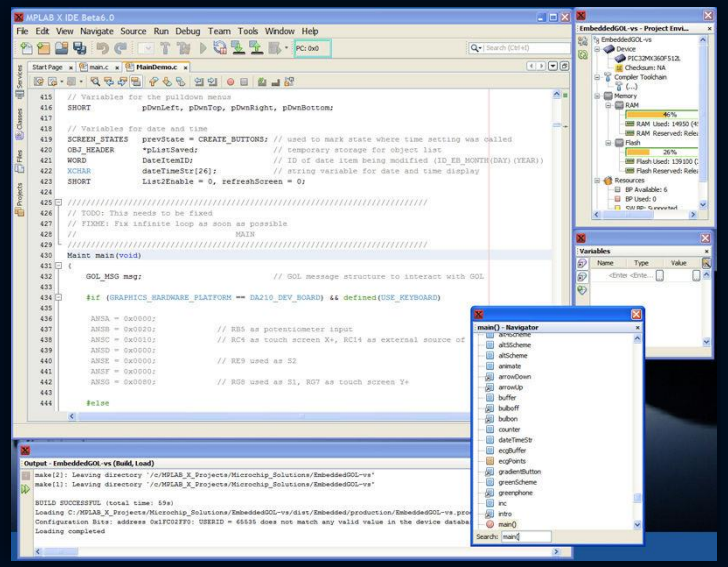
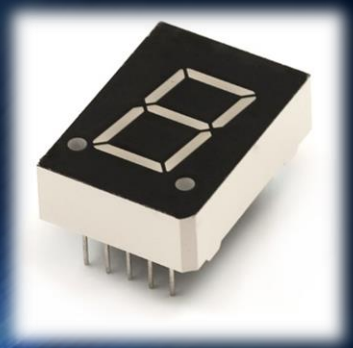
Lab Assignment Presentation

LEONARDO FUSSER (1946995)

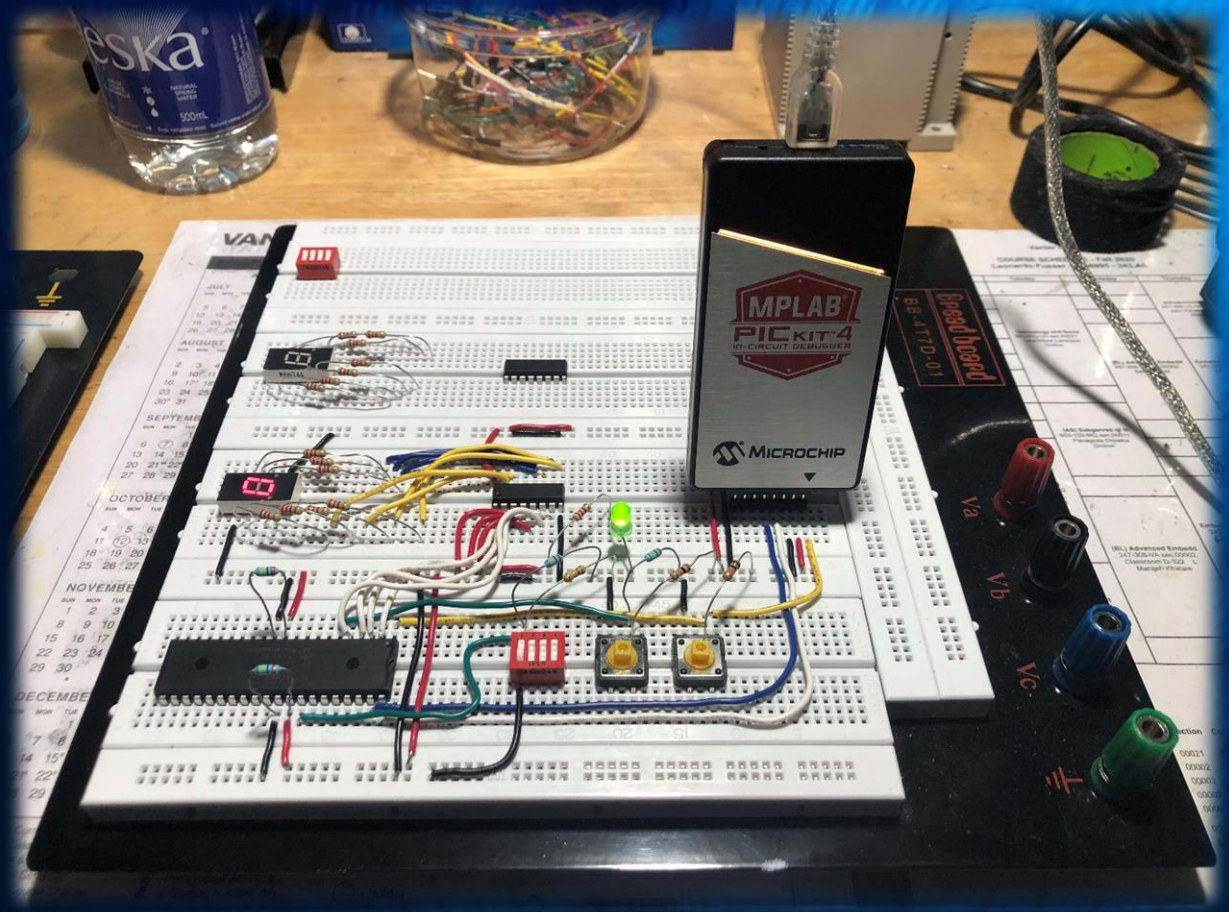
Today's Topics:

- *Hardware overview & approach*
- *Software overview & approach*
- *Problems encountered & solutions*
- *What I learned & what I could improve*





MPLAB® X IDE

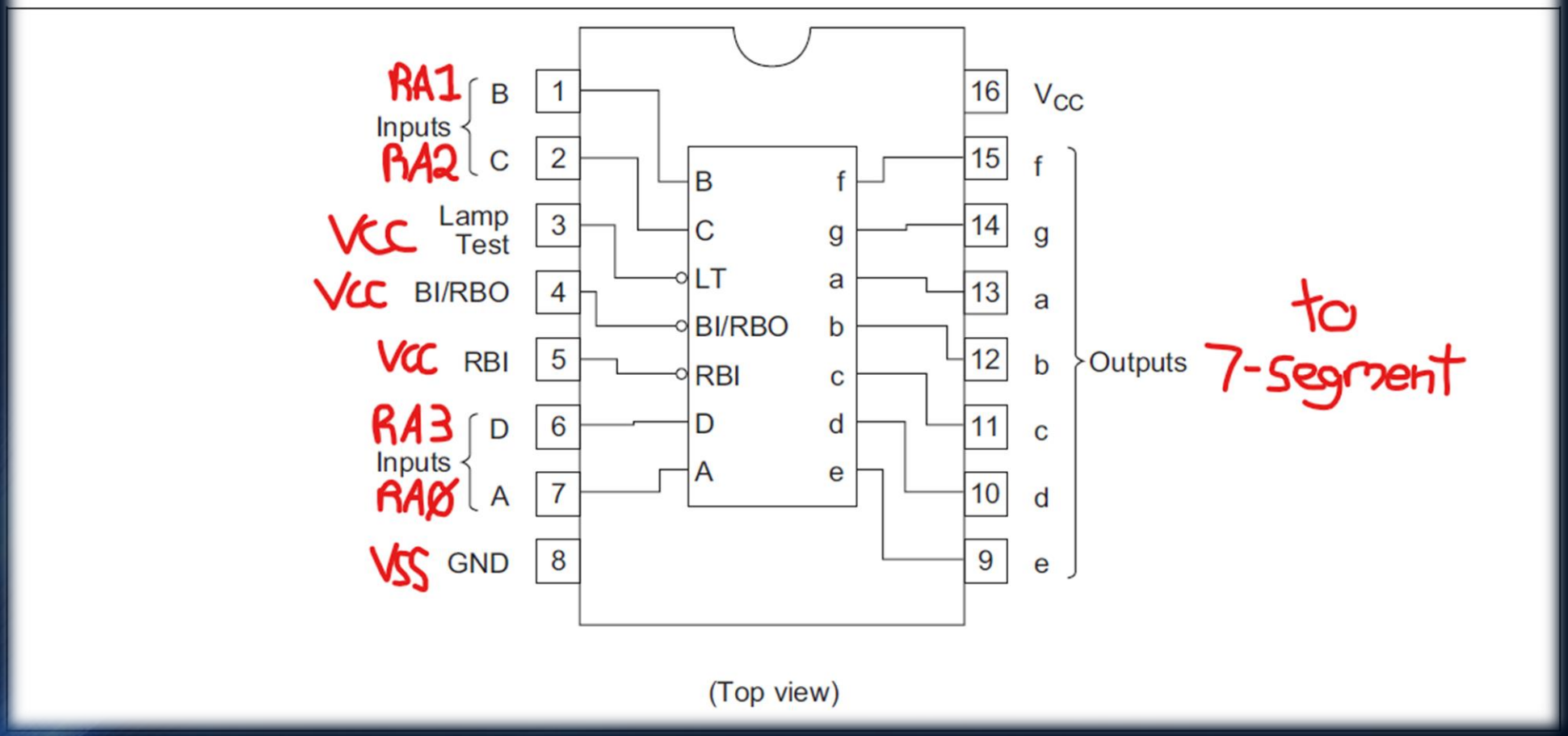


