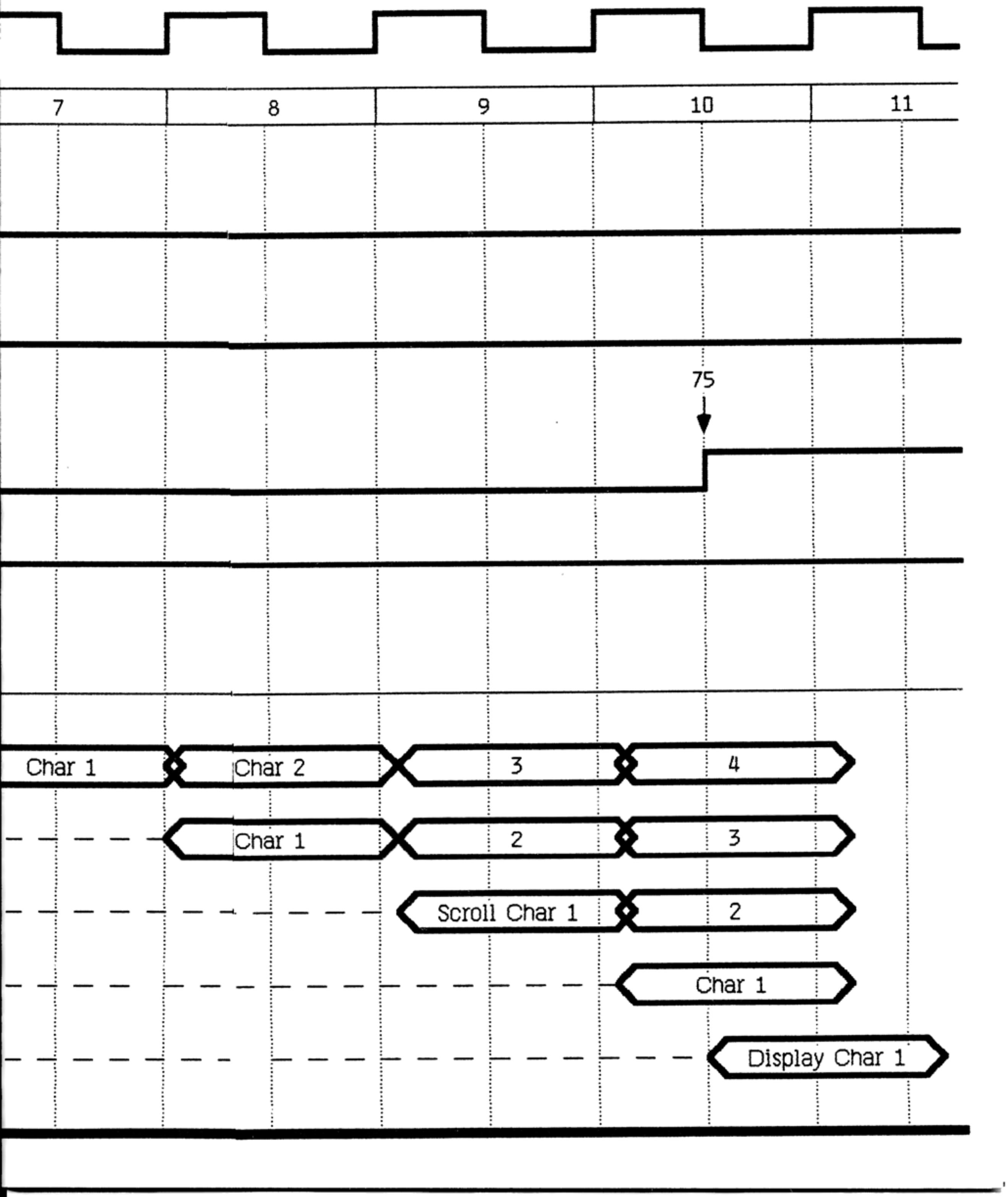



53
54
55
56
57
408
Object 0
Object 1
2
6
7
List Buffer 8
Object 0
Object 1
5
6
7
List Buffer 8
Object 0
4
5
6
7
Store LB 8
Sto


