Lab 8: Debugging Embedded Devices and Software

Summary:

Given pre-written code, isolate code and functional errors to create a working memory interfacing program.

Learning Objectives:

- Debug and fix pre-written code.
- Introduce a basic, but representative, microcontroller-to-memory interface
- Review EPROM specifications and operation
- Demonstrate the use of MON12 utility subroutines

Resources and Supplies:

- CML12S-DP256 development board
- 27512 EPROM
- PC with text editor, 68HC12 assembler, and MON12 monitor program
- *CML12S-DP256 MON12 Manual**
- *27512 data sheet**
- *Wiring Diagram Template**

All documents* are available on the class website

Important Reminders:

- Because wires may be cut and stripped in this lab, ***you must wear safety goggles***.
- During lab, you are responsible for knowing everything in the Background section. Failure to prepare for the lab could result in the TA subtracting points from your grade.
- It is your responsibility to save the programs you write.
- Pre-lab assignments must be completed before coming to the lab.

Background:

Debugging and Code-Correction

Debugging is a very important process in software and hardware development. Rarely in real world experience will you be able to create a product from scratch without experiencing some form of error or incorrect functionality. In many cases, you will find yourself correcting and debugging other people's code. It is important to divide coding errors into two categories. The first are errors in actual coding that cause the compiler or assembler to fail. These are usually corrected by examining the errors thrown by the building program. The other set are functional errors: errors in programming that while not throwing build errors do cause problems in functionality. For the second set of coding errors, you engage in the actual process of debugging. In relation to the HCS12 development board we are using, the best way to do this is to step through your program and examine register values. A good reference for debugging can be found at

http://en.wikipedia.org/wiki/Debugging

27512 EPROM.

The 27512 is a 512kbit (64K x 8) EPROM. Read through the data sheet to become familiar with the information available so you can refer to it as needed. This 64k-byte EPROM has 16 address bits. In this lab, you will only need to access addresses $FF00 and $FF1F (32 locations) using only 6 address signals from the MCU. You will need to determine how to wire the EPROM address signals to the appropriate MCU address pins and hardwired ground (0V) or power (5V) connections. You will also need to study the rest of the pins on the EPROM and determine the proper signal values for each input. Be sure to connect power and ground to the EPROM also.

ASCII Codes & MON12 OUTA Utility

The American Standard Code for Information Interchange (ASCII) specifies a correspondence between digital bit patterns and character symbols including 52 upper/lower case alphabetic characters, 10 numerical digits, and special graphic symbols. Each 7-bit ASCII code represents a character symbol, with codes 0-31 reserved for control codes. For example, $1000001 = 65_{10} = 41_{16}$ represents 'A'. A complete list can be found at en.wikipedia.org/wiki/ASCII. Using ASCII codes, a text string can be represented by a series of coded bytes. Note that character codes sent to the LCD display are also represented in ASCII.

The MON12 monitor program installs a variety of utility subroutines which can be accessed between $FF10 and FF67 (see MON12 Manual). One of these utilities is called OUTA and can be accessed by jumping to the subroutine at $FF4F. This subroutine will display the ASCII character represented by the value stored in accA. You will call this subroutine within your program to display ASCII codes read from the EPROM. Notice, your program will not know what 'OUTA' is, so you'll need to equate it to the address $FF4F.

Memory Interface

Rather than using the expanded mode of operation for the HCS12, an interface to external memory can be constructed using general purpose I/O ports. In this lab, Port A will be used as the 8-bit data bus, which for this lab will serve as an input port, receiving data from the external EPROM. 6 bits of Port K will be used as the address bus, to output to the EPROM the address that will be read.

Hardware and Connection Information

The EPROM will be inserted into the protoboard area on the AXM-0295 project board. To connect EPROM pins to the microcontroller, wires must be connected to wiring "headers" (connection points) on the project boards. Port A can be connected through the BUS-PORT on the CML12S-DP256 development board. Port K is connected to the project board through the big ribbon cable (where it is wired to the LEDs & other components) and can be accessed through the AUX1 and AUX2 ports on the AXM-0295 project board. Diagrams of these wiring headers are given below.

