Terence Henriod

HW 8

CPE 301: Microprocessors System Design

October 22, 2013

1. Determine the appropriate bit settings for UCSR0A, UCSR0B, UCSR0C,and UBRR0 to manage a serial interface using the following specific details:
   * use normal transmission speed (i.e., disable the x2 speed),

set UCSR0A bit 1 to 0

* + disable the multi-processor communication mode,

set UCSR0A bit 0 to 0

* + turn off the *RX* complete interrupt, turn off the *TX* complete interrupt, turn off the data register
  + empty interrupt, (In fact, turn off ALL USART0 interrupts.)

set UCSR0B bits 7:5 all to 0

* + turn on the receiver, turn on the transmitter,

set UCSR0B bits 4:3 both to 1

* + set the character size to 8 bits,

set UCSR0B bit 2 and UCSR0C bits 2:1 to 0b011 collectively

* + use the asynchronous USART mode,

set UCSR0C bits 7:6 both to 0

* + use no parity,

set UCSR0C bits 5:4 both to 0

* + use 1 stop bit,

set UCSR0C bit 3 to 0

* + set the baud rate to 115200 bits per second.

Place the value 8 in the UBRRH and UBRRL registers appropriately using the code found in the solution to #2

1. Create a function that initializes the USART based on the values determined in problem 1.

// when this function is called in setup() the appropriate actions are taken

// also listed above the function are the globals necessary for the program

/\*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Global Variables (Yes, these are global!)

Mostly used for register pointers that can't be declared constant.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\*/

volatile unsigned char \*myUCSR0A = (unsigned char \*) 0xC0; // Controls/Status

volatile unsigned char \*myUCSR0B = (unsigned char \*) 0xC1; // Controls/Status

volatile unsigned char \*myUCSR0C = (unsigned char \*) 0xC2; // Controls/Status

volatile unsigned char \*myUBRR0L = (unsigned char \*) 0xC4; // Baud Rate

volatile unsigned char \*myUBRR0H = (unsigned char \*) 0xC5; // Baud Rate

volatile unsigned char \*myUDR0 = (unsigned char \*) 0xC6; // Data Register

/\*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Constants

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\*/

#define RDA 0x80 // Receive Data Available

#define TBE 0x20 // Transmitter Buffer Empty

/\*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Function Implementations

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\*/

void U0init( unsigned long ubaud )

{

// variables

unsigned long FCPU = 16000000; //assumedfrequencyoftheprocessorinHzTODO:globalconstant

unsigned int tbaud;

// calculate the value to store in the USART Baud Rate Register (this must

// follow the calculation to be a compatible value)

tbaud = (((unsigned long)(FCPU / ((unsigned long) (16 \* ubaud)))) - 1);

// clear the USART Controls and Status Register A

\*myUCSR0A = 0x20; // Set the Data Register Empty and Frame Error bits.

// clear all other flags: Receive Complete, Transmit Complete,

// Data OverRun, Parity Error, Double TX speed, and Multi-

// Processor Communication Mode

// setup USART Controls and Status Register B

\*myUCSR0B = 0x18; // only set the Receiver Enable and Transmitter Enable

// clear the Interrupt Enables (significant bits) and

// set the character size to 5 bits (less significant) TODO:setto8 bits?

// setup USART Controls and Status Register C

\*myUCSR0C = 0x06; // set an 8 bit character size (2:1)

// while setting the UMSEL (7:6) to asynchronus mode, disabling

// the Parity Mode (5:4), set the Stop Bit Select to 0 (1-bit),

// and set the UCPOL (Clock POLarity) to zero to receive sampled

// data on the falling edge of the clock and change the

// data transmitted on the rising edge

// load the Baud Rate Registers

// because arithmetic was done in decimal, we use bitwise operations to keep the desired

// parts of the number without doing messy decimal to hex kinds of conversions

\*myUBRR0H = (tbaud >> 8); // shifts the calculated value right, leaving the most significant byte

\*myUBRR0L = (tbaud & 0xFF); // keeps only the 1 bits in low byte of the calculated value

}

1. Write the three serial I/O functions defined in Lab 8.

unsigned char U0kbhit()

{

// variables

unsigned char U0stat; // state of the data register (buffer)

// ascertain if the Read Data Available

U0stat = (\*myUCSR0A & RDA);

// return whether or not a

return U0stat;

}

unsigned char U0getchar()

{

// variables

unsigned char U0gdata; // will contain the read in character

// get the character data from the data register (buffer)

U0gdata = \*myUDR0;

// return the read character

return U0gdata;

}

void U0putchar( unsigned char U0pdata )

{

// wait for all data to be read in from the data register (buffer)

while ((\*myUCSR0A & TBE) == 0){};

// once buffer is clear, load the character to be transmitted into the buffer

\*myUDR0 = U0pdata;

}