Computer Technology I

Lab. 1 : How to use the PORTs, Digital input/output, Subroutine call

Author: ROEL DI VRIES, ANAS KWEFATI
Supervisor: ANDERS HAGGREN
Semester: Autumn 2019
Area: Computer Science
Course code: 1DT301

Contents

1 Task 1

For the first task the goal was to get a light blinking. This was done by setting the data direction register to output, and after that setting the LED port low.

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;       1DT301, Computer Technology 1
;       Date: 09-09-2019
;       Authors:
;               Roel de Vries
;               Anas Kwefati
;
;       Lab number 1
;       Title:  How to use the PORTS, digital IO, subroutine call
;
;       Hardware: STK600, CPU ATmega 2560
;
;       Function: Turn on LED 2
;
;       Input ports: None
;
;       Output ports: PORTB, used for LEDS
;
;       Subroutines: None
;       Included files: m2560def.inc
;
;       Other information: None
;
;       Changes in program:
;               09-09-2019 > file created
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"

main:
        SBI DDRB, 2 ; set pin to output
        CBI PORTB, 2; turn on led
```

Here is the flowchart :

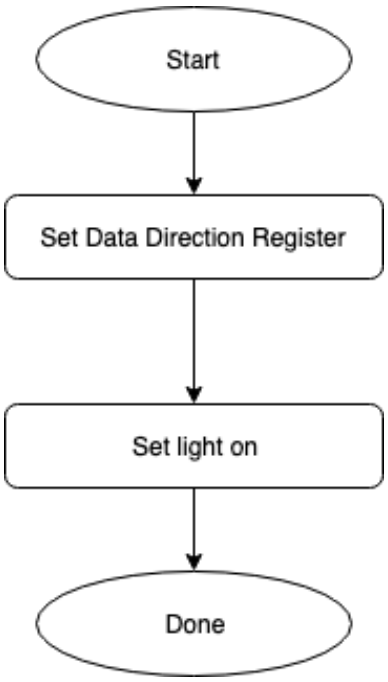


Figure 1: Task 1 flowchart

2 Task 2

For the second task the aim is to read switches and light to corresponding LED. This was done by using a data direction register

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
; Date: 2019-09-95
; Author:
;   Roel de Vries
;   Anas Kwefati
;
;       Lab number 1
;       Title:  How to use the PORTS, digital IO, subroutine call
;
;       Hardware: STK600, CPU ATmega 2560
;
;       Function: Turn on LED n if SWITCH n is pressed
;
;       Input ports: PORTD, used for the switches
;
;       Output ports: PORTB, used for LEDS
;
;       Subroutines: None
;       Included files: m2560def.inc
;
;       Other information: None
;
;       Changes in program:
;               09-09-2019 > file created
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

```
;TASK_2

;Load pre-configured files for the ports and memory adresses
.include "m2560def.inc"

;We first initialize everything

ldi r16, 0xFF ; load 0b1111 1111 to r16
out DDRB, r16 ; we set the Data Direction Register B to be ready to
    give an output to turn on the light (0 is on and 1 is off) so now
    we are outputting 1

ldi r17, 0x00 ; we load 0b0000 0000 to the register 17
out DDRD, r17 ; we set the Data Direction Register D to take an input

ldi r16, 0xFF ;
out PORTB, r16 ; we are setting the PORTB to give an output of 0b1111
    1111 like that the light is off (for LED 0 is on and 1 is off)

;Initialization is finished

; we are creating an infinite loop to check if the switch port is on or
     off
infinite_loop_switch:

        in r18, PIND ; The input data received from the input PIND is
            stored in the register r18
        out PORTB, r18 ;The PORTB will take the data of r18 and output
            it to the LED and turn on the light if it is correct

