

EET364 Microcontroller Systems

OIT Portland West, Winter 2011

Lab Assignment #3 – Serial Communications Interface Due January 27

Objective:

The student will demonstrate using the serial communication interface (RS232) using both polling and interrupt routines.

Equipment and Software needed:

- Dragon12-Plus-USB board and power supply, and USB cable.
- Computer running Windows 2000 or later or classroom computer
- AsmIDE installed on computer or EmbeddedGNU if you are using C.
- Heartbeat routine from lab assignment 1.

General Instructions:

DEBUG-12 uses serial port 0 for communication and prohibits application programs from setting up an interrupt service routine. In this unit, use serial port 1 to avoid confusion between DEBUG-12 and program communication. You might want to check your program first using the simulator. When you are using both serial ports you will need to connect to port 0 to download your program and give the "g" command to DEBUG-12. Then you will need to change jumper J42 (while the program is running) to select serial port 1 for the USB connection.

Part 1: (75% of the grade)

Before writing the interrupt routine, we will start by using a polling interface. In this part you will need to do the following:

1. Initialize the second serial port (SCI1) to 9600 bps.
2. Wait for a character to be in the receive buffer, then load it into register A. Go directly to the fourth step.
3. If a character is available in the serial port receiver buffer, load it into register A.
4. If the transmit buffer is empty, store the character (in register A) into the transmit buffer.
5. Go back to step three.

Make sure your program is written to exactly follow these steps.

Don't forget the heartbeat monitor interrupt.

This creates a simple program that will wait for a character to be typed and then output it in a continuous stream. When another key is depressed, the program will output a continuous stream of that character. Thus it is a good way to check operation of both the RS232 transmitter and receiver. If you see nothing being displayed, you can halt the program and see if it is waiting for a character to be

typed or (thinks) it is sending a character to the display. This can help debug any problems.

Using the oscilloscope, look at the transmit data from the microcontroller. You can locate this signal on the header that encircles the breadboard on the Dragon-12 board. Knowing the character that is being sent, identify a character frame. Sketch the voltages and time between the bits and incorporate into your lab report. You may want to select a character whose bit pattern will be easy to identify.

Part 2: (25% of grade for successful completion)

After your program is shown to work successfully, write an interrupt routine that will behave like the program you have just completed. Remember that the interrupt could be caused by an input character available, the output buffer empty, or both. The character received will need to be saved in a memory location, say *ECHOCH*, since the register contents will not be preserved between interrupts. The main task (the code after the initialization) should be:

```
loop:  wai                ; wait for interrupt
        bra      loop     ; repeat
```

To insure that characters are not output until the first time a key is pressed, do not set the TE control bit as part of the initialization, but set it within the interrupt service routine after a character is received. The interrupt routine should perform the following whenever an SCI interrupt occurs:

1. Turn on the TE control bit
2. If a character is available in the serial port receiver buffer, move it to *ECHOCH*.
3. If the transmit buffer is empty, move the character in *ECHOCH* into the transmit buffer.

Now you have a program with two interrupt service routines!

Again, don't forget the heartbeat monitor interrupt.

To turn in:

- Commented program listings
- The sketch of a character frame showing voltage levels and timing. A photograph of the oscilloscope screen will do as long as you mark the start of the frame. State the character being transmitted.