**EE475 Lab #1 Fall 2003**

**Using the COSMIC C Integrated Development Environment.**

The object of this lab is to learn how to set up a software project using the integrated development environment, build the software, and run the executable on the HC12 evaluation board. In this lab you will also learn how to call D-Bug12 subroutines from within your C program.

## Logging On:

1. Wake up the computer (or turn on the power if it is off).
2. Begin by pressing Ctrl+Alt+Delete (Windows XP).
3. Enter your username and password, and select the proper domain.

## Communicating with the EVB using HyperTerminal:

1. Get an HC12 EVB and power supply from the shelf. Attach the power cable and the proper serial cable from the computer. Make sure the serial port switchbox is set correctly.
2. Click on the HyperTerminal icon labeled `HC12.ht` on the desktop.
3. Move the cursor to the HyperTerminal window by clicking on the window with the mouse.
4. Turn the EVB power supply on.
5. Press the RESET button on the EVB and you should see the D-Bug12 monitor prompt.

## Setting Up a Cosmic Software Project:

1. Make a directory under `c:\EEClasses\EE475` called `temp`**`abc`**, where **`abc`** are your **initials**.
2. Copy the Lab #1 files (`dbug12.h`, `lab1.c`, `lab1.lkf`, `crts.s`) from the class web site to your `tempabc` directory.
3. Launch the Cosmic integrated development environment**: Start -> Programs -> Cosmic Tools -> IdeaCPU12.**
4. Click on the **Project** Menu and Select **New**.
5. Click on the **Setup** Menu and select **Working Directory.** Using the path editor, select the path to your `temp`**`abc`** directory and then click OK.
6. Right click on the Files icon in the left column. Select **Add File**. Go to the `temp`**`abc`** directory and **open** `lab1.c`.
7. Again, right click on the Files icon in the left column. Select **Add File**. Go to the `temp`**`abc`** directory and **open** `Crts.s` (you will have to change the *files of type* to 's' files (*.s) to see this file).
8. Click on the plus sign next to the Files icon to see what files have been added. You will notice the files are colored red. This means the files need to be compiled.
9. Double click on the `Lab1.c` file to bring it to the editor window.

## Configuring the tools:

1. Click on the plus sign next to the Tools icon to expand the tools section.
2. Click on the plus sign next to the **Compiler** icon to expand the compiler section.
3. Right click on the compiler options icon. This brings up a window where you can select any compiler options you want. Don't change anything and just click **OK** since we will use the default options.
4. Click on the plus sign next to the **Linker** icon to expand the linker section.
5. Right click on the linker options icon. This brings up a window, which you will have to edit.
    a. In the linker configuration window, check the box next to *Output to File*. Then click the **Find** button to the right. Go to the `tempabc` directory and then type `Lab1.h12` in the file name text box**.** Click on the **Save** button.
    b. In the linker configuration window, check the box next to *Command File.* Then click the **Find** button to the right. Go to the `tempabc` directory and click on `Lab1.lkf` which will put it in the file name text box. Click on the **Save** button.
    c. In the linker configuration window, check the box next to *Create Map File*. Then click the **Find** button to the right. Go to the `tempabc` directory and then type `Lab1.map` in the file name text box. Click on the **Save** button.
    d. Finally, in the linker configuration window click the **OK** button. Notice that `Lab1.h12` is put next to the Project Target File Name icon in the top part of the left column.

6. Right click on the linker command file to bring the command file to the editor. You will eventually edit this type of file to place code exactly where you want it in memory.
7. Click the plus sign next to the Builder icon to expand the builder section.
8. Right click on the builder options icon. This brings up a window, which you will have to edit.
    a. In the builder configuration window, check the box next to *Run Object Inspector*. Then click the **Options** button to the right. Click on the gray line that has: *output symbol table*. Click on the **OK** button.
    b. In the builder configuration window, check the box next to *Convert to S-Records*.
    c. Finally, in the builder configuration window click the **OK** button.

9. **Save** the project (lab1.prj) to your `tempabc` directory after configuring it so you won't have to reenter everything again!!

10. Now we are ready to build our project. To do this Click on the Builder icon (derrick symbol) which is the fourth button from the right on the tool bar. You will notice that two windows pop up. One window gives the symbol locations where the compiler put these symbols in memory. The other window (lots of hex data) gives the S-Record, which you will download to the HC12 evaluation board.

