Chapter 3: BUFFALO Monitor

By Prof. Taek Kwon

The board used in this course is called M68HC11EVB Evaluation Board or EVB in short. The EVB includes a monitor/debugging program called BUFFALO (Bit User Fast Friendly Aid to Logical Operation) which is contained in an EPROM chip (external to the MCU). The BUFFALO monitor allows the user to debug codes, download programs from a host computer, and emulate user code in a target system environment.

3.1 General Rules

BUFFALO monitor follows the following general rules.

- All numbers are interpreted as hexadecimal.
- Fields are separated by any number of space, comma, or tab characters.
- All input commands can be either upper or lower case lettering.
- A maximum of 35 characters may be entered on a command line.
- Command line errors may be corrected by backspacing (^H) or by aborting the command (^X). Here, ^H means pressing Ctrl and H key simultaneously.
- Pressing CR (Carriage Return) with empty command repeats the last command.

Terminal Keyboard Functions

^A	Exit transparent mode or assembler
^B	Send break command to host in transparent mode
^H	Backspace
^J	Line feed (f)

3.2 BUFFALO Commands

3.2.1 Register Modify: **RM**

The RM command is used to display and modify the content of MCU registers. The symbols used are:

P: Program counter
Y: Index register Y
X: Index register X
A: Accumulator A
B: Accumulator B
C: Condition Code Register
S: Stack Pointer

Examples:

>RM	Display all registers and modify PC
P-C600 Y-0028 X-4678 A-35 B-58 C-C0 S-0060	Display all registers
P-C600 C71D (CR)	Modify PC content to C71D
>RM B	Modify B register
P-C600 Y-0028 X-4678 A-35 B-58 C-C0 S-0060	Display all registers
B-58 64 (CR)	Modify B content to 64

<space bar> can be used to step through each register one by one to modify.

```
>RM
P-C600  Y-0028  X-4678  A-35  B-58  C-C0  S-0060
P-C600 <space bar>
Y-0028 <space bar>
X-4678 6342 <space bar>
A-35   78 <space bar>
B-58 <space bar>
C-CD <space bar>
S-0060 <space bar>                        After S-register it returns to the monitor.
>
```

3.2.2 Memory Modify: **MM** [address]

The MM command allows the user examine/modify contents in user memory at specified locations in an interactive manner. The MM command will also erase any EEPROM location, and will reprogram the location with the corresponding value. EEPROM locations appear as if they are in RAM except that the contents are retained even after power is removed.

Within the MM command, several sub-command exit and allow simple navigation of memory locations.

(Ctrl) J or (Space Bar)	Examine/modify next location.
(Ctrl) H or ^	Examine/modify previous location.
/	Examine/modify same location.
O	Compute branch instruction relative offset.

If an invalid address is specified, the invalid address message “rom” is displayed on the terminal.

Caution: **Never modify** the EEPROM location called MCU CONFIG register. It can disable the system ROM. The areas users can use are $b600 - $b7ff.

Examples:

```
>MM c000
c000 40 36 (Ctrl)J
c001 59 5F (Ctrl)J
c002 34 67 (Ctrl)J
c003 45 28 (Ctrl)H
c002 67 (Ctrl)H
c001 5F 22 /
c001 22 44 /
c001 44 (CR)

> MM c000
c000 36 5F 67 28 ........            You can examine next memory locations using (space bar).
```

3.2.3 Memory Display: **MD** [address1] [address2]

The MD command allows the user to display a block of user memory beginning at address1 and continuing to address2. If address2 is not entered, 9 lines of 16 bytes are displayed beginning at address1. If address1 is greater and address2, the display will default to the first address. If no addresses are specified, 9 lines of 16 bytes are displayed near the last memory location accessed.

At the right side group of columns, ASCII translation of binary data is displayed. If the value is a nonprintable character, "." is displayed.

Examples:

```
>MD c020 c030

C020    31 32 33 34 35 36 37 38 39 FF FF FF FF FF FF     123456789......

>MD

C020    31 32 33 34 35 36 37 38 39 FF FF FF FF FF FF     123456789......
C030    31 32 33 34 35 36 37 38 39 FF FF FF FF FF FF     123456789......
C040    31 32 33 34 35 36 37 38 39 FF FF FF FF FF FF     123456789......
C050    31 32 33 34 35 36 37 38 39 FF FF FF FF FF FF     123456789......
C060    31 32 33 34 35 36 37 38 39 FF FF FF FF FF FF     123456789......
C070    31 32 33 34 35 36 37 38 39 FF FF FF FF FF FF     123456789......
C080    31 32 33 34 35 36 37 38 39 FF FF FF FF FF FF     123456789......
C090    31 32 33 34 35 36 37 38 39 FF FF FF FF FF FF     123456789......
```

```
C0A0    31 32 33 34 35 36 37 38 39 FF FF FF FF FF FF     123456789......
```

3.2.4 Assembler/disassembler: **ASM** [address]

The ASM command initiates the line assembler/disassemble of EVB. [address] is the starting address of the assembler/disassemble operation. Each source line is converted into the proper machine language code and is stored in memory overwriting previous data on a line-by-line basis a the time of entry. In order to display an instruction, the machine code is disassembled and the instruction mnemonic and operands are displayed. All valid op codes are converted to assembly language mnemonics. All invalid opcodes are displayed on the terminal as "ILLOP".

