

## 9. Memory Interfacing

### 9.3.2. Timing Diagrams

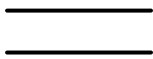
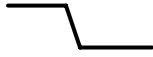
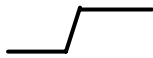

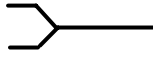
Symbol	Input	Output
	The input must be valid	The output will be valid
	If the input were to fall	Then the output will fall
	If the input were to rise	Then the output will rise
	Don't care, it will work regardless	Don't know, the output value is indeterminate
	Nonsense	High impedance, tristate, HiZ, Not driven, floating

Figure 9.18. Nomenclature for drawing timing diagrams.

### 9.5.4. Motorola MC9S12C32 External Bus Timing

In the *Expanded* modes,

Ports A,B contain address and data bus,

Port E contains the bus control signals, and

Port K contains the XAB19-14 address lines used in paging.

a 20-bit address allows access up to 1 Megabyte

MODC	MODB	MODA	Mode Description	PortA	PortB	MODx write ability
1	0	0	Normal Single Chip	In/Out	In/Out	Write once to Normal Expanded Narrow or Wide
1	0	1	Normal Expanded Narrow	A15-A8/ D7-D0	A7-A0	Cannot change mode
1	1	1	Normal Expanded Wide	A15-A8/ D15-D8	A7-A0/ D7-D0	Cannot change mode

Table 9.14. There are eight execution modes for the MC9S12C32.

Address	Bit 7	6	5	4	3	2	1	Bit 0	Name
\$0000	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	PORTA
\$0001	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0	PORTB
\$0002	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0	DDRA
\$0003	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0	DDRB
\$0008	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0	PORTE
\$0009	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	0	0	DDRE
\$000A	NOACCE	0	PIPOE	<b>NECLK</b>	<b>LSTRE</b>	<b>RDWE</b>	0	0	PEAR
\$000B	<b>MODC</b>	<b>MODB</b>	<b>MODA</b>	0	<b>IVIS</b>	0	EMK	EME	MODE
\$000C	PUPKE	0	0	PUPEE	0	0	PUPBE	PUPAE	PUCR
\$000D	RDPK	0	0	RDPE	0	0	RDPB	RDPA	RDRIV
\$000E	0	0	0	0	0	0	0	ESTR	EBICTL
\$0013	0	0	0	0	<b>EXSTR1</b>	<b>EXSTR0</b>	ROMHM	ROMON	MISC
\$001E	IRQE	IRQEN	0	0	0	0	0	0	IRQCR
\$0032	PK7	PK6	PK5	PK4	PK3	PK2	PK1	PK0	PORTK
\$0033	DDRK7	DDRK6	DDRK5	DDRK4	DDRK3	DDRK2	DDRK1	DDRK0	DDRK
\$0034	0	0	<b>SYN5</b>	<b>SYN4</b>	<b>SYN3</b>	<b>SYN2</b>	<b>SYN1</b>	<b>SYN0</b>	SYNR
\$0033	0	0	0	0	<b>REFDV3</b>	<b>REFDV2</b>	<b>REFDV1</b>	<b>REFDV0</b>	REFDV
\$0037	RTIF	PROF	0	LOCKIF	LOCK	TRACK	SCMIF	SCM	CRGFLG
\$0039	PLLSEL	PSTP	SYSWAI	ROAWAI	PLLWAI	CWAI	RTIWAI	COPWAI	CLKSEL
\$003A	CME	PLLON	AUTO	ACQ	0	PRE	PCE	SCME	PLLCTL

Table 9.15. MC9S12C32 registers used for external memory interfacing.

**IVIS** is the Internal Visibility bit. This bit determines whether internal bus signals can be seen on the external bus during accesses to internal locations. **NECLK** is the No External E Clock bit. If **NECLK**=0, then PE4 is the external E clock pin. The **LSTRE** is the Low Strobe (LSTRB) Enable bit. The signal LSTRB is used to implement 8-bit writes when running in expanded wide mode. **LSTRE**=1 means PE3 is configured as the LSTRB bus control output, otherwise PE3 is a general-purpose I/O pin. **RDWE** is the Read/Write Enable bit. If **RDWE**=1, PE2 is configured as the R/W pin. The R/W signal is 1 during read cycles and 0 during write cycles. If **RDWE**=0, then PE2 a general-purpose I/O pin. The **EBICTL** register contains one bit called **ESTR**, which determines whether the E clock behaves as a simple free-running clock or as a bus control signal that is active only for external bus cycles. We will set **ESTR** to 1 when interfacing external memory so that E stretches high during stretched external accesses and remains low during non-visible internal accesses. If **ESTR** is 0, then E never stretches (always free running). This bit has no effect in single-chip modes. The **EXSTR1 EXSTR0** bits in the **MISC** register specify the amount of stretch for all external memory cycles, as described in Table 9.16 and illustrated in Figures 9.35 through 9.38. While stretching, the CPU state machines are all held in their current state.

EXSTR1	EXSTR0	E clock stretch
0	0	none (default)
0	1	1
1	0	2
1	1	3

Table 9.16. E clock stretching for the external access space.

