SanDisk MultiMediaCard & Motorola 8 bit Microcontroller Interface Design Reference Example

SanDisk Corporation
140 Caspian Court
Sunnyvale, CA 94089
TEL: 408-542-0500 FAX: 408-542-0503
URL: http://www.sandisk.com

SanDisk® Corporation general policy does not recommend the use of its products in life support applications where in a failure or malfunction of the product may directly threaten life or injury. Per SanDisk Terms and Conditions of Sale, the user of SanDisk products in life support applications assumes all risk of such use and indemnifies SanDisk against all damages.

The information in this document is subject to change without notice.

SanDisk Corporation shall not be liable for technical or editorial errors or omissions contained herein; nor for incidental or consequential damages resulting from the furnishing, performance, or use of this material.

SanDisk products are covered or licensed under one or more of the following U.S. Patent Nos. 5,070,032; 5,095,344; 5,168,465; 5,172,338; 5,198,380; 5,200,959; 5,268,318; 5,268,870; 5,272,669; 5,418,752; 5,602,987. Other U.S. and foreign patents awarded and pending.

Lit. No. 80-13-00110 Rev. 2 3/99 *Printed in U.S.A.*

Table of Contents

1.0 Considerations for Interfacing to SanDisk Flash

The major consideration for interfacing the SanDisk MultiMediaCard is centered around the MultiMediaCard's Serial Peripheral Interface (SPI) mode and the 68HC11's port D, SPI interface.

The SPI interface consists of a Master device and at least one Slave device. In this example the Master device is the HC11 and the Slave device is the MultiMediaCard.

1.1 HC11 SPI Signals

There are 3 signals in the SPI interface that are controlled by the Master (HC11) and one signal controlled by the Slave (MultiMediaCard). The HC11 "Master" signals are defined as Slave Select (/SS), Serial Clock (SCK) and Master Out Slave In (MOSI). The HC11 defines the input signal from the Slave as Master In Slave Out (MISO).

The corresponding MultiMediaCard signal names and pin numbers are listed below.

Table 1-1

HC11	MultiMediaCard (SPI Mode)	MultiMediaCard Pin #
/SS	CS	1
MOSI	DataIn	2
Ground	Ground	3
-	VDD (+3.3V)	4
SCK	CLK	5
Ground	Ground	6
MISO	DataOut	7

1.2 MultiMediaCard Electrical Interface

The SanDisk MultiMediaCard source and signals operate at a voltage range between +2.7 to +3.6 VDC. This means that the SPI signals must go through some voltage translation if the HC11 is using a +5V source.

An LM317 Positive adjustable voltage regulator (fig. 1-1) is used to provide the MultiMediaCard a "clean" +3.3V source and voltage signal translations.

$$V_{out} = 1.25V\ (1 + R2/R1) \approx 3.3V$$

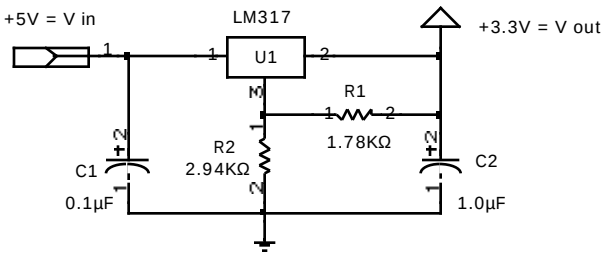


Figure 1-1

This example uses 2 Motorola, MPQ3904 Quad Amplifier Switching Transistor ICs to create 4 (3 master, 1 slave) "Invert the Inverter" step up or step down voltage translations for the SPI signals.

In addition, the MultiMediaCard powers up in MultiMediaCard mode which leaves the DataIn (pin 2) and DataOut (pin 7) lines in "Open Drain" (or floating) mode. The step up and step down circuits described here also provide the pull up resistors necessary to drive the card's DataIn and DataOut lines for this condition. If the circuit did not require voltage translation, the pull up resistors (≤ 10KΩ) would still be needed.

Each SPI output (master) signal from the HC11 uses 2 transistors from the MPQ3904 (fig. 1-2) to step down the +5V output to the +3.3V needed by the MultiMediaCard.

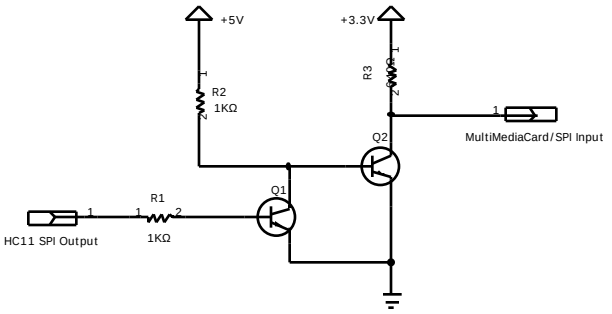


Figure 1-2

Similarly, the SPI input (slave) signal from the MultiMediaCard (DataIN) uses 2 transistors from the MPQ3904 (fig. 1-3) to step up the +3.3V output to the +5V needed by the HC11 signal (MISO).

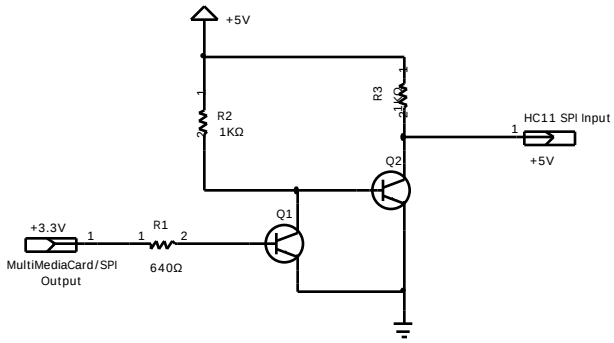


Figure 1-3

1.3 HC11 SPI Set Up

In order for the HC11 to communicate with the MultiMediaCard in SPI mode, the HC11's SPI system and registers must be initialized.

In this implementation, the HC11 registers are mapped to the $8000 address range. The Data Direction register for the HC11's Port D and SPI port resides at address $8009 (fig. 1-4). The /SS, SCK and MOSI SPI signals are set up as outputs from the HC11 (logic one in the corresponding bit place). The MISO SPI signal is set up as an input to the HC11 (logic zero in the corresponding bit place).

$ 8009

D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	1	1	0	1	0
		DDR5 /SS (output)	DDR4 SCK (output)	DDR3 MOSI (output)	DDR2 MISO (input)	DDR1 TXD (output)	DDR0 RXD (input)

Data = $3A

Figure 1-4 Data Direction Register for Port D

This set up is important prior to enabling the HC11's SPI system. Note that the slave select line (/SS) will be driven independent of the SPI system and becomes significant in the initialization of the MultiMediaCard.

The next register to be set up is the SPI Control register (SPCR) mapped to location $8028 (fig. 1-5). The 3 most important configuration bits here are SPE, MSTR, CPOL and CHPA. These last 3 bits determine the Master/Slave status, Clock Polarity and Clock Phase of the HC11. In order to communicate with the MultiMediaCard, the HC11 must be the Master (logic one in the corresponding bit place) and the SPI Clock polarity must be in sync with the Clock phase (both logic ones in the corresponding bit places) with the Clock phase setting equal to "1."

$ 8028

D7	D6	D5	D4	D3	D2	D1	D0
0	1	0	1	1	1	0	0
SPIE	SPE	DWOM	MSTR	CPOL	CHPA	SPR1	SPR0

Data = $5C

Figure 1-5 SPI Control Register

The SPIE, DWOM, SPR1 and SPR0 signals are strictly designer's choice. The current example uses a polling scheme to access SPI data (interrupts disabled) and Port D configured for a "push-pull" operation. The MultiMediaCard can operate at a clock speed from 0 to 20MHz in SPI mode. The SPI Bit Rate Select bits (SPR1 and SPR0) can be any setting since the HC11's input clock would have to be at least 160MHz (E ≥ 40 MHz) to exceed the MultiMediaCard's clock rate. In this example the HC11's clock was set to 9.8304

MHz (E = 2.458 MHz) to provide easy baud rate divisions for the RS232 port. The following table displays the SPI Bit Rate Select bits (SPR1 and SPR0) settings and the corresponding clock rates for the SPI clock.

Table 1-2

SPR1	SPR0	E ÷ by	SCK (out)
0	0	2	1.23MHz
0	1	4	614.4 KHz
1	0	16	153.6 KHz
1	1	32	76.8 KHz

Lastly, setting the SPE bit (bit 6) in the SPCR register to zero enables (turns on) the HC11's SPI system.

Once these configurations are set, the HC11 SPI system corresponds and operates as described in the MC68HC11 Reference Manual.

1.4 MultiMediaCard SPI Set Up

Once power is applied, before access to the MultiMediaCard can take place, the HC11 must send 64 SPI clock cycles (8 bytes) out the SPI port while the Slave select line is held high to initialize the card.

This can be accomplished by first setting the HC11's /SS line high by writing a '1' to bit 5 of the Port D register at address $8008 (fig. 1-6).

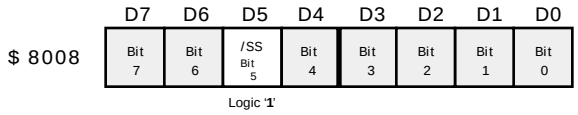


Figure 1-6 Port D Register

Since the HC11's SPI clock is only generated for exactly 8 cycles after a "write" to the SPI Data Register (SPDR, address $802A, fig. 1-7), $FF's can be written to that register simply to generate clock cycles (fig. 1-11). Ten writes of $FF data to the SPDR will provide the 74 clock cycles needed to initialize the MultiMediaCard, plus 6 extra cycles (80 cycles total).

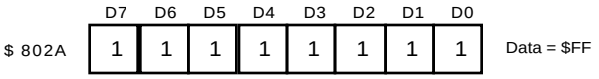


Figure 1-7 SPI Data Register

Note that the MultiMediaCard only requires a minimum continuous 8 cycle clock stream and that the delay between writes to the HC11's SPDR register does not cause timing or communications problems.

1.5 MultiMediaCard SPI Access

Even though the SPI communication protocol for the MultiMediaCard is described in the SanDisk MultiMediaCard Product Manual, it is probably worth reviewing here for clarity.

Basic communications with the MultiMediaCard in SPI mode always start by driving the 68HC11's /SS line low and keeping it in that state until a complete MultiMediaCard/SPI, two way, transaction is complete. This is accomplished by setting bit 5 in the HC11's port D register (address $8008) to '0.'

The key thing to remember about communicating with the MultiMediaCard in SPI mode is that **ALL** commands to the card are 6 bytes long (fig. 1-10).[1] The first byte is always the "command" byte and the command byte's upper bits (7 and 6) are always '0' and '1' (fig. 1-8).[2] For example, if you were to send the card Reset command (command 0), the first byte would be the zero command with bits 7 and 6 equal to '01' or the byte would equal $40.

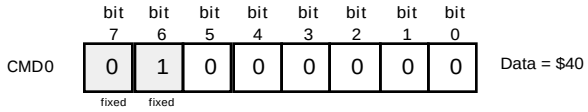


Figure 1-8

The MultiMediaCard "commands" are all listed in decimal format in the manual so it is not always easy to see the binary conversion. For example, Command 24 (Write a Block) is 24 decimal, $18 hexadecimal and $58 hexadecimal when bits 7 and 6 are figured into the byte (fig. 1-9).[3]

---

1 See Figures 1-10, 10-12, 10-13, 10-14, 10-16 & 10-20.
2 See Figures 1-8 & 10-12.
3 See Figures 1-9 & 10-16.

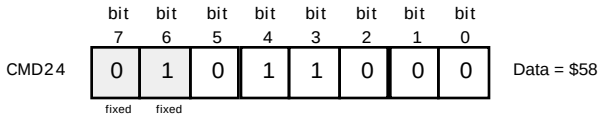


Figure 1-9

After the command byte is sent, it must always be followed by 4 argument or address bytes, with the most significant byte (MSB) sent first (fig. 1-10). Even if the command (as in command 0) does not have a corresponding argument, 4 "null" or "don't care" bytes must be sent.[4]

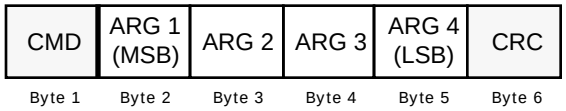


Figure 1-10

The last byte to be sent is the CRC byte. When the MultiMediaCard is powered up, the card is in MultiMediaCard mode. To put the card in SPI mode, the following steps must be taken.

- Pull the card's CS (/SS, pin 1) line low.
- Send the CMD0 (RESET) command and argument.

  ($40 00 00 00 00)

- Send the valid (since we are still in MultiMediaCard mode) CRC byte.

  ($95)

Since the command byte for CMD0 is always $40 and the command argument is always a "null" argument ($00 00 00 00), the CRC will always be $95.

The sequence above (followed by a valid response on pin 7) puts the card into the SPI mode until power is cycled to the card. Once in SPI mode, the default mode is "non-protected." This means that CRC calculations are disabled and the command protocol can be satisfied by sending an $FF in the CRC byte. As in the case of the argument bytes, the CRC byte (or $FF) must be sent even if CRC checking is disabled to ensure that the command sequence is 6 bytes long. In this example, all operations are accomplished in the "non-protected" mode with CRCs disabled.

Once the 6 byte command sequence is sent to the MultiMediaCard, the 68HC11 then must wait for the card to send "response byte" back via the MISO line.

Reading data from the SPI slave device on the 68HC11 requires some maintenance. It is probably wise to clear the 2 SPI read data buffers by a read to the HC11's SPSR register ($8029) followed by a read from the SPDR register ($802A) a couple of times before the next write to the SPDR. This ensures that the results of next SPI cycle will be the only data in the buffer.