rjmp infinite_loop_switch ;we repeat the process
```

Here is the flowchart for this task :

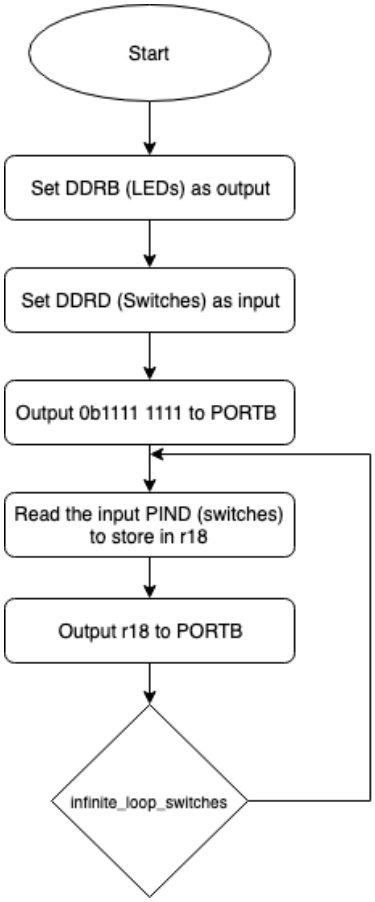


Figure 2: Task 2 flowchart

3 Task 3

In task 3 the goal was to turn on led 0, only if switch 5 was pressed. by checking if the bit for switch 5 is high we are able to turn the led on at the right moment

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;       1DT301, Computer Technology 1
;       Date: 09-09-2019
;       Authors:
;               Roel de Vries
;               Anas Kwefati
;
;       Lab number 1
;       Title:  How to use the PORTS, digital IO, subroutine call
;
;       Hardware: STK600, CPU ATmega 2560
;
;       Function: Turn on Led 0 when you press led 5
;
;       Input ports: PORTA, used for the switches
;
;       Output ports: PORTB, used for LEDS
;
;       Subroutines: None
;       Included files: m2560def.inc
;
;       Other information: None
;
;       Changes in program:
;               09-09-2019 > file created
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"

main:
        SBI DDRB, 0 ; set bit 0 to output
        CBI DDRA, 5 ; set bit 5 to input

lightloop:

        SBIS PINA, 5 ; check if button is pressed
        CBI PORTB, 0 ; if so, turn on led.
        SBIC PINA, 5 ; check if button is not pressed
        SBI PORTB, 0 ; if so, turn off led.
```

Here is the flowchart for this task :

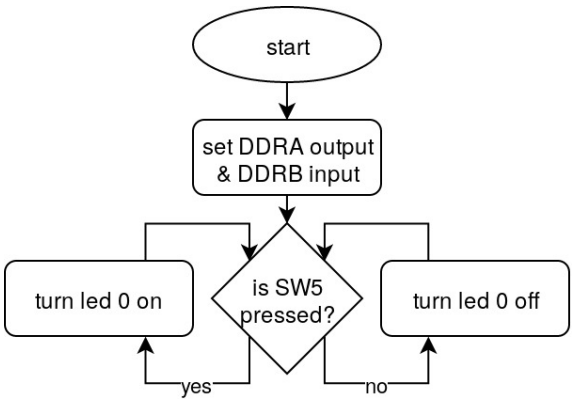
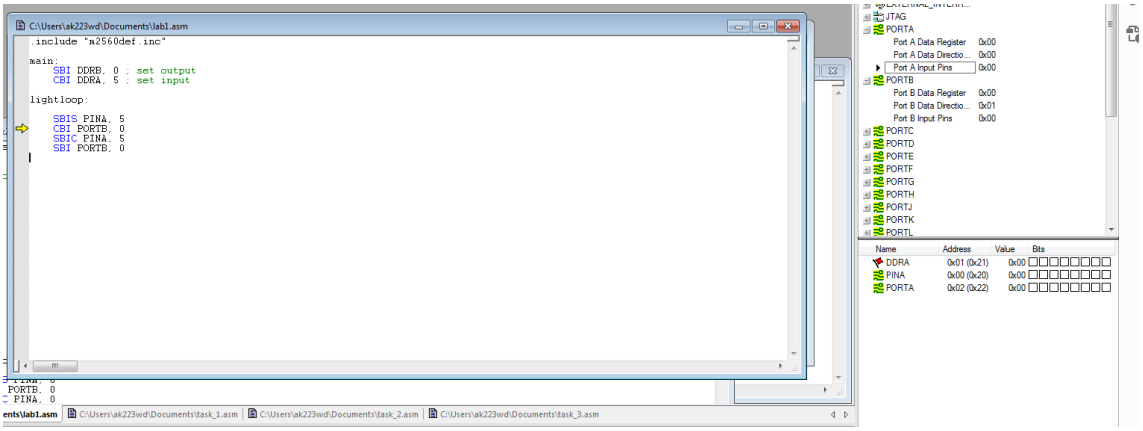


