Rapport

Laboratory Report

7 september 2017

Author: Caroline Nilsson
Daniel Alm Grundström
Termin: HT 2017
Course: 1DT301 - Computer Technology I

Innehåll

1 Introduktion

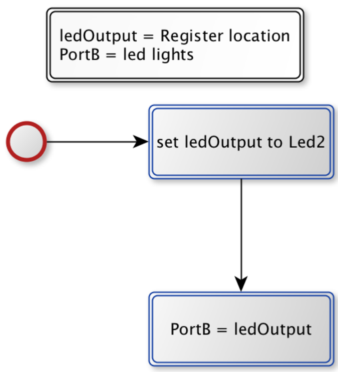
In the process of working with the laboratory assignments we started by doing research about the assembly language and the STK600 in order to better understand how to solve the different assignments. In each assignment we first created a pseudocode solution which we converted to flowchart diagrams, then it was rather simple to convert this into assembly language. Common for all assignments is also that we have been using the simulations to confirm that the program is working and completing the correct tasks.

2 Assignment 1 - Light LED2

In the first assignment we were to write an Assembly that lights up LED2 (which is the third light counting from the right).

Algorithm 1 Light LED2

procedure PSEUDOCODE
  $PortB = output$
  $Led2\ bitstring \rightarrow PortB$



Figur 1: Flowchart

The pseudocode (see algorithm 1) and the flowchart (see figure 1) shows that we first set PORTB as an output port. To light up `LED2` we then only need to write a value to the bit on PORTB that corresponds to LED2.

We started with the assumption that all bits in PORTB would be zero when the LEDs where turned off and as such wrote a 1 to the third least significant bit to light up LED2. When we tested the program on the hardware however, all LEDs except LED2 was turned on. If we understood this correctly, this was due to the pull-up resistor being activated on PORTB which made the LEDs light when their bit was 0 (as opposed to 1) on PORTB. We fixed this by simply inverting the value we wrote to PORTB (11111011_2 instead of 00000100_2).

The minimal number of lines required to write this program we think are 4 (unless there is some obscure trick). 2 lines are required to set the LED port as output: 1) write a value to a register and 2) write that value to the data direction register, and 2 lines for turning on the LED: 3) write the LED state to a register and 4) write the LED state to the output port. If the value written to the data direction register is reused when writing to the output port, the LED will not turn on because of the pull-up resistor will require that a zero is written to the bit corresponding to the LED that we want to light.

2.1 Assembly Program

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;    1DT301, Computer Technology I
;    Date: 2017-09-07
;    Author:
;                       Caroline Nilsson            (cn222nd)
;                       Daniel Alm Grundström       (dg222dw)
;
;    Lab number:        1
;    Title:             How to use the PORTs. Digital input /output.
;                       Subroutine call.
;
;    Hardware:          STK600, CPU ATmega2560
;
;    Function:          Lights LED2 on PORTB
;
;    Input ports:       N/A
;
;    Output ports:      PIN2 on PORTB
;
;    Subroutines:       N/A
;    Included files:    m2560def.inc
;
;    Other information: LEDs are configured to light when PINs on PORTB are set
;                       to 0. The default state, when no LED is lit must
;                       therefore be set to 0b1111_1111.
;
;    Changes in program:
;                       2017-09-01:
;                       Implemented flowchart design.
;
;                       2017-09-02:
;                       Added comments and .def for r16
;
;                       2017-09-07:
;                       Adjusts code to handle pull-up resistors on PORTB.
;                       Removes unnecessary loop that prevented program from
;                       exiting after LED2 had been turned on.
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"
.def ledOutput = r16