Note that in the HC11, any SPI cycle is started by a write to the SPDR register ($802A), even an SPI 'read' from the slave. Writing an $FF to the SPDR (fig. 1-11), to maintain a "high" level on the MOSI output line, toggles the SPI clock for 8 cycles. Checking the SPIF flag bit (bit 7) in the HC11's SPSR register indicates the end of the transfer and the SPDR can then be read to check for the MultiMediaCard's response byte.[5]

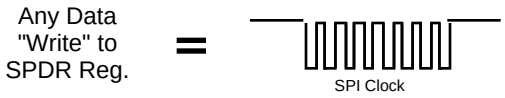


Figure 1-11

The MultiMediaCard's response byte (fig. 10-20) will always start (MSb) with a '0' in bit 7. If the data returned (response byte) is $00 then the rest of the MultiMediaCard SPI transaction can take place. If not, then the error needs to be determined and handled in accordance with the SanDisk MultiMediaCard manual.

If stored data is being read or written, the next few SPI HC11 read cycles can be handled in the same manner as the response byte, this time checking for an $FE (Data Token)[6] followed on the very next set of 8 bit cycles by stored memory data in a "read" operation. In a Data Read operation from flash memory, a Start Token will be first transmitted over the HC11's MISO line (fig. 1-12). This will immediately be followed by 1 block of stored data. Immediately following the stored data, the MultiMediaCard will transmit a 2 byte CRC, marking the end of the data transfer.

[4] See Figures 1-10, 10-12, 10-13, 10-14, 10-16 & 10-20.

[5] See Figures 1-11 & 10-9 through 10-23.

[6] See Figures 1-12, 10-17 & 10-22.

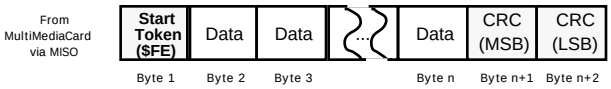


Figure 1-12

A Data Write operation from flash memory, is similar in structure to the Data Read, with the *HC11* sending the same sequence as in Figure 1-12 via the MOSI line. The end of a Data Write transfer is acknowledged by the Data Response[7] byte, via the HC11's MISO line will transmit a 2 byte CRC, marking the end of the data transfer.

When the entire MultiMediaCard SPI transaction is complete, with the receipt of a response byte or receipt of the 16 bit CRC as in a data read cycle, the /SS bit can now be set high by writing a '1' to bit 5 of the HC11's port D register (address $8008).

[7] xxx0 1011 = rejected & xxx0 0101 = accepted.

2.0 Basic Design Example

The basic design chosen for this example uses a Motorola 68HC11 8 bit Microcontroller in the "expanded multiplexed" mode[8] with 24K* of contiguous external ROM and 32K of contiguous external static RAM as show in the diagram below.

[8] Motorola MC68HC11 HCMOS Reference Manual, Rev.1,1990, section 2.6.

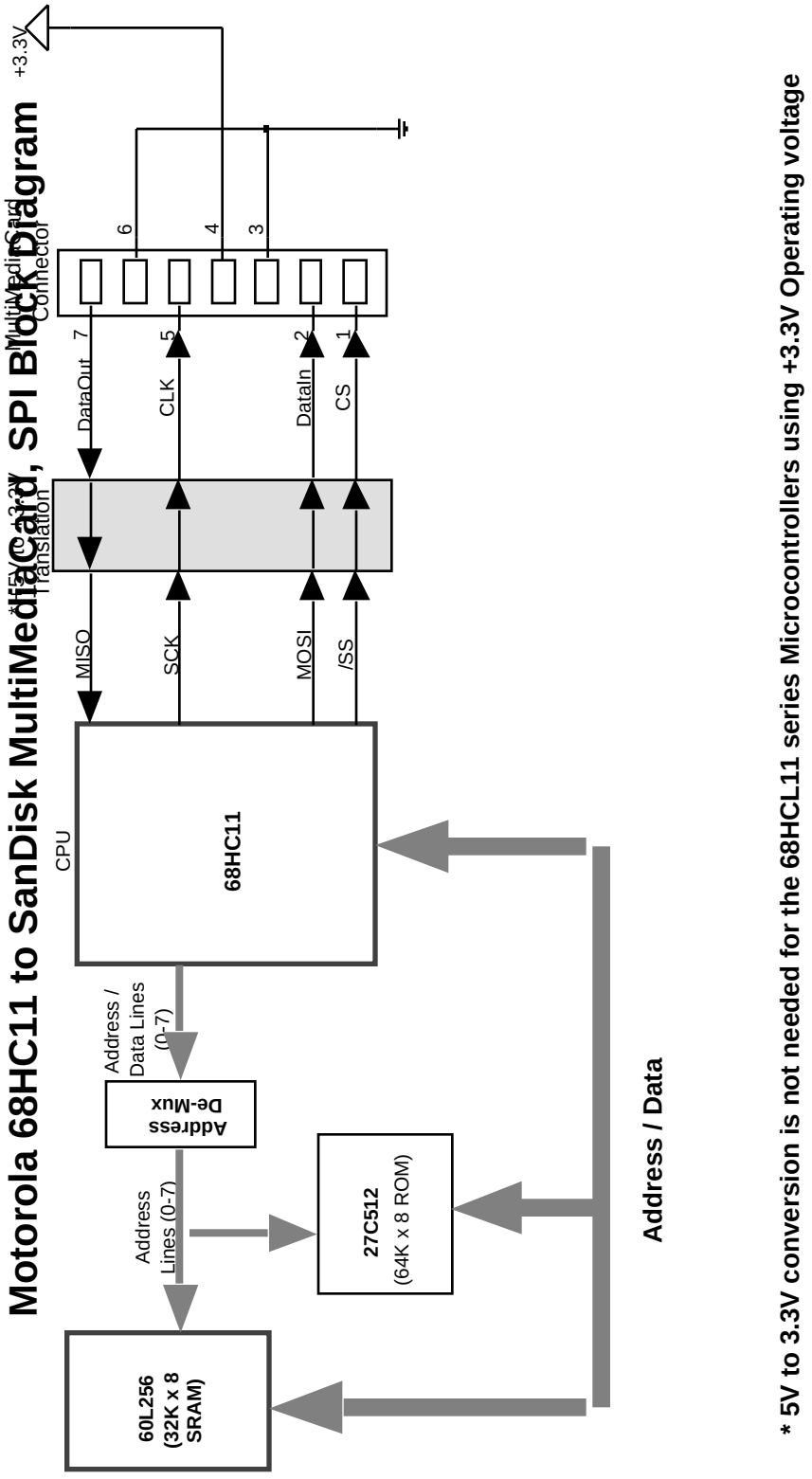


Figure 2-1 Basic Design Example

*Note: The upper 40K of a 64K x 8 bit PROM is discarded in this simple design, but could be reclaimed through a ROM “paging” scheme.

3.0 Design Example CPU

The CPU used in this example is the single-chip 8 bit Motorola MC68HC711E9 Microcontroller.

The 711E9 version of the 68HC11 family contains the following features that are of immediate interest for this design:

- 12k of UV Erasable internal PROM
- 512 bytes of internal RAM
- 512 bytes of internal EEPROM
- A built-in Serial Communications Interface
- 32 selectable "standard" Baud rates
- A built-in Serial Peripheral Interface
- A built-in 8 channel, 8 bit 'A' to 'D' converter
- A built-in 8 bit Pulse Accumulator/ Generator System

The 68HC11 is clocked by a 9.8304 MHz crystal (XTAL-1) which it internally divides to provide a 2.4576 MHz "machine cycle" timing reference (E[9]) for the CPU and external devices.

The 68HC11 can be set to either Expanded Multiplexed or Special Boot Strap[10] modes by switch SW1- (2) controlling the processor's MODA and MODB lines.[11]

The normal operating mode for the 68HC11 in the design example is the Expanded Multiplexed mode. In this mode, ports B and C are used as multi-purpose ports to provide a 16 bit multiplexed address and 8 bit data bus. Port C becomes AD0 - AD7 and port B becomes A08 - A15.

3.1 Port A (User Input Control Register)

Port A (PA0 - PA7) of the 68HC11 is used as the control pin port for communicating with the user (fig. 3-1). PA0 (bit 0) is connected to the open end of SW4 and a 4.7KΩ pull-up resistor when SW4 is closed a PA0 is pulled to ground and bit 0 of Port A becomes '0.' PA1 is connected to SW3 in exactly the same way as SW4 and provides a '0' at bit 1 of Port A when closed. SW4 is polled in software to initiate the audio "record" function and similarly, SW3 is polled to initiate the "Play" function.[12] PA7 is connected directly to the cathode of LED, D1. Writing a '0' to bit 7 of Port A turns on the LED. Both PA0 and PA1 are configured as inputs and PA7 is configured as an output.[13]

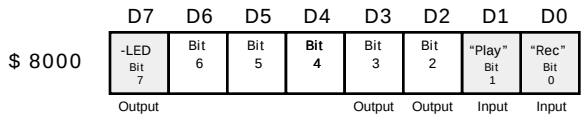


Figure 3-1 Port A Data Register

PA2 through 6 are unused and free for other use.

3.2 Ports B & C

Port B of the 68HC11 is used to provide the circuit with the high order address lines (A08 - A15).

Port C (AD0 - AD7) is used as the 8 bit data lines for the circuit. Chip select lines for RAM, ROM and the DAC are controlled by the AS (address strobe) and R/W (read/write) lines from the processor to ensure that these lines are accessed on data cycles only.

Additionally, Port C (AD0 - AD7) is demultiplexed by a 74HC373 (U3).[14] The 74HC373 latch is controlled by the AS (address strobe) line to provide the low order address lines (A00 - A07) for the circuit.

---

9 E = HC11's timing reference signal (bus clock).

10 Motorola MC68HC11 HCMOS Reference Manual, Rev.1,1990, section 3.5.4.

11 See Schematic (Figure 10-2) section G-3.

12 See Schematic (Figure 10-2) section E-1 & 2.

13 See Schematic (Figure 10-2) section A-3.

14 See Schematic (Figure 10-2) section D-5.

3.3 Port D

Port D is the 68HC711E9's serial port/general purpose port. This circuit uses PD0 as RXD (the receive line) and PD1 as the TXD (transmit line) for RS232 communications.

PD2 through PD5 of port D make up the Serial Peripheral Interface (SPI) port for the HC11 as follows.

PD2 = MISO

PD3 = MOSI

PD4 = SCK

PD2 = /SS

Since the HC11's interface to SPI functions as a system, it is unnecessary to toggle each corresponding bit in the port D register, with the exception of /SS.

3.4 Port E

Port E of the 68HC11 is used as the A to D port. Only AN0 is actually connected and polled for A/D input from the Microphone.

4.0 Memory Management

The "system" RAM, EEPROM and register controls are all internal to the 68HC11 and are shown in the Design Example's memory map (fig. 5-1).

4.1 RAM

The programmer, through the 68HC11, has the ability to re-map the HC11's internal RAM, EEPROM and register controls to any 4k boundary in a 64k memory map.

The 512 bytes of internal RAM are mapped from address $0000 to address $0200 hex via the 68HC11's "INIT" register (fig. 4-1).

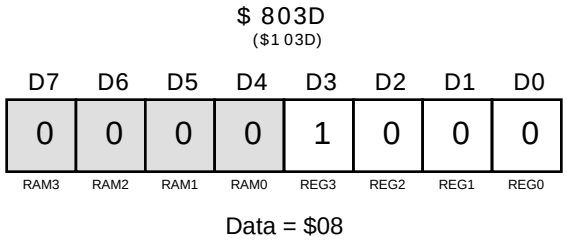


Figure 4-1 RAM & I/O Init. Register

"External" RAM is provided via an MCM60L256, 32k x 8 static RAM IC (U4). The "external" RAM is mapped between address $0000 and $0FFF in order to take advantage of the BCLR, BITA, BITB, BRSET and BSET instructions that assume "00" is the high order address byte. These instructions give the programmer the ability to bit manipulate RAM data that can then be read in from or "stuffed" out of any of the CPU ports.

Because of the larger amount of contiguous RAM needed for the fast accumulation of "Audio" input data and since the "external" RAM has a 4k byte address range limit, this design sacrificed the lower 512 bytes (overwritten by the internal HC11's RAM)[15] to limit the part count and avoid developing an address conversion scheme.

4.2 Internal ROM & EEPROM

Internal ROM and EEPROM are disabled via the 68HC11's CONFIG register (fig. 4-2). This was done to provide 24K of contiguous external ROM needed for the Host Developer's Tool Kit.

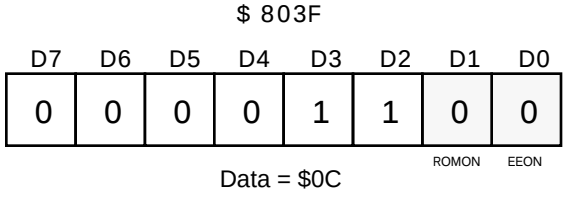


Figure 4-2 Configuration Control Register

4.3 External ROM

In this Design Example, external ROM is mapped between address $A000 and $FFFF and is intended to contain the system routines including the Host Developer's Tool Kit code.

[15] External address and data lines are automatically disabled by the HC11 when accessing internal RAM.

4.4 Control Registers

The 68HC11 control registers are mapped at the first 4k boundary above external RAM, between address $8000 and $8FFF. Setting the 68HC11 **INIT** register (fig. 4-3) to a hexadecimal value of $08 ensures that Internal RAM and registers are mapped correctly into this design.

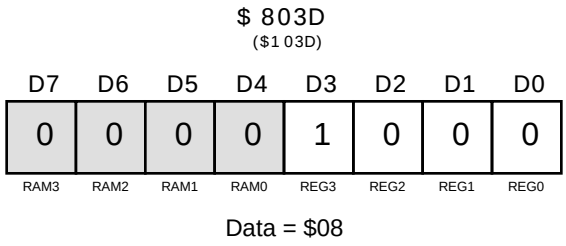


Figure 4-3 RAM & I/O Init. Register

4.5 Digital to Analog Converter

The 8 bit Digital to Analog Converter (DAC). is mapped at the first 4k boundary above the 68HC11 control registers. All access to the DAC is between addresses $9000 and $9FFF.

