# Introduction to Microcontrollers

Motorola M68HC11 Specs
Assembly Programming Language
BUFFALO

## Topics of Discussion

- Microcontrollers
- M68HC11
- Package & Pinouts
- Accumulators
- Index Registers
- Special Registers
- Memory Map
- I/O Registers
- Instruction Set
What is a Microcontroller?

- Computer on a chip:
  - CPU, memory, I/O devices, timing devices
- 68HC11 most widely used
- Used in automobiles, appliances, instruments, industrial controllers
- Control applications vs data processing

Microcontroller Systems

- Buffers: condition I/O to proper levels
- Bus: data, address, control signals
- Clock: Fixed-frequency timing signal
- MCU: CPU, Memory, Registers connected by internal bus
  - CPU: controls microcontroller operations
  - Memory: program & data storage
    - ROM, RAM, EEPROM
  - Registers: temporary storage for data manipulation
Motorola 68HC11
MicroController Unit

The CPU: 68HC11E9
- 8-Bit word size
- 6800 Instruction set w/extensions
- Modes of Operation:
  - Single Chip
  - Expanded
  - Special Boot
  - Test

What about memory?

On Chip Memory:
(Refer to Chapter 4 in HC11 Reference Manual)
- 12k EPROM
  - User Program & Data
- 512 of EEPROM
  - Semipermanent data, calibration tables
- 256-512 RAM
  - variables & temp
68HC11E9 Pin Connections

- Ground VSS Pin 1
- E Clock Pin 5
- PORTA Pins 34-27
- PORTB Pins 42-35
- PORTE Pins 43-50

Storage Capabilities

**Accumulators**
- A, B 8-bit accumulators, location of math and logic functions
- D 16-bit accumulator, actually same physical device as A and B taken together

<table>
<thead>
<tr>
<th>A7</th>
<th>A0</th>
<th>B7</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccA</td>
<td>AccB</td>
<td>AccD</td>
<td></td>
</tr>
<tr>
<td>D15</td>
<td>....</td>
<td>D0</td>
<td></td>
</tr>
</tbody>
</table>

**Index Registers**
- X, Y 16-bit registers, used for indexed addressing (may be used as general storage registers or counters)
Special Registers

P 16-bit program counter, keeps track of address of next instruction to be executed.

S 16-bit stack pointer, used for temporary storage of the program counter (needed for subroutines)

C 8-bit condition code register. Stores several 1-bit flags (status indicators), and interrupt masks

More 68HC11 info

• Input/Output: (Memory mapped)
  • 5 parallel digital I/O ports
  • RS-232 (connect to terminals or PCs)
  • High Speed Synchronous Serial
  • Parallel Handshake (Printer)

• Timer functions
  – Generate precise digital signals
  – Measure frequency and pulse width
  – Count external events
  – Real Time Clock

• 8-channel, 8-bit Analog to Digital Converter
I/O Registers

<table>
<thead>
<tr>
<th>Register</th>
<th>Base Address</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>PORTA</td>
<td>$1000</td>
<td>I-0,1,2 O-3,4,5,6 BI-7</td>
</tr>
<tr>
<td>PORTB</td>
<td>$1004</td>
<td>Output Only</td>
</tr>
<tr>
<td>PORTC</td>
<td>$1003</td>
<td>BI</td>
</tr>
<tr>
<td>DDRC</td>
<td>$1007</td>
<td>Def: 0=I, 1=O</td>
</tr>
<tr>
<td>PORTD</td>
<td>$1008</td>
<td>BI</td>
</tr>
<tr>
<td>DDRD</td>
<td>$1009</td>
<td>Input Only</td>
</tr>
<tr>
<td>PORTE</td>
<td>$100A</td>
<td></td>
</tr>
</tbody>
</table>

Instruction Set Introduction

Refer to Appendix A of HC11 Reference Manual

Move Instructions
- Load   Load value (or contents of memory location) TO Accumulators
- Store  Store values FROM Accumulators TO Memory
- Transfer Transfer contents of Accumulators
- Stack Instructions Push and Pull
- Clear Instructions Clear Accumulators
Arithmetic Instructions
   Addition, Subtraction, Division

Logic Instructions
   AND, OR, NOT

Shift Instructions
   Shift Left, Shift Right, Rotate Left, Rotate Right

Control Instructions
   Branches

Addressing Modes

Immediate Mode

Direct Addressing
   (8-Bit) “Page 0 Addressing”

Extended Addressing
   (16-Bit)

