Blinking LED (Assembler)

Executed by:Kevin Anum
Mailbox:108
Embedded System Lab
Natural Science Laboratory, Jacobs University Bremen

September 5, 2017

Prelab

1.Study the datasheet Chapter I/O Ports, find out how to use the above three registers

There are three I/O memory address for each port.

1. DDRx-Data Directory Register contains the direction of the port where input is logic zero and the output is a logic one.(read and write)
2. Pinx-Contains the input value of the port.(read only)
3. PORTx-Contains the current value of the port.(read and write)

were x is the name of the port.Each port pin consists of three register namely :

1. DDxn-DDxn are accessed at the DDRx I/O address
2. PINxn-PINxn are accessed at the PINx I/O address
3. PORTxn-PORTxn are accessed at the PORTx I/O address

The DDxn in the DDRx Register selects the dirrection of the pin.If DDxn is 1 the Pxn (I/O-port pin) is configured as an output pin.If DDxn is a logic zero the Pxn is configured as an output.

The pull-up resistor is activated when the PORTxn is 1 and the DDxn is 0.To produce a 1 at Port pin DDxn and Port xn should be 1.To produce a 0 at the port pin DDxn should be 1 and PORTxn should be zero.

Writing a logic one to PINxn toggles the value of PORTxn

2.Study the assembly instructions LDI, OUT, SBI, CBI, JMP/RJMP, CALL/RCALL, RET, DEC, BRNE, CLI and try to understand my assembly examples.

1. BRNE-checks if the value is zero
2. CALL/RCALL-call a the subroutine
3. CLI -Disables

4. DEC - Subtract one from the value.
5. JMP/RJMP-jumps to place instructed.
6. SBI - Set bit in I/O Register
7. LDI-Loads the value of the second argument in the first argument.
8. RET-Indicates the end of the subroutine.
9. OUT-Outputs the second argument into the first argument.
10. CBI-Clear bit in I/O register.

```
.include "m328def.inc"
.org 0x0000
           RJMP begin; jump to begin
.org 0x0034 ;initialize stack
 begin:    CLI
           LDI    R16,low(RAMEND);insets address of lower end of the ram in
                                register 16
         OUT    SPL,R16;inserts value of register 16 into the
         stack pointer low
           LDI     R16,high(RAMEND);insets address of higher end of
                                 the ram in register 16
         OUT    SPH, R16;inserts value of register 16 into
                          the stack pointer high
         LDI    R16,0xFF ;loads 0xFF into register 16
         OUT    DDRD, R16;outputs value in register 16
                           into the DDRD
           LDI     R16,0xFF;loads 0xFF into register 16
         OUT    PORTD, R16;outputs value in register 16
         into the PORTD
           RCALL  Delay;Call subroutine Delay
           LDI     R16,0x00;loads 0x00 into register 16
         OUT     PORTD, R16;outputs value in register 16
                            into the PORTD
           RCALL  Delay; Calls Subroutine Delay
Delay:     LDI     R17, 0x02;insetrs 0x02 in register 17
loop:    DEC     R17 ; subtract one from value in subroutine
         BRNE    loop ;checks if it is zero
         RET  ;return to main routine
```

In the routine above we create an output high then a delay and then an out put low.

3.The CPU clock is in the range of 20Mhz, calculate how many CPU clock cycles you need to have 1 second delay. Assume implementing each assembly instruction need one CPU clock cycle, change the code in the last examples such that the Delay subroutine produce 1 second delay

$$f = \frac{1}{T} \tag{1}$$

where T is period and f is frequency

$$\frac{1}{5 \times 10^{-8}} = 20 \times 10^{6} \tag{2}$$

50000 CPU clock cycles create a delay of a second.

```
.include "m328def.inc"
.org 0x0000
           RJMP begin; jump to begin
.org 0x0034
 begin:    CLI
           LDI    R16,low(RAMEND)
         OUT    SPL,R16
           LDI     R16,high(RAMEND)
         OUT    SPH, R16
         LDI    R16,0xFF
         OUT    DDRD, R16
           LDI     R16,0xFF
         OUT    PORTD, R16
           RCALL  Delay
           LDI     R16,0x00
         OUT     PORTD, R16
           RCALL  Delay
Delay:     LDI     R17, 0x98
loop1:     LDI     R18,0xff
loop2:     LDI     R19,0xff
loop3:     DEC     R19
```

```
BRNE    loop3
DEC     R18
BRNE    loop2
DEC     R17
BRNE    loop1
RET
```

Introduction

In this lab we were introduced to AVR Assembly language,Arduino Uno Board and controlling the digital I/O ports of ATmega 328.