An 8 bit data write to any address between $9000 to $9FFF will cause the DAC to output a voltage, based on the following formula, out to the input of one of the OP amps in the MC33174 (U8).

$$V_{DAC\ out} = +5V * (data\ in / 256)$$

5.0 Design Example HC11 Memory Map

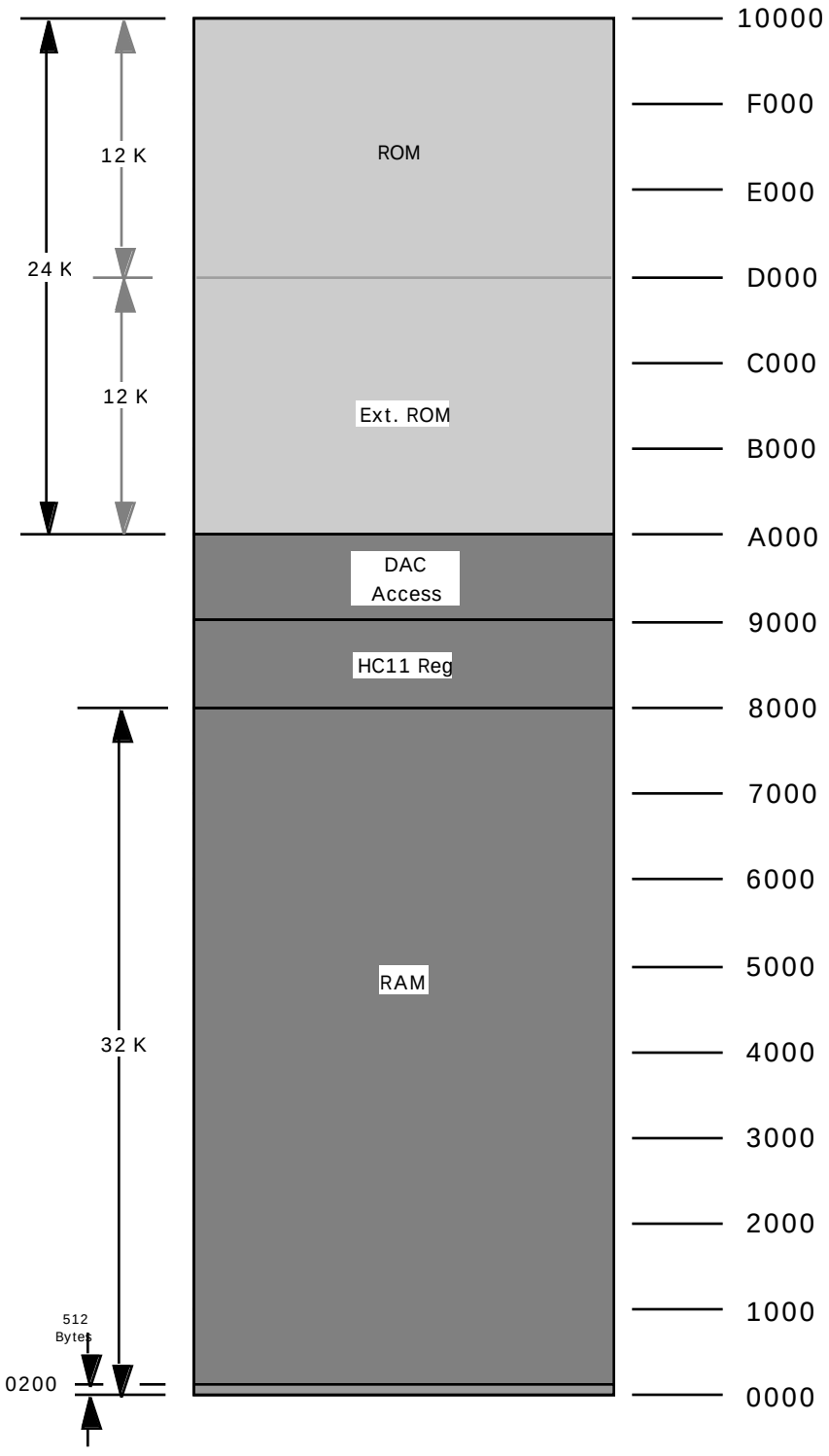


Figure 5-1 HC11 Memory Map Design Example

6.0 Design Example Configuration & Control Logic

6.1 Address and Data Logic

This Design Example is intended to be configured to use the address and data lines by setting the 68HC11 in the expanded multiplexed mode.[16] Using the expanded multiplexed mode sacrifices access to the general purpose I/O ports B & C and is accomplished by setting both the external MODA (pin 3) and MODB (pin 2) pins of the 68HC11 to a logical "1" (+5 V).

The lower byte address lines (A00 - A07) are extracted from multiplexed address & data lines (AD0-AD7) from the 68HC11 by a 74HC373 latch. The 68HC11's Address Strobe line (AS, pin 4) provides the output enable signal for the latch to capture and hold a valid address during the address cycle.

6.2 PAL Decode Logic

To map the DAC, external RAM and external ROM into the HC11's memory map, certain control signals must be modified or generated. In the design example an ICT Inc. PEEL™ 22V10AZ PAL[17] was used to provide these control signals.

The PEEL 22V10AZ was used because of its flexibility, speed and ability to be reprogrammed without UV erasure.

ICT also makes the PEEL 22LV10AZ that is nearly identical to the 22V10AZ except that it operates at low voltage ranges (between +2.7 and +3.6V) that exactly mirror that of the MultiMediaCard for low power/voltage designs.

6.3 RAM Equations

The RAM used in the design example is a 60L256 static RAM. This RAM requires 2 control signals.

The "WE" (Write Enable) signal is tied directly to the HC11's "R/W" (pin 6) and requires no other logic.

In this design example the RAM's "OE" (Output Enable) and "CE" (Chip Enable) are tied together and controlled by the PAL generated "-RAMEN" signal. The logic for "-RAMEN" (active low) is as follows.

-RAMEN = A15 or (**NOT** E)[18]

6.4 ROM Equations

The ROM used in the design example is a 27C512 PROM. This PROM requires two control signals and one modified address line to map into the HC11's memory map.

The "/E" (chip enable) signal is tied to ground and requires no other logic.

In this design example the ROM's "/G" (Output Enable) is controlled by the PAL generated "-ROMEN" signal. The logic for "-ROMEN" (active low) is as follows.

-ROMEN = (**NOT** A15) **or** ((**NOT** A14) **and** (**NOT** A13)) **or** (**NOT** E) or (**NOT** R/W)[19]

The ROM address must be modified to map into the $A000 to $FFFF address range. This is done so that access to ROM addresses are sequential and contiguous within the IC.

RA15 = "0"

RA14 = A14 **and** A13

RA13 = (**NOT** A13)

RA12 = A12

---

16 Motorola Microprocessor, Microcontroller and Peripheral Data book, Vol. II, MC68HC24 Port Replacement Unit (PRU), 1988, pp.3-1719.

17 The PEEL ICT 22V10A from ITC can also be used.

18 'E' is the 68HC11's bus frequency clock output. 'E' low = address portion, 'E' high = data portion. Motorola M68HC11 Reference Manual, 1990, pp.2-11.

19 'E' is the 68HC11's bus frequency clock output. 'E' low = address portion, 'E' high = data portion. Motorola M68HC11 Reference Manual, 1990, pp.2-11.

6.5 DAC Access Equations

The Texas Instrument 8 bit Digital to Analog Converter (TLC7524) used in the design example to take the stored data from memory and convert it back into an analog output to reproduce the original sound.

Access to the TLC7524, DAC is controlled by its /WR (pin 13) and /CS (pin 12) pins, both controlled by the -CE signal from the ICT, PEEL 22V10AZ PAL (U5). The logic for "-CE" (active low) is as follows.

-CE = (**NOT** A15) **or** A14 **or** A13 **or** (**NOT** A12) **or** (**NOT** E)

All PAL equations are listed below in "ABEL" format.

Equations:

RAMEN_ = A15 # !E;

ROMEN_ = (!A15 # (!A14 & !A13)) # !E # !R_W;

CE_ = !A15 # A14 # A13 # !A12 # !E;

RA15 = 0;

RA14 = A14 & A13;

RA13 = !A13;

RA12 = A12;

7.0 Serial Communications

Port D of the 68HC11 functions as the serial communications port for the processor. PD0 (pin 20) acts as the single Receive Data (RXD) line and PD1 (pin 21) acts as the single Transmit Data (TXD) line for NRZ format serial communications port on the 68HC11.[20]

The TXD and RXD signals are ported through the Maxim, MAX232, RS232 driver/receiver chip. These signals are then connected to a mini DIN 8 connectors.[21]

Some useful addresses for serial communications are provided below:

Table 7-1 Addresses for Serial Communications

Address	68HC11 Serial Comm. Register
$802B	BAUD
$802C	SCCR1
$802D	SCCR2
$802E	SCSR
$802F	SCDR

[20] Complete descriptions of the serial interface and controls are provided in the Motorola MC68HC11A8 HCMOS Single-Chip Microcontroller Technical Data book, 1988, pp.5-1 to 5-11.

[21] See Schematic fig. 10-2, section F-6.

8.0 Power Supply & Special Modes

8.1 Power Supply

The heart of the 5 volt supply for this design example is a simple 7805, linear DC to DC converter. The output of the 7805A provides a fixed 5 volt supply at approximately 200mA.

The design example power supply requires a +9 volt (DC) source to be connected to the input of the 7805 (V_in).

In addition to the 7805, an LM317 Positive adjustable voltage regulator is used to provide a "clean" +3.3V source for the MultiMediaCard and voltage signal translations.

8.2 Low Battery Detection

Low battery detection is accomplished using a Seiko, S-8054HNM low voltage detector (Q3).

R4 and R5 provide the voltage reference division for the "low battery detection" circuit. Q3 triggers (logic "0") if the voltage across R5 drops below 4.5 to 4.7 volts, providing ground (thus completing the circuit) for the low battery LED (LED1). R4 (10KΩ ± 1%) & R5 (26.1KΩ ±1%) divide the "9Vin" battery input to provide the voltage across R5 for input to Q3 ($V_{5\ Ref}$ = [9Vin x R5] / [R4+R5] ≈ 9Vin x 0.723).

LED D1 will be lit when the low battery condition is met (when Q3 triggers) and will remain dark while the battery ($V_{5\ Ref}$) is above 6.22 to 6.50 volts.

8.3 Low Voltage Reset

The 68HC11's EEPROM is susceptible to corruption if VDD (supply voltage) falls to less than +5V. Q1, the Seiko S-8054HN low voltage detector, is used to pull the Design Example system reset (/RESET) line low, in accordance with the 68HC11 specification to protect the EEPROM data. Q1 triggers (logic "0") if the +5V line drops below 4.2 to 3.8 volts.

8.4 Design Example Programming Mode

Motorola provides no operating code in the internal EEPROM or RAM of the 68HC11 (U2).

Motorola provides a "Special Bootstrap" mode that allows the programmer/developer to load and then execute a program in the 68HC11 RAM via the serial communications port (PD0 & PD1) at 9600 baud.[22]

Closing both switch, SW1- 1 and 2 (setting them to the "on" position) sets the 68HC11 in the Design Example to the "Special Boot Strap mode" by pulling the 68HC11 MODA & MODB lines to a logic "**0**."

TXD and RXD are ported through the MAX232, RS232 driver/receiver chip. These signals are then connected to the mini DIN 8 connector and become the port for the 9600 baud "Special Boot Strap" mode when SW1- switches 1 and 2 are closed. A short program that will initialize the EEPROM and allow the developer to then download a second program directly into the 68HC11 EEPROM is intended to be loaded into the 256 bytes of 68HC11 internal RAM using this method.

This method allows the design example to be used as a "primitive" development system to test code.

[22] Motorola MC68HC11A8 HCMOS Single-Chip Microcontroller Technical Data book, 1988, pp.2-3 & 2-5 to 2-8.

9.0 Basic Record/Play Operation

At power up, the firmware initializes the system to map RAM between $0000 - $7FFF, the internal registers to $8000 - 8FFF and the DAC to $9000 - 9FFF. It then sets up the SPI port in poling mode with CHPA and CPOL, both equal to one in the HC11 SPCR register ($8028).

Once the system is set up, the firmware resets the MultiMediaCard flash over the SPI port by sending the Command 0 ($40 00 00 00 00 95)[23] and Command 1 sequence for the MultiMediaCard in SPI mode and waits in a loop polling bit '0' of port A for a button press.

Upon detection of a button press from SW4, the A/D converter is configured for Single channel, continuous scan of A/D channel '0' in the ADCTL register ($8030). The A/D converter channel '0' is connected to the output of one of the MC33174 op amps that amplifies the input from MIC1, the condenser microphone.

The program takes eight samples from the microphone and immediately stores them in sequential RAM locations starting at address $0200. Once written to RAM, the program loops back to read another eight samples from the microphone and stores them in the next consecutive RAM address locations. This loop continues until RAM has been filled to address $7FFF (end of RAM).

At the end of the "RAM write" sequence, the program issues a "Write Block Command" (Command 24) to the MultiMediaCard via the SPI port. Once this command has been acknowledged with a $00 response from the card, a data token ($FE) is sent followed by the first 512 bytes of data stored in RAM ($0200 - $03FF), followed by a "dummy" CRC ($FF). After this block write is completed, the program checks for the Data response from the MultiMediaCard (xxx00101 binary). Then the sequence starts over again with the next MultiMediaCard block address ($0400) until the entire contents of RAM have been transferred to the MultiMediaCard Flash memory.

The program then waits in a loop polling bit '1' of port A for a button press. Upon detection of a button press from SW3, the program issues a "Read Block Command" (Command 17) to the MultiMediaCard via the SPI port. Once this command has been acknowledged with a $00 response from the card, the program waits for a data token ($FE) and begins transferring the following data into RAM staring at address $0200. After the MultiMediaCard SPI Read Block cycle has completed with the 16 bit CRC (discarded), the program issues the "Read Block command" again for the next consecutive block of flash memory ($0400) and stores that data in the next consecutive locations in RAM. This loop continues until RAM is filled (approximately 63 block reads).

Finally, the program reads the data stored at RAM address $0200 and writes it to the DAC address $9000. This data is translated to an analog voltage and amplified by one of the MC33174 op amps that becomes an input to the speaker SPKR1. The next RAM address is read and output to the DAC with an appropriate delay equal to time between reads of the A/D converter.

