

ASSEMBLER AND MONITOR

Introduction and Objectives

This laboratory introduces you to the following MC9S12 assembly language programming tools:

- 1) The Freescale MC9S12 assembler **as12**. This runs on the PC.
- 2) The **D-Bug 12 monitor** that runs on the MC9S12.

An assembler takes assembly language code in a form that a human can reasonably read with a little practice, translates it to machine code which a microprocessor can understand, and stores it in a **‘.s19’** file which the MC9S12 monitor program can understand. In this lab we will use the Freescale as12 assembler, which is on the PCs in the Digital/Microcontroller lab. The as12 assembler is also included on the CD-ROM which comes with your MiniDRAGON+ board.

The **as12** assembler produces an output file with an **.s19** extension which can be loaded into and run on the MC9S12. The **D-Bug12** monitor, running from 9S12 EPROM, load S19 record into the 9S12 and provides some tools for debugging loaded programs.

Pre-Lab

Read this entire lab, and answer all of the question in the pre-lab section **before coming to the lab**. The pre-lab questions are repeated on a separate page to help you prepare for lab.

The D-Bug 12 monitor allows you to interact with the 9S12 microcontroller. You can use it to load programs, to find out what is in the 9S12’s registers and memory, to change the contents the registers and memory, and many other things. The **D-Bug 12** commands of interest for this lab are:

ASM, BF, BR, NOBR, G, HELP, LOAD, MD, MDW, MM, MMW, RM, RD, and T.

Read the description of these command in [Chapter 5 of the D-Bug 12 Reference Guide](#).

Questions to answer before lab:

- What is the difference between the MM and MMW commands?
- How would you change the values of the X accumulator to 0x55AA? (There are several ways to do this; you only need to find one way.)

Assembly language is used to write programs for the 9S12. An assembly **language instruction** will be converted by the assembler into a single **machine instruction** for the 9S12. The assembly language commands of interest to us for this lab are:

LDAA, STAA, TAB, ABA, and ASRA.

Read the descriptions of the commands in the [MC9S12 CPU12 Core Users Guide](#). At this point you will not understand everything the guide says about these instructions, but you should understand enough to get you through this lab.

Figure 1 is a simple program for the 9S12. We will use it to illustrate how to use the assembler, the simulator, and the D-Bug 12. Note: We will always use the SWI instruction to end a program. This instruction will transfer control to the 9S12 from the program to the D-Bug 12 monitor.

```
; 68HC12 demo program.
; Bill Rison 1/18/06
; This is a program to add the numbers in memory locations $2000 - $2003, divide the sum by four,
; and store the result in memory location $2004.
```

```
                title    "LAB 1 Demo Program"

prog:           equ     $1000        ; Start program at beginning of RAM
data:           equ     $2000        ; Memory area for data

                org     prog         ; set program counter to 0x1000
                ldaa   input         ; Get first input data into ACC A
                adda   input+1       ; add next value
                adda   input+2       ; add next value
                adda   input+3       ; add next value
                lsra                   ; Divide by 2
                lsra                   ; Divide by 2
                staa   result        ; Save the result
                swi                    ; End program (return to D-Bug 12 monitor)

                org     data
input:          dc.b   $15,$63,$24,$3F ; first input data
result         ds.b   1                ; reserve one byte for results
```

Figure 1. An assembly language program used to get started with the as12 and D-Bug 12.

Question to answer before lab:

- What are the contents of the A register after each instruction of the program shown in Figure 1 executes?

We will use D-Bug 12 to explore the memory of the 9S12 on your evaluation board. The memory map for you MC9S12 is shown in page 24 of the [MC9S12DP256B Device Users Guide](#). (Look at the figure labeled “Normal Single Chip”). Much of the memory is used by the D-Bug 12 monitor. For this class you can use RAM from 0x1000 to 0x2FFF, and EEPROM from 0x0400 to 0x0FFF.

The getting started.pdf file which comes on the CD with your MiniDRAGON+ board shows how to install and use the AsmIDE software on your personal computer. Use the AsmIDE is also discussed in Section 3.6 of Huang. Use the AsmIDE or a text editor to enter the program shown in Figure 1 before coming to lab and save it under the name lab01.asm.

To assemble the program double-click on the AsmIDE icon on your desktop. Open your file with the [File:Open](#) menu option. Assemble with the Build:Assemble menu option.

The **as12** assembler will create a file called lab01.lst which shows what op codes were generated by the assembler, and a file called lab01.s19, which is the file you will use with the 9S12 evaluation board.

The Lab

Create a directory for this course (say, D:\EE308). Inside this directory create a subdirectory for this lab (say, D:\EE308\Lab01). Change to this directory, type in the lab01.s program, and assemble it with the commands shown above.

Questions on the output of the assembler

- Look at the lab01.lst file. Where will the machine code for the instruction staa average be stored?
- What machine code is generated for the staa average mnemonic? Is this what you expected? (Look up the STAA instruction in your [MC9S12 Core Users Guide](#) to determine what code this instruction should generate.)
- At what address will average be located in the 9S12 memory?

Connect your MiniDRAGON+ board to your computer using the included serial cable. Make sure AsmIDE is running, and that Terminal window is active. Power up the MiniDRAGON+ board. You should see the following in the Terminal window:

```
D-Bug12 4.0.0b29
Copyright 1996-2005 Freescale Semiconductor
For Command type "Help"
```

- Use the MM command to put 0x55 into memory location 0x2000 and 0xAA into memory location 0x2001. Use the MD command to verify that this was done.
- Use the MMW command to put 0x55 into memory location 0x2100 and 0xAA into memory location 0x2101. Use the MD command to verify that this was done.
- Use the BF command to load 0x55 into memory location 0x2800 to 0x2FFF. Use the MD command to verify that it worked.
- Use the ASM 1000 command to enter the following simple program at address 0x1000. (What does this program do?).

```
clra
clrb
inca
incb
aba
swi
```

- Use the RM command to change the value of the Program Counter to 0x1000, the address of the first instruction of the simple program.
- Use the Trace T to trace (single step through) the first four instructions of the simple program. Verify that the values in A and B are what you expect.
- Use the G 1000 command to run your entire program.

Load your program into the EVB. To do this, type LOAD into the terminal window, then use the Build:Download menu option to send the program to the EVB.

In the Terminal window type ASM 1000 followed by the ENTER key. You should see the first instruction of your program, along with its address and machine code. Each time you hit ENTER you should see the next instruction in the program. To exit this mode, type a period before hitting ENTER.

You can trace (execute your program one step at a time) through your program by setting the PC (Program Counter) to the address of the first instruction of your program, and then use the T (Trace) command. Trace through your program and observe what is happening to the register and memory. (Use the MD command to display memory.)

How do the contents of the A register compare to what you predicted in the Pre-lab after the execution of each instruction?

When you are done tracing through your program, reload it and run the entire program by giving the command: G 1000. Verify that the program worked correctly.

Change the two lsra instructions to asra instructions. Rerun your program. Does it give a different result? Why?