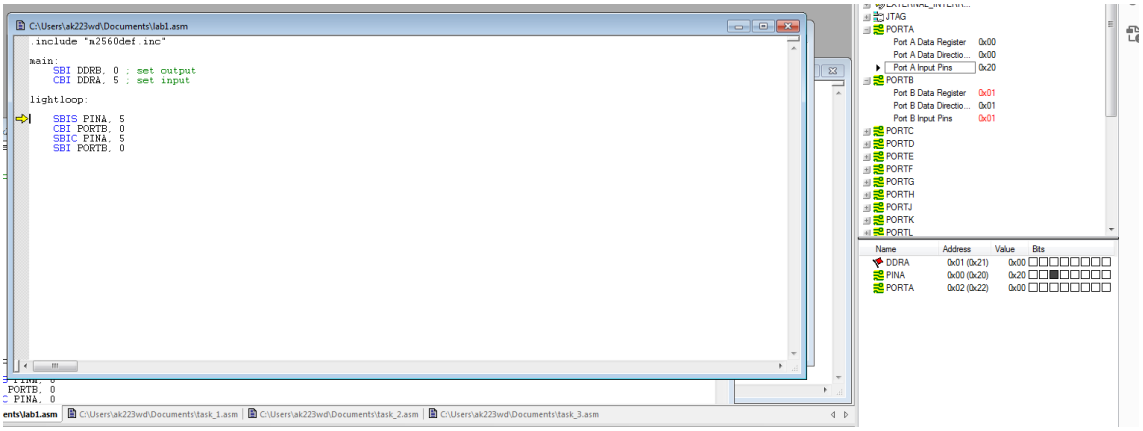
Figure 3: Task 3 flowchart

4 Task 4

In task 4 we needed to run the task 3 code in the simulator. In the first screenshot you can see that PINA is low, and the LED is off, in the second screenshot a bit of PINA has been set to high, and the led has turned on as well.



Both the LED and the button are off.



The switch, and with that also the LED, has turned on.

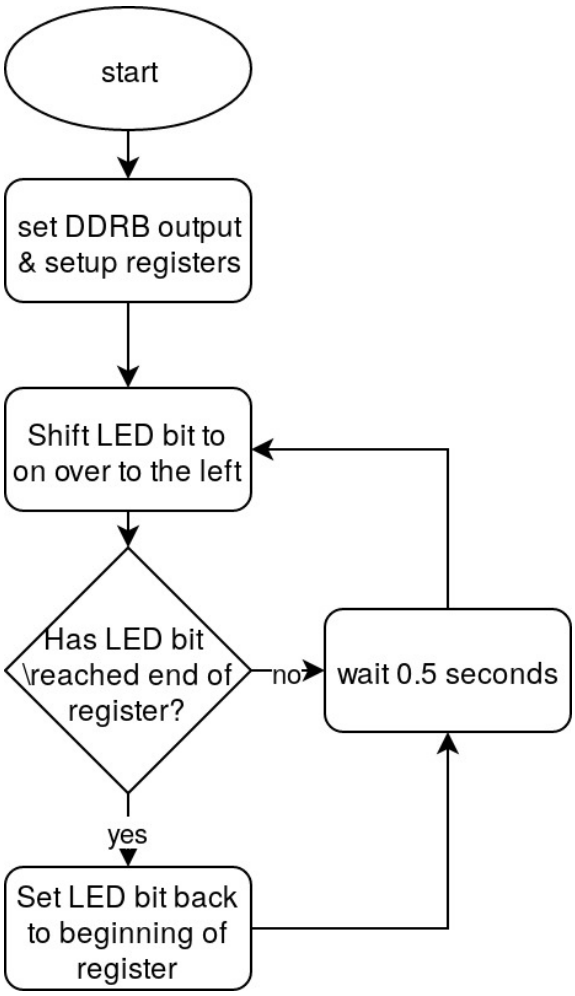
5 Task 5

For task 5 we needed to create a ring counter. This was done by creating a loop which constantly shifts the PORTB register one sideways with a delay.

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;       1DT301, Computer Technology 1
;       Date: 09-09-2019
;       Authors:
;               Roel de Vries
;               Anas Kwefati
;
;       Lab number 1
;       Title:  How to use the PORTS, digital IO, subroutine call
;
;       Hardware: STK600, CPU ATmega 2560
;
;       Function: Creates a ring counter, updates every 0.5 seconds
;
;       Input ports: None
;
;       Output ports: PORTB, used for LEDS
;
;       Subroutines: Timer
;       Included files: m2560def.inc
;
;       Other information: None
;
;       Changes in program:
;               09-09-2019 > file created
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"

main:
        ; Initialize SP, Stack Pointer
        ldi r20, HIGH(RAMEND) ; R20 = high part of RAMEND address
        out SPH,R20 ; SPH = high part of RAMEND address
        ldi R20, low(RAMEND) ; R20 = low part of RAMEND address
        out SPL,R20 ; SPL = low part of RAMEND address