MCU Registers

The relevant MCU registers for this lab are:

$0000	Port A data register
$0002	Port A data direction register
$0032	Port K data register
$0033	Port K data direction register

Wiring Diagrams:

In preparation for constructing a circuit on a protoboard, it is good to plan your circuit on a wiring diagram, like the one shown in Fig. 1. A wiring diagram is a sketch of the components you will use with lines showing where wires must be run to complete the circuit. It is good to include both signal wiring as well as power/ground wiring in our diagram. Indicate the boundaries of the chip(s) in your circuit and label their pins numbers. Also be sure to show where the inputs and outputs will be taken. This example wiring diagram was drawn using the *Wire Diagram Template* document available on the class website. The example uses color coded wiring, red for power, green for ground, and orange for signals. Your wiring diagrams should be done in pencil so you can easily make changes, although once you are sure your diagram is

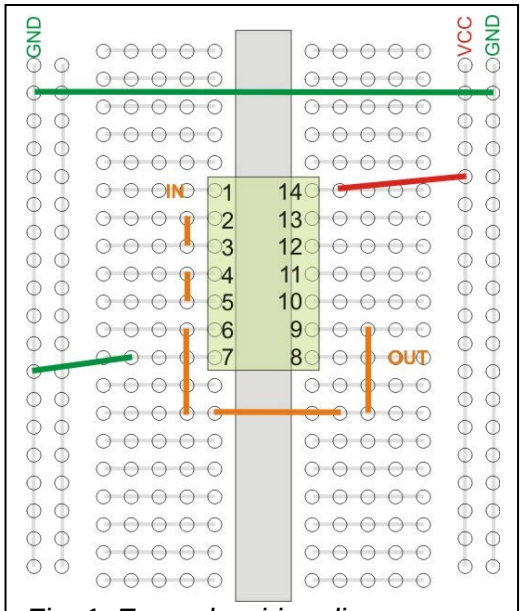
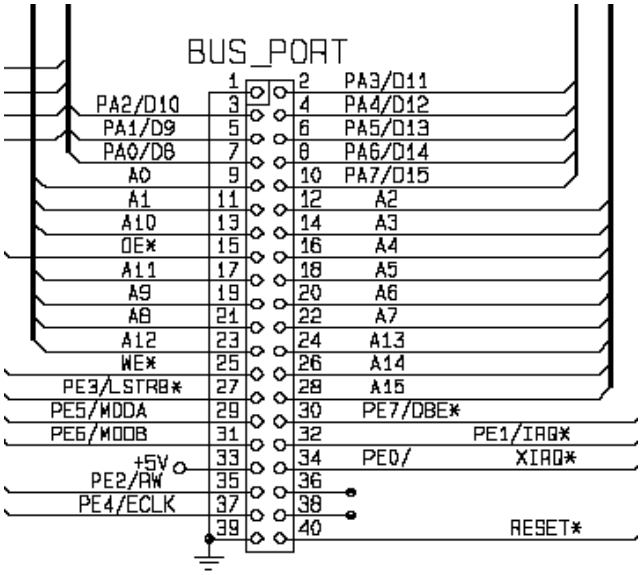


Fig. 1. Example wiring diagram.

correct, you may want to retrace some lines in color (ink/marker/etc.) to help clarify your drawing before you attempt to implement it on a protoboard.

