Corwin Perren – 931759527
ECE 152 – Thursday, 10:00am
5/8/12
Assignment 2

Short Answer: (16 points)

- (4 points) Write a one sentence definition for each of the following:
  - Tool Chain
    - A set of tools such as a linker or debugger that chain their inputs and outputs together so that you can take a source file and convert it into a working program.
  - Makefile
    - A file used to tell the compiler how to compile a source file for a specific operating system, hardware device, language standard, etc…
- (4 points) What do the following register and bit names stand for in reference to the AT90USB USART datasheet (attached)? Describe what each one does with one sentence in your own words.
  - RXENn
    - The RXENn, or Receive Enable Bit, is used to enable the receive capabilities of a particular UART on the atmel chip.
  - TXCn
    - The TXCn bit, or USART Transmit Complete bit, is a flag set when there is no longer any data left in the UDR data register and the last data present in the buffer has been sent.
- (2 points) Explain in a few sentences the differences between using the UDRE1 bit and the TXC1 bit for knowing if data has been/can be sent using USART1.
  - The UDRE1 bit is set whenever the UDR buffer is empty, whereas the TXC1 bit is set when data is finished being sent. Using the UDRE1 bit for knowing when to send data is better because when it's set, you know for a fact that the buffer has nothing in it. The TXC1 bit being set could mean that data is currently in the buffer, but what last was in there was sent successfully, meaning you could overwrite unsent data, which is not a good thing.
- (4 points) What do the following register and bit names stand for in reference to the AT90USB datasheet (attached)? Also for each write one sentence, in your own words, what each does.
  - OCR0A
    - This is one of the output/compare registers. A value is placed into it for comparison against a counter or timer value.
  - CS02:0

- This is the set of bits used to set the timer/counter clock rate. Depending on the value set here, the timer/counter can be made to update at a scaled rate compared to the speed of the processor. This can be useful when highly accurate measurements are not needed.

- (2 points) Describe the difference between = and == in C.
    - The single equals sign in C assigns a value or variable to some other variable. It makes something equal to something else. The double equals sign is used to check whether something is equal to something else and returns the outcome.

Coding: (80 points)

- (8 points) Write C statement(s) that will accomplish each of the following:
    - Clear only bit 3 of register UCSR1A
        - USCR1A &= 0b11111011;
    - Set only bit 7 of register UCSR1B
        - USCR1B |= 0b10000000;
    - Make only the lowest 3 bits of TCNT0 a 0b101
        - TCNT0 |= 0b10100000;

- (4 points) Give the proper C syntax for an enum of type states with the options of: idle, running, walking.

```
enum states {
        idle,
        running,
        walking
} currentstate;
```

- (8 points) Write the correct syntax for a while loop in C that will loop if the value of x is less than 23.

```
while(x < 23){
        //Do stuff
        x++;
}
```

- (10 points) Write the correct syntax for an if statement with an else condition. The if condition should be executed if variable y is equivalent to the character q or the character or Q. y should be reassigned to the character of z if true and the character a if not true.

```
if((y == 'q') | (y == 'Q')){
        y = z;    //Or   y = 'z'   instructions seemed unclear…
```

```
}else{
        Y = a;    //Or   y = 'a'    same reason as above….
}
```

- (12 points) What is the result for each of the following in the format specified? ASCII chart is attached

(0b10101010 | 0b11110000) & 0b11001100	**Binary: 0b11001000**
0b11111111 & (~(1<<4))	**Binary: 0b11101111**
(0b11111111 & 0b11111000) | 0b00000010	**Binary: 0b11111010**
'a' & 0b11011111	**ASCII: 'A'**

- (14 Points) For every line of the code below, explain what each line does in details. Do not skip any lines.

`unsigned char SendByteUART (unsigned char data){`	This is the line that creates a function named "SendByteUART" that receives an unsigned char as a argument and returns an unsigned char upon returning.
`if (0 != (UCSR1A & (1 << UDRE1))){`	This if statement checks the buffer empty bit in the UART flag register is set. If it is, it runs the it's code o if not it moves to the else.
`UDR1 = data;`	This puts whatever is in the unsigned char "data" into the transmit buffer UDR1.
`return(0);`	This returns a 0 for the function, meaning that the data was transmitted successfully.
`}`	This is the ending bracket for the if statement.
`else{`	This is the else statement that is run if the transmit buffer already has data in it.
`return(1);`	This returns a 1 from the function, meaning there wa an error and the data was not sent.
`}`	This is the closing bracket for the else statement.
`}`	This is the closing bracket for the SendByteUART function.
`unsigned char SendStringUART(unsigned char *data){`	This line creates a function named "SendStringUART" that receives a pointer to a string or array of unsigne chars as an argument with an unsigned char return type.
`unsigned char i;`	This creates the variable i, of type unsigned char.

for (i=0;i<strlen(data);i++)	This for loop makes the variable i equal to zero and adds one to that variable each time it loops through its code until it hits the length of the data contained i the string/array passed in as an argument.
SendByteUART(data[i]);	This moves through each element in the string/array and passes it into the previous SendByteUART function
return(0);	This returns a 0 from the function once sending is complete, meaning it was successful.
}	This is the closing bracket for the SendStringUART function.

- (6 points) Assuming the code in question 11, what would happen if the main function attempted to send the string "All your base are belong to us!" using the SendStringUART() function as written? The system clock is 1MHz (fast) and the USART baud rate is 9600 (slow).
    - Attempting to send the string with the code in its current state would cause either random gibberish or only small portions of the string to be sent. This happens because the processor is updating significantly faster than the uart can send, and will read that the buffer is empty twice in a row, and then overwrite the data in the middle of being sent.

- (8 Points)How would you change the code in question 11 to allow for proper transmission of the string, "All your base are belong to us!"?

    - To combat this problem, you would place a "_delay_ms();" call after the "UDR1 = data;" line, and modify the delay amount until the string went through properly while not slowing your code too much.

- (10 points) Write a function that will setup Timer 0 to function with a divide by 256 clock setting and trigger a timer compare match A at a count of 101. The timer should also be set to CTC mode. Leave the external OC0A pin disconnected.

```
unsigned char SetupTimer(void){
        TCCR0A = (1 << WGM01);      //Set CTC Mode
        TCCR0B &= 0b11111000;       //Zeros out last three bits of clock
        TCCR0B |= 4;                //Sets clock mode to 256 divide
        OCR0A = 101;                //Sets compare match to 101
        TCNT0 = 0;                  //Zeros compare register
        return 0;                   //Returns non-error
}
```

Extra Credit:

- (4 points) In complete sentences, please explain the hardest thing so far in the course and the easiest.

    - The hardest thing to do in this course is keep track of all of the small things spread across multiple registers in order to make a specific piece of hardware function. Once you've finally gotten all of the bits set correctly, the rest of the code isn't nearly as difficult. The easiest part of this course would be writing the main function for all the projects so far. As I said before, that part is not difficult once the hardware is playing nice.