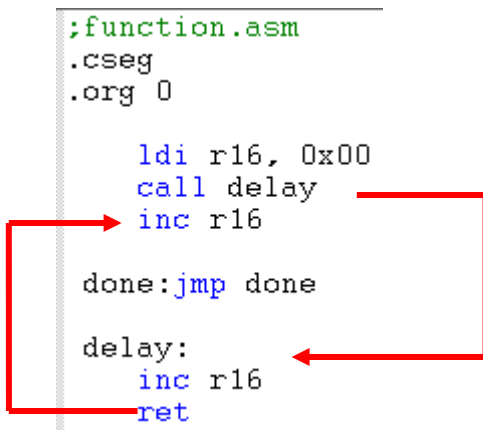


Lab 4 Simple Function Call and Input Instructions

I. Simple Function Call

In the first year programming language courses such as CSc 110, CSc 111, we wrote many functions. In assembly language, the term procedure, function, or subroutine can be used interchangeably. A function call implies a transfer (branch, jump) to an address representing the entry point of the function, causing execution of the body of the function. When the function is finished, another transfer occurs to resume execution at the statement following the call. The first transfer is the function call (for invocation), the second transfer is the return (to get back to the calling program). Together, this constitutes the processor's call-return mechanism.

In today's lab, we are going to write a simple function – no parameter passing, no returned value. Create a new project named lab4. Type the following code:



```

;function.asm
.cseg
.org 0

    ldi r16, 0x00
    call delay
    inc r16

done: jmp done

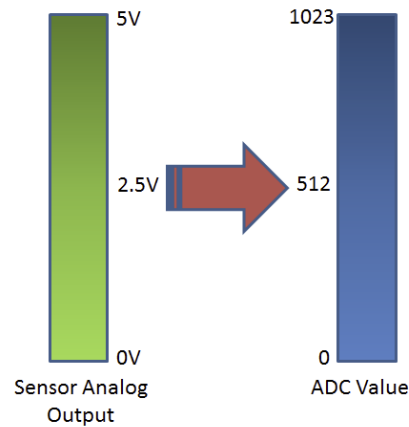
delay:
    inc r16
    ret
  
```

Observe how **pc** (program counter) is changed. How program is transferred to the function “delay” and is returned to the statement (**inc r16**) after the call. The value in **r16** is changed in the function call.

II. Exercises1: download blink_lab4.asm, change the code to a function call.

III. Analog Inputs from Push Buttons

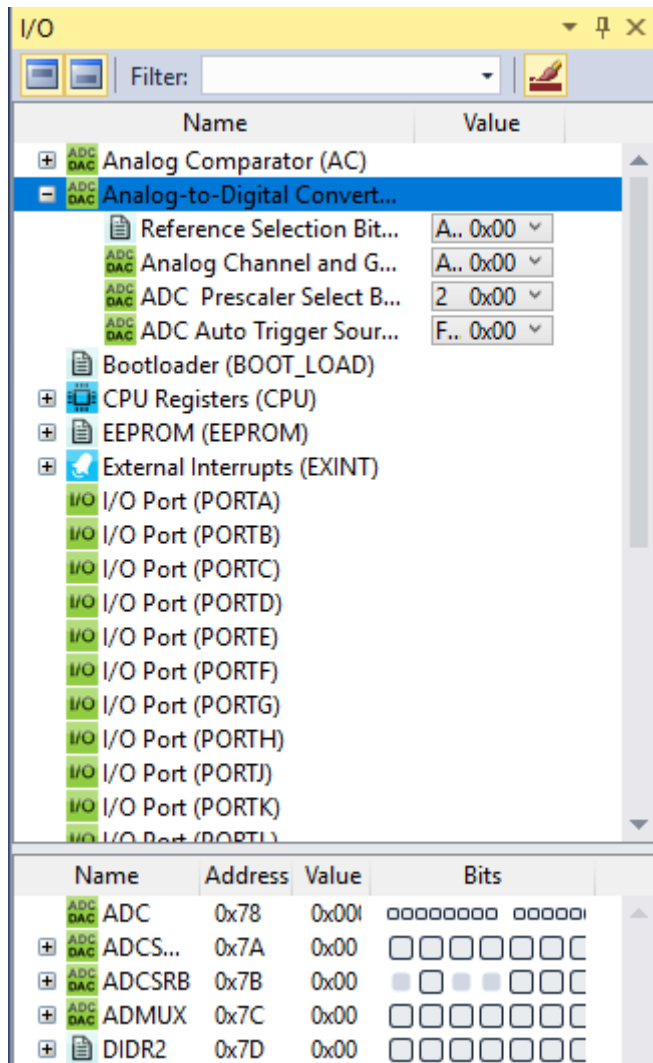
The five push buttons (exclude the *RST*- reset button) on the LCD shield are connected to pins of Port F and Port K on the board. Analog means continuous time signal and we need to use the built-in Analog to Digital Convertor (ADC) to convert continuous electronic analog signal to digital value first. When a signal is sampled, the ADC converts the analog value to a 10-bit digital value as shown below:



ATMEGA16/32

- 8 channels » 8 pins
- 10 bit resolution
- $2^{10} = 1024$ steps

The following is the I/O View of the ADC:



The memory addresses in SRAM are shown in the diagram below:

Name	Address	Value	Bits
ADC	0x78	0x00	00000000 00000001
ADCS...	0x7A	0x00	00000000 00000000
ADCSRB	0x7B	0x00	00000000 00000000
ADMUX	0x7C	0x00	00000000 00000000
DIDR2	0x7D	0x00	00000000 00000000
DIDR0	0x7E	0x00	00000000 00000000

Several registers must be set up properly before the ADC starts to digitize an analog signal. They are: ADCSR A/B (Control and Status Register A/B), ADMUX (Multiplexer Selection Register: selects a channel, as ADC can handle up to 8 channels) and ADC (10-bit ADC Data register). ADC is split into two 8-bit registers ADCL (8-bits) and ADCH (only 2-bits are used)

Expand the “+” sign for details. Give them a symbolic name using the following code:

```
; Definitions for using the Analog to Digital Conversion
.equ ADCSRA_BTN=0x7A
.equ ADCSRB_BTN=0x7B
.equ ADMUX_BTN=0x7C
.equ ADCL_BTN=0x78
.equ ADCH_BTN=0x79
```

Set the proper values for those I/O registers:

Name	Address	Value	Bits
ADC	0x78	0x00	00000000 00000000
ADCS...	0x7A	0x00	00000000 00000000
AD...	0x00	0x00	00000000 00000000
AD...	0x00	0x00	00000000 00000000
AD...	0x00	0x00	00000000 00000000
ADIF	0x00	0x00	00000000 00000000
ADIE	0x00	0x00	00000000 00000000
AD...	0x00	0x00	00000000 00000000
ADCSRB	0x7B	0x00	00000000 00000000
ADMUX	0x7C	0x00	00000000 00000000
REFS		0x00	00000000 00000000
AD...		0x00	00000000 00000000
M...		0x00	00000000 00000000
DIDR2	0x7D	0x00	00000000 00000000
DIDR0	0x7E	0x00	00000000 00000000

A brief description of the registers:

ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

ADCL and ADCH – The ADC Data Register

ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
(0x79)	–	–	–	–	–	–	ADC9	ADC8	ADCH
(0x78)	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	

ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
(0x79)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
(0x78)	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	

To initialize the ADC, several I/O registers should be set up properly. These are: **ADEN** bit in **ADCSRA**, reference signal bits **REFS1:S0** in **ADMUX**, the channel to be selected. The **ADLAR** bit affects the presentation of the ADC conversion result in the ADC Data Register. When this bit is 1, the result is left adjusted. Otherwise, it is right adjusted.

```
; enable the ADC & slow down the clock from 16mHz to ~125kHz, 16mHz/128
ldi r16, 0x87 ;0x87 = 0b10000111
sts ADCSRA_BTN, r16

ldi r16, 0x00
sts ADCSRB_BTN, r16 ; combine with MUX4:0 in ADMUX_BTN to select ADC0 p282

; bits 7:6(REFS1:0) = 01: AVCC with external capacitor at AREF pin p.281
; bit 5 ADCL_BTNR(ADC Left Adjust Result) = 0: right adjustment the result
; bits 4:0 (MUX4:0) = 00000: combine with MUX5 in ADCSRB_BTN ->ADC0 channel is used.
ldi r16, 0x40 ;0x40 = 0b01000000
sts ADMUX_BTN, r16
```

ADCSRA: 0x87 = 0b 1000 0111 (means enable the ADC and slow down the ADC clock from 16mHz to ~125kHz, 16mHz/128).

ADMUX: 0x40 = 0b 0100 0000 (means use the AVCC with external capacitor at AREF pin and use the right adjustment since ADCLAR is 0, ADC0 channel is used.) The reason for the right adjustment is that the analog signal is digitized to a 10 bits value. ADCH:ADCL is 2 bytes (16bits). The digitized value is either left-adjusted or right-adjusted.

```

check_button:
    ; start a2d
    lds r16, ADCSRA_BTN

    ; bit 6 =1 ADSC (ADC Start Conversion bit), remain 1 if conversion not done
    ; ADSC changed to 0 if conversion is done
    ori r16, 0x40 ; 0x40 = 0b01000000
    sts ADCSRA_BTN, r16

    ; wait for it to complete, check for bit 6, the ADSC bit
wait:    lds r16, ADCSRA_BTN
        andi r16, 0x40
        brne wait

    ; read the value, use XH:XL to store the 10-bit result
    lds DATAL, ADCL_BTN
    lds DATAH, ADCH_BTN

    clr r23
    ; if DATAH:DATAL < BOUNDARY_H:BOUNDARY_L
    ;     r23=1 "right" button is pressed
    ; else
    ;     r23=0
    cp DATAL, BOUNDARY_L
    cpc DATAH, BOUNDARY_H
    brsh skip
    ldi r23,1
skip:    ret

```

In Single Conversion mode, turn on the ADC start conversion bit to start.

In Single Conversion mode, this bit is 0 if the conversion is in progress, 1 if the conversion is completed.

IV. Exercises: download button.asm and write the delay function you did in the exercise II.

This lab is derived from the Chapter 6 of the book (Some Assembly Required by Timothy S. Margush) and section 26 (ADC – Analog to Digital Converter) of the datasheet of ATMEGA 2560 (See AVR Resources on connex)

The schematic on the LCD shield is posted below:

<http://www.dfrobot.com/image/data/DFR0009/LCDKeypad%20Shield%20V1.0%20SCH.pdf>