56
57
58
59
60
421
408
421
Object 1
2
3
4
8
Object 0
Object 1
2
3
Refresh 1
List Buffer 8
Object 0
Object 1
2
3
7
Store LB 8
Store Object 0
Store Object 1
2


Partial
Cycle
61
62
0
1
2
455, 0
454
5
6
Object 7
Object 4
5
6
Object 7
Refres
Refresh Addr 1
Object 4
5
6
Obje
3
Store Object 4
5

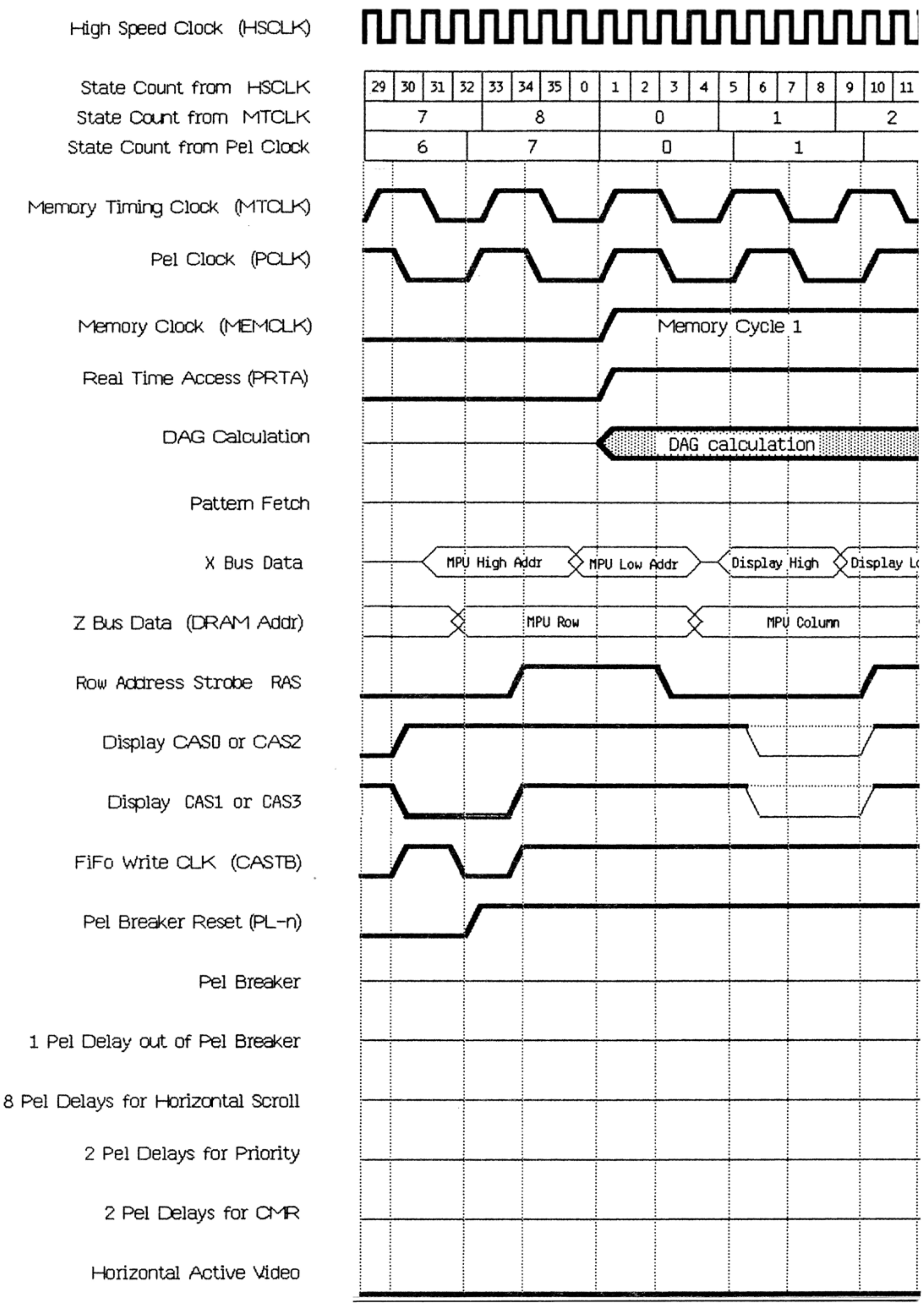

3
4
5
6
7
38
n 2
Refresh 3
NOP
NOP
NOP
Char 1
ct 7
Refresh Addr 2
Refresh Addr 3
5
Store Object 7