The loop continues until all RAM locations have been written to the DAC and in turn to the speaker.

[23] Since the card powers up in MultiMediaCard mode a valid CRC is required for CMD0. Once sent, the card is in SPI mode and valid CRCs are not required.

10.0 Block Diagrams and Timing Diagrams

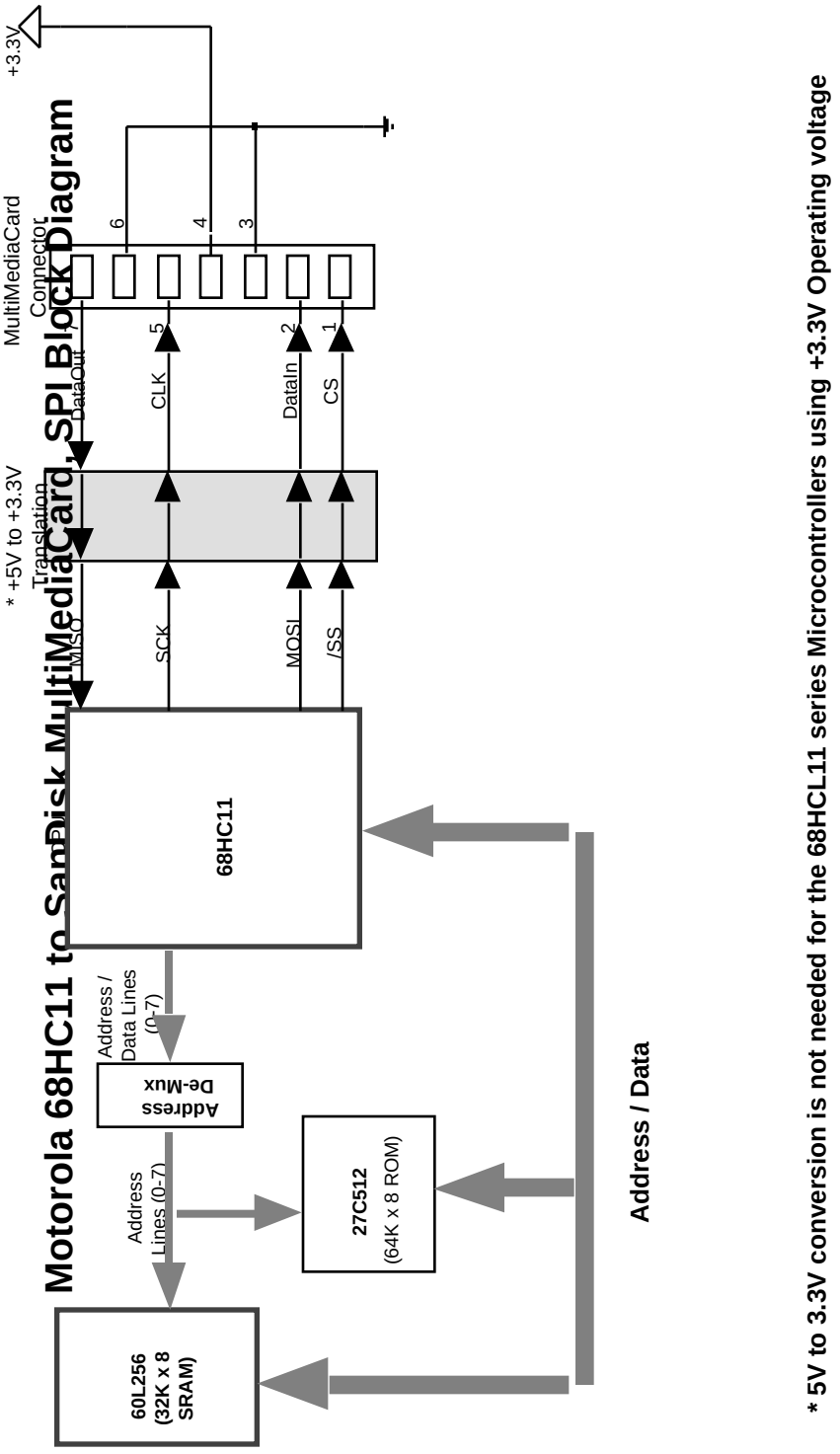


Figure 10-1 Block Diagram

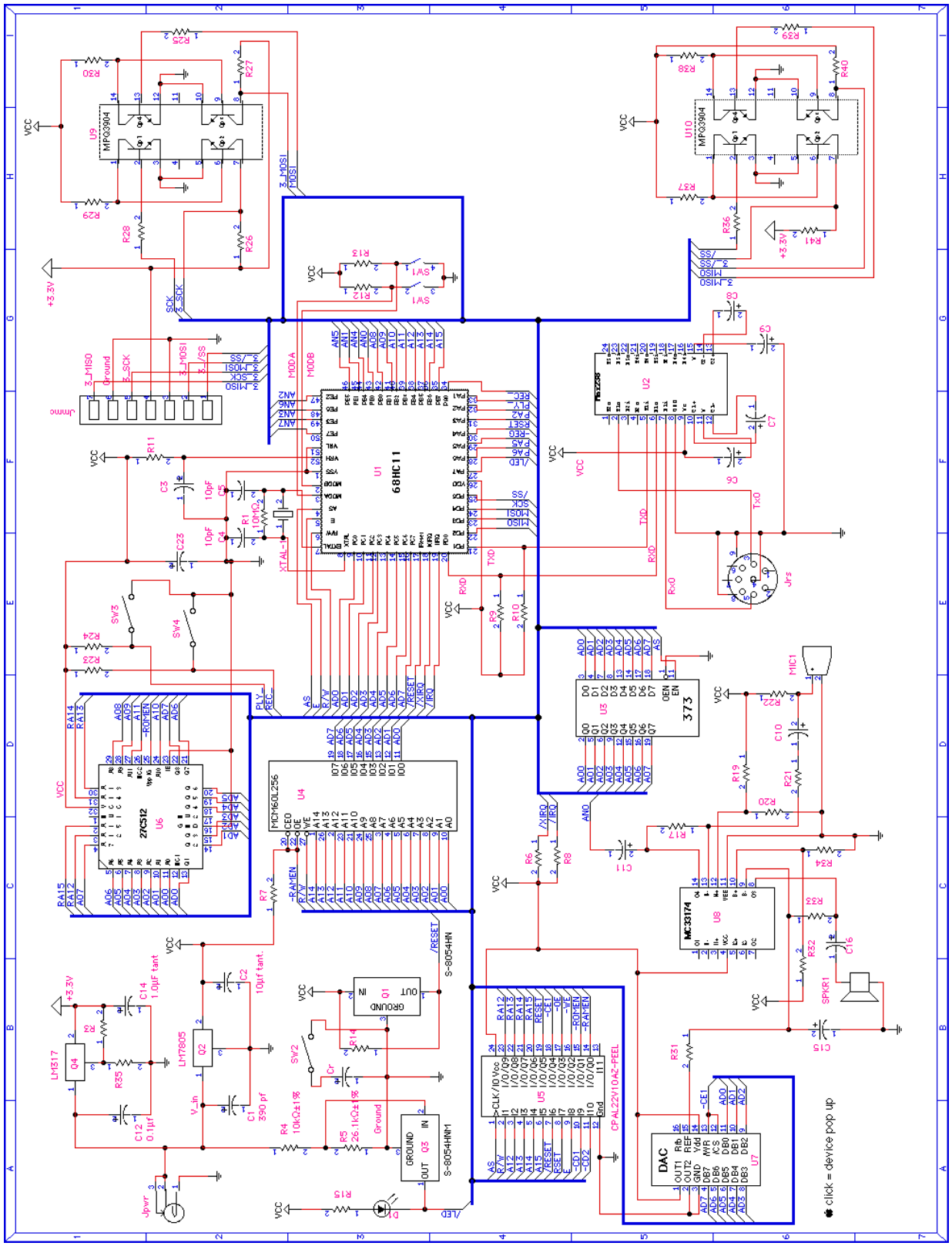
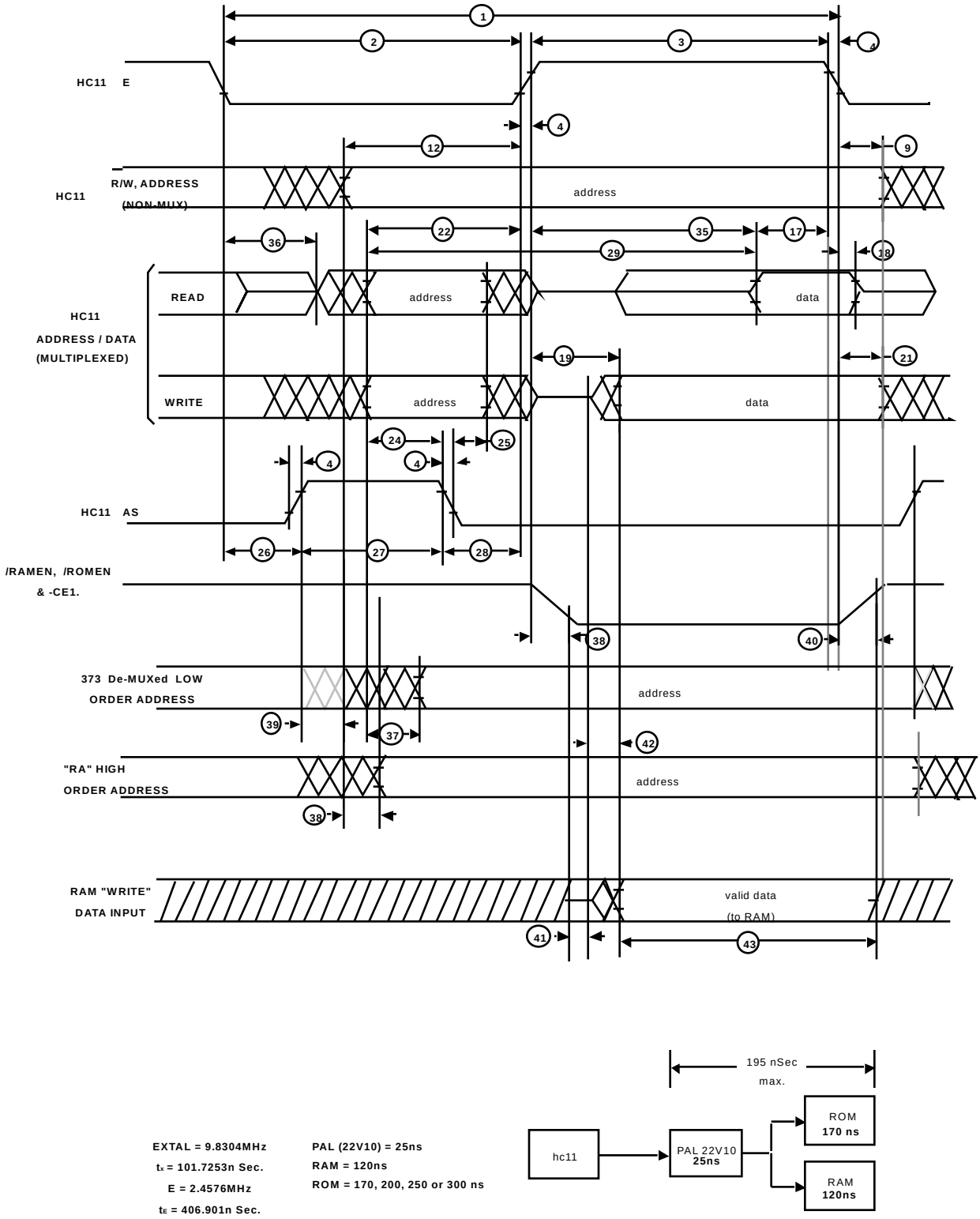


Figure 10-2 Schematic



EXTAL = 9.8304MHz
t_x = 101.7253n Sec.
E = 2.4576MHz
t_E = 406.901n Sec.

PAL (22V10) = 25ns
RAM = 120ns
ROM = 170, 200, 250 or 300 ns

Figure 10-3 Timing Diagram

Table 10-1 Timing Diagram Definitions

Characteristic	Signal #	MIN	MAX
EXTAL (MHz)	-	9.8304	-
E (MHz)	-	2.4576	-
E Cycle Time (nS)	1	406.901	-
Pulse Width, E Low (nS)	2	180.451	-
Pulse Width, E High (nS)	3	175.451	-
E & AS Rise & Fall Time (nS)	4	-	20
Address Hold Time (nS)	9	21.3626	-
Non-Muxed Address Valid Time to E Rise (nS)	12	59.0879	-
Read Data Setup Time (nS)	17	30	-
Read Data Hold Time (nS)	18	10	71.3626
Write Data Delay Time (nS)	19	-	116.363
Write Data Hold Time (nS)	21	21.3626	-
Muxed Address Valid Time to E Rise (nS)	22	49.0879	-
Muxed Address Valid Time to AS Fall (nS)	24	2.72526	-
Muxed Address Hold Time (nS)	25	21.3626	-
Delay Time, E to AS Rise (nS)	26	41.3626	-
Pulse Width, AS High (nS)	27	72.7253	-
Delay Time, AS to E Rise (nS)	28	41.3626	-
MPU Address Access Time (nS)	29	214.538	-
MPU Access Time (nS)	-	-	145.451
Muxed Address Delay [Previous Cycle MPU Read] (nS)	36	71.3626	-
Address de-Mux [74HC373] Enable to Data Out Time (nS)	37	-	35
PAL PEEL 22V10AZ Propagation Delay Time (nS)	38	-	25
RAM [60L256] Read, Address Access Time (nS)	-	-	120
RAM [60L256] Read, Enable Access Time (nS)	-	-	120
RAM [60L256] Read, Disable to Output High-Z Time (nS)	-	0	60
RAM [60L256] Address Valid to End Write Time (nS)	-	100	-
RAM [60L256] Data Valid to End of Write Time (nS)	-	50	-
RAM [60L256] Enable to End Write Time (nS)	-	80	-
RAM [60L256] Write Recovery Time (nS)	-	10	-
ROM [27C512] Read, /G Enable Access Time (nS)	-	-	75
ROM [27C512] Read, Disable to Output High-Z Time (nS)	-	0	60

