EE475 Lab #4 Fall 2003

Foreground and Background Processing

The object of this lab is to implement a simple program with a foreground process and a background process running concurrently on the HC12 evaluation board. You will need to create a new project for the following assignments as you did for previous labs.

Preliminaries

1. Make a temporary local folder for your work: `c:\EEClasses\EE475\tempxxx`.
2. Launch the Cosmic CPU12 program. You will need to create a new project as you did in Lab #3. If you still have a copy of the `lab3.prj` project, open it, then use `Save As`... to make a new `lab4.prj` project file for this week without having to re-enter all the parameters. *Remember that you will need to remove the `lab3.c` file from the project file list and edit the linker directive (`.lkf`) file to use lab4.o, etc.*
3. Change the program start address specified in the `.lkf` file to be 0x1000.

Exercise #1:

Using the D-bug12 manual, determine the `UserIRQ` location in the RAM interrupt vector table on the HC12 EVB. Knowing this location will allow you to set the interrupt vector in your program *without* using D-bug12's routines. This is what you would normally do when programming an embedded system in C since the vector table will be at a known location.

Verify the location using a C program to write into that address and observe the table using the monitor.

Hint: In your C program use the `SetUserVector()` routine supplied by D-bug12 (see Lab #1) and set the `UserIRQ` interrupt routine address to some number other than zero (e.g. 0x2222). Then look at the RAM table (you discovered where it started in lab #1) using the monitor's `md` command and see what address contains 0x2222 (everything else will be zeros). Remember that in your `main()` routine you will need to end the program with the line:

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

This will give back the D-Bug12 prompt so you can look at the memory. If you don't do this you will have to hit the reset switch, but this zeros out the RAM vector table!

→ Show the instructor where the UserIRQ address is located in the RAM vector table - What address is this? Give your answer in hexadecimal.

Exercise #2:

Implement the following foreground/background system. For the background task, implement an infinite loop that prints out an integer that is incremented by 1 on each line:

```
1
2
3
4
…
```

Figure out a way to put a delay within the loop so that next integer appears approximately once every second. Create the interrupt-driven foreground task so that the statement:

```
An IRQ Interrupt Occurred
```

is printed out whenever you press the IRQ switch.

To do this you will have to define an interrupt service routine (ISR) function by using the type qualifier `@interrupt`. This is done as follows:

```
@interrupt void your_function_name(void)
{
      … your ISR code …
}
```

Then, to put the address of this interrupt function where you want it (in the address location you just determined in Exercise #1), you can again use the `SetUserVector` monitor function.

```
DBug12FNP->SetUserVector(UserIRQ,(Address) your_function_name);
```

Keep in mind that after your program sets the interrupt vector to point to your function, you will need to enable (unmask) interrupts on the HC12 using:

```
_asm("cli");
```

→ Demonstrate your program for the instructor: integers being printed out in ascending order, and the statement **“`An IRQ Interrupt Occurred`”** printed out when the IRQ switch is pressed. Also show that your program runs correctly after the RESET button is pressed.

Exercise #3:

Now modify your program so that you can set the interrupt vector table *without using the D-bug12* `SetUserVector` *function*. This requires the use of an absolute address and some interesting pointer flim-flam--which is necessary if the target system does not have the D-bug12 monitor routines.

The expression

```
*(void **) 0x2222 = your_function_name ;
```

will cause the starting address of your ISR to be stored into location 0x2222 (where you need to replace 0x2222 with the *correct* address in the vector table!). Can you parse the expression and understand what it is doing?

Another approach is to define a macro to store a function address into the table. Try the following `#define` statement in your file:

```
#define  SetVector(isr,address)    (*(void **)(address)=(isr))
```

You will then use this defined macro in your code as follows:

```
SetVector(your_function_name, 0x2222);
```

where `0x2222` must be replaced with the address of the IRQ vector in the table.

→ Demonstrate this new version of your program for the instructor. Show that the address of the ISR is properly stored in the interrupt vector table. Also show that your program runs correctly after the RESET button is pressed.

Exercise #4:

You probably noticed that the interrupt routine gets called repeatedly if you hold down the IRQ button. This is because the default behavior for the IRQ line is *level sensitive*: the interrupt is asserted whenever the IRQ pin is pulled low.

The HC12 can be programmed to be negative-edge sensitive for the IRQ line instead of level sensitive. Look at the HC12 documentation and/or Prof. Cady's HC12 textbook to find out how to program the chip for edge-triggered IRQ operation. Modify the initialization section of your program to enable the edge sensitive behavior.

→ Demonstrate the edge-sensitive behavior of the program for the instructor: only one message should appear when the IRQ button is pressed and then the background count should continue.

Exercise #5:

It is usually good procedure to avoid spending lots of time within the interrupt service routine, since the ISR prevents any background task from running at all.

Modify your program to minimize the ISR duration: in the ISR just set a variable (a *flag*) that the background routine can poll regularly to determine if an interrupt has occurred. If the background routine finds the flag variable set, it should print out the interrupt message, reset the flag, and then continue its integer increment loop.

Think about the pros and cons of this program compared to the original program, and provide some comments regarding these issues in your report.

Lab Report

The lab report is to be written up in the Memo format. Be sure to put the *lab number* in the Memo header along with your name and date. For each exercise, answer the given questions and demonstrate your understanding of the exercise. Include **commented** file excerpts and the instructor verification sheet to get credit for the lab.
→ This lab report is due the beginning of the lab period in one week.

Instructor Verification Sheet
Lab #4   Fall 2003

Student Name:

	Instructor Signature	**Date**
#1 Identify and verify address of UserIRQ		
#2 ISR and background routine running (using SetUserVector)		
#3 ISR and background routine running (using manual table address method)		
#4 Edge-sensitive interrupt configuration.		