**CAMOSUN COLLEGE**

**ELECTRONICS DEPTARTMENT**

**ECET 165 - LAB 3**

**Switch Counter**

This lab involves a number of concepts. We have to build an 8 bit counter, whose output will appear on 8 LEDs connected to PORTC. Wire the LEDs to PORTC as shown in the video. Include a picture of your awesome wiring for marking. The counter will increment once on each press of the on board switch. A binary counter will go through a binary sequence for each press of the switch.

0000 0001

0000 0010 etc.

If the value in the byte becomes:

1111 1111

and the button is pressed, the value will become:

0000 0000

This is called rollover. If this was a digital counter IC, when this event occurs a carry bit is generated.

1111 1111

1 0000 0000

The 1 preceding the byte of 0’s would be the carry bit incrementing the next most significant byte of a counter. When using counter IC’s, this bit is used as an input to the next counter to increment it. In this lab we are not required to do this since we are only implementing an 8 bit counter.

Some of the code you will need to accomplish this lab has already been developed in lab2.

Firstly, you need a delay routine for switch debouncing and secondly, you should already know how to detect the switch states. Create a new project file and open the template file you saved in lab 2. Make sure the new project and file is named 165L03CounterXX.s where XX is your initials.

In terms of initialization, we will need to do the following.

PORTC has to be setup to be all digital outputs. Remember **0 (zero)** is for **O**utput and **1 (one)** is for an **I**nput. Each bit position in the port can be either an input or output. The uC sets them up by looking for the pattern of 1’s and 0’s in the TRISC register.

Note that the switch input is on PORTB bit 4.

Let’s go back and look at the counter. There is a simple instruction that operates on any register which is ideal for our purpose.

**INCF f**

Where f represents any file register. What this instruction will do, is to increment the contents of the file register that follows the command by one. This would mean that the file register chosen will contain a binary representation of the number of times the switch as has been pressed. We need only declare the address of a register for this counter and the delay loop variables. This is done near the beginning of your code: e.g.

innerCount EQU 0x00 ; Access RAM GPRs are available 0x00 to 0x5F

outerCount EQU 0x01

COUNTER EQU 0x02:

:

INCF COUNTER, F,A

If we assume the counter had a value of :

1000 1010

before the incf instruction, it should have a value of:-

1000 1011

after the instruction. We need to be able to display this value on the LEDS on PORTC.

We can accomplish this very easily using the following code:

MOVF COUNTER,W,A ; Move the value in COUNTER to the WREG

We now just have to transfer the contents of the W register to PORTC. So the code we would need is:

MOVWF PORTC,A ; display on PORTC

Optional Challenge:

Change the three instruction above to just one instruction that gives the same result on PORTC.

Picture of your awesome wiring:

A close-up of a circuit board

Description automatically generated with low confidence

Screen shot of your commented code:

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;Lab 3 - SWITCH COUNTER

;ECET165 Embedded Microcontrollers

;E165L03CounterAH.s

;Jan 24, 2023

;Aaron Huinink

;This program increments a counter with each press of the onboard switch.

;The switch is debounced to improve the counter accuracy.

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; PIC18F57Q43 Configuration Bit Settings

; Assembly source line config statements

; CONFIG1

CONFIG FEXTOSC = ECH ; External Oscillator Selection (EC (external clock) above 8 MHz)

CONFIG RSTOSC = EXTOSC ; Reset Oscillator Selection (EXTOSC operating per FEXTOSC bits (device manufacturing default))

; CONFIG2

CONFIG CLKOUTEN = OFF ; Clock out Enable bit (CLKOUT function is disabled)

CONFIG PR1WAY = ON ; PRLOCKED One-Way Set Enable bit (PRLOCKED bit can be cleared and set only once)

CONFIG CSWEN = ON ; Clock Switch Enable bit (Writing to NOSC and NDIV is allowed)

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

; CONFIG3

CONFIG MCLRE = EXTMCLR ; MCLR Enable bit (If LVP = 0, MCLR pin is MCLR; If LVP = 1, RE3 pin function is MCLR )

CONFIG PWRTS = PWRT\_OFF ; Power-up timer selection bits (PWRT is disabled)

CONFIG MVECEN = ON ; Multi-vector enable bit (Multi-vector enabled, Vector table used for interrupts)

CONFIG IVT1WAY = ON ; IVTLOCK bit One-way set enable bit (IVTLOCKED bit can be cleared and set only once)

CONFIG LPBOREN = OFF ; Low Power BOR Enable bit (Low-Power BOR disabled)

CONFIG BOREN = SBORDIS ; Brown-out Reset Enable bits (Brown-out Reset enabled , SBOREN bit is ignored)

; CONFIG4

CONFIG BORV = VBOR\_1P9 ; Brown-out Reset Voltage Selection bits (Brown-out Reset Voltage (VBOR) set to 1.9V)

