Intro to microcontrollers

Lab 2 raport

Creator: Sampo Kajaste 1881563

TEKIJÄ:

SISÄLTÖ

[1 Goal 4](#_Toc95609630)

[1.1 Objectives 4](#_Toc95609631)

[1.2 Instructions 4](#_Toc95609632)

[2 Code 5](#_Toc95609633)

[3 Self-reflect 9](#_Toc95609634)

# Goal

A microcontroller requires communicating with other devices. It should receive data, process them, and send out the results. There are many ways to connect a microcontroller to other chips. A popular method for exchanging data between a microcontroller and a device is through Input and Output (I/O) ports. In this lab, you learn how to read digital inputs from push button switches and how to check the output of a program through LEDs.

## Objectives

write assembly programs that:

• Read data from input ports

• Write data to the output ports.

• Implement loops and if-then-else statements using control instructions.

## Instructions

Write a ping-pong program. The program receives two inputs through SW1 and SW2 in PIC18 explorer board. SW1 is connected to RB0 and SW2 is connected to RA5. The program turns on/off eight LEDs that are connected to RD0~RD7. Initially, the LED in the far right is on and it moves to the left. When it reaches to the far left, if SW1 is pressed, then the direction changes and the on LED moves towards the right. Similarly, when the on LED reaches to the far right, if SW2 is pressed, then the direction changes. If none of the switches is pressed, then the direction of the on LED does not change.

# Code

; PIC18F87J11 Configuration Bit Settings

; Assembly source line config statements

#include "p18f87j11.inc"

; CONFIG1L

CONFIG WDTEN = OFF ; Watchdog Timer Enable bit (WDT disabled (control is placed on SWDTEN bit))

CONFIG STVREN = OFF ; Stack Overflow/Underflow Reset Enable bit (Reset on stack overflow/underflow disabled)

CONFIG XINST = OFF ; Extended Instruction Set Enable bit (Instruction set extension and Indexed Addressing mode disabled (Legacy mode))

; CONFIG1H

CONFIG CP0 = OFF ; Code Protection bit (Program memory is not code-protected)

; CONFIG2L

CONFIG FOSC = HS ; Oscillator Selection bits (HS oscillator)

CONFIG FCMEN = ON ; Fail-Safe Clock Monitor Enable bit (Fail-Safe Clock Monitor enabled)

CONFIG IESO = ON ; Two-Speed Start-up (Internal/External Oscillator Switchover) Control bit (Two-Speed Start-up enabled)

; CONFIG2H

CONFIG WDTPS = 32768 ; Watchdog Timer Postscaler Select bits (1:32768)

; CONFIG3L

CONFIG EASHFT = ON ; External Address Bus Shift Enable bit (Address shifting enabled, address on external bus is offset to start at 000000h)

CONFIG MODE = MM ; External Memory Bus Configuration bits (Microcontroller mode - External bus disabled)

CONFIG BW = 16 ; Data Bus Width Select bit (16-bit external bus mode)

CONFIG WAIT = OFF ; External Bus Wait Enable bit (Wait states on the external bus are disabled)

; CONFIG3H

CONFIG CCP2MX = DEFAULT ; ECCP2 MUX bit (ECCP2/P2A is multiplexed with RC1)

CONFIG ECCPMX = DEFAULT ; ECCPx MUX bit (ECCP1 outputs (P1B/P1C) are multiplexed with RE6 and RE5; ECCP3 outputs (P3B/P3C) are multiplexed with RE4 and RE3)

CONFIG PMPMX = DEFAULT ; PMP Pin Multiplex bit (PMP port pins connected to EMB (PORTD and PORTE))

CONFIG MSSPMSK = MSK7 ; MSSP Address Masking Mode Select bit (7-Bit Address Masking mode enable)

pattern equ 0x25

counter equ 0x26

delay\_count1 equ 0x27

delay\_count2 equ 0x28

org 0x0

goto start

start:

;configuration of PORTD(LED) PORTB0(push button1), and PORTA5(push button2)

bsf WDTCON,ADSHR ;Shared SFR

setf ANCON0 ;bsf ANCON0,PCFG4 <--- this should be used instead

bcf WDTCON,ADSHR

movlw 0x00

movwf TRISD ;LEDs are connected to PORTD

bsf TRISB, 0

bsf TRISA, 5

start\_loop\_left:

; code starts with automatically putting the lights to go from right to left

call shift\_left

;if SW1 is pressed, then RB0 is zero and program will skip loop left and go for the right loop

btfsc PORTB, 0

bra start\_loop\_left

bra start\_loop\_right ;sw1 is pressed, pattern will change

start\_loop\_right:

call shift\_right

btfsc PORTA, 5

bra start\_loop\_right

bra start\_loop\_left

shift\_left:

; program moves 8 to counter so the loop will go threw 8 times

movlw .8

movwf counter

; assignment for the pattern so right most led is first lit

movlw 1

movwf pattern

loop:

; pattern is moved to port d for leds

movff pattern, PORTD

; program shifts the patterns bit to left for next round

rlncf pattern

; delay enables user to see the lights easier

call delay

; decrement and branching for loop

decf counter,f

bnz loop

return

shift\_right:

; program moves 8 to counter so the loop will go threw 8 times

movlw D'8'

movwf counter

; assignment for the pattern so left most led is first lit. Used bit representation just to try it

movlw B'10000000'

movwf pattern

loop2:

; pattern is moved to port d for leds

movff pattern, PORTD

; program shifts the patterns bit to right for next round

rrncf pattern, f

; delay enables user to see the lights easier

call delay

; decrement and branching for loop

decf counter,f

bnz loop2

return

delay:

; fairly quick counter assigning 250 to WREG and moving it to both counters.

movlw .250

movwf delay\_count1

silmukka:

movwf delay\_count2

kierros:

; nop just to waste clock cycles

nop

decf delay\_count2

bnz kierros

decf delay\_count1

bnz silmukka

return

end

# Self-reflect

Writing the program was fairly simple. Only problems I had with this was the understanding of hex digit representation and moving the bit left or right. Once I found the command for moving the bit it was only trivial writing of the program. This exercise was fun to make and I would like to make more of these in the future.