Programming the CME11E9-EVBU, Keypad, and LCD

The purpose of this lab is to become familiar with the Motorola 68HC11 Lab Trainer kit, related software (Buffalo monitor, TExaS simulator) and learn to program the system.

Objectives

- Understanding the CME11E9-EVBU Development Board from Axiom Manufacturing,
- Uploading, running, and observing a pre-assembled program using AxIDE and the Buffalo monitor
- Assembling a pre-written program so that it runs on the TExaS simulator,
- Modifying the program to run in the RAM address space of the Axiom EVBU, testing, and demonstrating it,
- Becoming familiar with the Keypad and LCD display mounted on the Axiom EVBU, including:
    - Running a pre-assembled program to echo back the identity of each depressed key,
    - Modifying, reassembling, and testing the above program,
- Modifying, the given program to respond to a different key sequence.

Turn-In Requirements:

As specified in the lab policies handout.

References

1. Motorola, Inc, *M68HC11 E Series Programming Reference Guide*, (MC68HC11ERG/AD.
2. Motorola, Inc, *MC68HC11E Family Technical Data*, MC6811E/D-Rev 3, 2000.
3. Motorola, Inc, *Buffalo© Monitor for HC11 Development Boards*.
4. Axiom Manufacturing, *CME11E9-EVBU Development Board*, 1999.
5. The "Tips" handout for uploading assembled files with Buffalo.

Equipment for this lab:

- 68HC11 trainer kit, to include 68HC11 EVBU, host PC, cabling, TExaS, AxIDE and Buffalo software
- A 3.5-inch floppy disk (supplied by the student).

Background Discussion

Uploading and Running Assembled Programs

- Source files are <*filename* >.RTF
- Assembled Object Files are names <*filename*>*.s19*
- Loading (See *tips.pdf* file and *Buffalo Monitor* Manual for details)
    - Type the buffalo command "load T",
    - Use AxIDE's "*upload*" menu and follow the dialogue boxes to select the *.s19* file to upload,
    - Use the Buffalo command "*go* <*start-address*>" or "*call* <*start-address*>" to start the program; NOTE:
        - "GO" does not return to the buffalo monitor after the program terminates,
        - "CALL" returns to the Buffalo monitor after the program terminates, if you ended the program with an *RTS* instruction (Return from Subroutine).
- If necessary, you can always abort the program by hitting the RESET button on the EVBU board.

Assembling Programs

For TExaS

- Start your Program section at "*org $6000*"
- Create your .RTF and .UC files as previously described in EE-3170 lecture and handouts. Note: the versions of TExaS in Room 619 EERC have an Axiom CME11E9 option in the "*mode -> processor"* menu. Select it, if not already chosen as the default.
- Assemble and simulate your file.

For the CME11E9EVBU

- Re-"org" your source file to run in the CME11E9 RAM space (e.g. "org 2000")
- In order for TExaS to create a .s19 object file, you must click on: the "*assemble -> options*" menu and then in the dialogue box, check mark the "*automatically create a *.s19 file*" box.
- Your file should assemble properly, but will not simulate on TExaS because of the changed "org".
- When you exit TExaS the .s19 file should be in the same directory as your source code file.

The Keypad

A keypad is physically just a set of Single-Pole, Single-Throw (SPST) switches at the junctions of horizontal and vertical lines. In this respect, it closely resembles the architecture of a most ROM technologies (as described in the EE-3170 lecture). The major difference is that the diodes or FETs are replaced with mechanical switches.

To read a keypad, the program outputs a "1" to each row (one at a time). It then reads the column values.

- IF the column returns a non-zero value, then a key in the "current" row has been pushed.
- THEN the program examines the contents of the returned word to determine which column contains the pushed key.

The identity of the pushed key is indicated by its both row and column number. The connection of the keypad to the CME11E9 board is described on page 15 of the *CME11E9-EVBU* manual.

The LCD Display

The LCD display receives ASCII characters as well as command characters from the program. It has both a data register and a command register. The connection of the LCD display to the CME11E9 board is described on page 14 of the *CME11E9-EVBU* manual. The Character Codes and Command Codes are described on page 22 of the *CME11E9-EVBU* manual.

Instructor Verification

Name: _____ Signature: _____

Name: _____ Signature: _____

Schedule

This lab activity should be finished during one lab meeting.

Notes

- In this lab, all numbers such as addresses and data are given in hexadecimal format ("hex") unless otherwise indicated. In completing the lab, record all information in hexadecimal unless directed differently.

- Make sure your development EVBU board is connected to power (green LED on board ON) and the serial port of the EVBU is connected to the serial port of your development PC containing the AXIDE software, configured to the correct port at the correct baud rate, etc.

- Buffalo is the monitor program initiated as the Axiom EVBU board is reset when operating in either single chip mode(on-off) or expanded mode(off-off). Note: This is the default configuration for the EVBU board assuming that you have not modified the ROMON flag (more on this later).

- Buffalo commands are *not case sensitive*. You will need to refer to Reference [3] frequently.

- Figure 4-2 in the E Series Technical Data Manual depicts the E9 memory map (this figure can also be found on page 4 of the Programming Reference Guide. Also refer to the EVBU memory map on page 12 of the CME11E9-EVBU Development Board manual)). These figure indicate that 512 bytes of RAM exist in locations $0000-01FF.

  However, some of these memory locations are used by Buffalo, and should **not** be used in user programs. Buffalo may use any RAM locations from $0000 to $00FF. Also, the upper portion of RAM, $01FF down, is normally used for system stack.