With no cycle stretching, the external E clock follows the internal E clock.

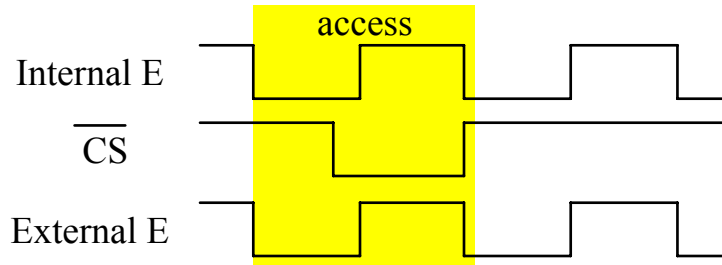


Figure 9.35. CS timing with no cycle stretching.

With 1 cycle stretching, the access cycle is 500 ns long:

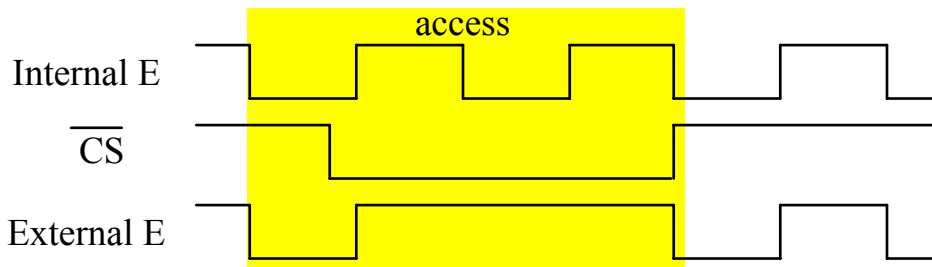


Figure 9.36. CS timing with 1 cycle stretching.

With 2 cycle stretching, the access cycle is 750 ns long:

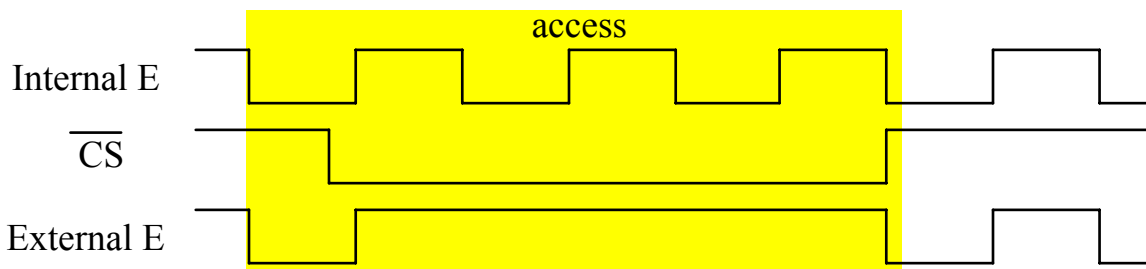


Figure 9.37. CS timing with 2 cycle stretching.

With 3 cycle stretching, the access cycle is 1000 ns long:

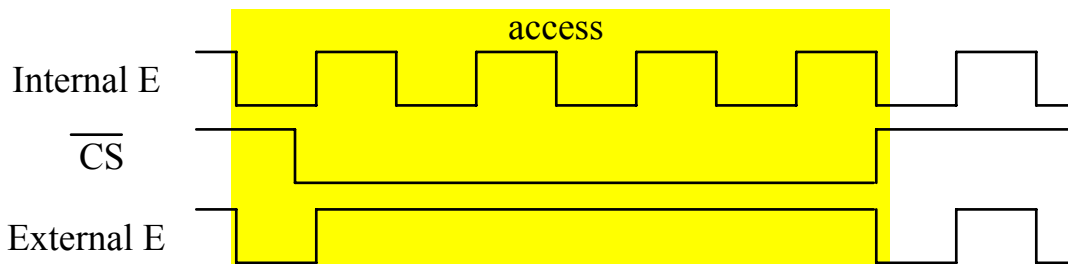


Figure 9.38. CS timing with 3 cycle stretching.

The E clock is a timing signal of the MC9S12C32, meaning the rising and falling edges occur at predictable and precise times.

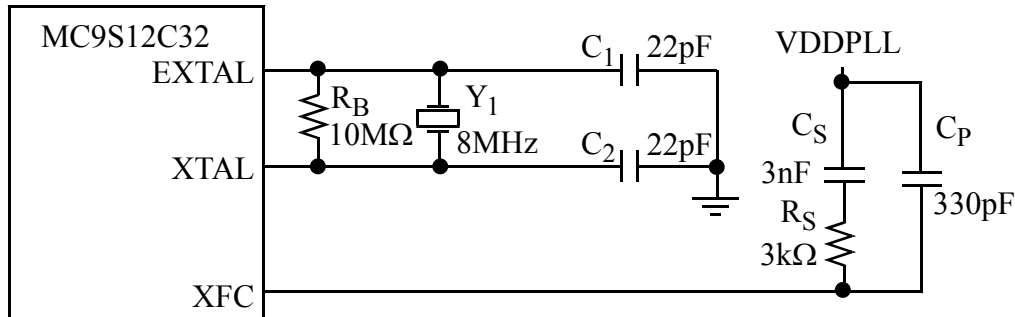


Figure 9.39. MC9S12C32 clock circuit, values taken from the Technological Arts Nanocore12.

