Lab05 - ASM

Introduction to Embedded Systems - University of Nebraska

Dat Nguyen

Contents

1 Introduction **3**

2 Program Description **3**
2.1 Program 1 - Configure UART with RX Interrupt 3

3 Summary **4**

4 Appendix **4**
4.1 Main program 4
4.2 Experiment #1 4

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/csWgLcUvTQo

```
1  void myHardDelayUsec(uint16_t delayInUsec) {
2    uint16_t unused;
3    asm volatile (
4      "loop:              \n\t"
5      "sbiw %0,1          \n\t" // 2 clk from subi. (
6      "nop                \n\t" // 1 clk
7      "nop                \n\t"
8      "nop                \n\t"
9      "nop                \n\t"
10     "nop                \n\t"
11     "nop                \n\t"
12     "nop                \n\t"
13     "nop                \n\t"
14     "nop                \n\t"
15     "nop                \n\t"
16     "nop                \n\t"
17     "nop                \n\t"
18     "brne loop          \n\t" // 2 clk. Last loop is 1clk.
19     : "=w" (unused)
20     : "0" (delayInUsec) // placeholder. at first, gcc will load delayInUsec into 2
         register and pass it to %0
21   );
22 }
```

The delay is implemented by delaying the program by 1uS multiple time. Upon enter the delay method, gcc load the 16bit delay value to 2 register and replace %0 in the assembly template. Multiple nop instruction are executed to model 1uS delay.

1. Port PB1 is configured as output and toggle with a time frame in between of 100uSec. Figure below display the output signal of pin PB1.

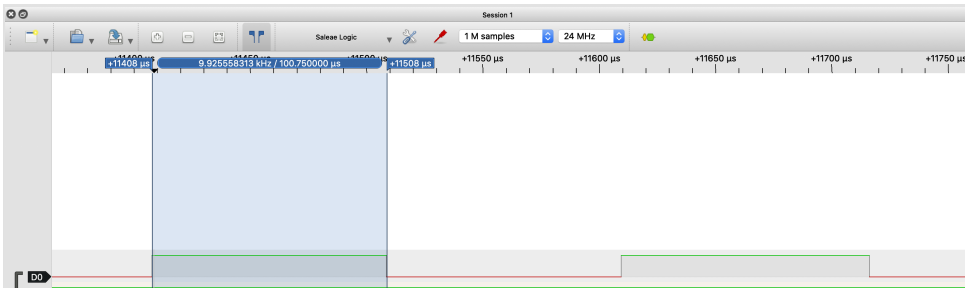

Figure 1: UBRR value for different baudrate

2. Similarly, below table contains timing mesurements of myHardDelayUsec function for different input arguments: 1, 10, 100, 1000, 10000.

Expected delay (uS)	Actual delay(uS)	Error (%)
1	1.75	-0.75
10	10.75	-0.075
100	106.6	-0.066
1000	1007.33	-0.00733
10000	10069	-0.0069
43459	43712	-0.0058216

Figure 2: UBRR value for different baudrate

3. The accuracy fluctuate with 7 or below percentage error with an exception of 75 percentage error when executing 1uS delay. Reducing number of nop from 12 to 11 will result in a better delay below 300 uS but will create a larger error as the delay value increase.

4. Based on the delay value N, each loop execution will result in average of 16 cycles(SBIW, NOP, BRNE). Upon enter and exiting the method, loading the 16bit values to register will result in 4 cycles(2 STD instruction) with 1 cycle of the BRNE at the end. In total, the number of cycle can be formulated as follow for N ¿ 0

$$cycles = (N-1) * 16 + 4 + 1$$

3 Summary

This lab introduced the usage of inline ASM in arduino to optimize the number of CPU cycles in a program.

4 Appendix

4.1 Main program

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

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

4.2 Experiment #1

```
#include <Arduino.h>

#define DELAY_VALUE 43459
/*
 * the free register are r16 - r31
 */
volatile uint8_t *ioPORTB, *ioDDRB;
```

```
8
9  void myHardDelayUsec(uint16_t delayInUsec) {
10   uint16_t unused;
11   asm volatile (
12     "loop:           \n\t"
13     "sbiw %0,1       \n\t" // 2 clk from subi. (
14     "nop             \n\t" // 1 clk
15     "nop             \n\t"
16     "nop             \n\t"
17     "nop             \n\t"
18     "nop             \n\t"
19     "nop             \n\t"
20     "nop             \n\t"
21     "nop             \n\t"
22     "nop             \n\t"
23     "nop             \n\t"
24     "nop             \n\t"
25     "nop             \n\t"
26     "brne loop       \n\t" // 2 clk. Last loop is 1clk.
27     : "=w" (unused)
28     : "0" (delayInUsec) // placeholder. at first, gcc will load delayInUsec into 2
           register and pass it to %0
29   );
30 }
31
32
33 void experiment01() {
34   ioPORTB = (uint8_t *)0x25;
35   ioDDRB = (uint8_t *) 0x24;
36   //make PB1 as output
37   *ioDDRB = 0x02; // DDRB[7:0] 0000 0010
38   while (1) {
39     asm volatile (
40       "ldi r16, 0x02 \n\t" // 1 cycle. load r16 with 0x02.
41       "sts 0x25, r16 \n\t" // store content in r16 to add 0x25 (PORTB), 2 cycle
42     );
43     myHardDelayUsec(DELAY_VALUE);
44     asm volatile (
45       "ldi r17, 0x00 \n\t"
46       "sts 0x25, r17 \n\t"
47     );
48     myHardDelayUsec(DELAY_VALUE);
49   }
50 }
```