Indexed Addressing
   Use index registers X,Y
Immediate Mode

data immediately follows instruction
Use # before number constant

Example:
LDAA #$64       loads value $64 into AccA
LDAA #100       same as above 100 = $64

AccA $64

Direct Addressing (8-bit) Mode

8-bit address of data follows instruction
Also called Page 0 Addressing because assumes most significant nibble is 00

Example:
LDAA $64       loads contents of memory address $0064 into AccA

AccA $0064

$0065
**Extended Addressing (16-bit) Mode**

16-bit address of data follows instruction

Example:

```
LDAA $100A
```
loads contents of memory address $100A into AccA

Also:

```
PORTE EQU $100A
```
```
LDAA PORTE
```
same as above

```
AccA $1009 $100A
```

---

**Indexed Addressing Mode**

the operand’s address is the content of the index register plus the *constant* given in the instruction

Example:

```
LDX #$1000
```
loads value $1000 into X

```
LDAA $A,X
```
loads value at memory location \([[$1000] + $0A]\) into AccA

```
AccA $1009 $100A
```

---
Programming the 68HC11

- Know your design goals & criteria
- Write code and save as text file with .A11 extension
- Either:
  1. Assemble code with AS11.exe
     
     AS11.EXE filename.a11 -L CRE >filename.lst
     
     Load .LST file
  
  or

  2. Use onboard assembler

Getting Started

First make sure you have a copy of AS11.EXE (from the lecture webpage- see Lecture7)

For easier use put it in the directory where you have your .A11 files (like your floppy disk)

Next run a MS-DOS Command Prompt Window by selecting Start:Run: command.exe

This brings up a command prompt window. Change the working directory to the location of your .A11 files.
Assemble your .A11 file using the command line:
    As11.exe filename.a11 -L cre>filename.lst

This creates a .lst file which you can use for de-bugging. It also creates a .S19 file which is loaded into the HC11.

Open your .LST file in notepad and check for errors. HINT: If your .S19 file is 0 bytes, you probably had errors in your .A11 file.
Sample Code

A11File

Comments

* in first column
; after commands

Labels in first column

Commands tabbed to right

Assembling count.a11

COUNT A11  Count pulses at an input.
*    Two digit bcd output.
*    Bruce Hoeppner  11/10/92
*    Bounceless input at bit 0 of Port E
*    Output to Port B
PORTB   eq $1004
org  $C000  ;origin in user RAM
* Initialize
MAIN    cd
sta PORTB
* Loop while input = 0
WAIT0   ldat $100a  ;read input
         andb #$01    ;mask off 7 msbs
         bne WAIT0
* Loop while input = 1
WAIT1   ldat $100a  ;read input
         andb #$01    ;mask off 7 msbs
         beq WAIT1
         adda #$01    ;increment AccA
         daa ;adjust for bcd
         staa $1004  ;write to Port B
         jmp WAIT0

Created after using As11.exe
Useful for finding errors in code
INNOTOUT.A11

INNOTOUT.A11  Read word, NOT it, Write it

* Definitions
PORTB equ $1004

* Load program into 8k user RAM
START org $C000

* Read 8-bit word from PortE into AccA
ldaa $100a

* Complement the word.
coma

* Write the word to PortB
staa PORTB

* Jump back to beginning of program
jmp START

* End of INNOTOUT.A11

Bruce Hoeppner 01 JAN 94

Read an 8-bit word from PortE.
Complement the word.
Write the word to PortB.
Loop Continuously.
Essentially this turns the HC11 into an
Octal Inverter.

Definitions
PORTB equ $1004

----------------------------------------

Load program into 8k user RAM
START org $C000

Read 8bit word from PortE into AccA
ldaa $100a

Complement the word.
coma

Write the word to PortB
staa PORTB

Jump back to beginning of program
jmp START

End of INNOTOUT.A11
Mult4bit.lst

Assembling mult4bit.lst

0001 * MULT4BIT.A11 Multiply using repeated addition
0002  * P = M x N
0003  * Bruce Hoeppner 10/7/89
0004
0005 c000                      org $C000   ;origin in user RAM
0006
0007 c000 f6 10 0a ldab $100a   ;load M & N into AccB
0008 c003 17 tba           ;Copy B to A
0009 c004 c4 0f andb #$0f   ;Mask off M from N
0010 c006 44 lsra           ;Move M to 4 LSBs
0011 c007 44 lsra
0012 c008 44 lsra
0013 c009 44 lsra           ;Move M to 4 LSBs
0014 c00a 84 0f anda #$0f   ;Mask off N from M
0015 c00c b7 d0 00 staa $d000   ;Store accA in temp
0016 c00f 4f clra           ;clear accumulator A
0017
Continued on next slide