**Observation:** Microcontroller data sheets suggest a PCB layout pattern for interfacing the crystal.

Figure 9.40 draws a simplified timing diagram, showing both the read and write timing. The bus cycle time is  $t_1$ . For example, the 8 MHz crystal (period=125ns) in Figure 9.39 will create a  $t_1$  of 250ns without cycle stretching. Let OSCCLK be the frequency of the crystal, and let PLLCLK be the frequency of the PLL. The SYNR and REFDV registers determine the PLL frequency,

$$PLLCLK = 2 * OSCCLK * \frac{(SYNR + 1)}{(REFDV + 1)}$$

The CLKSEL register contains the PLL Select Bit, PLLSEL. When PLLSEL is 1, the system clocks are derived from PLLCLK (bus clock frequency is PLLCLK/2). When PLLSEL is 0, the system clocks are derived from the crystal oscillator (bus clock frequency is OSCCLK/2). Let  $t_{cyc}$  be the period of the bus clock.

$$t_{cyc} = \frac{2}{OSCCLK} \text{ (if PLLSEL = 0) } \quad \text{or} \quad t_{cyc} = \frac{2}{PLLCLK} \text{ (if PLLSEL = 1)}$$

We can execute software that activates the PLL, creating a bus cycle time as short as 40ns (25MHz). Program 9.1 shows the sequence of steps required to engage the PLL. First, it sets the SYNR and REFDV registers to specify the PLL frequency. Next, it sets the PLLCTL register. The Clock Monitor Enable Bit (CME) is set to enable the clock monitor, so that slow or stopped clocks will cause a clock monitor reset sequence. The Phase Lock Loop On Bit (PLLON) is set to turn on the PLL. The Acquisition Bit (ACQ) is set to select the high bandwidth filter. The Self Clock Mode Enable Bit (SCME) is set so the detection of crystal clock failure forces the MCU in Self Clock Mode.

```
void PLL_Init(void){
    CLKSEL = 0x00;           // make sure PLL is deselected
    SYNR = 24; REFDV = 7;   // PLLCLK=2*OSCCLK*(SYNR+1)/(REFDV+1)
    PLLCTL = 0xD1;         // Turn on PLL
    while((CRGFLG&0x08) == 0){ // Wait for PLLCLK to stabilize.
        CLKSEL |= 0x80;     // Switch to PLL clock
    }
}
```

Program 9.1. MC9S12C32 code to change the E clock from 8 to 25 MHz.

**Checkpoint 9.4:** Modify program 9.1 to run at 24MHz.

When stretching the E clock, confusion sometimes arises. Changing the bus cycle time with the PLL will modify both halves of the bus cycle  $t_3$  and  $t_4$ . On the other hand, when a bus cycle is stretched, times  $t_1$ ,  $t_4$  are increased, while  $t_3$  is fixed. The rising edge of the E clock occurs at the same time regardless of stretching, which will be at  $\frac{1}{2}t_{cyc}$ . Notice in Figures 9.35 through 9.38 that there is a stretched E clock and an unstretched E clock. The

clocks used by SPI, SCI, PulseAcc, PWM, and TCNT are derived from the unstretched E clock. External memory accesses use the stretched E clock. In other words, the only activity affected by cycle stretching is the time for external memory accesses and hence the execution of the instructions causing those accesses. Internal memory accesses, the timer and I/O functions are not affected by cycle stretching. In Figure 9.40, the time delays with numerical values (e.g., 7ns, 11ns, etc.) are parameters of the MC9S12C32 and not affected by the PLL or cycle stretching. Let  $n$  be the number of stretches (0, 1, 2, or 3) as specified by the **EXSTR1 EXSTR0** bits. When interfacing external memory, we need to know when the E clock rises and when it falls.

$$\uparrow E = \frac{1}{2}t_{\text{cyc}}$$

$$\downarrow E = t_1 = (n+1) * t_{\text{cyc}}$$

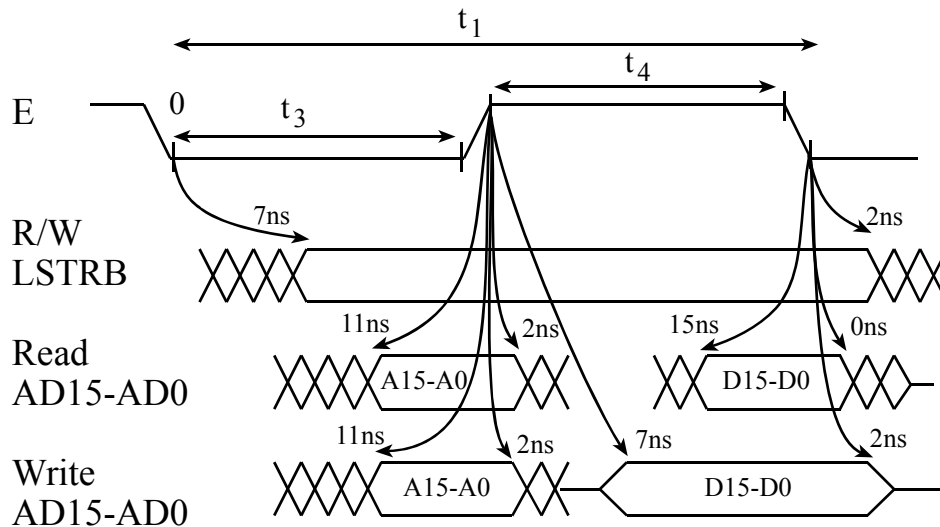


Figure 9.40. Simplified bus timing for the MC9S12C32 in expanded mode.

