Lab05 - UART

Introduction to Embedded Systems - University of Nebraska

Dat Nguyen

Contents

1 Introduction

This lab provides hand-on activity to give in-depth knowledge of UART.

2 Program Description

2.1 Program 1 - Configure UART with RX Interrupt

Video demo: https://youtu.be/CoZB$_E$6$HYQA$

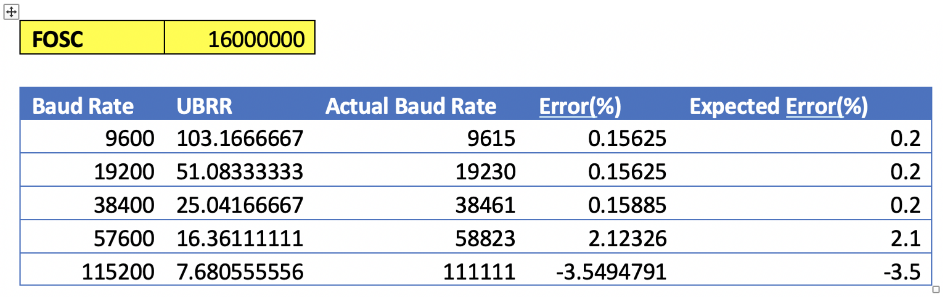
The RX interrupt is configured by setting the RX complete interrupt enable bit (RXCIE0) and global interrupt. pUCSRnA register is cleared at initialization step. Method mySerialBeginWithInterupt in the code configure the Serial interface of the program. The ISR handler is implemented by handling the USART_RX_vect to write back to the TX what was received. Note that polling implemented in mySerialWriteOne can be skip in experiment 01.

```
void mySerialWriteOne(uint8_t data) {
  #define UDREn 5 // USART Data Register Empty

  /* Wait for empty transmit buffer */
  while ( !( (*pUCSRnA) & (1<<UDREn)) );

  /* Put data into buffer, sends the data */
  *pUDRn = data;
}

ISR(USART_RX_vect, ISR_BLOCK) {
  uint8_t rxData = *pUDRn;
  mySerialWriteOne(rxData);
}
```

- UBRR value is computed as below where FOSC is the clock speed of 16MHz

```
#define BAUD2UBRR(baud) FOSC/16/baud-1
```

The table display the UBRR value and error for each baudrate value. Note that the recorded error were very accurate based of the datasheet.



FOSC	16000000

Baud Rate	UBRR	Actual Baud Rate	Error(%)	Expected Error(%)
9600	103.1666667	9615	0.15625	0.2
19200	51.08333333	19230	0.15625	0.2
38400	25.04166667	38461	0.15885	0.2
57600	16.36111111	58823	2.12326	2.1
115200	7.680555556	111111	-3.5494791	-3.5

Figure 1: UBRR value for different baudrate

- Waveform in figure 2 show the TXD signal when transmitting character ’F’ at baudrate 19200. The waveform is as expected as there is 1 low start bit, 8 databit, and 1 high stop bit at the end displayed. In this case, the character ’F’, $(70)_{10}$, or $(01000110)_2$. Note that by default the data package is sent as the LSB is transfered first.

