ECE 362 Lab Verification / Evaluation Form

Experiment 5

Evaluation:

IMPORTANT! You must complete this experiment during your scheduled lab period. All work for this experiment must be demonstrated to and verified by your lab instructor *before the end* of your scheduled lab period.

STEP	DESCRIPTION	MAX	SCORE
Step 1	Quantifying Software Delay Error	5	
Step 2	Observing Bus Signals and Identifying Machine Cycles	14	
Step 3	Measuring Bus Signal Timing Parameters	3	
Step 4	Thought Questions	3	
	TOTAL	25	

Signature of Evaluator: _____

Academic Honesty Statement:

IMPORTANT! Please carefully read and sign the Academic Honesty Statement, below. ***You will not receive credit for this lab experiment unless this statement is signed in the presence of your lab instructor.***

"In signing this statement, I hereby certify that the work on this experiment is my own and that I have not copied the work of any other student (past or present) while completing it. I understand that if I fail to honor this agreement, I will receive a score of ZERO and be subject to possible disciplinary action."

Printed Name: _____ Class No. __ __ __ __ - __

Signature: _____ Date: _____

Experiment 5: Microprocessor Bus Timing Analysis

Students will work in teams of two to complete the in-lab portion of this experiment – all steps must be completed during scheduled lab period

Instructional Objectives:

- To learn how a mixed signal oscilloscope (MSO) can be used for logic and timing analysis
- To investigate delay routines, and the actual associated timing
- To observe and analyze microcontroller instruction timing on a machine cycle level

References:

- *MC9S12C Family Data Sheet,* pp. 123-125 – on 9S12C References page
- *Multiplexed External Bus Interface (MEBI) Block User Guide* – on 9S12C References page
- *HCS12 External Bus Design* (AN2287) – on Homework page
- *Examples of External Bus Design* (AN2408) – on Homework page
- *Lecture Module 2-B* – on Notes page
- *Agilent InfiniiVision 3000 X-Series Oscilloscopes User Manual* – on Lab Experiments page

Prelab Preparation:

- Read the description of the lab exercises thoroughly
- Carefully review Lecture Module 2-B

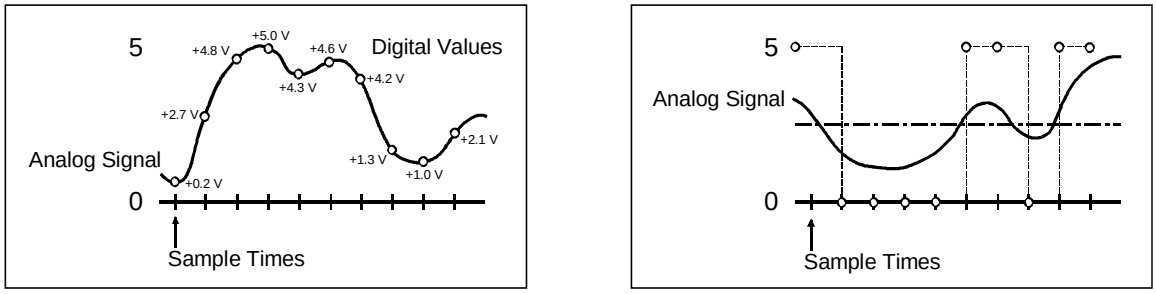
Introduction

Digital circuits are great…when they work. But we all know that they usually don't work the first time, and we must therefore have a development tool to use for designing and debugging digital circuits. Logic analyzers have long been the tool of choice, allowing us to see the logical level of signals at discrete points in time.

In this experiment we will learn how to use the digital oscilloscopes with built-in logic analyzers ("mixed signal oscilloscopes" or MSOs) available in lab. We will start by "instrumenting" a simple software delay routine. This will provide an opportunity to observe the accuracy with which timing intervals can be measured with the digital oscilloscope. We will then use the built-in 16-channel logic analyzer in conjunction with the digital oscilloscope to examine HCS12 multiplexed bus transactions. This will provide an opportunity to observe the differences between read and write cycles as well as the pipelined nature of the processor.

Measurement of Logic Signals

We will begin by investigating how a "mixed signal" oscilloscope (MSO) works. The MSOs in lab combine a 4-channel digital oscilloscope (DSO) with a 16-channel logic analyzer (LA). The DSO samples the input signal at a rate up to 1 GSa/s (billion samples per second) and displays the actual (sampled) waveform. An oscilloscope's bandwidth is typically described as the lowest frequency at which input signal sine waves are attenuated by 3 dB (-30% amplitude error). The bandwidth of the MSOs in lab is 100 MHz. The LA also samples the input signals, but for each sample determines whether the input voltage corresponds to logic "0" or logic "1", when compared against the threshold level for the selected logic family (i.e., TTL, CMOS, etc.). Thus the LA is used to show the logical state of a signal at discrete points in time and the DSO is used to show the actual voltage levels of a signal at discrete points in time (see Figure 1).