Hardware Overview

THE 7-SEGMENT DISPLAY, 7448 & DIP SWITCH INTERFACING TO PIC16F887

The 7448

Pin Arrangement



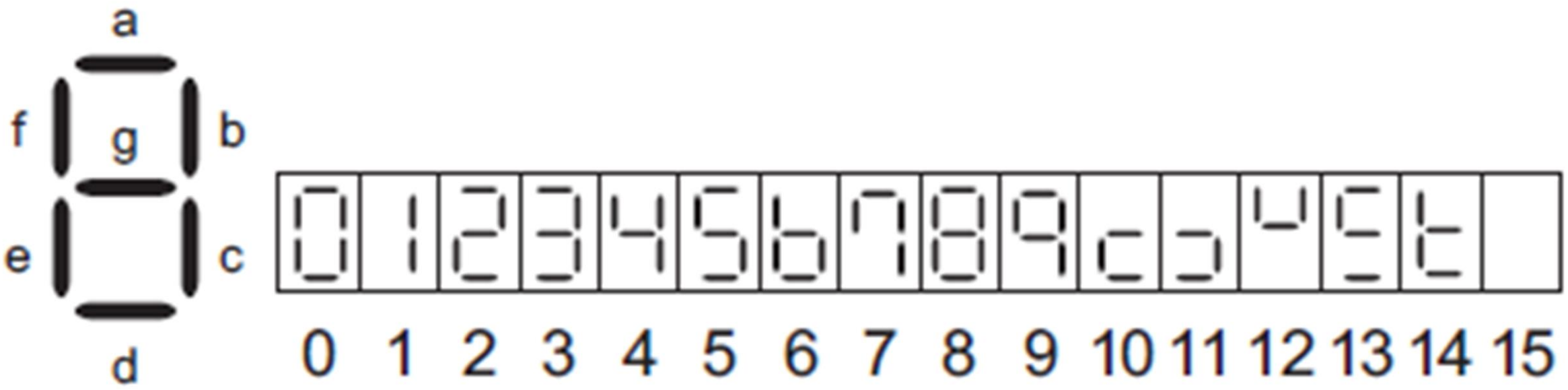
Connections to 7448 decoder/7-segment display driver