We were introduced to basic instructions like DEC,RJMP,ADD to use as operations in the micro-controller.A micro-controller is a chip the contains a CPU,timer and memory.A micro-controller has several components these include:

- Timer Module
- Analog I/O Modules
- Digital Module
- Serial Module

In this lab we used the ATmega328 micro-controller.We also obtained some important information about the ATmega 328 below are a few examples:

1. Interrupt
   In storing application programs it is important to note that spaces 0x0000 to 0x0032 in the program memory are reserved for the interrupt vector table.

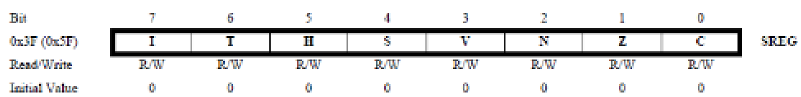

Figure 1: The figure shows an example of a status register(SREG).

To enable interrupt the bit 7-I needs to set to one (block I will be shaded if it is set to one) using the assembly instruction SEI or the interrupt can be disabled using the the instruction CLI which sets bit 7-I to zero(block I will not be shaded).

2. The Stack pointer
   The stack is used to store temporary data,local variables and returning address after interrupts and subroutine calls.The stack grows from higher to lower memory locations).The Stack Pointer always points to the top of the stack.In AVR the stack pointer works as two 8-bits register defined as SPL (contains the lower 8-bits) and SPH (contains the high 8-bits).

3. Register
   In the AVR Atmega328 Micro-controller registers sixteen to thirty-two are of interest to us as they are the registers in which we can perform operations.

With the help of the information above the adriuno uno and ATmel we were able to causes a LED light to blink continuously.

Circuit design

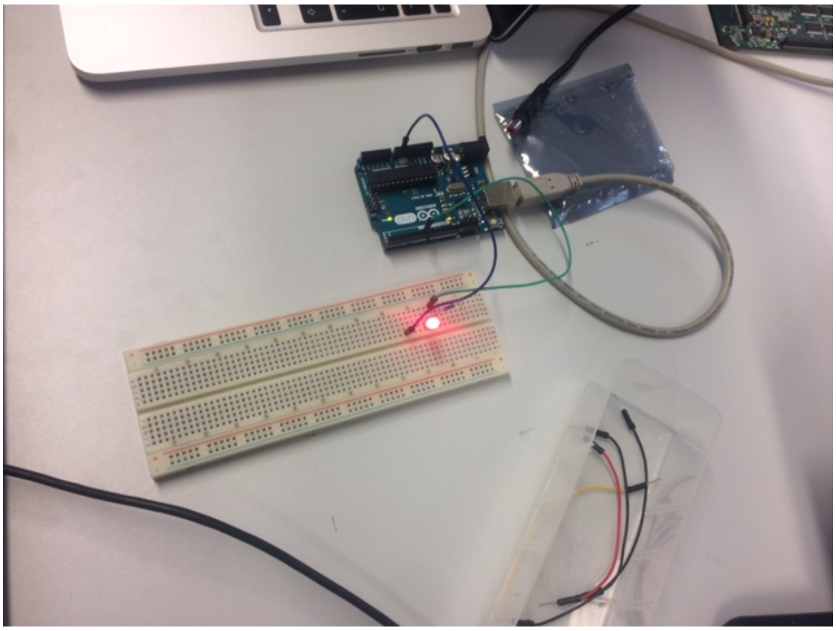

Figure 2: The figure shows the image of the circuit implemented in the lab.