  Therefore, *you should limit your use of RAM in this lab and other labs or projects to locations $0100-01C0* if using Buffalo. This is only true if you launch your hardware application using a *call* by Buffalo. If you use the *GO* command of buffalo and expect to have to RESET the board to regain control, then all of the lower RAM is available for your application.

Memory Reads and Writes:

Figure 4-2 in the E Series Technical Data Manual depicts the E9 memory map (this figure can also be found on page 4 of the Programming Reference Guide. Also refer to the EVBU memory map on page 12 of the CME11E9-EVBU Development Board manual)). These figure indicate that 512 bytes of RAM exist in locations $0000-01FF.

However, some of these memory locations are used by Buffalo, and should **not** be used in user programs. Buffalo may use any RAM locations from $0000 to $00FF. Also, the upper portion of RAM, $01FF down, is normally used for system stack.

Therefore, you should limit your use of RAM in this lab and other labs or projects to locations $0100-01C0 if using Buffalo. This is only true if you launch your hardware application using a call by Buffalo. If you use the GO command of buffalo and expect to have to RESET the board to regain control, then all of the lower RAM is available for your application..

We will now use EVBU and Buffalo to read and write to portions of the HC11's CME-EVBU memory map.

Make the AXIDE Terminal window active. Make sure you have an active Buffalo prompt (>) (after seeing Buffalo identifying message, hit enter). This should appear after a hardware reset while in single-chip mode (on-off). Note this disables the EVBU from accessing the expanded memory features seen on page 12 of the CME-EVBU manual.

Using the MD command, display RAM locations $0110-0119. Record the contents in table below.

Location	**110**	**111**	**112**	**113**	**114**	**115**	**116**	**117**	**118**	**119**
Content (hex)										

Using the MM command, change RAM locations $0110-0119 to the following.

Location	**110**	**111**	**112**	**113**	**114**	**115**	**116**	**117**	**118**	**119**
Content (hex)	**0A**	**0B**	**0C**	**0D**	**0E**	**0F**	**10**	**20**	**30**	**40**

Using the MD command, display RAM locations $0110-0119 and verify that the contents in ram locations $0110-0119 are correct.

Use the BF command to Block Fill $0110-0119 with the value 5A.

Using the MD command, display RAM locations $0110-0119. Verify the operation by filling out the table below.

Location	**110**	**111**	**112**	**113**	**114**	**115**	**116**	**117**	**118**	**119**
Content (hex)										

Use the MOVE command to move the contents of locations $E280-E289 to locations $0110-0119.

Using the MD command, display RAM locations $0110-0119 and $E280-E289 and verify the operation.

We will learn some more Buffalo commands in the next lab. The only additional command you need to know for this lab is the "GO" (or "G") command. Typing G followed by an address causes the program execution to begin at this specified address.

Upload and Test a Keyboard and LCD Program

UPLOAD and RUN program *echo-1.s19*. This program reads from the keypad, and for every key pressed, it echoes that key's character back to the LCD display. NOTES:

- This program is "orged" at 2000.
- Use the "G 2000" command to execute the program.
- This program does not terminate, so you have to reset the EVBU when you want to exit.

1. One key acts as a "Display Reset" key. What is the ASCII value of the key? _____

2. Demonstrate to the TA that you have it running. **T.A. INITIALS**: _____

Edit, Assemble, and Test the Keyboard and LCD Program

Program *echo-1.RTF* is the source code for *echo-1.s19*. Do the following:

1. Copy *echo-1.RTF* to a new file named *echo-2.RTF*.

2. Using TExaS, edit *echo-2.RTF* to change the "Display-Reset" key to be the pound-sign (#) key. Also, change the original Display Reset key to simply echo its character.

3. Demonstrate to the TA that you have it running. **T.A. INITIALS**: _____

Modify, Assemble, and Run the Keyboard and LCD Program

1. Program *echo-1.RTF* is the source code for *echo-1.s19*. Do the following:

2. Copy *echo-1.RTF* to a new file named *echo-3.RTF*.

3. Using TExaS, edit *echo-3.RTF* to change the "Display-Reset" if the key sequence "ABC" is typed.

4. Demonstrate to the TA that you have it running. **T.A. INITIALS**: _____

Do the Following to document your program (Note: comments, structure, and readability count!!!)

1. Hand-in a complete *Flowchart* and for the entire program (*echo-3*.RTF). We expect the flowchart will require multiple layers, so include a *Hierarchy Roadmap*.

2. Hand in the listing of the complete, well-commented source-code that *matches the flowchart*. Remember, if the TA can not read your code with "reasonable" effort, then it is wrong.

3. Demonstrate to the TA that you have it running. **T.A. INITIALS**: _____

When you turn in the lab report, please answer the following questions. Provide flow charts and/or assembly language programs as necessary to explain your answers.

Questions

1. Name 2 types of on-chip memories available in 68HC11 family.

2. Why are some chips called "microprocessors" and others called "microcontrollers."

3. Suppose acc A contains $80 and acc B contains $10. If subroutine "`key_lookup`" is called, what will acc B contain when the program exits the subroutine.

4. "`INIT_LCD`" subroutine is slow to execute because the display controller is completely re initialized. If you have to clear the screen often, it will be very inefficient to use "`INIT_LCD`." Provide a way of clearing the display without reinitializing it. Hint: use "`put_lcd`" subroutine.

5. The keypad reading algorithm is inefficient because we can't output signals to port E of 68HC11. The program uses port D to energize one row of the keypad at a time to read a key. Suppose we can program ports D and E as both input and output ports, provide an efficient algorithm to read the keypad.