; Set PORTB to output
ldi ledOutput, 0xFF
out DDRB, ledOutput

; Turn on LED2 on PORTB
ldi ledOutput, 0b1111_1011
out PORTB, ledOutput
```

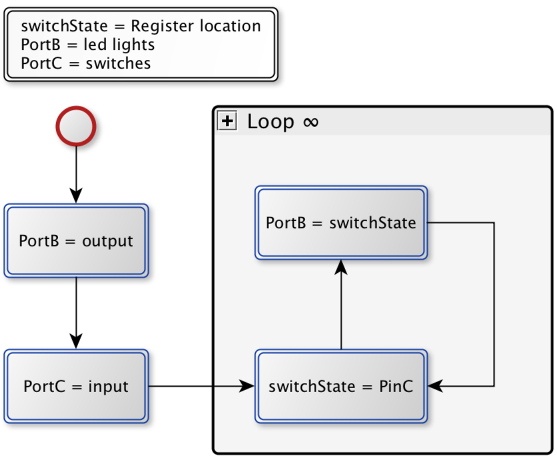
3 Assignment 2 - Switch light corresponding LED

In the second assignment we were to write a program that waits for a switch to be pressed and then lights up the corresponding LED. For example if switch 3 is pressed LED 3 should light up. The way we interpreted the assignment was that the LED should stay on for as long as the switch is pressed and turn off when the switch is released.

We figured the easiest way to do this was to simply redirect the input from the switches to the LEDs. Since both the input from the switches and the output to the LEDs are bit strings were each bit corresponds to a switch/LED

Algorithm 2 Switches pressed lights corresponding LED

procedure PSEUDOCODE
  $PortB = output$
  $PortC = input$
  repeat
    $PortC\ value \rightarrow switchState$ $\triangleright$ $switchState = register\ location$
    $switchState \rightarrow PortB$
  until ∞



Figur 2: Basic flow in order to read switches and light corresponding LED

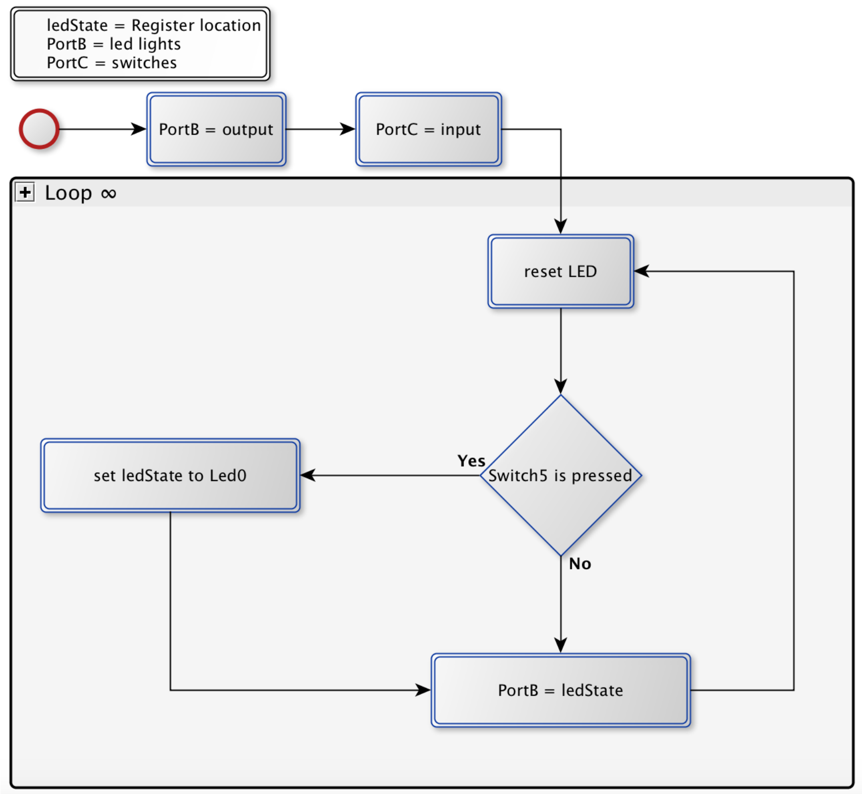
3.1 Assembly Program

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;    1DT301, Computer Technology I
;    Date: 2017-09-07
;    Author:
;                        Caroline Nilsson            (cn222nd)
;                        Daniel Alm Grundström       (dg222dw)
;
;    Lab number:         1
;    Title:              How to use the PORTs. Digital input /output.
;                        Subroutine call.
;
;    Hardware:           STK600, CPU ATmega2560
;
;    Function:           Reads input from the switches SW0..SW7 and lights the
;                        corresponding LED when a switch is pressed. (SW0 lights
;                        LED0, SW1 lights LED1 and so on)
;
;    Input ports:        PORTC
;
;    Output ports:       PORTB
;
;    Subroutines:        N/A
;    Included files:     m2560def.inc
;
;    Other information:  N/A
;
;    Changes in program:
;                        2017-09-01:
;                        Implemented flowchart design.
;
;                        2017-09-02:
;                        Adds header and comments.
;
;                        2017-09-07:
;                        Adjusts code to handle pull up resistor on PORTB.
;                        Changes switch port to PORTC.
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"
.def switchInput = r16
.def dataDir = r17

; Set PORTB (LEDs) as output
ldi dataDir, 0xFF
out DDRB, dataDir

; Set PORTC (switches) as input
ldi dataDir, 0x00
out DDRC, dataDir

loop:
    in switchInput, PINC        ; Read input from switches
    out PORTB, switchInput      ; Output switch input to LEDs
    rjmp loop
```