Function Table

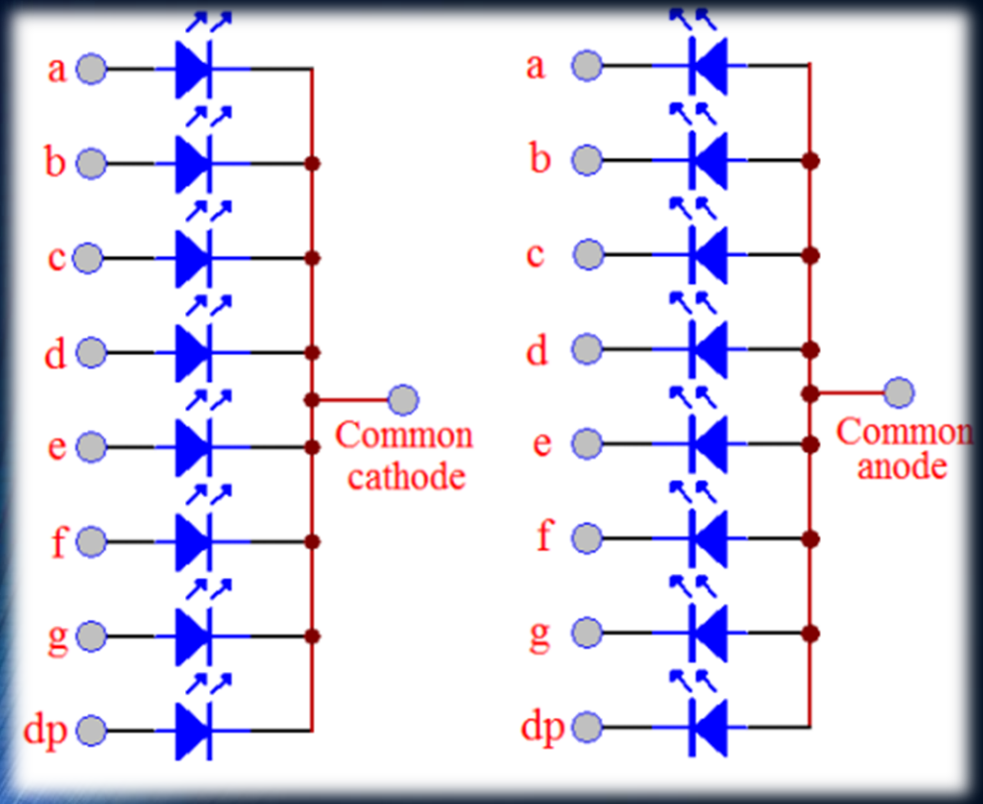
Decimal or Function	Inputs						BI / RBO	Outputs							Note
	LT	RBI	D	C	B	A		a	b	c	d	e	f	g	
0	H	H	L	L	L	L	H	H	H	H	H	H	H	L	
1	H	X	L	L	L	H	H	L	H	H	L	L	L	L	
2	H	X	L	L	H	L	H	H	H	L	H	H	L	H	
3	H	X	L	L	H	H	H	H	H	H	H	L	L	H	
4	H	X	L	H	L	L	H	L	H	H	L	L	H	H	
5	H	X	L	H	L	H	H	H	L	H	H	L	H	H	
6	H	X	L	H	H	L	H	L	L	H	H	H	H	H	
7	H	X	L	H	H	H	H	H	H	H	L	L	L	L	
8	H	X	H	L	L	L	H	H	H	H	H	H	H	H	1
9	H	X	H	L	L	H	H	H	H	H	L	L	H	H	
10	H	X	H	L	H	L	H	L	L	L	H	H	L	H	
11	H	X	H	L	H	H	H	L	L	H	H	L	L	H	
12	H	X	H	H	L	L	H	L	H	L	L	L	H	H	
13	H	X	H	H	L	H	H	H	L	L	H	L	H	H	
14	H	X	H	H	H	L	H	L	L	L	H	H	H	H	
15	H	X	H	H	H	H	H	L	L	L	L	L	L	L	
BI	X	X	X	X	X	X	L	L	L	L	L	L	L	L	2
RBI	H	L	L	L	L	L	L	L	L	L	L	L	L	L	3
LT	L	X	X	X	X	X	H	H	H	H	H	H	H	H	4