7
8
9
10
11
75
Char 1
Char 2
3
4
Char 1
2
3
Scroll Char 1
2
Char 1
Display Char 1

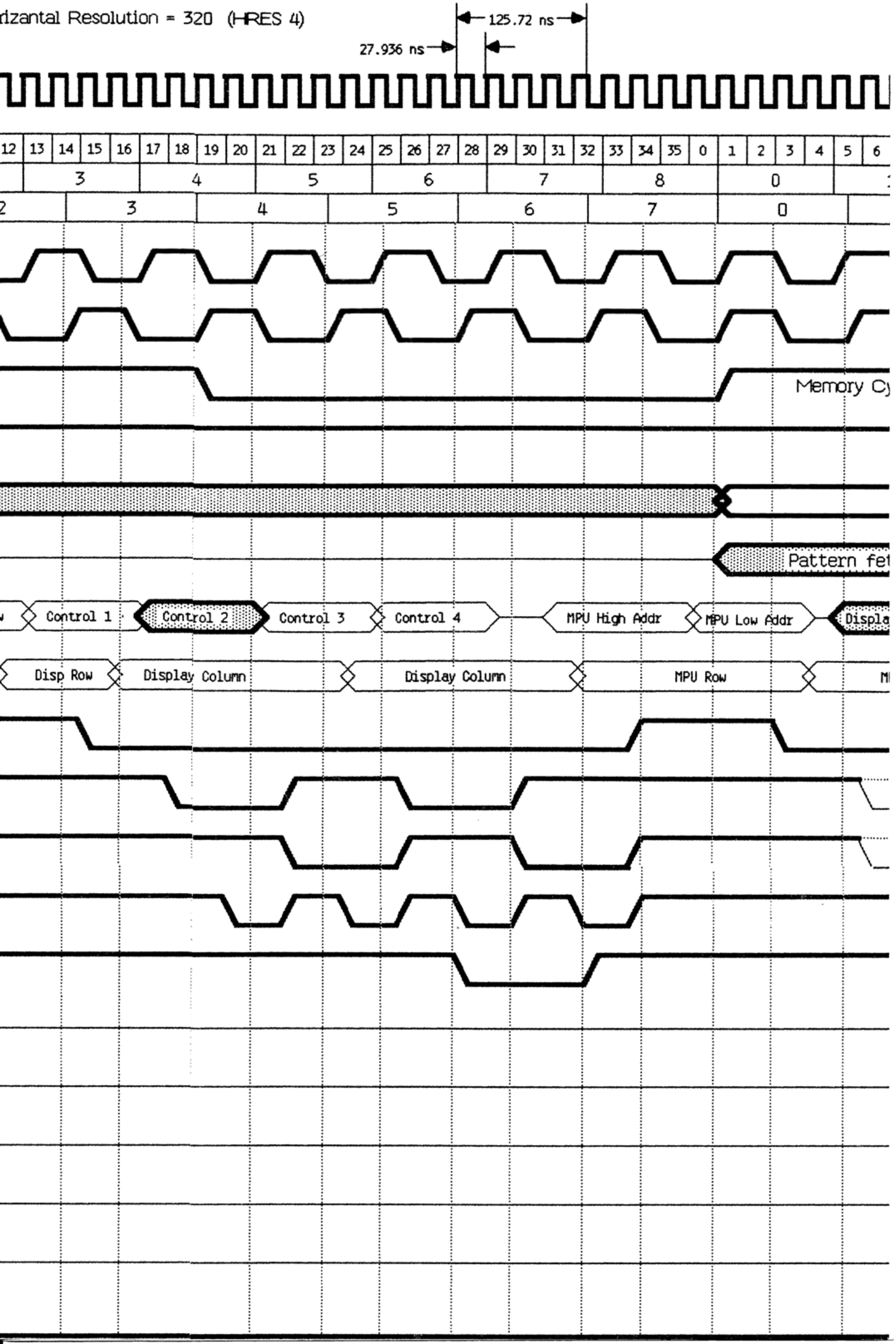