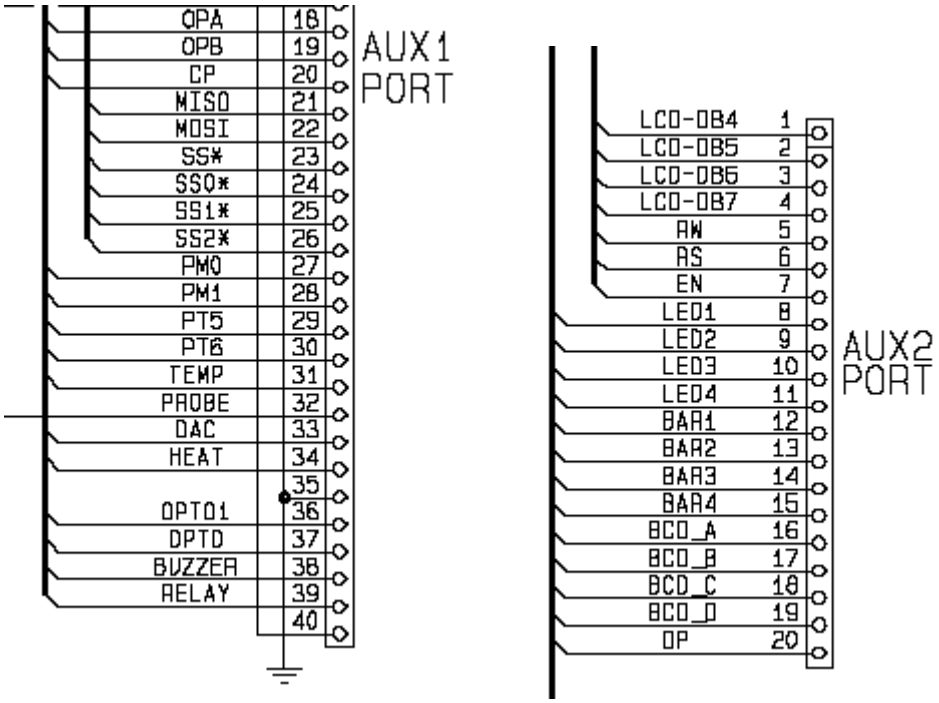
Wire Connection Header Pin Descriptions

Header Pin	PortA Pin
2	PA3
3	PA2
4	PA4
5	PA1
6	PA5
7	PA0
8	PA6
10	PA7



BUS_PORT of the CML12S-DP256 development board.

Header Pin	PortK Pin
AUX2, 8	PK0
AUX2, 9	PK1
AUX2, 10	PK2
AUX2, 11	PK3
AUX1, 39	PK4
AUX1, 38	PK5



AUX ports of the AXM-0295 project board. Notice that header pins are named after the device they connect to on the project board rather than the MCU port they are connected to.

Project Description:

In this lab, each lab team will be given a 64k-byte EPROM that has an ASCII coded text message stored between $FF00 and $FF20. You must install the EPROM into the protoboard space on your development board and correctly wire it to the MCU, power, etc. in order for software you prepare to be able to read the stored ASCII message from the EPROM. Once the EPROM is correctly wired, your program should sequentially read though the EPROM addresses from $FF00 - $FF1F, and after each byte is read the OUTA MON12 utility subroutine should be called to display the ASCII value on accA. Each stored text message ends with a '.' (ASCII code $2E), which will be used as an indicator to end the scanning loop.

It is important to note that in this lab you are not asked to write the program yourself. Rather, when you arrive in lab the TA will give you a program designed to read from the EPROM. However, the code will contain compiler and functional errors, and it will be your task to debug these errors and demonstrate a working program capable of reading from the EPROM.

Here are a few additional important notes.

- Addresses $4000 to $7FFF are not available for program/data storage due to the changes you will make in hardware settings on the development board. Programs can begin at $1000.
- The EPROM chip has a read access time specified on the data sheet. Any attempt to read the EPROM should have sufficient delay between the time the address is set and when data is read. A simple delay loop of ~1msec should solve any problems.
- Notice that the message can be read with only 5 address bits (i.e., the message on the EPROM does not use all the available memory space). The 6th Port K bit (PK5) can be left unconnected, in which case you'll have to hardwire the corresponding address pin on the EPROM. Alternatively, since PK5 is connected to a buzzer on the project board, it could be used as an indicator that the program loop has reached the end. That way, if everything is designed properly, the buzzer will sound when your message is ready to be read from the monitor.