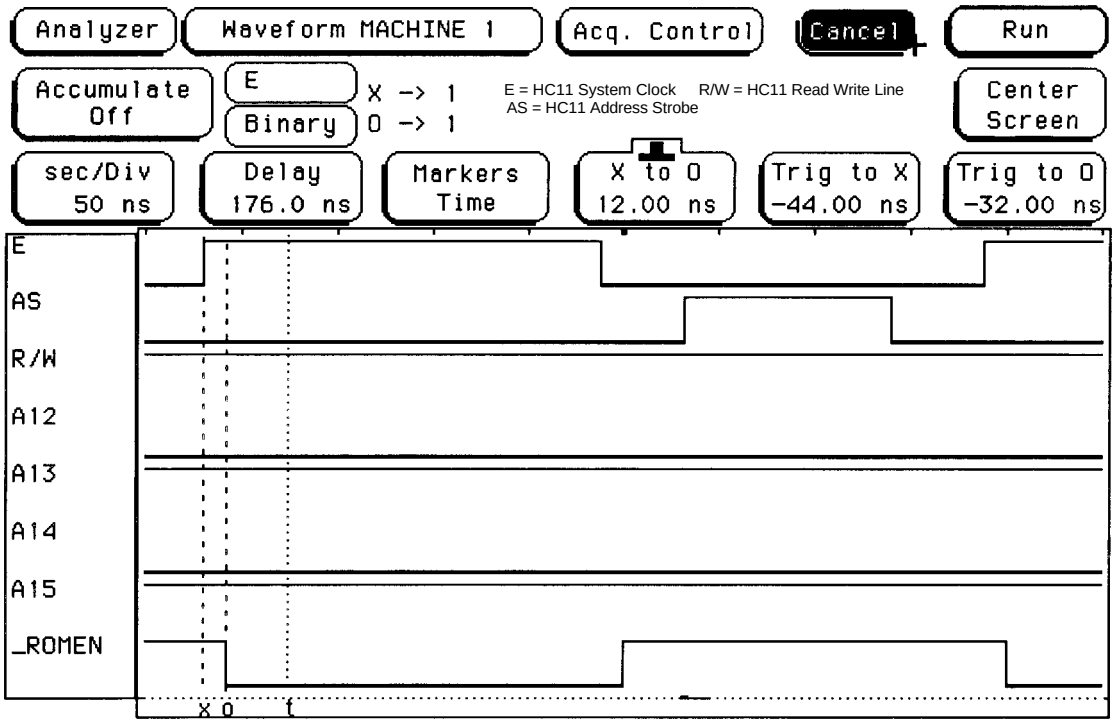


Figure 10-4 ROM Read Cycle

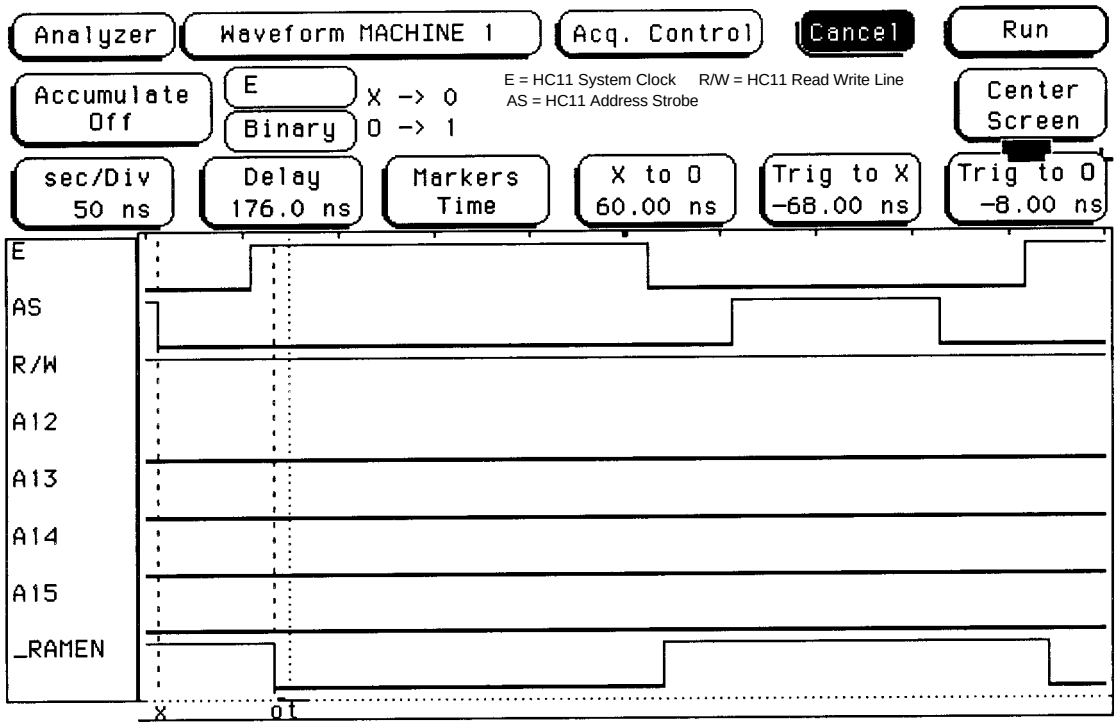


Figure 10-5 RAM Read Cycle

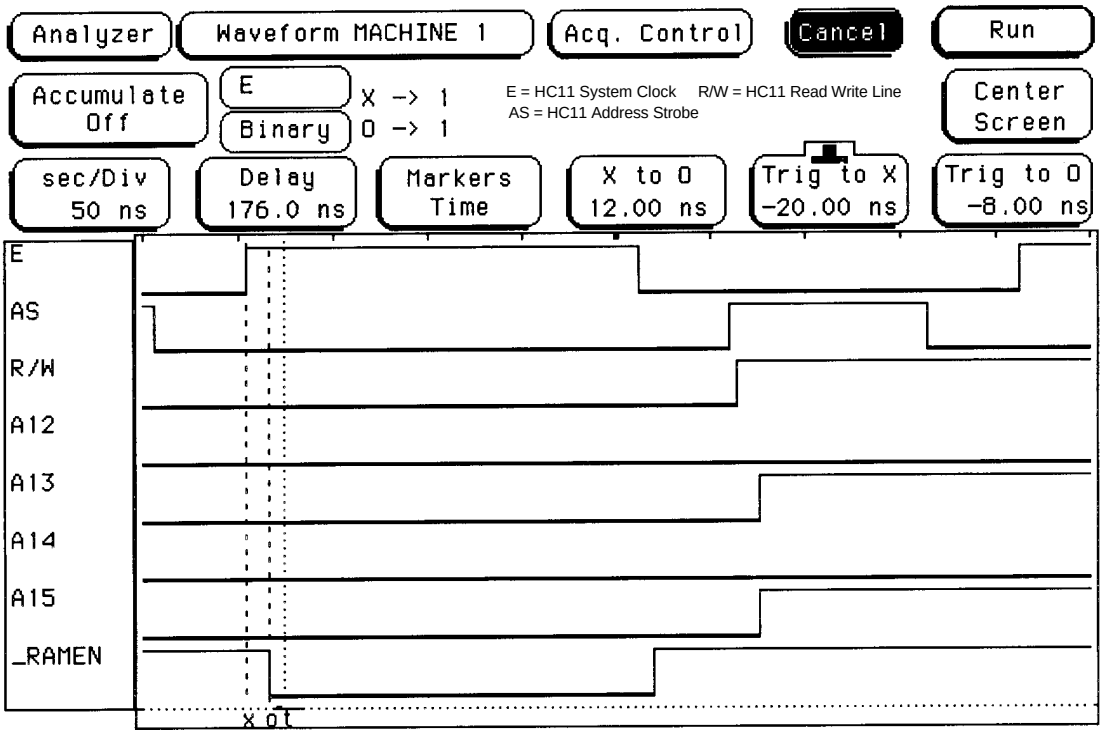


Figure 10-6 RAM Write Cycle

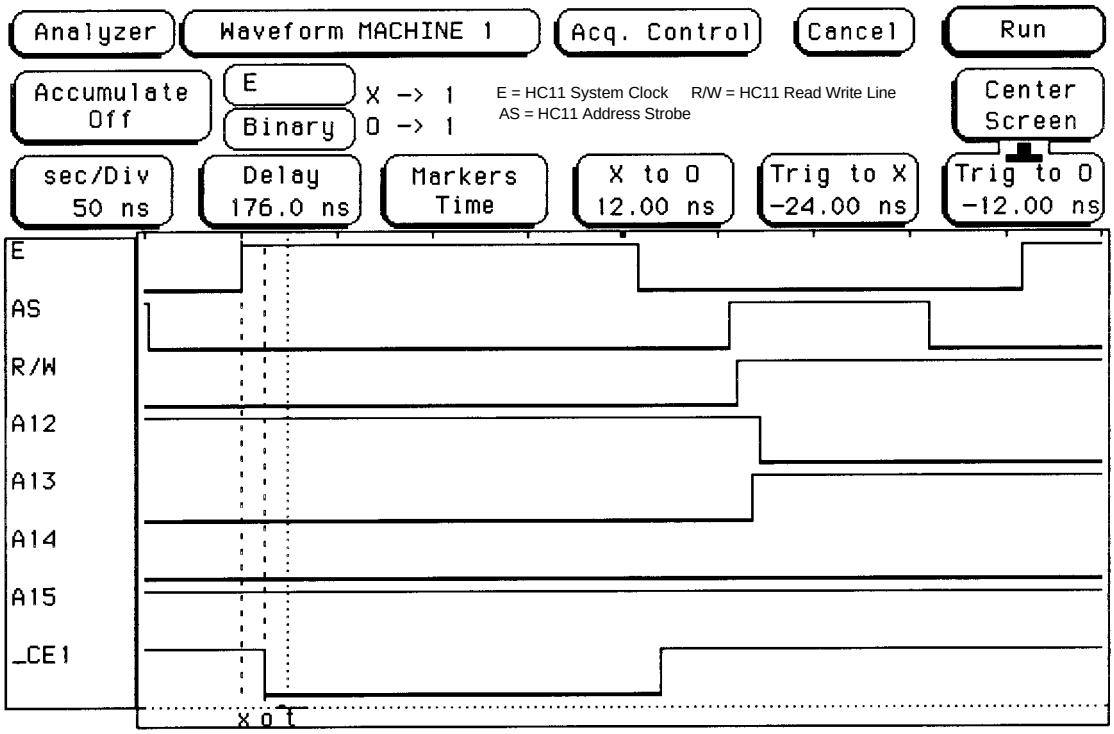


Figure 10-7 DAC Write Cycle

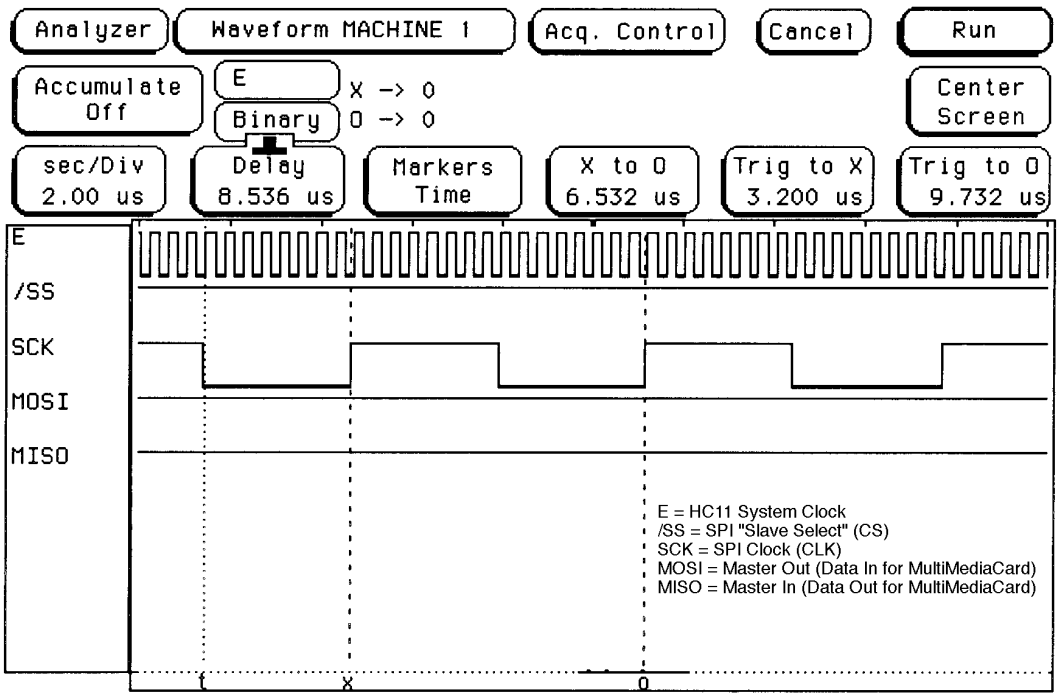


Figure 10-8 HC11 “E” Clock versus SPI “SCK” Clock

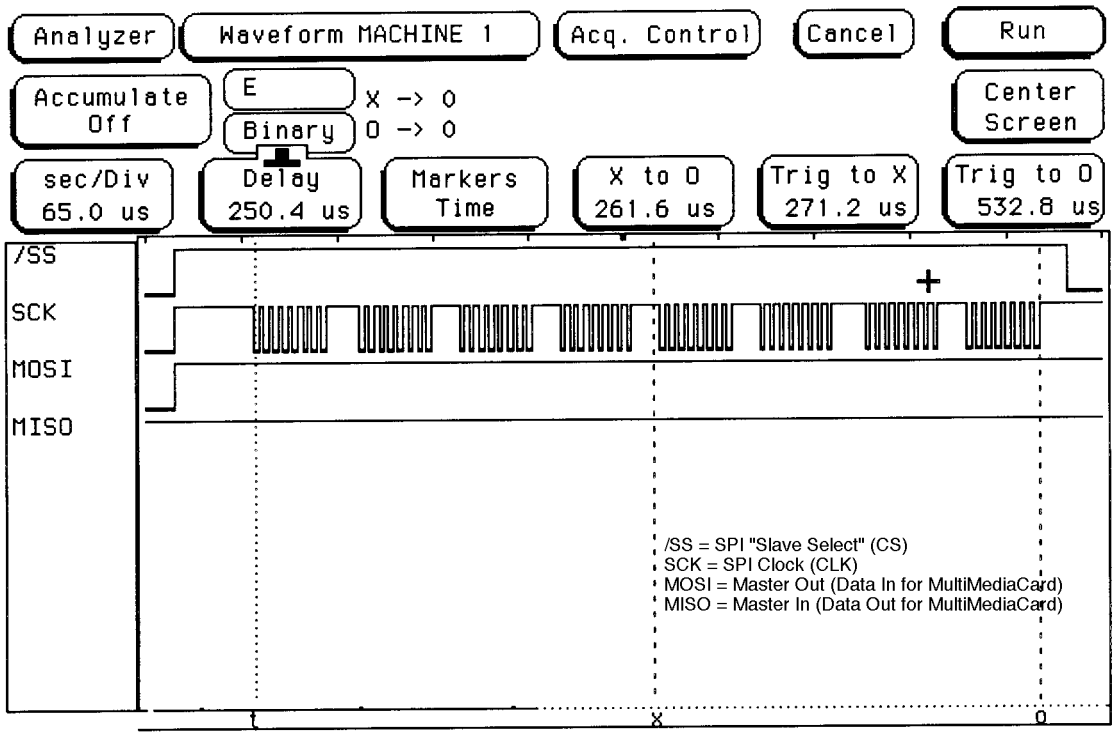


Figure 10-9 MultiMediaCard/SPI 80 Clock Cycle Start-Up