Raster Memory System Timing Diagram

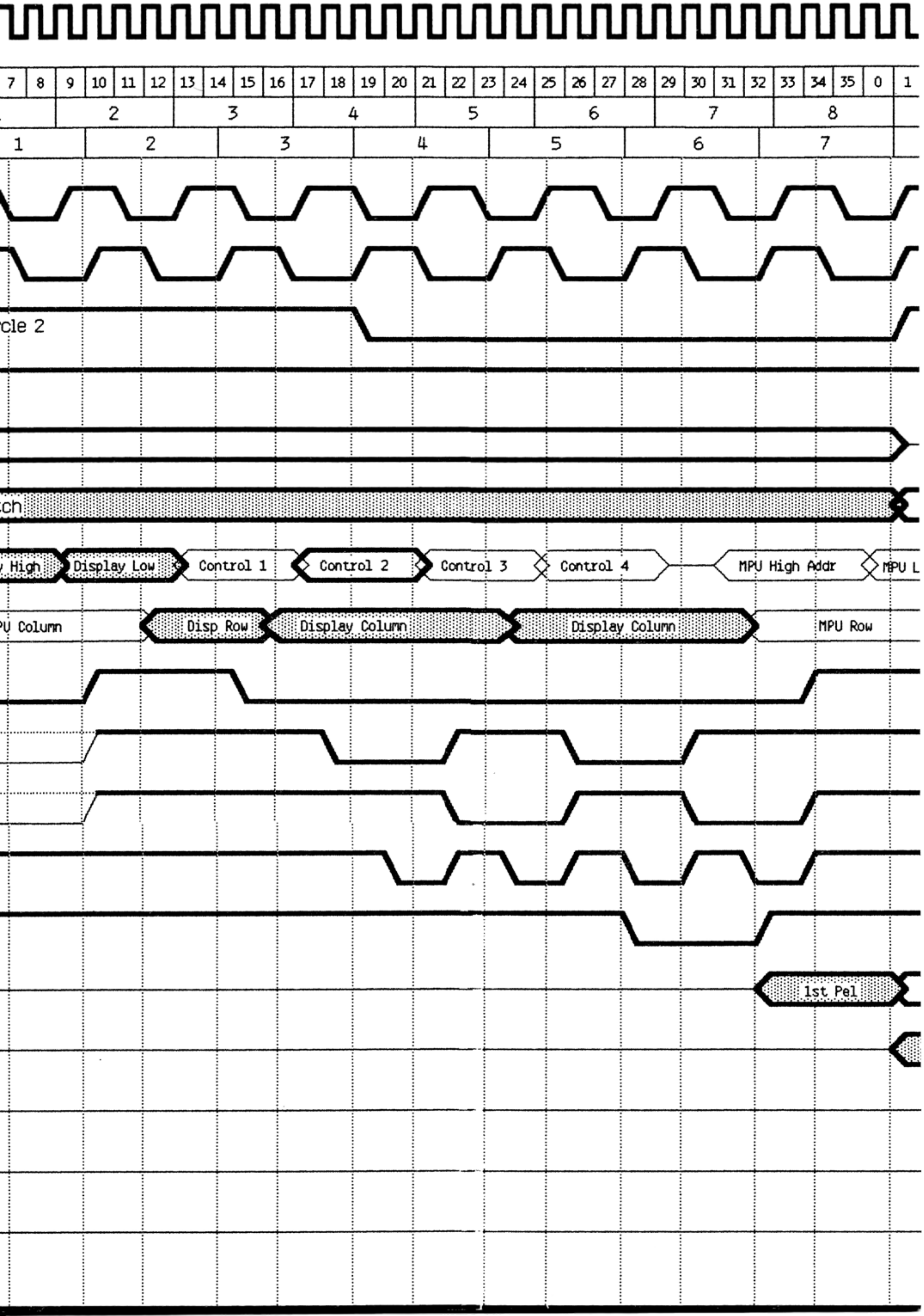