Pre-lab Assignment:

- Complete the steps described in the Pre-lab sheet near the end of this document. Each student must complete his/her own pre-lab before coming to the lab and hand it in to the lab TA at the beginning of the lab.
- Read through the Laboratory Assignment so you know what to expect in the lab. Also read through the *27512 EPROM Data Sheet* to become familiar with where to find information in that document.

Laboratory Assignment:

This lab consists of one part. A check-off sheet is included at the end of this lab document.

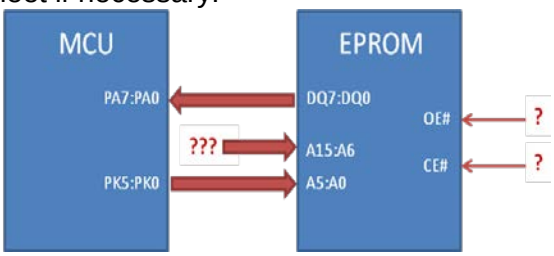
- Print the check-off sheet. Where indicated, you must record your results on the check-off sheet. After you successfully finish the lab, show the TA your results and ask him to sign the check-off sheet.

Preparation -Hardware:

Based on your wiring diagram from the Prelab, insert the EPROM into the protoboard on the project board and wire it correctly. Do not forget to connect the EPROM to power and ground, either from the power supply or from the AUX1-PORT.

By default, the CML12S-DP256 development board uses an external "jumper" that sets the 68HCS12 in an expanded operating mode. This permits the board to access external RAM at addresses $4000 to $7FFF. To use Port A as a general purpose I/O port, this hardware setting must be changed by completing the following steps.

1. Turn off the power to the 68HC12 development board.
2. Remove the MEM_EN jumper from the DP256 development board. So the jumper does not get lost, insert it back with only one side connected to one pin and the other side dangling.
3. Restore power to the 68HC12 development system.
4. Wire correctly between the MCU and the EPROM as shown in the figure below. Be aware of connections to the address pins on the EPROM: configure those addresses that are not connected to Pork K to the correct logic needed to access memory location from $FF00 - $FF1F. Also correctly connect the CE# and OE# pins so you can use the EPROM in read mode; see the datasheet if necessary.



Connections between MCU and EPROM

Making this change will means the external RAM is not available, so you can not store any code between $4000 and $7FFF.

At the end of this lab you will need to replace the MEM_EN jumper (with the power off) to return the development board to its default configuration.

Preparation -Software:

1. Using any text editor, write the assembly code for the counter program described above and save it as a .asm file in your secure programs directory.
2. Launch the ***AsmIDE*** utility from the 68HC12 program folder under the 'Start' menu.
3. Within AsmIDE, select *File* → *Open* to open your .asm file.
4. From the *Build* menu, select *Assemble*. to assemble your program and generate the *.lst* *.s19* files. At this point, you will encounter compiler errors. Record these build errors.
5. Remove any syntax errors that are indicated by the assembler until you obtain an error free assembly. Remember that you also need to set the port directions correctly for Port A and K. A complete *.s19* file is only produced when the assembly proceeds to conclusion with no errors. Print your final *.lst* file using the text editor and include this in your lab report.