**Observation:** Most of the activities (such as executing instructions, memory access, TCNT, SCI, SPI) are affected by the PLL. Conversely, RTI interrupts always operate using the oscillator crystal.

In expanded narrow mode, the data bus is only 8 bits (on pins PA7-PA0), and the low address is available throughout the cycle (on pins PB7-PB0). In narrow mode, we need only one octal latch to capture the A15-A8 address, as shown in Figure 9.41. In expanded wide mode, the data bus is 16 bits, and we need two octal latches to capture the A15-A0 address from PA7-PA0 on the rising edge of E. The 74FCT374 octal flip flop is chosen because of its speed and the fact that it captures on the rising edge of the clock. The setup and hold times for the 74FCT374 are 2ns and 1.5 ns respectively. In Figure 9.40, we see the 6812 maintains the address 11ns before and 2 ns after the rise of E, so the setup and hold times of the octal D flip-flop are satisfied. The propagation delay from clock to output valid is [2,6.5] ns.

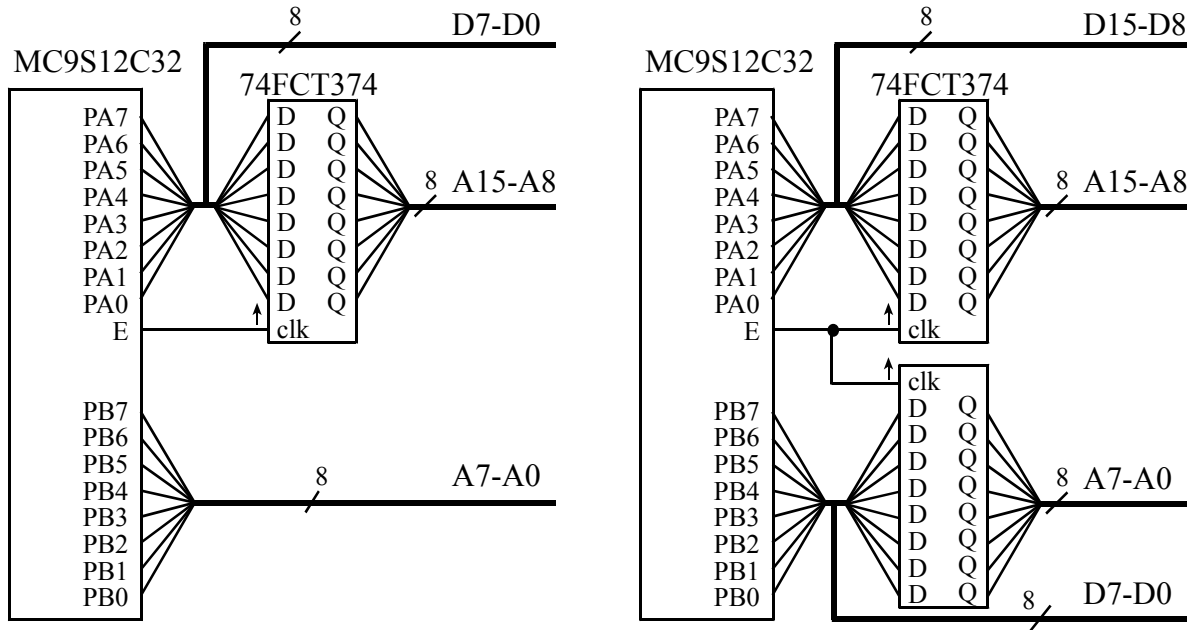


Figure 9.41. Address latch for the MC9S12C32 in expanded narrow and wide modes.

There are three important timing intervals we must determine from the microcomputer timing diagram. The first is the address available, AA, interval. This interval defines when during the cycle the address is valid. In particular, AdV is the time when the address lines A15-A0 are valid, and AdN is the time when the address lines are no longer valid. In order to calculate when the address is available, we first consider the 74FCT374 octal D flips used to capture the address. In expanded wide mode, all 16 bits of the address are clocked into the 74FCT374 on the rising edge of E, so the address is available during the second half of the cycle and continues to be valid until the next rising edge of E. The +[2,6.5] occurs because of the delay in the 74FCT374's.

$$AA = (AdV, AdN) = (\frac{1}{2}t_{cyc} + [2,6.5], t_1 + \frac{1}{2}t_{cyc})$$

In expanded narrow mode the least significant address lines are not latched, so AdV is determined by the latched address and AdN is determined by the unlatched address.