4 Assignment 3 - Swift5 lights LED0

Algorithm 3 Light LED0 when switch5 is pressed

procedure PSEUDOCODE
  $PortB = output$
  $PortC = input$
  repeat
    $reset\ ledState$ ▷ $ledState = register\ location$
    if $Switch5\ is\ pressed$ **then**
      $ledState = LED0\ bit\ string$
    $ledState \rightarrow PortB$
  until ∞



Figur 3: Flowchart

4.1 Assembly Program

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;   1DT301, Computer Technology I
;   Date: 2017-09-07
;   Author:
;                       Caroline Nilsson            (cn222nd)
;                       Daniel Alm Grundström       (dg222dw)
;
;   Lab number:         1
;   Title:              How to use the PORTs. Digital input /output.
;                       Subroutine call.
;
;   Hardware:           STK600, CPU ATmega2560
;
;   Function:           Turns on LED0 when SW5 is held down.
;
;   Input ports:        PORTC
;
;   Output ports:       PORTB
;
;   Subroutines:        N/A
;   Included files:     m2560def.inc
;
;   Other information:  N/A
;
;   Changes in program:
;                       2017-09-01:
;                       Implemented flowchart design.
;
;                       2017-09-04:
;                       Minor refactoring. Adds header and comments.
;
;                       2017-09-07:
;                       Adjusts code to handle pull up resistor on PORTB.
;                       Changes switch port to PORTC.
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"
.def dataDir = r16
.def ledState = r17

; Set PortB as output
ldi dataDir, 0xFF
out DDRB, dataDir

; Set PortC as input
ldi dataDir, 0x00
out DDRC, dataDir

loop:
    ser ledState                ; Set bits in LED state so LEDs are turned
                                ; off when button is released

    sbis PINC, PINC5            ; If SW5 is pressed down (PINC5 bit is zero)
        ldi ledState, 0xFE      ;   then set LED0 state to turned on