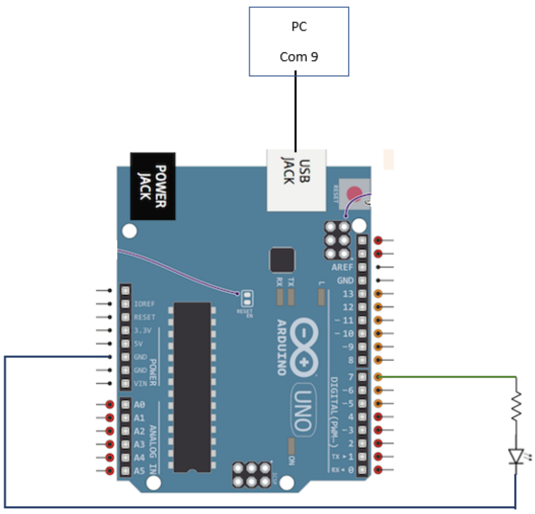

Figure 3: The figure shows a sketch of the circuit implemented in the lab.

There is a voltage supply in the circuit if there is a high(0xFF) at PORTD,all ports at D, (the DDRD is a high 0xFF and PORTD is a high 0x00) and there is no voltage supply when there is a low (0x00) at PORTD (the DDRD is a high(0xFF) and a PORTD is low (0x00)).This scenario causes the LED light to flash.

Code

```
.include "m328def.inc"
.org 0x0000
           RJMP begin; jump to begin
.org 0x0034 ;initialize the stack.(0x0000 to 0x0034 is reserved for the
             interrupt vector table)
 begin:  CLI
         LDI    R16,low(RAMEND);insets address of lower end of the ram in
                                   register 16
         OUT    SPL,R16;inserts value of register 16 into the
                        stack pointer low
         LDI     R16,high(RAMEND);insets address of higher end of
                                    the ram in register 16
         OUT    SPH, R16;inserts value of register 16 into
                           the stack pointer high
loop:    LDI    R16,0xFF ;loads 0xFF into register 16 and a loop is created
         OUT    DDRD, R16 ;makes port D a output
           LDI     R16,0xFF;loads 0xFF into register 16
         OUT    PORTD, R16;loads 0xFF into port D.(LED light switches on)
         RCALL  Delay; calls sub routine called delay
          LDI     R16,0x00;loads 0x00 into register 16
         OUT     PORTD, R16; outputs 0x00 into Port D (LED light goes off)
           RCALL  Delay
           RJMP   loop; jump to loop (infinte loop has been created)
Delay:     LDI     R17, 0x30;loads 0x30 into register 17
loop1:     LDI     R18,0xff;loads 0xff into register 18
loop2:     LDI     R19,0xff;loads 0xff into register 19
loop3:     DEC     R19;decreases value in register 19 by one
           BRNE    loop3;checks if the value in register 19
                         is zero if not it goes back to loop 3
           DEC     R18;decreases value in register 18 by one
           BRNE    loop2;checks if the value in register 18
                         is zero if not it goes back to loop 2
```

```
DEC     R17;decreases value in register 17 by one
BRNE    loop1;checks if the value in register 17
            is zero if not it goes back to loop 1
RET ;returns to the address at the stack pointer
```

In the prelab we saw that 20×10^6 cycles are,produced in 1 second.We can see that in the subroutine Delay the most frequent instructions used are BRNE and DEC.From the data sheet it can be seen that BRNE takes half a cycle and DEC takes one cycle.So in loop 3 there are $255 \times 1.5 = 382.5 cycles$.In loop 2,loop 3 is repeated 255 times.The number of cycles (this is an approximate value since we ignore the line LDI R19,0xff) $382.5 \times 255 = 97537.5 cycles$.Loop 3 is loop 2 255 times so the number of cycles excluding the LDI lines are $97537.5 \times 255 = 24872062.5 cycles$.This value is larger that $20 \times 10^6 cycles$ so we gradually decreased the value in register 17 till we got a delay of one second.The program counter increases by one after an instruction.

Conclusion

In conclusion this lab introduced us to AVR assemble language we were able to understand and apply the three I/O memory addresses in a port.We also learn a few commands and how to obtain data from the data sheet given.