Digital Oscilloscope Sampled Signal

Logic Analyzer Sampled Signal

Figure 1. DSO vs. LA Interpretation of Input Signals

NOTE: The in-lab steps of this experiment which follow should only be completed with a lab partner during your scheduled lab period. Do not attempt to complete these steps during the evening office hours or at any time other than your scheduled lab period. Failure to follow these instructions will result in a score of zero for this experiment.

Step 0: Station Setup Verification

Before starting the in-lab portion of this experiment, check to make sure the 9S12C128 mircontroller board (supplied) and MSO are connected as detailed in Table 1. Also, make sure the DSO 'scope probes and the LA pods are grounded appropriately. Both lab partners should check (double-check) the connections before proceeding.

Table 1. Microcontroller-MSO Connections

Signal Name	*9S12C128 Pin*	*MSO Input*
ECLK	PE-4	DSO-1
R/W′	PE-2	DSO-2
LSTRB′	PE-3	DSO-3
Port B Pin 0	PB-0	DSO-4 and LA-0
Port B Pin 1	PB-1	LA-1
Port B Pin 2	PB-2	LA-2
Port B Pin 3	PB-3	LA-3
Port B Pin 4	PB-4	LA-4
Port B Pin 5	PB-5	LA-5
Port B Pin 6	PB-6	LA-6
Port B Pin 7	PB-7	LA-7
Port A Pin 0	PA-0	LA-8
Port A Pin 1	PA-1	LA-9
Port A Pin 2	PA-2	LA-10
Port A Pin 3	PA-3	LA-11
Port A Pin 4	PA-4	LA-12
Port A Pin 5	PA-5	LA-13
Port A Pin 6	PA-6	LA-14
Port A Pin 7	PA-7	LA-15

Step 1. Quantifying Software Delay Error

Download and unzip the project **Lab5-Step1** from the course website. Open **`main.asm`** in Code Warrior and examine the program. Note that it uses the delay subroutine described in Lecture Module 1-C to "toggle" the state of `Pin 0` of `Port B` (`PB-0`) every `NMS` milliseconds, thus producing a square wave with a period of `2*NMS` milliseconds (or, a frequency of 1/(`2*NMS`)).

First, using techniques outlined in Module 1-C, calculate a suitable value for `ILC`, the inner loop count that provides an overall delay of approximately one millisecond. Show your derivation in the space provided below, along with your rationale for choosing a specific value.

Derivation of `ILC` (note –target systems run at 12.5 MHz, which means each instruction cycle is 80 ns)

Run the program and observe the display − only DSO channel 4 and LA channel 0 (both of which should be connected to PB-0 on the microcontroller board) should be active. Use the built-in MSO functions to measure the square wave frequency, period, and duty cycle, for values of `NMS` ranging from 5 to 100. Complete the table below and calculate the error for each case.

Table 1. Square wave measurements for ILC = _____

`NMS`	*estimated period (ms)*	*estimated square wave freq (Hz)*	*measured period (ms)*	*measured duty cycle (%)*	*measured square wave freq (Hz)*	*error (%)*
1	2	500 Hz				
5	10	100 Hz				
10	20	50 Hz				
25	50	20 Hz				
50	100	10 Hz				

Based on these observations, try "tweaking" the value of `ILC` and re-run the five test cases. Comment on any measurable changes in the percent error calculated.

Table 2. Square wave measurements for ILC = _____

`NMS`	*estimated period (ms)*	*estimated square wave freq (Hz)*	*measured period (ms)*	*measured duty cycle (%)*	*measured square wave freq (Hz)*	*error (%)*
1	2	500 Hz				
5	10	100 Hz				
10	20	50 Hz				
25	50	20 Hz				
50	100	10 Hz				

Step 2. Observing Bus Signals and Identifying Machine Cycles

In this step we will investigate the internal timing parameters of the HCS12 address and data busses. As mentioned in class, by default the HCS12 operates in (Normal) Single Chip Mode. However, in order to interface an external memory component to the HCS12, it must operate in Expanded Mode. Though we will not interface external memory to the microcontroller in this experiment, we will operate the HCS12 in **Normal Expanded Wide Mode** with *internal visibility turned on.* This will allow the internal bus activity to be observed on Ports A, B, and E.