    out PORTB, ledState         ; write state to LEDs
    rjmp loop
```

5 Assignment 4

Algorithm 4

procedure PSEUDOCODE

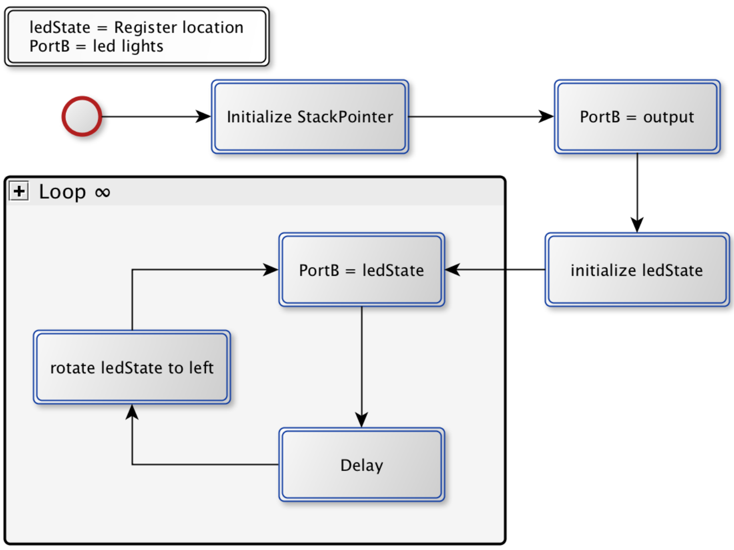
Figur 4: Flowchart

5.1 Assembly Program

6 Assignment 5 - Waterfall

Algorithm 5 Waterfall simulation using LEDs

procedure PSEUDOCODE
  $Initialize\ stack\ pointer$
  $PortB = output$
  $Initialize\ ledState$ $\triangleright$ $ledState = register\ location$
  repeat
    $ledState \rightarrow PortB$
    $Delay$
    $rotate\ ledState\ to\ left$
  until ∞



Figur 5: Flowchart

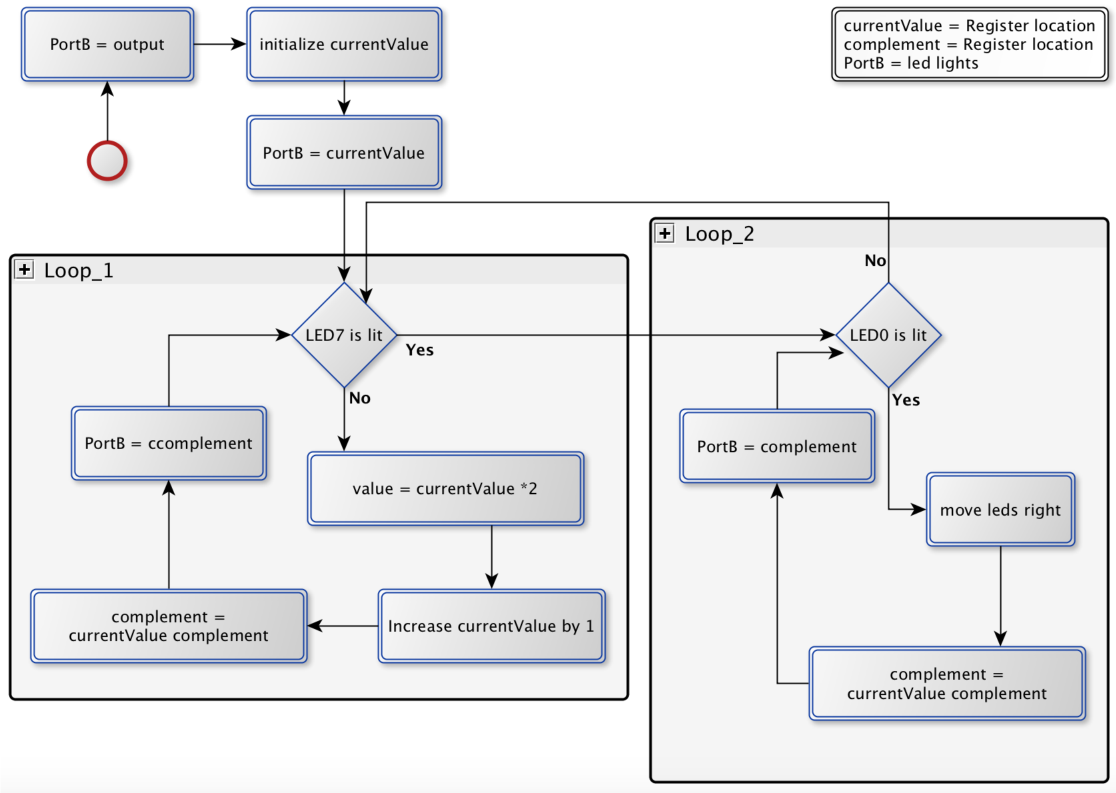
6.1 Assembly Program

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;    1DT301, Computer Technology I
;    Date: 2017-09-07
;    Author:
;                        Caroline Nilsson            (cn222nd)
;                        Daniel Alm Grundström       (dg222dw)
;
;    Lab number:         1
;    Title:              How to use the PORTs. Digital input/output.
;                        Subroutine call.
;
;    Hardware:           STK600, CPU ATmega2560
;
;    Function:           Repeatedly lights LEDs sequentially right to left.
;
;                        I.e:
;                        0000 0001 -> 0000 0010 -> 0000 0100 -> ... ->
;                        1000 0000 -> 0000 0001 -> 0000 0010 -> ...
;
;    Input ports:        N/A
;
;    Output ports:       PORTB
;
;    Subroutines:        delay - delays execution
;    Included files:     m2560def.inc
;
;    Other information:  Since a subroutine is used, the stack pointer must
;                        be initialized so the processor knows where in the
;                        code to jump when the subroutine returns.
;
;    Changes in program:
;                        2017-09-01:
;                        Implements flowchart design
;
;                        2017-09-04:
;                        Adds header, comments and some minor refactoring
;
;                        2017-09-07:
;                        Adjusts code to handle pull up resistor on PORTB.
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"
.def dataDir = r16
.def ledState = r17
.equ INITIAL_LED_STATE = 0xFF

; 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

; Set PORTB to output
ldi dataDir, 0xFF
out DDRB, dataDir

ldi ledState, INITIAL_LED_STATE          ; Set initial LED state

loop:
    out PORTB, ledState                  ; Write state to LEDs
    rcall delay                          ; Delay to make changes visible
    rol ledState                         ; Rotate LED state to the left
    rjmp loop

; Generated by delay loop calculator
; at http://www.bretmulvey.com/avrdelay.html
delay:
    push r18
    push r19
    push r20

    ldi  r18, 4
    ldi  r19, 12
    ldi  r20, 52
L1: dec  r20
    brne L1
    dec  r19
    brne L1
    dec  r18
    brne L1
    rjmp PC+1

    pop r20
    pop r19
    pop r18
    ret
```