H: high level, L: low level, X: irrelevant

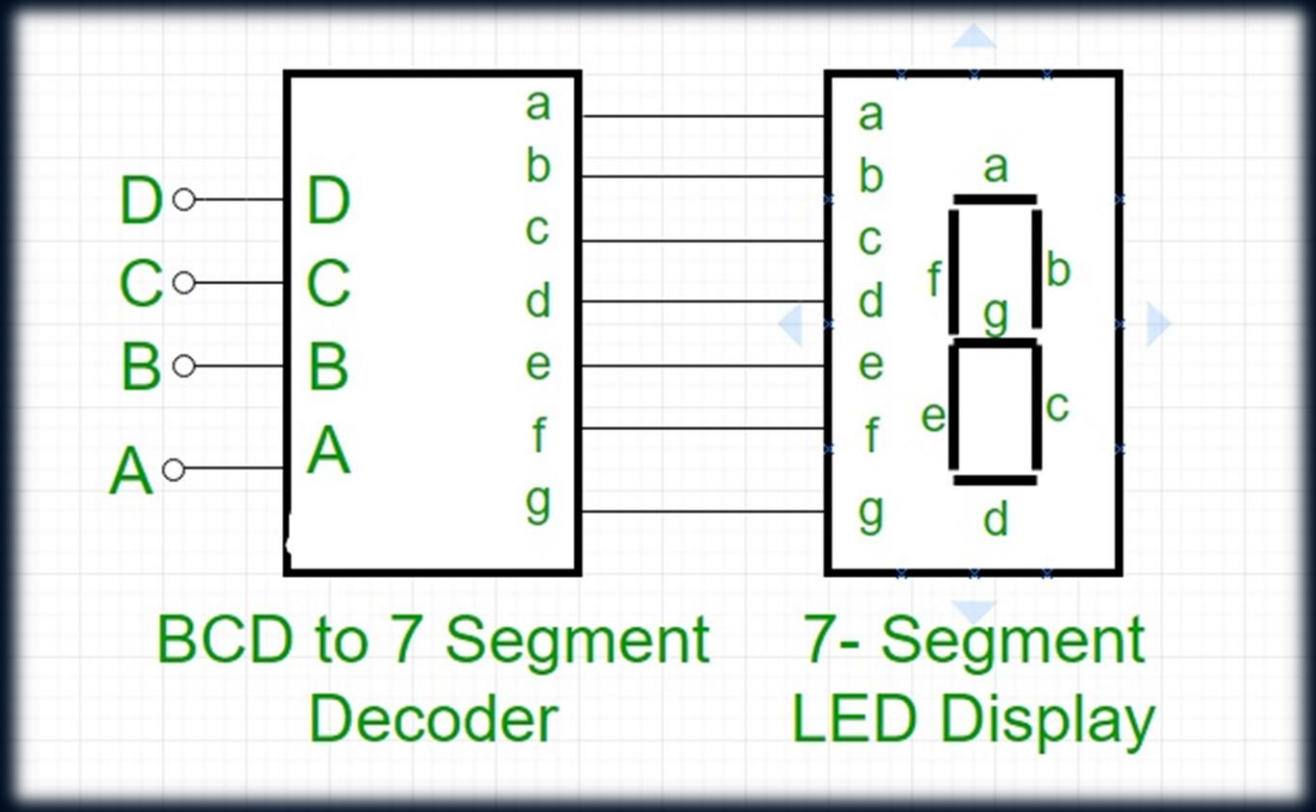
7448 truth-table