$$AA = (AdV, AdN) = (\frac{1}{2}t_{cyc} + [2,6.5], t_1 + 2)$$

The second important timing interval is read data required, RDR. During a read cycle the data is required by the MC9S12C32. Thus to determine the data required interval, we look in the MC9S12C32 data sheet. For data required, the worst case is the longest interval.

$$RDR = \text{Read Data Required} = (t_1 - 15, t_1)$$

The last important timing interval we get from the microcomputer is write data available, WDA. During a write cycle, the data are supplied by the MC9S12C32. Thus, to determine the data available interval we look in the MC9S12C32 data sheet. We will specify the worst case timing. For write data available, the worst case is the shortest interval.

$$WDA = \text{Write Data Available} = (\frac{1}{2}t_{cyc} + 7, t_1 + 2)$$

**Observation:** The speed of CMOS logic is strongly dependent on capacitive load. The 9S12C32 and 74FCT374 timings are specified for capacitive loads of 50pF. If the actual load is larger than 50pF, the timing will be significantly slower.

Table 9.17 presents these three intervals calculated for a 250ns cycle time with 0,1,2,3 stretches. The address times are given for expanded narrow mode with the shortest interval.

n	0	1	2	3
$\uparrow E = \frac{1}{2}t_{cyc}$	125	125	125	125
$\downarrow E = t_1$	250	500	750	1000
AA	(131.5, 252)	(131.5, 502)	(131.5, 752)	(131.5, 1002)
RDR	(235, 250)	(485, 500)	(735, 750)	(985, 1000)
WDA	(132, 252)	(132, 502)	(132, 752)	(132, 1002)

Table 9.17. Timing intervals for the MC9S12C32 with a 4 MHz clock.

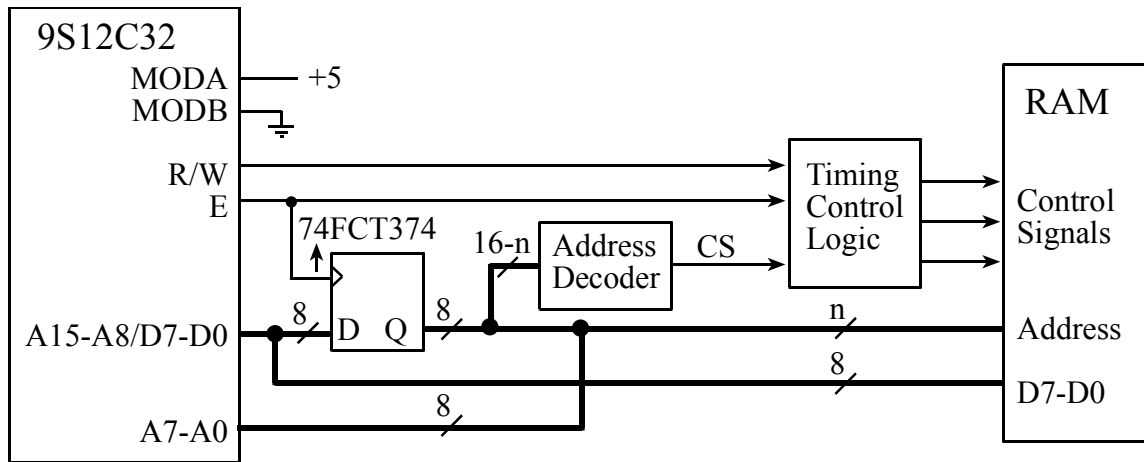


Figure 9.44. General approach to memory interfacing on a MC9S12C32 in narrow expanded mode.

### 9.7.2.3. 8K RAM Interface to a MC9S12C32

The design goal of this section is to interface an 8K external RAM to the MC9S12C32, minimizing cost. The least expensive way to interface external RAM to the MC9S12C32 is to use expanded narrow mode. We will place the 8K RAM at \$8000-\$9FFF, because there are no internal devices at these addresses. The address map of our system will be

- \$0000-\$03FF I/O ports
- \$3800-\$3FFF Internal RAM
- \$4000-\$7FFF Internal EEPROM
- \$8000-\$9FFF External RAM
- \$C000-\$FFFF Internal EEPROM

First, we design a minimal cost address decoder for \$8000 to \$9FFF using a 74FCT139. We need addresses A15 and A14 to differentiate our external RAM from the other three devices. The positive logic chip select will be  $A15 \cdot \overline{A14}$ . Just like the MC68HC812A4 interfaces earlier, we must select the proper amount of cycle stretching. Recall that for a MC9S12C32 running at 4 MHz  $\uparrow E$  will be 125ns,  $\downarrow E$  will be  $(n+1) \cdot 250ns$  where  $n$  is the number of stretches. The propagation delay from clock to output change through the 74FCT374 has a minimum of 2ns and a maximum of 6.5ns. In expanded narrow mode, the address available interval begins with the high address being clocked into the 74FCT374 and ends with the low address end time, which is 2ns after  $\downarrow E$ . Thus,

$$AA = (AdV, AdN) = (125+[2,6.5], \downarrow E+2) = ([127,131.5], \downarrow E+2)$$

The high address, which is used by the address decoder, is available all the way through until the next rise of  $E$ .