## Downloading the S19 File to the EVB:

1. Copy the contents of the S-Record Window (highlight all the lines and select 'copy').
2. Move the cursor to the HyperTerminal communications window, type **load**, and press enter.
3. Pull down the **Edit** menu and click on **Paste to host**. This will send what you copied from the S-Record Window to the EVB. Wait until the prompt reappears.

## Run the Program:

1. Run the program by typing *g 4000* in the HyperTerminal window (g<space>4000).
2. To interact with the program, type a line next to the prompt and hit return. You should see the line echoed back in all caps.

**Note:** To get the command prompt back, you will need to add the following line to the end of the program (this is similar to the exit(0) function in C, which gets you back to the D-bug12 prompt):

```
_asm("swi");
```

Otherwise you will have to press the reset button on the evaluation board to be able to interact and load new/revised programs to the board.

---

**Problems to do in lab:** For the following lab problems you need to demonstrate a working program to get credit for it. Have the instructor sign the verification sheet for each working assignment that is completed. Each problem is worth 10 points.

**Problem #1**: Modify the `Lab1.c` program so that in addition to echoing the input string it will also return each letter of the input on a separate line. You will want to look in the EE475 Lab Manual, which contains the document "Using and Extending D-Bug12 Routines". This will list the functions that D-Bug12 supplies that you can use in your C programs when running them on the HC12 evaluation board.

**Note:** Any D-Bug12 C function you call needs to be preceded by: DBug12FNP-> e.g. type DBug12FNP->printf("..."); when you want to use the printf function.

For example, your program should take the input:

- **Hello**

And return: (don't worry that they are all caps)

- **H**
- **E**
- **L**
- **L**
- **O**

**Problem #2:** Modify the `Lab1.c` program so that the input string is also echoed in *reversed* order. That is, an entry of

**Hello**

results in an output of

**OLLEH**

**Problem #3**: Modify the `Lab1.c` program so that it will also return the location of the "RAM vector interrupt table." Have the program print the following statement:

**The RAM vector interrupt table is located at address xxxx.**

The value xxxx is what you need your program to determine. Make sure that you print out the address in hexadecimal with 4 hexadecimal numbers.

Normally, the interrupt vector table is located at 0xffce. Since this location is in EPROM, it can't be modified when using D-Bug12. What D-Bug12 does when an interrupt occurs is to check a specific location in RAM to see if there is an address other than zero. If it is zero, it goes to the default interrupt routine specified by D-Bug12. If there is some other address, it goes to that address. Thus we can modify where the evaluation board goes for its interrupt service routines.

Hint: Use the SetUserVector() routine supplied by D-Bug12.

**Factory-Configuration Memory Map**

| Address Range | Description | Location |
|---|---|---|
| $0000 - $01FF | CPU Registers | on-chip (MCU) |
| $0800 - $09FF<br>$0A00 - $0BFF | user code/data<br>reserved for D-Bug12 | 1K on-chip RAM (MCU) |
| $1000 - $1FFF | user code/data | 4K on-chip EEPROM (MCU) |
| $4000 - $7FFF | user code/data | 16K external RAM (U4, U6A) |
| $8000 - $9FFF<br>$A000 - $FD7F<br>$FD80 - $FDFF<br>$FE00 - $FE7F<br>$FE80 - $FEFF<br>$FF00 - $FF7F<br>$FF80 - $FFFF | available for user programs*<br>D-Bug12 program<br>D-Bug12 startup code*<br>user-accessible functions<br>D-Bug12 customization data*<br>available for user programs*<br>reserved for interrupt and reset vectors | 32K external EPROM (U7, U9A) |

*Code in these areas may be modified. Requires reprogramming of the EPROMS.

**Lab Report: Due at THE START OF THE LAB PERIOD NEXT WEEK.**

The lab report is to be written up in the Memo format. Each student should submit a separate lab report. For each problem, write a short description of what you did to solve the problem. Include **commented** C code *excerpts* for each problem and attach this to the memo.

**Note:** You will need to include the instructor verification sheet to get any credit for the lab.

**Instructor Verification Sheet**
**EE475 Lab #1**
**Fall 2003**

**Student Name:**

| | **Instructor Signature** | **Date** |
|---|---|---|
| Assignment #1 runs correctly | | |
| Assignment #2 runs correctly | | |
| Assignment #3 returns correct RAM vector address | | |

**Note:** This verification sheet must be signed by the instructor and submitted with the lab report to get any credit for the lab.