Lab 7: Timing and Control in the M68HC11 EVB

1. Introduction

The system clock synchronizes all of the activity of the 68HC11 microcontroller. During each clock cycle the microcontroller does a specific operation. This can be either the transfer of one byte of data, transfer of an address, or some kind of computation. The nature of the operation depends on the instruction executed and the other information included as part of the instruction. This other information can include data, immediate addresses, extended addresses, branch offsets, etc. Each instruction contains a fixed number of bytes and requires a specific number of cycles. In this lab, we study the cycle-by-cycle operations that take place during the execution of instructions.

2. Analysis of a Simple Program

This section looks at the following very simple program:

Program 1

Address	Mnemonic	Operand
`$C010`	`LDAA`	`#$30`
`$C012`	`BRA`	`$C010`
`$C014`	`SWI`	

This program is an infinite loop, hence the instructions `LDAA` and `BRA` are repeated continuously. The cycle-by-cycle operations for this program are summarized in Table 1. There are two important points. In its second cycle, the `BRA` instruction puts the offset `rr` on the data bus. The offset is calculated according to the equation:

Desired Destination Address = Origin Address + 0002 + offset

As you already know, this offset calculation is done for you automatically when you enter the code `BRA $C010` into the assembler. Knowing how the calculation is made allows you to verify the offset calculation by examining the memory location corresponding to `rr` (in this case $C013). The second point to observe is that during the third cycle of the `BRA` instruction, the address bus puts out $FFFF while, at the same time, the data bus is "unknown". This means that any hex number 00-FF can be present on the data bus at that time.

Figure 1 illustrates these changes. Note the plots of the data lines are not exactly equal to the binary digits in the last columns of Table 1 for Opcode/Data. This is because when the 68HC11 is operating in expanded mode, the address values A0-A7 and the data values D0-D7 are shared on the same lines by *time multiplexing*. During the first half of an E-clock cycle, the multiplexed lines carry the address values A0-A7. You can see this in Figure 1: when the E-clock is high, D0 is the same as A0, D1 is the same A1, and D2 is the same A2. During the second half of the E-clock cycle, the multiplexed lines carry the data values D0-D7. Look again at Figure 1: when the E-clock is low, lines D0-D2 are the same as the last three bits in the "Opcode/Data in Binary" column.

Program 1				
Instruction	Address	Address in Binary	Opcode/Data	Opcode/Data in Binary
LDAA	$C010	1100 0000 0001 0000	$86	1000 0110
	$C011	1100 0000 0001 0001	$30	0011 0000
BRA	$C012	1100 0000 0001 0010	$20	0010 0000
	$C013	1100 0000 0001 0011	$FC	1111 1100
	$FFFF	1111 1111 1111 1111	–	xxxx xxxx

Table 1: Address bus & data bus changes during each cycle

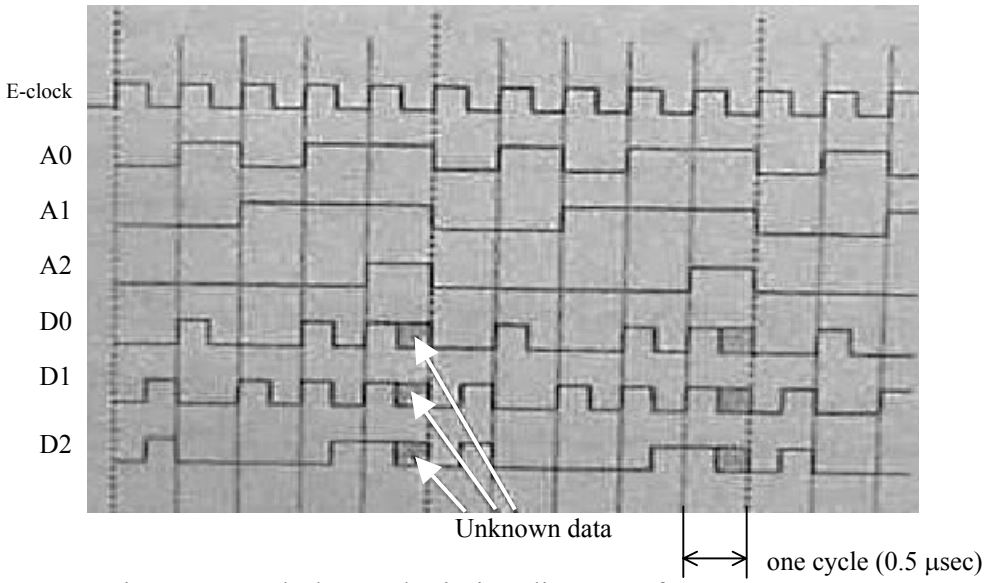

Figure 1: Cycle-by-cycle timing diagram of Program 1

The goal for this portion of the lab is to verify the diagram of Figure 1 for Program 1. To do so you need to attach the oscilloscope leads to different pins on the components on the EVB.

Prelab: Download the data sheet for the HM62256 32K x 8-bit CMOS static RAM chip from the course web page. Determine the correct pins you will need to verify the timing diagrams for lines A0, A1, A2, D0, D1, D2, and the E-clock. Note that the RAM chip data sheet shows "I/O0" and "I/O1" for data lines D0 and D1, respectively.

Lab work: Before powering up the EVB, place a "Chip Clip" on the 62256 RAM chip (the U4 chip on the EVB). Set the oscilloscope's trigger source to Channel 2, and then connect the Channel 2 probe to observe address line A2. Use the Channel 1 probe to measure the required signals while Channel 2 shows the A2 signal. Turn on the EVB and connect the oscilloscope leads, making sure not to short the scope leads across pins! Then enter the program and run it. Make a timing diagram of the observed signals A0, A1, A2, D0, D1, D2, and the system clock. Your diagram will be similar to – but not exactly like – Figure 1.

Lab Report: Copy the timing diagram, summarize how you made the measurements, and describe any problems encountered. Verify that the signals on the data lines during the first half of the E-clock cycles are the same as the address signals.

3. Analysis of a Program Involving Store Operations

Program 2 incorporates the `STAA` instruction. This uses the R/$\overline{W}$ line of the 68HC11 to control what is read from and written to external memory. You can observe this on pin 27 of IC U4 (the $\overline{WE}$ line on the static RAM chip).

Program 2

```
Addr    Label    Opcode    Operand    Comments
C010    Loop     LDAA      #$03       ; Get number 3
                 STAA      $02        ; Write the number into result address
                 BRA       Loop       ; Keep doing it forever
```

Prelab: Make a table for Program 2 in the same format as Table 1.

Lab work: Assemble Program 2 and make the measurements necessary to draw a timing diagram that shows the E-clock, address signals A0, A1, A2, data signals D0, D1, D2, and the R/$\overline{W}$ line.

Lab report: Show the table and the timing diagram in your report. How do they compare? Identify the specific cycle where the `STAA` instruction writes to memory address `$0002`. How are the address signals A0-A7 and data signals D0-D7 time multiplexed? Why do the data signals not appear on the address lines?