Video Pipe line Delay Bit Plane

Hc

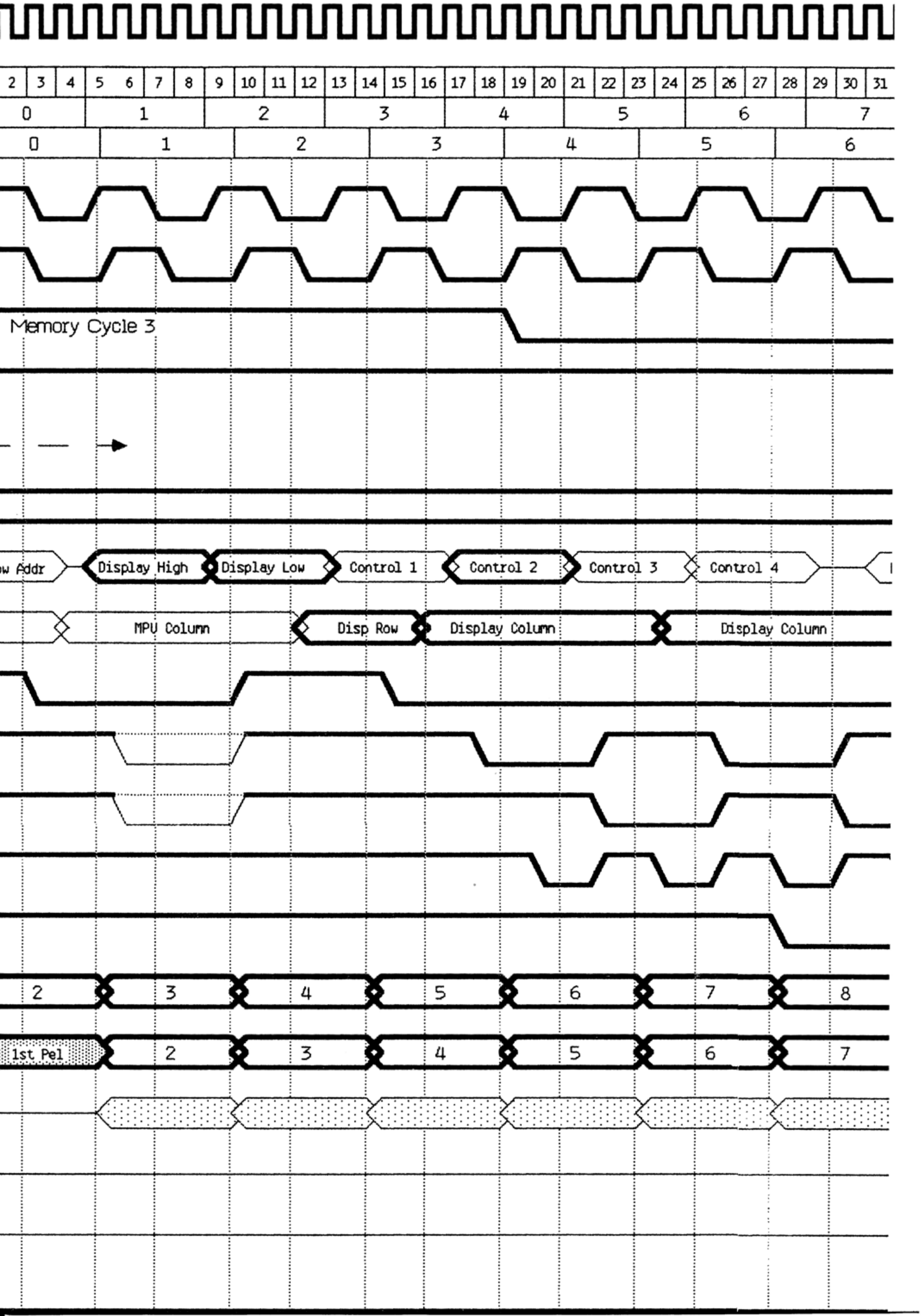


rizantal Resolution = 320 (HRES 4)