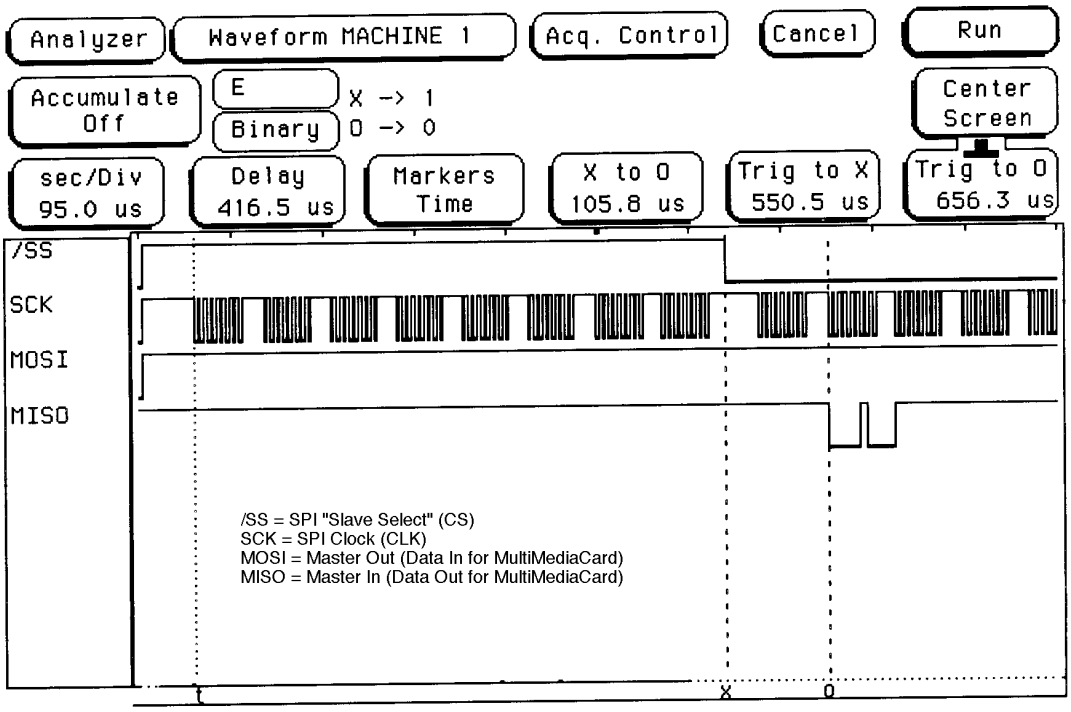


Figure 10-10 80 Clock Cycles Plus MultiMediaCard/SPI Card Response

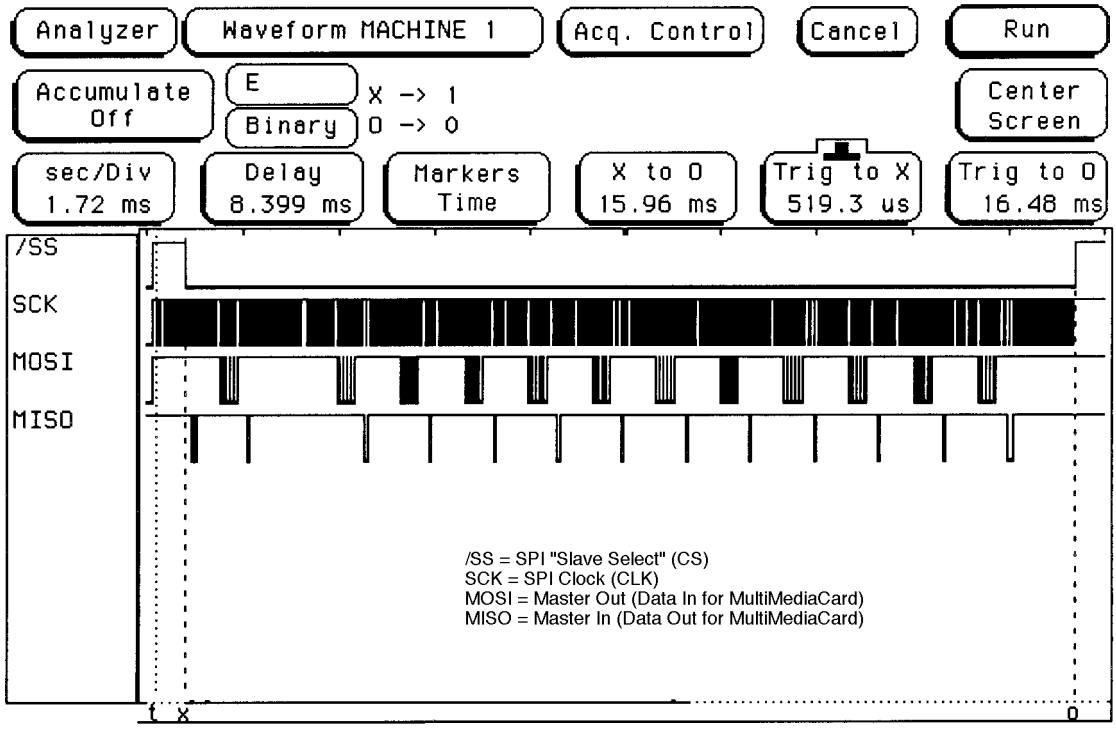


Figure 10-11 MultiMediaCard/SPI Initialization Sequence

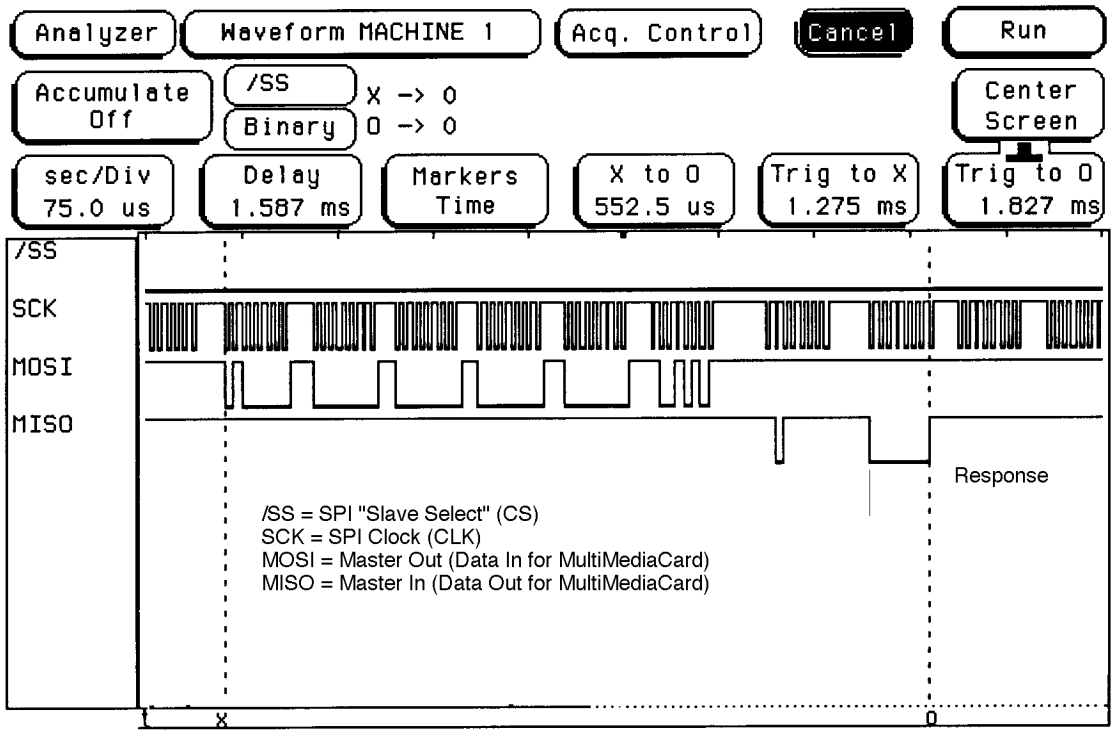


Figure 10-12 MultiMediaCard/SPI “Command 0” Plus Card Response

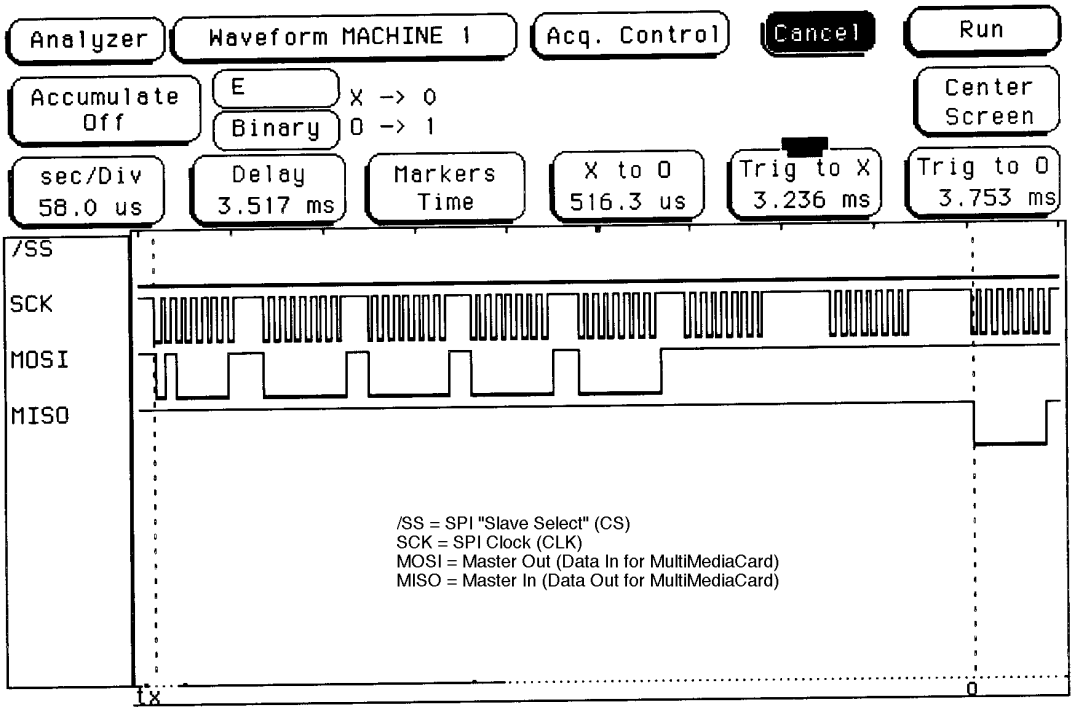


Figure 10-13 MultiMediaCard/SPI “Command 1” Plus Negative “Busy” Response

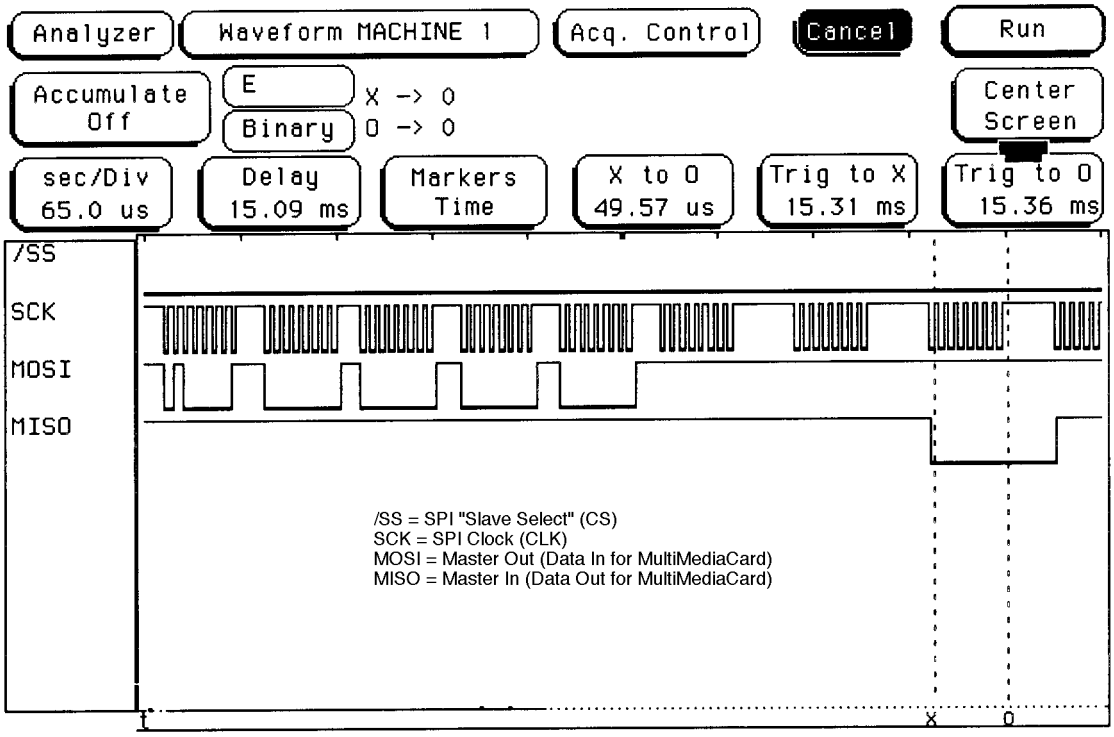