        LDI r20, 0xFE ; set all bits but the last one high
        OUT PORTB, r20 ; move to led register
        call timer

lightloop:
        LSL r20 ; shift bits one to the left
        BRCS setbit ;set rightmost low bit to high if carry bit is set
lightloopcont:
        OUT PORTB, r20 ; move to led register
        call timer
        call lightloop ; loop indefinitely

setbit:
        SBR r20, 1 ; set bit high
        CLC ; and clear carry bit so we can check again.
        call lightloopcont

timer:
; Generated by delay loop calculator
```

```
; at http://www.bretmulvey.com/avrdelay.html
        ldi  r17, 5
    ldi  r18, 20
    ldi  r19, 175
L1: dec  r19
    brne L1
    dec  r18
    brne L1
    dec  r17
    brne L1
        ret
```

Here is the flowchart for this task :



Task 5 flowchart

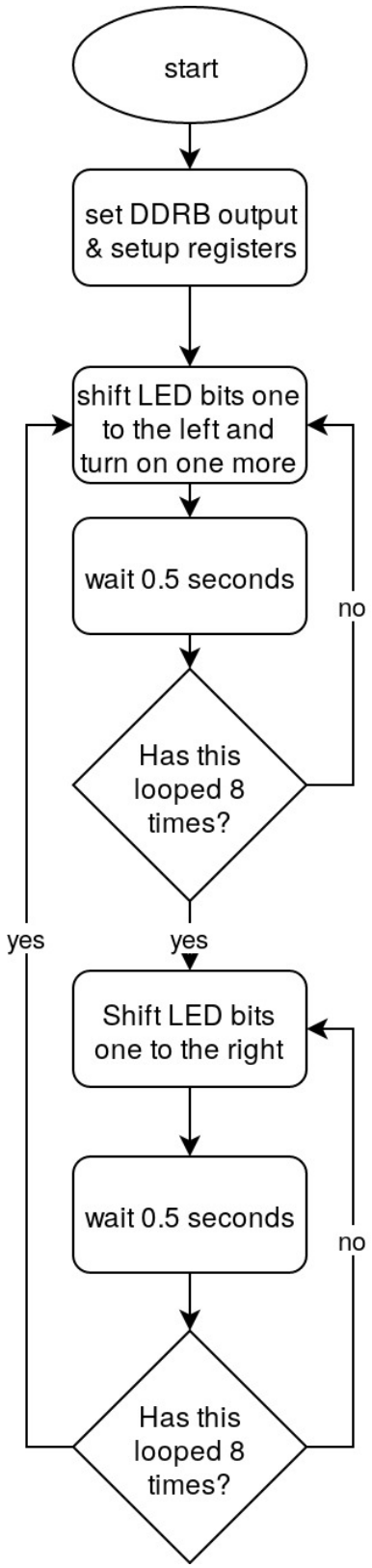
6 Task 6

The Johnson counter was created by using two smaller loops who constantly call eachother. the first which increases the amount of leds on, and a second which decreases the amount of leds on.

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;       1DT301, Computer Technology 1
;       Date: 09-09-2019
;       Authors:
;               Roel de Vries
;               Anas Kwefati
;
;       Lab number 1
;       Title:  How to use the PORTS, digital IO, subroutine call
;
;       Hardware: STK600, CPU ATmega 2560
;
;       Function: Creates a johnson counter, updates every 0.5 seconds
;
;       Input ports: None
;
;       Output ports: PORTB, used for LEDS
;
;       Subroutines: Timer
;       Included files: m2560def.inc
;
;       Other information: None
;
;       Changes in program:
;               09-09-2019 > file created
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"

main:
        ; Initialize SP, Stack Pointer
        ldi r21, HIGH(RAMEND) ; R20 = high part of RAMEND address
        out SPH,R21 ; SPH = high part of RAMEND address
        ldi R21, low(RAMEND) ; R20 = low part of RAMEND address
        out SPL,R21 ; SPL = low part of RAMEND address

        ldi r16, 8 ; set counter register
        SBR r17, 255; set light state
        OUT DDRB, r17 ;move to led register

incloop:
        LSL r17 ; shit register left
        OUT PORTB, r17 ; and move it to led register
        call timer
        dec r16 ; decrease counter
    brne incloop ; once counter is finished, move on.
        ldi r16, 8
        call decloop

decloop:
        LSR r17 ; shit register right
        SBR r17, 128 ;set leftmost bit high
        OUT PORTB, r17 ; and move it to led register
        call timer
```

```
        dec r16
    brne decloop ; once counter is finished, move on.
        ldi r16, 8
        call incloop

timer:
; Generated by delay loop calculator
; at http://www.bretmulvey.com/avrdelay.html
        ldi  r18, 5
    ldi  r19, 20
    ldi  r20, 175
L1: dec  r20
    brne L1
    dec  r19
    brne L1
    dec  r18
    brne L1
    rjmp PC+1
        ret
```

Here is the flowchart for this task :



Task 6 flowchart