$$AA_{15-8} = ([127,131.5], \downarrow E+125)$$

To guarantee proper timing on the read and write cycles, we have three options. We must synchronize E2,  $\overline{E1}$ , or both  $\overline{G}$  and  $\overline{W}$ . The RAM has both a positive logic (E2) and a negative logic ( $\overline{E1}$ ) chip select. Since we are going to use the positive logic chip select for the timing, we will use the negative logic chip select for the address decoder. We choose to synchronize E2 because it is fastest (connected directly to E). In summary, we will have

- E2        positive logic synchronized to E
- $\overline{E1}$     negative logic unsynchronized address decoder
- $\overline{G}$        negative logic unsynchronized
- $\overline{W}$        negative logic unsynchronized

With time-multiplexed signals, we have to be careful to avoid address/data collisions during a read cycle. All read and write timing will be controlled by the E clock (without any gate delays) by connecting the E directly to E2. In this way, the memory data output will not collide with the microcomputer address output during a read cycle when E=0. Assuming 9 ns 74FCT139 gate delay max and 1.5 ns delay minimum,  $\overline{E1}$  falls at  $[127,131.5]+[1.5,9]$  ns and rises at  $\downarrow E+125+[1.5,9]$  ns. The worst case timing is the latest (maximum=140.5) time for  $\downarrow \overline{E1}$  and the earliest (minimum= $\downarrow E+126.5$ ) time for  $\uparrow \overline{E1}$ .  $\overline{G}$  will be grounded, so it is removed from the timing equation. Entering this information into the memory timing

$$\begin{aligned} \text{Read Data Available} &= ( \text{later} ( \text{AdV}+t_{AVQV} , \downarrow \overline{E1} + t_{E1LQV} , \uparrow E2 + t_{E2HQV} , \downarrow \overline{G} + t_{GLQV} ) , \\ &\text{earlier} ( \text{AdN}+t_{AXQX} , \uparrow \overline{E1} + t_{E1HQZ} , \downarrow E2 + t_{E2LQZ} , \uparrow \overline{G} + t_{GHQZ} ) ) \\ &= ( \text{later} (131.5+150, 140.5+150, 125+150) , \text{earlier} ( \downarrow E+2+20, \downarrow E+126.5, \downarrow E ) ) \end{aligned}$$

During a read cycle the data is required by the 6812. Thus to determine the read data required interval, we look in the 6812 data sheet. For read data required, the worst case is the longest interval. Recall that

$$\text{RDR} = \text{Read Data Required} = (\downarrow E - 15, \downarrow E)$$

We must choose the number of cycle stretches to make the read data available interval overlap the read data required interval. We must make  $\overline{W}=1$  during a read cycle. Thus,

Address	$131.5+150 \leq \downarrow E - 15$	and	$\downarrow E+2+20 \geq \downarrow E$
$\overline{E1}$	$140.5+150 \leq \downarrow E - 15$	and	$\downarrow E+126.5 \geq \downarrow E$
E2	$125+150 \leq \downarrow E - 15$	and	$\downarrow E \geq \downarrow E$

The  $\overline{E1}$  timing ( $305.5 \leq \downarrow E$ ) tells us that 1 extra cycle is needed to stretch the access time to 500ns. If the read data available did not overlap the read data required, then we would have to:

- increase the number of cycle stretches; or
- slow down the 6812 by increasing the E period; or
- decrease  $t_{E1LQV}$  by spending more money on a faster RAM chip.

The beginning of RDA is determined by  $\downarrow \overline{E1}$  and the end by  $\downarrow E2$

$$\text{Read Data Available} = ( \downarrow \overline{E1} + 150, \downarrow E2 ) = (290.5, 500).$$

Data is written into the memory when E2=1,  $\overline{E1}=0$ , and  $\overline{W}=0$ . During a write cycle, the data are supplied by the 6812. For write data available, the worst case is the shortest interval. Recall that

$$\text{WDA} = \text{Write Data Available} = ( \frac{1}{2}t_{\text{cyc}} + 7, \downarrow E+2 ) = (132, \downarrow E+2)$$

Since we have chosen to synchronize E2 to the E clock,

$$\text{Write Data Required} = (\downarrow E - 60, \downarrow E)$$

The number of cycle stretches will also affect whether or not the write data available interval overlaps the write data required interval. The worst case delay selects the largest WDR interval

$$132 \leq \downarrow E - 60 \quad \text{and} \quad \downarrow E \leq \downarrow E + 2$$

Thus,

$$192 \leq \downarrow E$$

Therefore the 1-cycle stretch needed for the read cycle also is also sufficient for the write cycle. Although, the write cycle could have operated with no stretches, we have to choose 1-stretch so both read and write timing is satisfied.

We combine the command, and timing aspects of the design to find the boolean expression that will generate the control signals: E2,  $\overline{E1}$ ,  $\overline{G}$ , and  $\overline{W}$ .