Testing:

6. Connect the CML12S-DP256 board to power and to your computer using the wiring bundle attached to your computer.

Press the reset button on the development board. You should see a message to "PRESS KEY TO START MONITOR…" in your terminal window. Press the ENTER key and you should see:

a. Axiom MON12 - HC12 Monitor / Debugger V256.5
b. Type "Help" for commands.
c. >_

7. Type '**load**' at the prompt and press the ENTER key.
8. Select the '**Upload**' option. Then click on the '**Browse**' button and select the ***.s19*** file that was created from the assembly process. Once you have selected the file, click 'OK'.
9. Use the 'Modify CPU Register Contents' command (***RM***) to set the Program Counter to point at address $1000. To do this type RM press enter, and then enter the address.
10. Type ***G***, and press Enter to start the program execution.
11. At this point, you will notice the program not functioning as expected. Record the errors you encountered.
12. Use debugging techniques to isolate the functional errors and fix them appropriately. Once your program is correctly displaying the message, record the message in the check off sheet.
13. When you are satisfied that the program is operating correctly, ask the TA to check a demonstration of your program. Ask the TA to check off on your lab check-off sheet.

Final Tasks:

- Turn off the power supply and anything else that you might have turned on.
- **Replace the MEM_EN jumper (with the power off) to return the development board to its default configuration.**
- Disconnect and return the microcontroller interface board to the lab closet.
- Return the EPROM to the TA.

Discussion Points

As explained in the *Lab Report Guide*, you should address these discussion points in a designated section of your report.

1. What is the maximum message length that can be stored in the EPROM and displayed correctly using 6 bits of Port K?
2. Describe how you would modify your code and the wiring of the EEPROM to read from locations $F00F thru $FA0F. Please describe ALL changes. How many address lines are needed for this configuration?
3. What will happen if there is no delay loop when the data is read after the address is set?

PRE-LAB 8

Due: At the beginning of lab.

Student Name: _____ **Lab. Section (time):** _____

1. Read the Background and Project Description sections of this lab. These sections contain a lot of information that is important to understand before the lab begins. Read them 2 or 3 times if necessary.

2. The 27512 EPROM chip contains 28 pins. One of them is NC (no connect). The remaining 27 will have to be connected when you wire the EPROM to the project board. Refer to the 27512 data sheet to answer the following questions.

   a) How should the chip enable input (CE#) be connected?

   b) OE#/Vpp pin is used to program (write) data to the EPROM. How should it be connected in read mode?

   c) The 64k-byte EPROM has memory addresses from $0000 to $FFFF. How would you connect these address pins so that you can access memory locations $FF00 - $FF20 using only 5 address bits, PK5 – PK0? Specify which pins are connected to 5V (‘1’), 0V (‘0’) or to a Port K pin.

EPROM Addr pin	A15	A14	A13	A12	A11	A10	A9	A8
connection								

EPROM Addr pin	A7	A6	A5	A4	A3	A2	A1	A0
connection								

3. Prepare a wiring diagram needed for this lab. Using the wiring diagram template provided in previous labs, prepare a wiring diagram for the EPROM that specifies all connections to the EPROM. Be sure to include power and ground connections. For all connections to wiring headers on the development or project boards, be sure to include the header name and pin number for each connection.

   Show the wiring diagram to the TA before beginning your lab, and attach it to your lab report.

4. The following is a program whose purpose is to jump to a subroutine with a delay loop and increment a counter to 10 and end. It has some coding and functional errors. Identify the errors and fix the program accordingly by writing a correct version of the code.

```
;Lab 8 Prelab #3 debugging
                ORG     4000
                LDAA    #$10
                LDAB    #0
LOOP            INC     A
                JSR     time
                DECA
                BNE     LOAP
                SWI

TIMER           LDAA    #$FF
OLOOP           LDX     $FFFF
ILOOP           DECX
                BNE
                DECA
                BNE     OLOOP
                RTS
```

LAB 8 CHECK-OFF SHEET

Student Name: _____ **Lab. Section (time):** _____

Complete this sheet as you complete the lab. Remember to have the TA check off each section of the assignment. This sheet must be included in your lab report.

Part 1: EPROM interfacing and access.

Step 4. Please list the compiler errors.

Step 11: Please list functional errors you encountered:

Step 12. What is the displayed EPROM message?

_____

Part 1: TA sign off
Part 1: EPROM interfacing and access. Initial_____