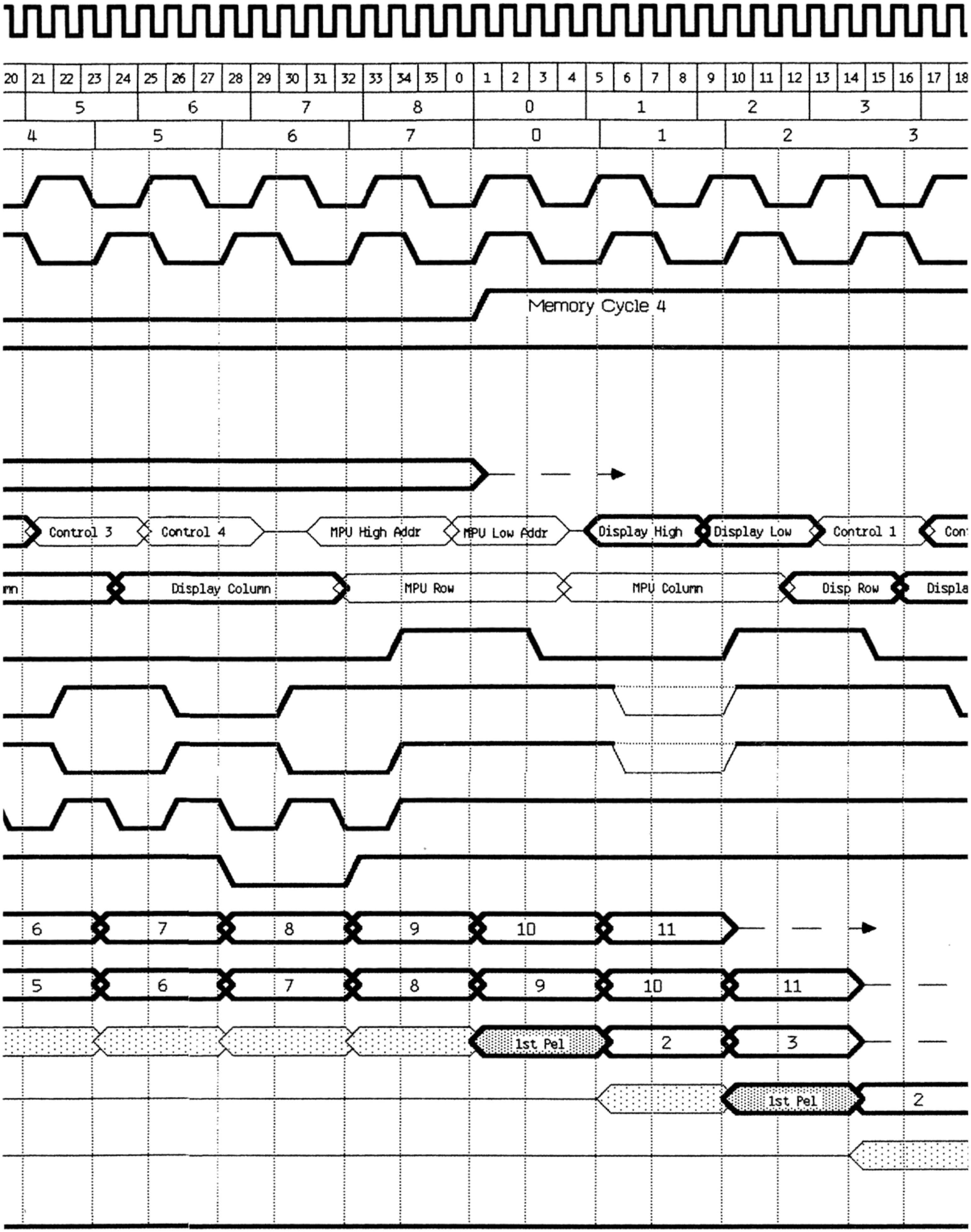


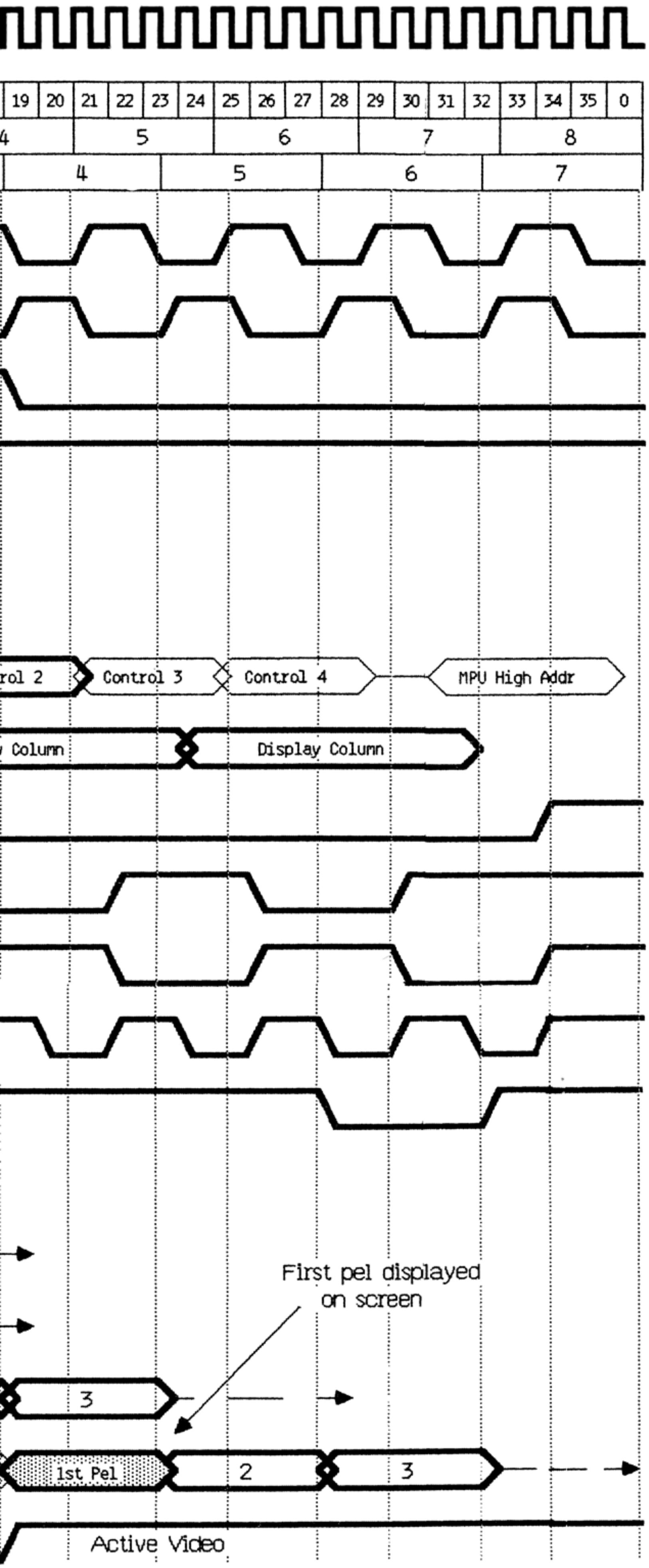


7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
0
1
2
3
4
5
6
7
8
1
2
3
4
5
6
7
cle 2
ch
High
Display Low
Control 1
Control 2
Control 3
Control 4
MPU High Addr
MPU L
PU Column
Disp Row
Display Column
Display Column
MPU Row
1st Pel



2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6
Memory Cycle 3
w Addr
Display High
Display Low
Control 1
Control 2
Control 3
Control 4
MPU Column
Disp Row
Display Column
Display Column
2 3 4 5 6 7 8
1st Pel
2 3 4 5 6 7


20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
5 6 7 8 0 1 2 3
4 5 6 7 0 1 2 3
Memory Cycle 4
Control 3
Control 4
MPU High Addr
MPU Low Addr
Display High
Display Low
Control 1
Con
mn
Display Column
MPU Row
MPU Column
Disp Row
Displa
6 7 8 9 10 11
5 6 7 8 9 10 11
1st Pel
2
3
1st Pel
2


19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
0
4
5
6
7
8
4
5
6
7
rol 2
Control 3
Control 4
MPU High Addr
Column
Display Column
First pel displayed on screen
3
1st Pel
2
3
Active Video