Figure 10-14 MultiMediaCard/SPI "Command 1" Plus Positive Response

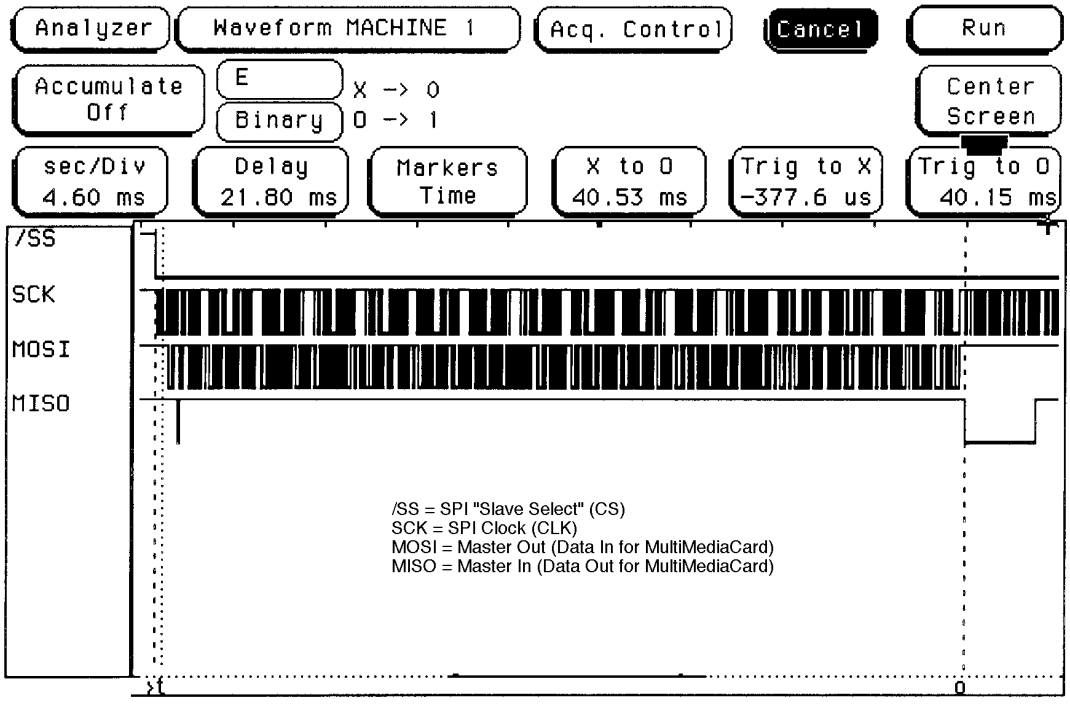


Figure 10-15 MultiMediaCard/SPI "Command 24, Block Write" Sequence

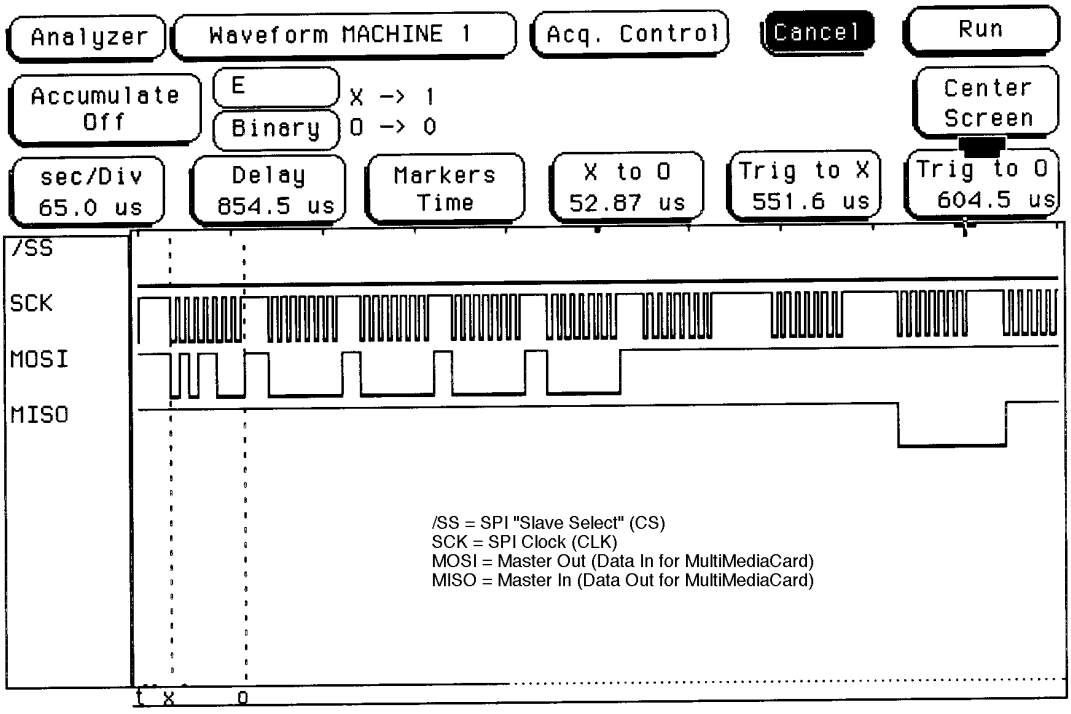


Figure 10-16 MultiMediaCard/SPI "Block Write" Command Plus Positive Response

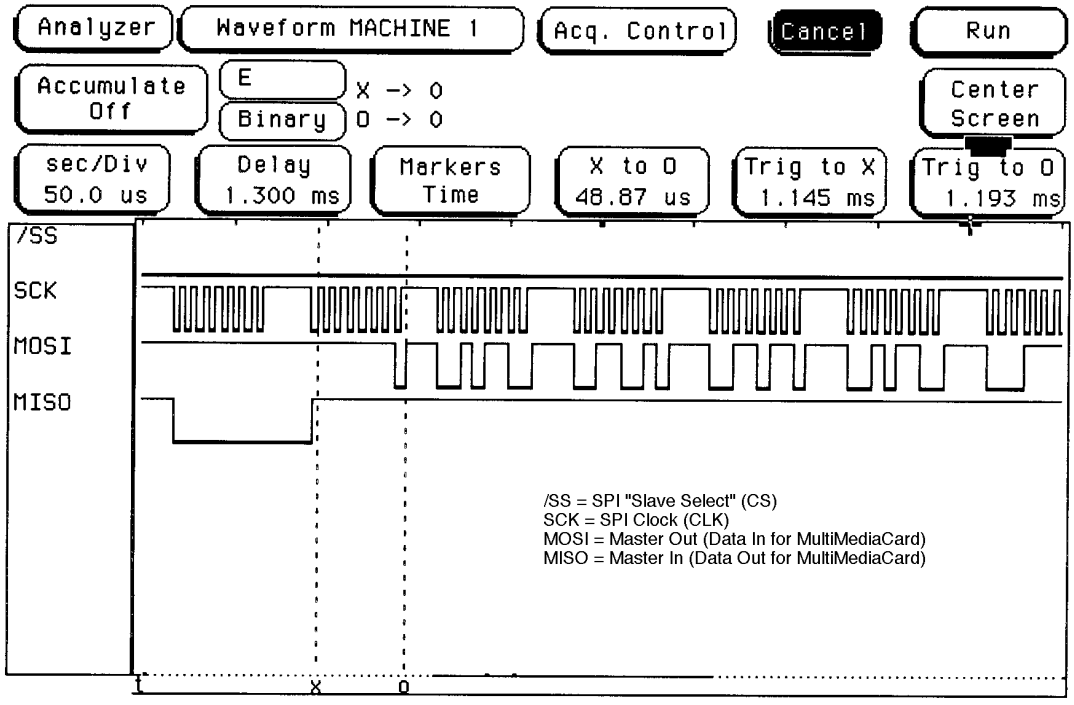


Figure 10-17
MultiMediaCard/SPI Positive Response to "Block Write" Command Plus $FE "Start Byte" to Data