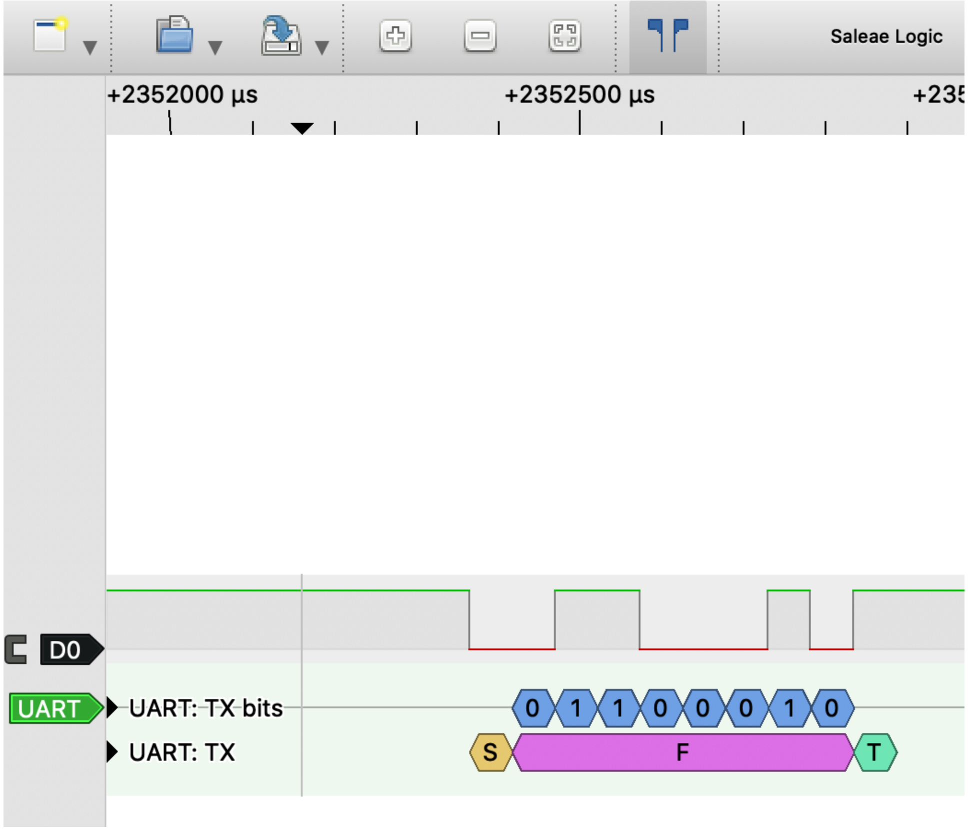


Figure 2: TXD Signal For Transmitting Character 'F' at 19200 baudrate

2.2 Program 2 - Implement Serial.Begin and Serial.Write

Video demo: https://youtu.be/LAqL2zW7DDo

Serial.begin is implemented similar as previous experiement where interrupts aren't enable. The following code attempts to mimic Serial.begin

```
void mySerialBegin(uint32_t baudrate) {
  #define TXEN0 3
  #define RXEN0 4
  #define RXCIE0 7 // RX complete interupt enable bit

  #define UCSZ0_01 1

  uint32_t ubrr = BAUD2UBRR(baudrate);
  *pUBRR0H = (uint8_t) (ubrr >> 8);
  *pUBRR0L = (uint8_t) ubrr;

  *pUCSRnA = 0x00;

  // b[1]Enable receiver and b[0]transmitter
  *pUCSR0B = (1<<RXEN0) | (1<<TXEN0);

  // Set frame format: 8bit data, default 1stop bit
  *pUCSR0C = (3 << UCSZ0_01);
}
```

Serial.write is implemented by repeatedly using mySerialWriteOne function and constantly polling to ensure their is no buffered TX data in the pipeline.

```
void mySerialWrite(uint8_t * msg) {
  while ((*msg) != 0) {
    mySerialWriteOne(*msg);
    msg++;
  }
}
```

3 Summary

This lab introduced UART0 peripheral and provided handon activity attempt to mimic the implementation of Serial.begin and Serial.write. As a result, using mySerialBegin method, I were able to use the builtin Serial.write(). Similarly, using builtin Serial.begin(), I were able to use mySerialWrite() function.

4 Appendix

4.1 Main program

```
#include <stdint.h>;
#include "expriments.h";

int main(void) {
  init();
  // experiment01();
  experiment02();
  return 0;
}
```

4.2 Experiment #1 and #2 code

```

#include <Arduino.h>
#include "avr/interrupt.h"

#define FOSC 16000000 // Clock speed
#define BAUDRATE 9600
#define BAUD2UBRR(baud) FOSC/16/baud-1

volatile uint8_t *pUBRR0L,
                 *pUBRR0H,
                 *pUCSRnA,
                 *pUCSR0B, // USART Control and Status Register 0 B
                 *pUCSR0C,    // USART Control and Status Register 0 C
                 *pUDRn,
                 *pSREG
;

void myHardDelay(uint32_t ms) {
  volatile int16_t count;

  while (ms) {
    for (count = 0; count < 835; count++);
    ms -= 1;
  }
}

void mySerialBegin(uint32_t baudrate) {
  #define TXEN0 3
  #define RXEN0 4
  #define RXCIE0 7 // RX complete interupt enable bit

  #define UCSZ0_01 1

  uint32_t ubrr = BAUD2UBRR(baudrate);
  *pUBRR0H = (uint8_t) (ubrr >> 8);
  *pUBRR0L = (uint8_t) ubrr;

  *pUCSRnA = 0x00;

  // b[1]Enable receiver and b[0]transmitter
  *pUCSR0B = (1<<RXEN0) | (1<<TXEN0);// | (1 << RXCIE0);, b[7]Set RX complete interupt enable

  // Set frame format: 8bit data, default 1stop bit
  *pUCSR0C = (3 << UCSZ0_01);
}

void mySerialWriteOne(uint8_t data) {
  #define UDREn 5 // USART Data Register Empty

  /* Wait for empty transmit buffer */
  while ( !( (*pUCSRnA) & (1<<UDREn)) );

  /* Put data into buffer, sends the data */
  *pUDRn = data;
}

void mySerialWrite(uint8_t * msg) {
//  Serial.print("strlen "); Serial.println(strlen(data));
  while ((*msg) != 0) {
    mySerialWriteOne(*msg);
    msg++;
  }
}

void configure_register() {
  pUCSRnA = (uint8_t *) 0xC0;
  pUCSR0B = (uint8_t *) 0xC1;  // USART Control and Status Register 0 B
  pUCSR0C = (uint8_t *) 0xC2;  // USART Control and Status Register 0 C
  pUBRR0L = (uint8_t *) 0xC4;
  pUBRR0H = (uint8_t *) 0xC5;
  pUDRn   = (uint8_t *) 0xC6;
  pSREG = (uint8_t *) 0x5F; // GLOBAL interupt
}

void mySerialBeginWithInterupt(uint32_t baudrate) {
  #define TXEN0 3
  #define RXEN0 4
  #define RXCIE0 7 // RX complete interupt enable bit

  #define UCSZ0_01 1

  uint32_t ubrr = BAUD2UBRR(baudrate);
  *pUBRR0H = (uint8_t) (ubrr >> 8);
  *pUBRR0L = (uint8_t) ubrr;

  *pSREG |= (0x80); // turn on global interrupt

  *pUCSRnA = 0x00;

  //b[7]Set RX complete interupt enable, b[1]Enable receiver and b[0]transmitter
  *pUCSR0B = (1<<RXEN0) | (1<<TXEN0) | (1 << RXCIE0);

  // Set frame format: 8bit data, default 1stop bit
  *pUCSR0C = (3 << UCSZ0_01);
}
```

```
ISR(USART_RX_vect, ISR_BLOCK) {
  uint8_t rxData = *pUDRn;
  mySerialWriteOne(rxData);
}

void experiment01() {
  configure_register();
  mySerialBeginWithInterupt(BAUDRATE);
  mySerialWrite("Start of program #1 \n");
  while(1);
}

void experiment02() {
  configure_register();
  mySerialBegin(BAUDRATE);
  while(1){
    mySerialWrite("Testing\n");
    myHardDelay(1000);
  }
}
```