*Common Error: A capacitive load occurs both by the physical layout of the PCB as well as with each input pin the output must drive. The 9S12C32 and 74FCT374 timings are specified for capacitive loads of 50pF. If the actual load is larger than 50pF, the timing will be significantly slower.*

E	R/W	A15,A14	RD	WR	E2	$\overline{E1}$	$\overline{G}$	$\overline{W}$	
0	0	00 01 or 11	0	0	X	1	X	X	Address not on the chip
1	0	00 01 or 11	0	0	X	1	X	X	
0	1	00 01 or 11	0	0	X	1	X	X	
1	1	00 01 or 11	0	0	X	1	X	X	
0	0	10	0	0	0	0	X	0	Write cycle
1	0	10	0	1	1	0	X	0	
0	1	10	0	0	0	0	X	1	Read cycle
1	1	10	1	0	1	0	0	1	

Table 9.27. Combined timing table for the MCM60L64 RAM interface.

$$E2 = E$$

$$\overline{E1} = A15 \bullet \overline{A14}$$

$$\overline{G} = 0$$

$$\overline{W} = R/W$$

The next step is to build the interface. MODC, MODB, MODA are set to start the system in Special Single Chip mode.

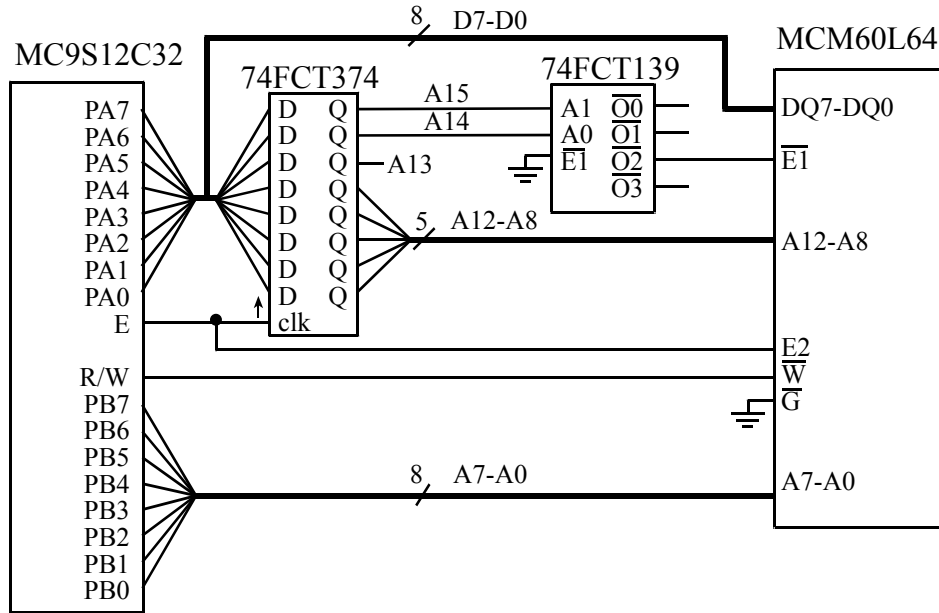


Figure 9.63. Interface between the MC9S12C32 and the 60L64 8K RAM chip.

To verify proper read cycle timing we draw the *Combined Read Cycle Timing Diagram*. We need to verify that read data available overlaps read data required before we build the design.