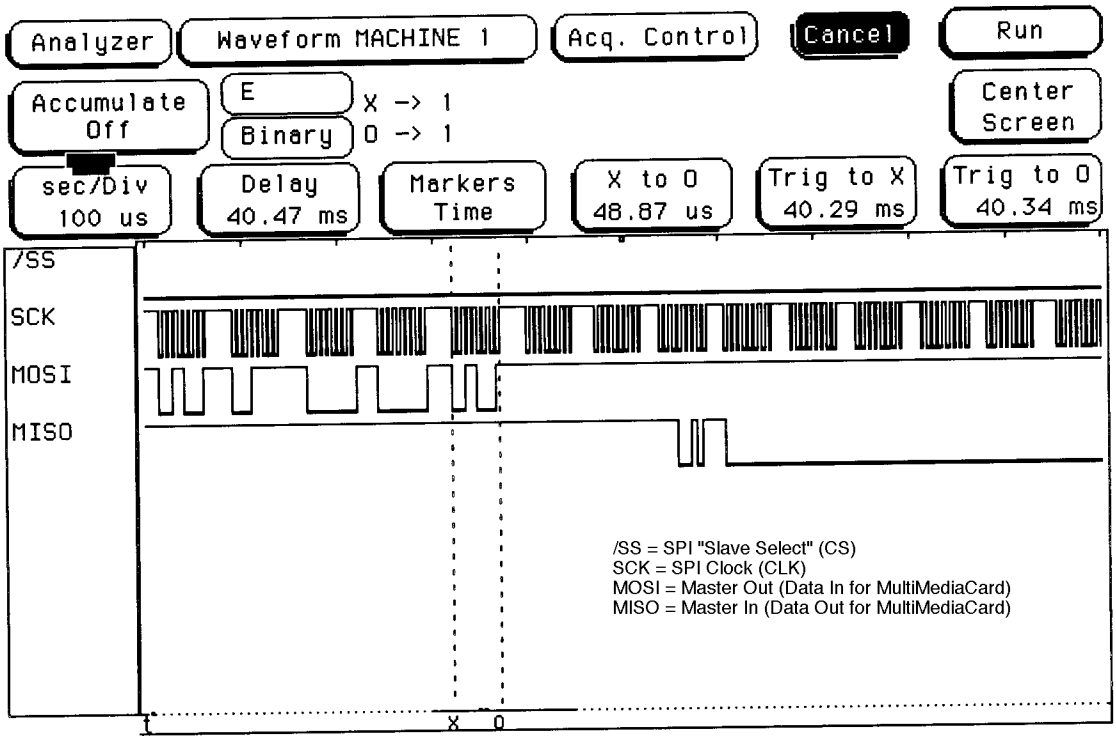


Figure 10-18 MultiMediaCard/SPI "Block Write" End Sequence

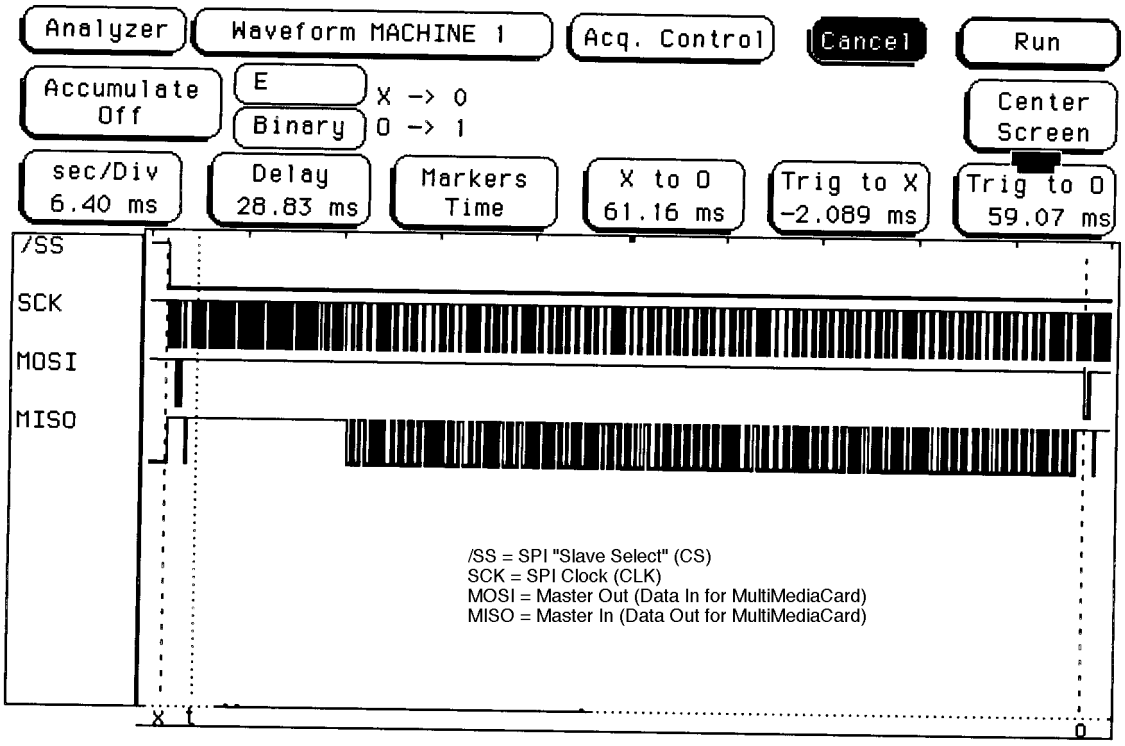


Figure 10-19 MultiMediaCard/SPI "Command 17, Block Read" Sequence

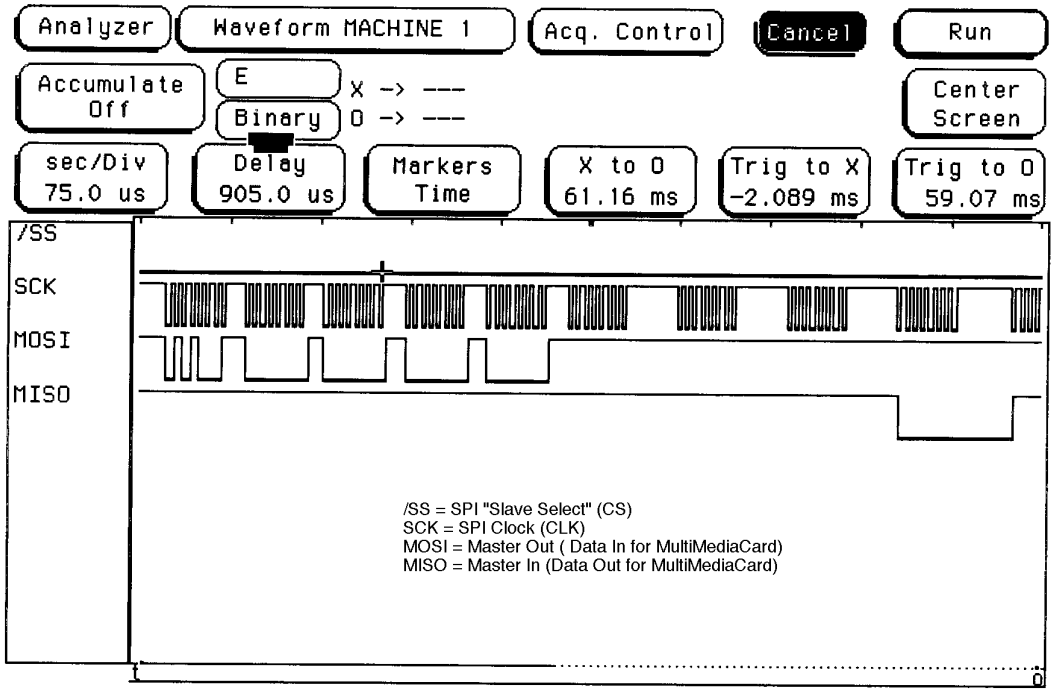


Figure 10-20 MultiMediaCard/SPI "Block Read" Command Plus Positive Response

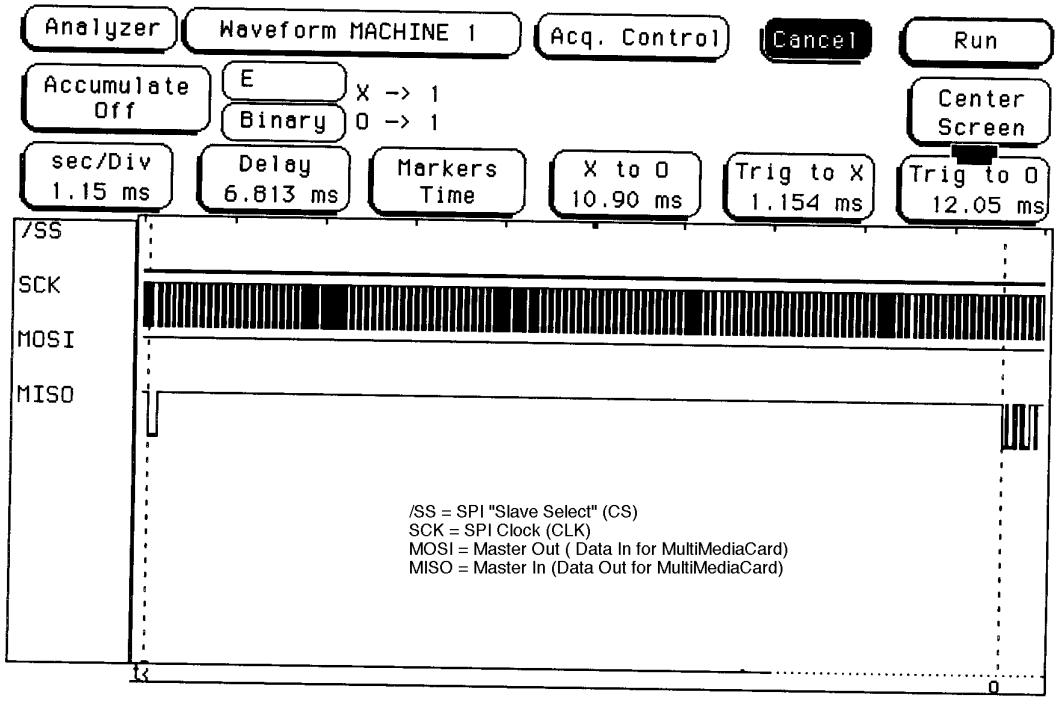


Figure 10-21
MultiMediaCard/SPI Card Positive Response to "Block Read" Command Plus Start of Data

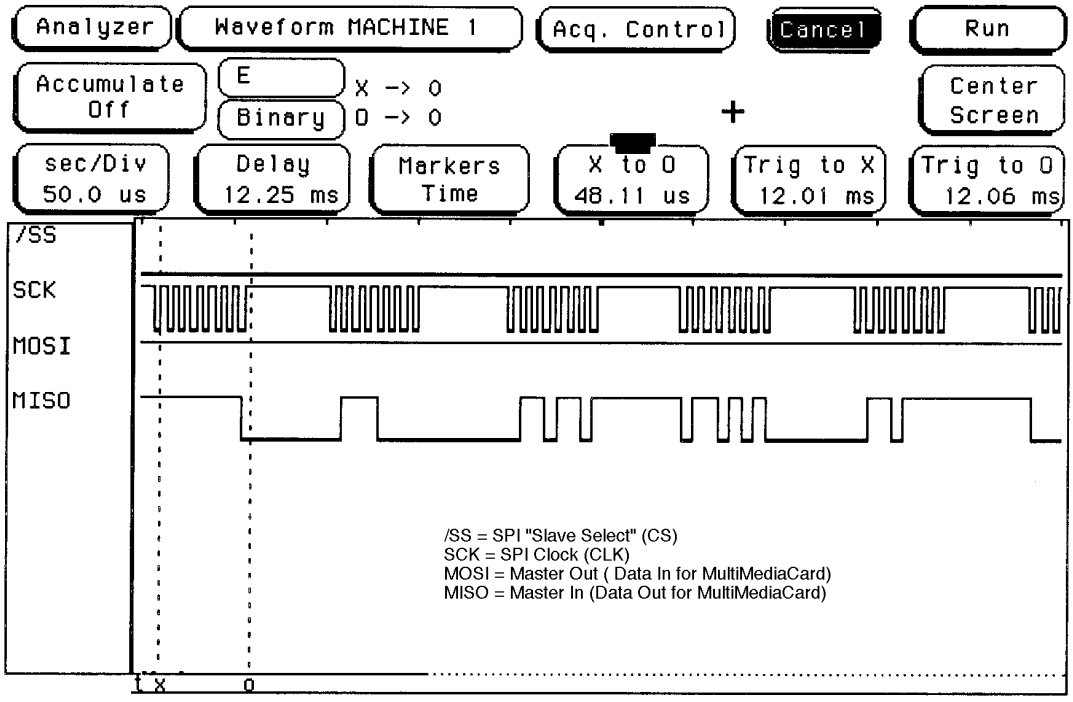


Figure 10-22
MultiMediaCard/SPI Card $FE "Start Byte" Plus First Data to "Block Read" Command

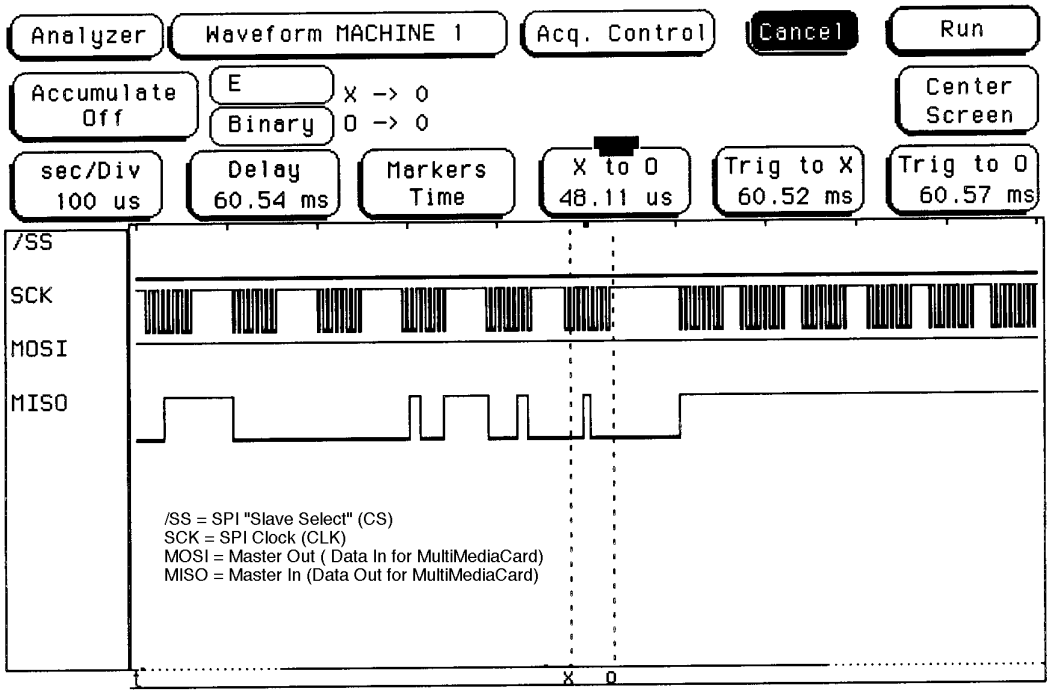


Figure 10-23 MultiMediaCard/SPI "Block End" End Sequence Plus CRC

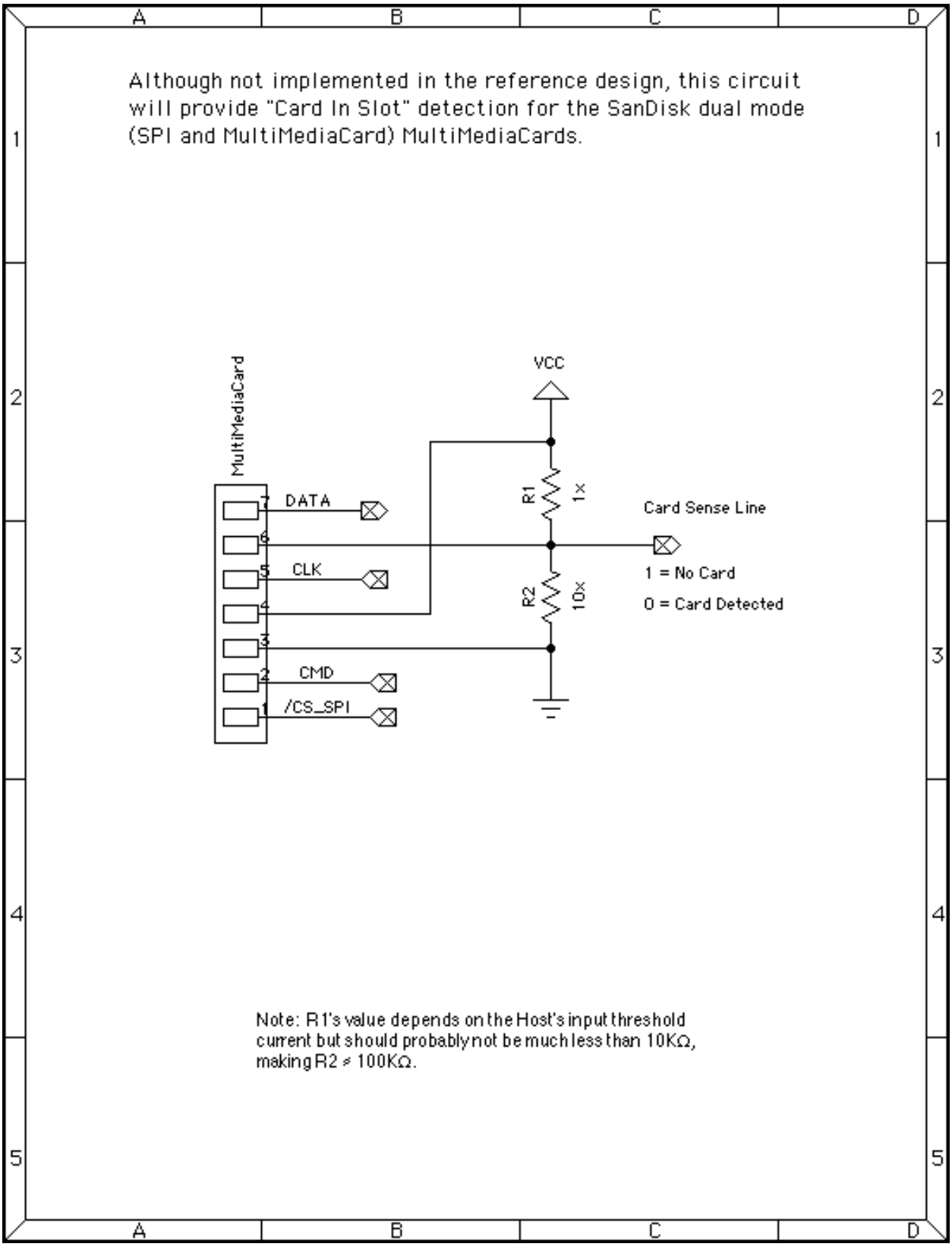


Figure 10-24 Card Detect Circuit