7 Assignment 6 - Johnson counter

Algorithm 6 Johnson counter simulation using LEDs

procedure PSEUDOCODE
    $PortB = output$ ▷ $complement = register\ location$
    $Initialize\ currentValue$ ▷ $currentValue = register\ location$
    repeat ▷ $Loop\_1\ (count\ up)$
        if $LED7\ is\ lit$ **then**
            $Continue\ at\ Loop\_2$
        else
            $currentValue = currentValue \times 2$
            $Increase\ currentValue\ by\ 1$
            $complement = complement\ of\ currentValue$
        $complement \rightarrow PortB$
        $Delay$
    until ∞
    repeat ▷ $Loop\_2\ (count\ down)$
        if $LED0\ is\ lit$ **then**
            $Continue\ at\ Loop\_1$
        else
            $currentValue = Shift\ right$
            $complement = complement\ of\ currentValue$
        $complement \rightarrow PortB$
        $Delay$
    until ∞



Figur 6: Flowchart

7.1 Assembly Program

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;    1DT301, Computer Technology I
;    Date: 2017-09-07
;    Author:
;                        Caroline Nilsson            (cn222nd)
;                        Daniel Alm Grundström       (dg222dw)
;
;    Lab number:         1
;    Title:              How to use the PORTs. Digital input /output.
;                        Subroutine call.
;
;    Hardware:           STK600, CPU ATmega2560
;
;    Function:           Lights LEDs as a Johnson counter in an infinite loop.
;
;                        I.e:
;                        0000 0001 -> 0000 0011 -> 0000 0111 -> ...
;                        1111 1111 -> 0111 1111 -> 0011 1111 -> ...
;
;    Input ports:        N/A
;
;    Output ports:       PORTB
;
;    Subroutines:        delay - delay execution
;
;    Included files:     m2560def.inc
;
;    Other information:  N/A
;
;    Changes in program:
;                        2017-09-02:
;                        Implements flowchart design
;
;                        2017-09-04:
;                        Adds header and comments
;
;                        2017-09-07:
;                        Adjusts code to handle pull up resistor on PORTB.
;                        Changes code to use shift left instead of multiplying
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"
.def dataDir = r16
.def currentValue = r17                  ; Current value of Johnson counter
.def complement = r18

; 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

; Set PORTB as output
ldi dataDir, 0xFF
out DDRB, dataDir

; Set and output initial value
ldi currentValue, 0x00
rcall led_out

count_up:
    sbis PORTB, PINB7                    ; If LED7 is lit (i.e. all LEDs lit)
        rjmp count_down                  ;     then start counting down

    ; Get next johnson value by multiplying by 2 and adding 1
    lsl currentValue
    inc currentValue

    rcall led_out                        ; Output complement of current value
    rcall delay_500ms                    ; Delay to make changes visible
    rjmp count_up                        ; Continue counting up

count_down:
    sbic PORTB, PINB0                    ; If LED0 is unlit (i.e. all LEDs unlit)
        rjmp count_up                    ;     then start counting up

    lsr currentValue                     ; Shift to right to get previous
                                         ; johnson value

    rcall led_out                        ; Ouput complement of current value
    rcall delay_500ms                    ; Delay to make changes visible
    rjmp count_down                      ; Continue counting down

; Writes the complement of 'currentValue' to PORTB
led_out:
    mov complement, currentValue
    com complement
    out PORTB, complement
    ret

; Generated by delay loop calculator
; at http://www.bretmulvey.com/avrdelay.html
delay:
    push r29
    push r30
    push r31

    ldi  r31, 4
    ldi  r30, 12
```

```
        ldi   r29, 52
L1:     dec   r29
        brne  L1
        dec   r30
        brne  L1
        dec   r31
        brne  L1
        rjmp  PC+1

        pop  r31
        pop  r30
        pop  r29
        ret
```