Mult4bit.lst

0018 c010 cl 00 multiply cmpb $00   ;check for N = 0
0019 c012 2e 03 bgt addem        ;branch to addem when
0020  * accB > 0
0021 c014 7e e0 0a jmp $e00a      ;when N = 0 you are done
0022  * ;go back to BUFFALO
0023
0024 c017 bb d0 00 addem        ;accA = accA + M
0025 c01a 5a decb        ;decrement accB
0026  * ;accB = accB - 1
0027  * ;N = N - 1
0028
0029 c01b b7 d0 01 staa $d001   ;store result P
0030
0031 c01e 20 f0 bra multiply
0032
0033 c020 end                  ;end of MULT.A11
Bit User Fast Friendly Aid to Logical Operations

- BUFFALO Quick Reference (9-19)
- Numeric Values assumed HEX
- ASM <addr> Onboard assembler
  - <CR> Next line of code
  - <backspace> abort onboard assembly
- G <addr> Go, execute
- help

More BUFFALO

LOAD T download assembled file
MD <addr> memory display
MM <addr> memory modify
  - <space> next memory location
  - <CR> return to BUFFALO
RM register modify
T <n> trace (NOTE: set P before trace)
Using the On-board Assembler

0011  * Definitions
0012 0014  PORTB equ $1004  * Only recognizes HEX values
0015 0016  * Load program into 8k user RAM
0017 0018  c000             START     org  $C000
0019 0020  b6 10 0a          ldaa $100a
0021 0022  c003 43            coma
0023 0024  b7 10 04          staa PORTB  * Write the word to PortB
0025 0026  7e 00 00          jmp START  * Jump back to beginning of program

* Only recognizes HEX values
ASM C000 <CR>
ldaa 100a <CR>
coma <CR>
staa #1004 <CR>
jmp C000

More Fun and Educational Reading

- HC11 Reference Manual
- BUFFALO Monitor Quick Users Reference
- 68HC11 Instruction Set Introduction
- Sample Programs
- Search Motorola’s Website for 68HC11
Introduction to Microcontrollers

Experiment #7: 68HC11 Introduction

Purpose:
Learn to use microprocessors and microcontrollers, particularly the Motorola 68HC11.
Learn to program in 68HC11 assembly language.
Become familiar with programming the M68HC11.

Preparation:
Read the entire section of this laboratory exercise in this Laboratory Manual. Also read and familiarize yourself with the sections in the Class Notes pertaining to Microprocessors.
Prepare data for each experiment section of this lab. Indicate a specific test plan for each experiment.
Edit a text file containing the 68HC11 assembly language program: COUNT.A11.
Download AS11.EXE from the class website. Assembly the program using the AS11.EXE assembler.

>AS11 COUNT.A11   -L CRE  >COUNT.LST

Check COUNT.LST for errors. Correct any errors, and re-assemble.
Design a two digit (decimal) 7-segment display to connect to HC11-143 PortB connector.
Paste, tape or staple copies of your LST files into your notebook. Reminder: Bring to lab: a printout of COUNT.LST, a floppy disk containing your source file, COUNT.A11, and object code file, COUNT.S19.

Experiment Procedure:

1. 68HC11 Start-Up and Test
Measure the E clock frequency and duty cycle of the M68HC11EVB
Simple Output:
Use the BUFFALO monitor's Memory Modify (MM) command to change an output port.
Simple Input:
Use the BUFFALO monitor's Memory Display (MD) and/or Memory Modify (MM) commands to change and display the binary value at an input port.

2. On-Board Assembly -- Arithmetic: Multiplication using repeated addition.
Enter program MULT4BIT.A11 into the EVB's memory using the on-board assembler (ASM C000).
Connect PortE pins to logic switches. N is 4 lsbs. M is 4 msbs.
Run the program using the Go (G C000) command.
Use the Memory Display (MD) command to display the product at address $D001 (P).
Repeat steps b through d for a several values of M and N.

3. Uploading Programs -- Counter program.
Uploading programs from PC to EVB. (If PC is not available enter COUNT.A11 into the EVB using the on-board assembler.)
Turn off power to the EVB.
Connect PortE bit 0 to a push-button on the CADET. (A pull-up resistor is needed.)
Connect your two-digit display to PortB of the EVB.
Execute the Count program.