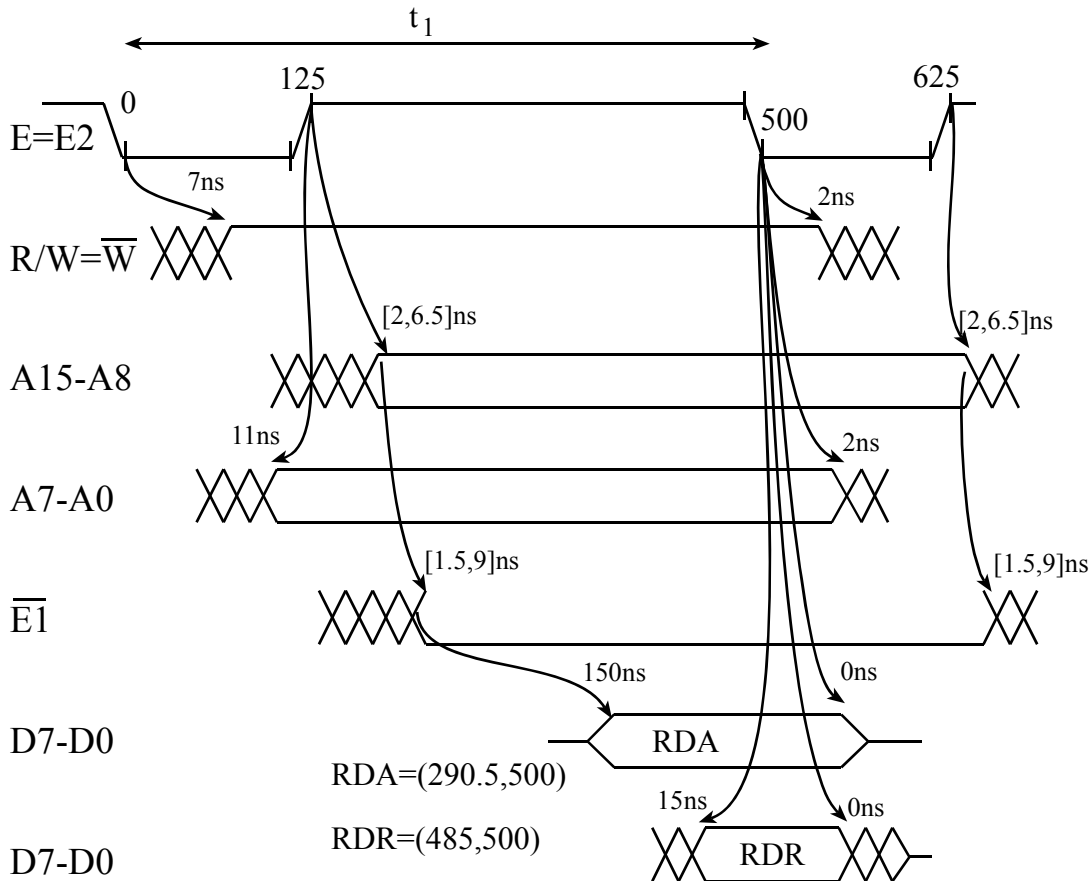


Figure 9.64. Read timing of the MC9S12C32 and the 60L64 8K RAM chip.

Similarly, to verify proper write cycle timing we draw the *Combined Write Cycle Timing Diagram*

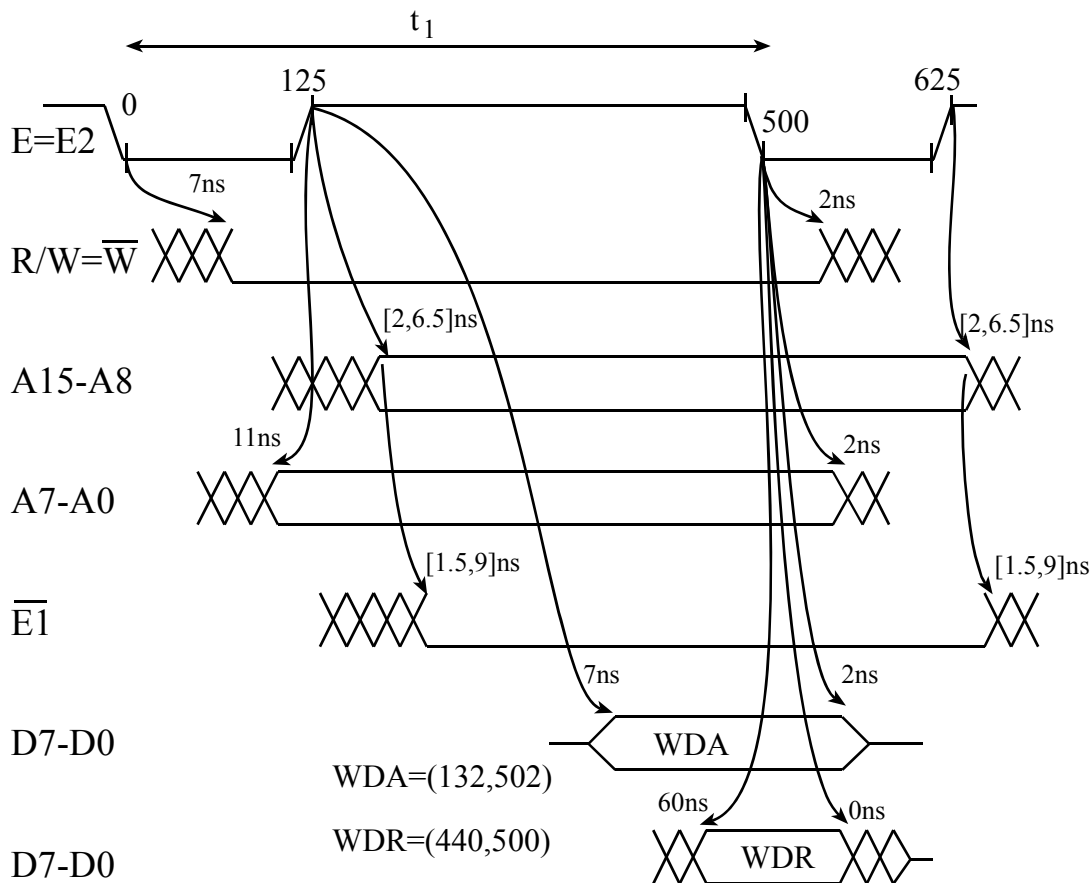


Figure 9.65. Write timing of the MC9S12C32 and the 60L64 8K RAM chip.

Notice that write data available overlaps write data required. After reset, the MC9S12C32 will be running in Special Single Chip mode. The compiler generates initialization code to place the RAM at \$3800 to \$3FFF. Program 9.4 will change the mode from Special Single Chip to Normal Expanded Narrow. Internal visibility is turned off to simplify debugging (IVIS=0). The PEAR register is set to enable the R/W and E clock.

```
void RAM_Init(void){
    MODE = 0xA0;           // normal expanded narrow mode
    MISC = (MISC&0xF3)|0x04; // 1-cycle stretch on external
    PEAR = 0x0C;          // enable E, R/W, LSTRB(NA)
}
```

Program 9.4. Software to configure the MC9S12C32 for the external RAM.