Download and unzip the project **Lab5-Step2** from the course website. Open **`main.asm`** in Code Warrior and examine the program. Note that the program includes startup code that places the HCS12 into **Normal Expanded Wide Mode**, followed by a series of instructions the read data from and write data to SRAM. The instructions and associated machine code executed in a loop are listed in Table 2.

Table 4. Memory Access Loop Instructions

Address	*Object Code*	*Assembly Mnemonic*	*Clock Cycles*
0900	A7	NOP	1
0901	A7	NOP	1
0902	B6 0A 00	LDAA $0A00	3
0905	F6 0A 01	LDAB $0A01	3
0908	7A 0A 02	STAA $0A02	3
090B	7B 0A 03	STAB $0A03	3
090E	7C 0B 00	STD $0B00	3
0911	B7 81	EXG A,B	1
0913	7C 0B 82	STD $0B82	3
0916	20 E8	BRA LOOP	2

Run the program and "autoscale" the MSO; set the display to trigger on DSO channel 0. Stop the capture and scale the DSO channels at the top of the display to fit the available space. To more easily interpret the LA data, set up the "digital" channels to be displayed as a "bus" (refer to pp. 114-116 of the MSO *User Manual* to see how to do this): "Bus 2" (B_2) should select D8-D15, while "Bus 1" (B_1) should select D7-D0. Adjust the horizontal timing interval (via "fine" adjustment) to 40ns/div – this will align the bus cycles to the display grid. Finally, scroll the display horizontally "back in time" to find the fetch of the `NOP` instructions at the beginning of the loop. Refer to Figure 2 for an illustration of the initial MSO display configuration desired.

Note the following before proceeding. The 9SC128 microcontroller utilizes a 16-bit multiplexed address/data bus; for all memory read and write operations the address is emitted on the bus during the first half of the cycle, while the data is transacted on the bus during the second half of the cycle. In order to interface an external memory component to the microcontroller, we would have to make use of a *transparent latch* (or an edge-triggered D-register) to make the address and data available at the same time (as detailed in Lecture Module 2-B). Because the external bus interface is initialized for "wide" expanded mode, the microcontroller (always) reads two bytes at a time (high byte on Port A, low byte on Port B). For writes, there exists the need to accommodate writes of single bytes in addition to word writes; the `LSTRB′` signal ("low byte strobe") is provided for the purpose of handling byte writes to word-wide SRAMS correctly.