CONFIG ZCD = OFF ; ZCD Disable bit (ZCD module is disabled. ZCD can be enabled by setting the ZCDSEN bit of ZCDCON)

CONFIG PPS1WAY = ON ; PPSLOCK bit One-Way Set Enable bit (PPSLOCKED bit can be cleared and set only once; PPS registers remain locked after one clear/set cycle)

CONFIG STVREN = ON ; Stack Full/Underflow Reset Enable bit (Stack full/underflow will cause Reset)

CONFIG LVP = ON ; Low Voltage Programming Enable bit (Low voltage programming enabled. MCLR/VPP pin function is MCLR. MCLRE configuration bit is ignored)

CONFIG XINST = OFF ; Extended Instruction Set Enable bit (Extended Instruction Set and Indexed Addressing Mode disabled)

; CONFIG5

CONFIG WDTCPS = WDTCPS\_31 ; WDT Period selection bits (Divider ratio 1:65536; software control of WDTPS)

CONFIG WDTE = OFF ; WDT operating mode (WDT Disabled; SWDTEN is ignored)

; CONFIG6

CONFIG WDTCWS = WDTCWS\_7 ; WDT Window Select bits (window always open (100%); software control; keyed access not required)

CONFIG WDTCCS = LFINTOSC ; WDT input clock selector (WDT reference clock is the 31.0 kHz LFINTOSC)

; CONFIG7

CONFIG BBSIZE = BBSIZE\_512 ; Boot Block Size selection bits (Boot Block size is 512 words)

CONFIG BBEN = OFF ; Boot Block enable bit (Boot block disabled)

CONFIG SAFEN = OFF ; Storage Area Flash enable bit (SAF disabled)

CONFIG DEBUG = OFF ; Background Debugger (Background Debugger disabled)

; CONFIG8

CONFIG WRTB = OFF ; Boot Block Write Protection bit (Boot Block not Write protected)

CONFIG WRTC = OFF ; Configuration Register Write Protection bit (Configuration registers not Write protected)

CONFIG WRTD = OFF ; Data EEPROM Write Protection bit (Data EEPROM not Write protected)

CONFIG WRTSAF = OFF ; SAF Write protection bit (SAF not Write Protected)

CONFIG WRTAPP = OFF ; Application Block write protection bit (Application Block not write protected)

; CONFIG10

CONFIG CP = OFF ; PFM and Data EEPROM Code Protection bit (PFM and Data EEPROM code protection disabled)

PROCESSOR 18F57Q43

#include <xc.inc>

#define BUTTON RB4 ;register b bit 4

PSECT udata\_acs ; program section unassigned data in access bank

global outerCount ; Define outerCount variable

global innerCount ; Define innerCount variable

global COUNTER ; Define COUNTER variable

outerCount: DS 1 ; set datasize 1 byte for all

innerCount: DS 1

COUNTER: DS 1

PSECT resetVec, class=CODE, reloc=2

resetVec:

goto main

PSECT code

main: ;program starts here

;from lab 2, set i/o ports

BANKSEL PORTC

clrf PORTC ;clear portc

BANKSEL TRISC

clrf TRISC,a ;make port C an output port

BANKSEL TRISB

bsf TRISB4 ;make button bit input

BANKSEL ANSELC

clrf ANSELC,a ;make led output port digital

BANKSEL ANSELB

bcf ANSELB4 ;make button bit digital

BANKSEL WPUB

bsf WPUB4 ;enable weak pullup for button

clrf WREG,A ;clear the working reg

clrf COUNTER,A ;clear counter register

clrf PORTC,A ;clear portc to turn off all LEDs at start

waitForPush:

BTFSC BUTTON ;is button pressed?

GOTO waitForPush ;no - loop again

call delay20ms ;yes - delay ~20ms

incf COUNTER, F, A ;increment counter by 1

movf COUNTER, W, A ;move value in counter to wreg

movwf PORTC,A ;display on portc

waitForRelease:

btfss BUTTON ;is button released?

goto waitForRelease ;no - loop again

goto waitForPush

delay20ms:

movlw 30 ;send a value of 30 to WREG

movwf outerCount, A ;send 30 to outerCount reg

movlw 28 ;send a value of 28 to WREG

movwf innerCount, A ;send 28 from wreg to outerCount reg

delayOuter:

delayInner:

decfsz innerCount, F, A ;decrement innerCount, leave in the file reg in the access bank

goto delayInner ;if innerCount is not 0, go back to delayInner

decfsz outerCount, F, A ;decrement outerCount, leave in file reg in access bank

goto delayOuter ;if outerCount is not 0, go back to delayOuter

nop ;no op

nop ;no op

return ;return to waitForPush

end resetVec