Assembler Syntax Rule

(a) All numerical values are assumed to be hexadecimal.
(b) Operands must be separated by one or more space or tab characters.
(c) Any characters after a valid mnemonic are ignored.

Addressing Mode Designation

(a) Immediate addressing: "#"	LDAA #20
(b) Index addressing: ","	LDAA 2,X

(c) Direct and extended addressing: One to four digit numbers.

One or two digits ------> direct addressing	LDAA 3A
Three or four digits ------> extended addressing	LDAA 34B6

(d) Relative offset is computed by assembler, so the user simply needs to supply the address to be branched.

Subcommands

/	Stay in the same address location.
^	Move to the previous sequential address location and disassemble.
(CR)	Assemble the current line and then disassemble the next opcode address.
(Ctrl)A	Exit the assembler mode of operation.

Examples:

```
>ASM c000
c000 STOP $FFFF
      > ldaa #55
        86 55
c002 STOP $FFFF
      >staa c0
        97 c0
c004 STOP $FFFF
      >lda 0,x
        AE 00
```

```
c006 STOP $FFFF
        >bra c500

Branch out of range
c006 STOP $FFFF                    The branchable range is offset of 0 to 7f. Exceeding
        >BRA c030                  this range gives the error message: "Branch out of
          20 28                    range".
c008 STOP $FFFF
        >(Ctrl)A                   Assembler operation is terminated
>
```

3.2.5 Breakpoint set: **BR** [-] [address]

The BR command sets the address into the breakpoint address table. During program execution, a halt occurs to the program execution immediately preceding the execution of any instruction address in the breakpoint table. A maximum of four breakpoints may be set. After setting the breakpoint, the current breakpoint addresses, if any, are displayed. Whenever the G, Call, or P commands are invoked, the monitor program inserts breakpoints into the user code at the address specified in the breakpoint table.

Examples

```
>BR c003                           Set breakpoint at address location c003.
C003 0000 0000 0000
>
>BR c003 c005 c007 c009            Set four break points.
 c003 c005 c007 c009

>BR                                Display all current breakpoints.
 c003 c005 c007 c009

>BR -                              Remove all breakpoints.
0000 0000 0000 0000
>BR -c009                          Remove breakpoint at address c009.

>BR - c009                         Clear breakpoint table and add c009.
```

3.2.6 Go: **G** [address]

The G command allows the user to run user program. The user may optionally specify a starting address where execution is to begin. If the starting address is not specified, execution starts at the current program counter (PC) address location. Program execution continues until a breakpoint is encountered, or the EVB reset switch is pressed.

Example

```
> G C000                        Run the program starting at location C000.
```

3.2.7 Trace: T [n]

The T command allows the user to monitor program execution on an instruction-by-instruction basis. The user may optionally execute n instruction at a time by entering a counter value n (up to $ff). Execution starts at the current program counter (PC).

Examples

```
>RM P                                     Set PC first.
P-C007 Y-7982 X-FF00 A-44 B-70 C-C0 S-0054
P-C007 C000  (CR)

>T                                             Trace one instruction.

Op-86
P-C002 Y-DEFE X-FFFF A-44 B-00 C-00 S-004B


>T  1                                          Trace one instruction.

>T 3                                           Trace three instructions.
```

3.2.8 Move: **MOVE** [address1] [address2] [destination]

The MOVE command allows the user to copy/move memory to new memory location. If the destination is not specified, the block of data residing from addrewsw1 to address2 will be moved up one byte. Using the MOVE command on EEPROM locations will program EEPROM cells.

The MOVE command is useful when programming EEPROM. As an example, a program is created in user RAM using the assembler, debugged using the monitor, and then programmed into EEPROM with the MOVE command.

Example

```
>MOVE E000 E7FF C000            Move data from locations $E000-$E7FF to locations
                                $C000-$C7FF.
```

3.2.9 Proceed/Continue: **P**

This command is used to proceed or continue program execution without having to remove assigned breakpoints. This command is used to bypass assigned breakpoints in a program executed by the G command.

Example

```
> BR c002 c004              Set breakpoints at c002 c004.

> G c000                    Start execution from c000.

The execution breaks at C002.
> P                         Continue execution of next instruction.

The execution breaks at C004.
> P
```

3.2.10 Help: **HELP**

The HELP command enables the EVB command information to be displayed on the terminal for quick reference.