Linnéuniversitetet

1DT301, Computer Technology I, autumn 2019.
Lab. 2: Subroutines

Goal for this lab:
Learn how to program and use subroutines with Assembly language.

Development environment: **AVR Studio4 or AVR Studio6**

You will find information on the course web page at MyMoodle

The laboratory work has to be done in groups of maximum 2 students per group.

You have to write comments that explain the code in each program. The structure of each program has to be explained with Flow Charts for each program. (each task)

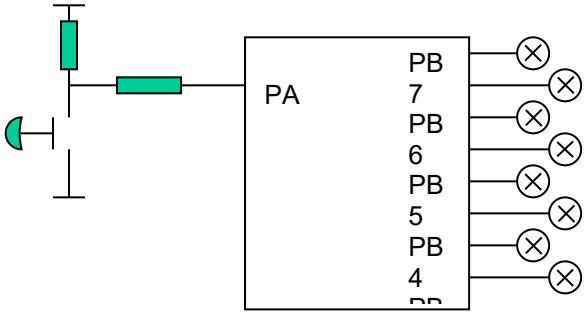
Presentation of results:

Present each task for the teacher when you have solved the task. A written report of all assignments should be submitted after each lab, containing the code and a brief description of results. The report must also include flowcharts for all programs.
The report should be sent to the lab teacher within 1 week, thus before next week lab.
Use text in the programs (comments) to explain the function. Each program must also have a header like the example below.

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;               1DT301, Computer Technology I
;               Date: 2016-09-15
;               Author:
;                               Student name 1
;                               Student name 2
;
;               Lab number: 1
;               Title:          How to use the PORTs. Digital input/output. Subroutine call.
;
;               Hardware:   STK600, CPU ATmega2560
;
;               Function: Describe the function of the program, so that you can understand it,
;                       even if you're viewing this in a year from now!
;
;               Input ports: Describe the function of used ports, for example on-board switches
;                               connected to PORTA.
;
;               Output ports: Describe the function of used ports, for example on-board LEDs
;                               connected to PORTB.
;
;               Subroutines:                    If applicable.
;               Included files:                 m2560def.inc
;
;               Other information:
;
;               Changes in program: (Description and date)
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

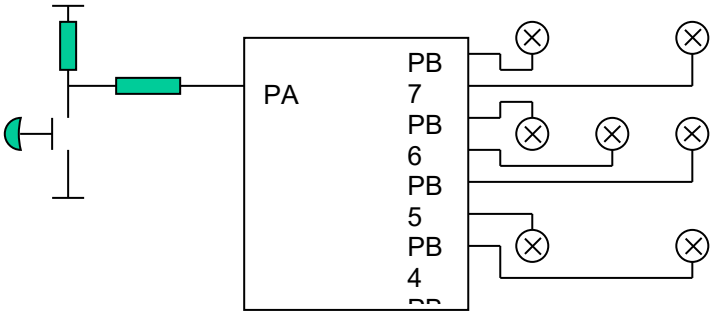
Task 1:
Switch – Ring counter / Johnson counter

Write a program which switch between Ring counter and Johnson counter. You should not use Interrupt in this lab. The pushbutton must be checked frequently, so there is no delay between the button is pressed and the change between Ring/Johnson. Use SW0 (PA0) for the button. Each time you press the button, the program should change counter.



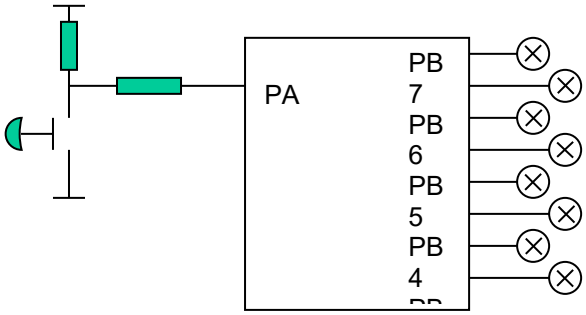
Task 2:
Electronic dice

You should create an electronic dice. Think of the LEDs placed as in the picture below. The number 1 to 6 should be generated randomly. You could use the fact that the time you press the button varies in length.



Task 3:
Change counter

Write a program that is able to count the number of changes on a switch. As a change we count when the switch SW0 goes from 0 to 1 and from 1 to 0, we expect therefore positive and negative edges. We calculate the changes in a byte variable and display its value on PORTB.



Task 4:
Delay subroutine with variable delay time

Modify the program in task 5 in Lab 1 to a general delay routine that can be called from other programs. It should be named **wait_milliseconds.** The number of milliseconds should be transferred to register pair R24, R25.

Using subroutines:

When calling subroutines in a program, the return jump address will be stored on the stack. That is handled automatically by the processor. But for this to work the Stack Pointer has to be initialized in the program. It can for example be done by the following instructions in the initialization part of the program.

```
.equ    SPH     =0x3E
.equ    SPL     =0x3D
.equ    RAMEND  =0x21FF         (Atmega16:  RAMEND = 0x45F)

; initialize the Stack Pointer (SP) to the highest address in SRAMs (RAMEND)

        ldi     r16, HIGH(RAMEND)   ; MSB part av address to RAMEND
        out     SPH, r16            ; store in SPH
        ldi     r16, LOW(RAMEND)    ; LSB part av address to RAMEND
        out     SPL, r16            ; store in SPL
```

RAMEND, SPH and SPL is defined in the file m2560def.inc for the ATMega2560 CPU which can therefore be included in the program with the following line. The three .EQU states above will thus not be needed:

```
.include “m2560def.inc”                 ; Define names for ATmega2560
```

You can also write the correct address instead of the SPH, SPL and RAMEND as follows:

```
ldi r20, HIGH(0x21FF)     ; R20 = high byte of RAMEND address, 0x21
out 0x3E,R20              ; SPH = high part of pointer to STACK
ldi R20, LOW(0x21FF)      ; R20 = low byte of RAMEND address, 0xFF
out 0x3D,R20              ; SPL = low part of pointer to STACK
```

or like this:

```
ldi  r16, 0x21            ; r16 = high byte of RAMEND address, 0x21
out  0x3E,r16             ; 0x3E = SPH = high part of pointer to STACK
ldi  r16,0xFF             ; r16 = low byte of RAMEND address, 0xFF
out  0x3D,r16             ; 0x3D = SPL = low part of pointer to STACK
```