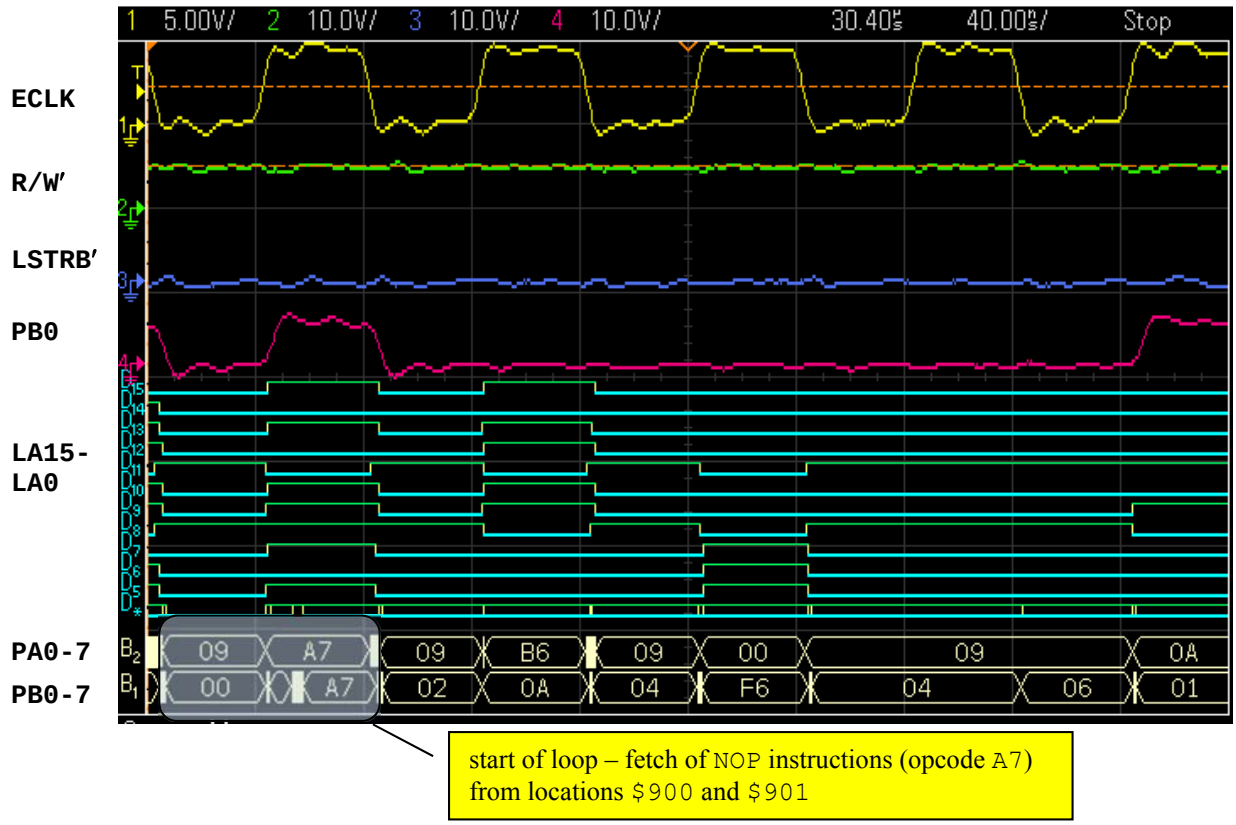


Figure 2. Initial MSO Display of Multiplexed Bus Activity Desired

You don't have to proceed very far down "memory lane" (as you scroll the bus activity display horizontally in time) to realize that the bus activity associated with data reads and writes is displaced (delayed) relative to the fetches of the corresponding load and store instructions. This is because the processor is *pipelined*, i.e. it "fetches ahead" so that it can begin decoding and preprocessing the instruction bytes in advance of its actual execution. Note that several cycles go by before a previously fetched load or store instruction performs the data transaction. Likewise, several cycles go by before the branch instruction is executed, effecting the transfer of control to the beginning of the loop. One last thing to note is that many multi-cycle instructions perform "internal CPU" operations on some of the clock cycles associated with their execution, with no meaningful activity reflected on the external bus during those cycles (in many of these cases, the bus may contain something "random" or it may simply "stay in the same state").

Finally, the "fun" part! Referring to the "disassembled" program listing provided in Table 4, scroll horizontally in time through the captured data (beginning with the state shown in Figure 1) and identify what actions the CPU/memory are performing on a cycle-by-cycle basis. You will most likely need to refer to the *CPU12 Reference Manual* on an instruction-by-instruction basis to determine the nature of the "internal" cycles performed. You and your lab partner should do this "as a team", splitting up responsibility as equitably as possible (suggest alternating roles every other instruction or so). Record your bus cycle identifications and comments in Table 5. Recall that the "address phase" occurs when `ECLK` is low, while the "data phase" occurs when `ECLK` is high. The "first three ticks" have been completed for you; based on Table 4, a total of 24 clock cycles should be consumed for each iteration of the loop.

Table 5. Bus Cycle Identification

Clock Cycle	*Address Phase*	*Data Phase*	R/W′	LSTRB′	*Cycle Type / Description / Comments*
1	09 00	A7 A7	H	L	fetch of two consecutive NOP instructions from addresses $900 and $901
2	09 02	B6 0A	H	L	fetch of first two bytes of LDAA $0A00 instruction
3	09 04	00 F6	H	L	fetch of last byte of LDAA instruction and first byte of LDAB instruction
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					

Step 3. Measuring Bus Signal Timing Parameters

In this step we will use the cursor tool (in conjunction with the built-in "Δ" function) to estimate some of the key timing parameters associated with the multiplexed expansion bus of the 9SC128. To complete this step, you will need to scroll the captured display in time in order to find bus cycles that facilitate these measurements (e.g., execution of "load data" or "store data" cycles). Refer to Module 2-B (and the homework you completed for prelab) to identify the parameters listed in Table 6 (refer to the *A.C. Timing Parameter* chart in Module 2-B for published values).

Table 6. Measurement of Timing Parameters

Timing Parameter	*Published Value (ns)*	*Measured Value (ns)*
t_{CYC}	–	80
t_{AD}		
t_{MAH}		
t_{OH}		
t_{RS} *available*		
t_{DDW}		
t_{WH}		

Step 4. Thought Questions

1. Why do the DSO and LA plots show the same signal (i.e. PB-0) changing state at slightly different times?

_____

_____

2. For the DSO data, why doesn't the transition look "square" (i.e. what causes the "rounding" of the transitions and the apparent "ringing" or undershoot/overshoot)?

_____

_____

3. Explain the role of the `LSTRB′` signal in an expanded wide CPU-memory interface employing word-wide SRAMs (Reference: *AN2287 HCS12 External Bus Design*, p. 9).

_____

_____