Complete possible outputs on 7-segment display

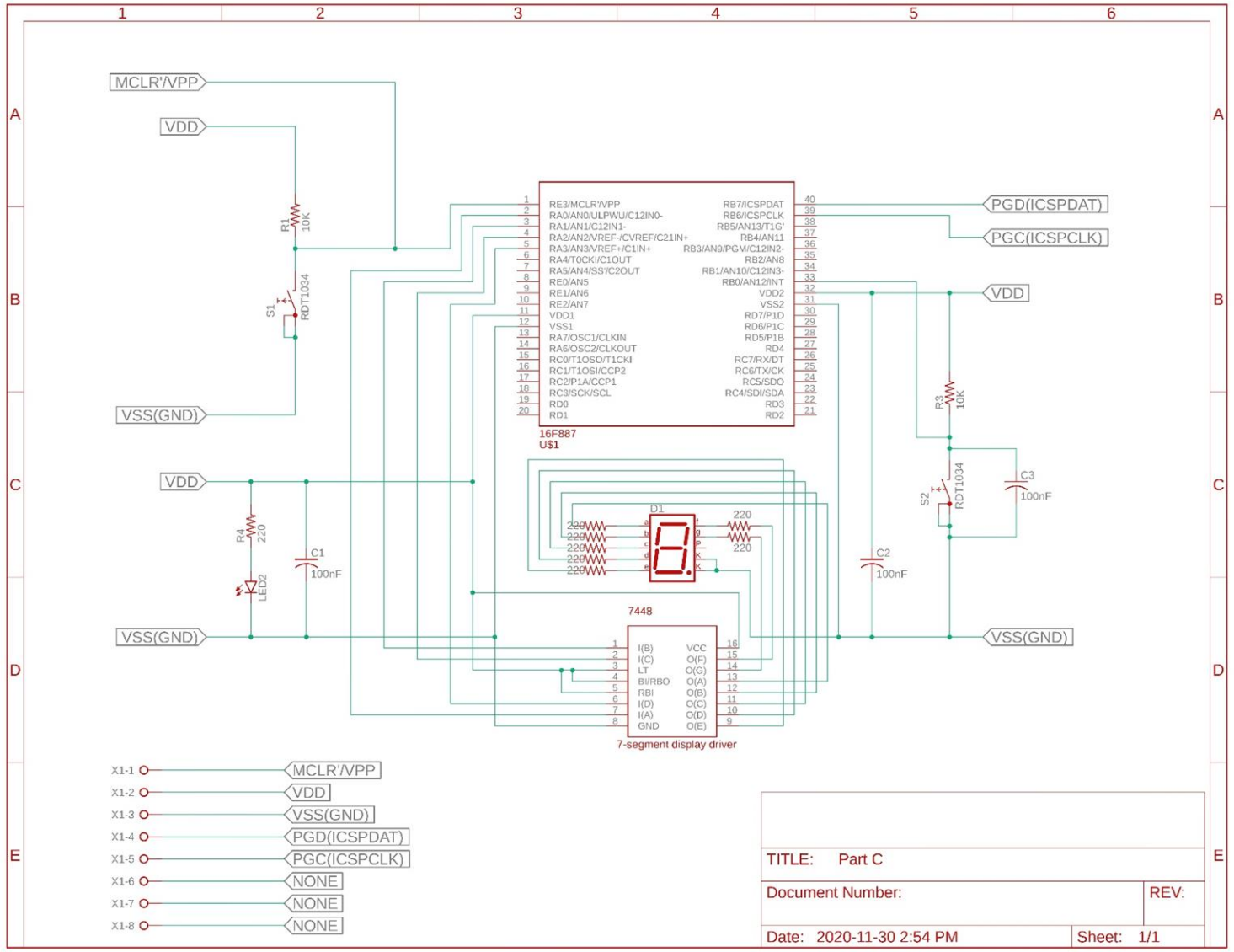


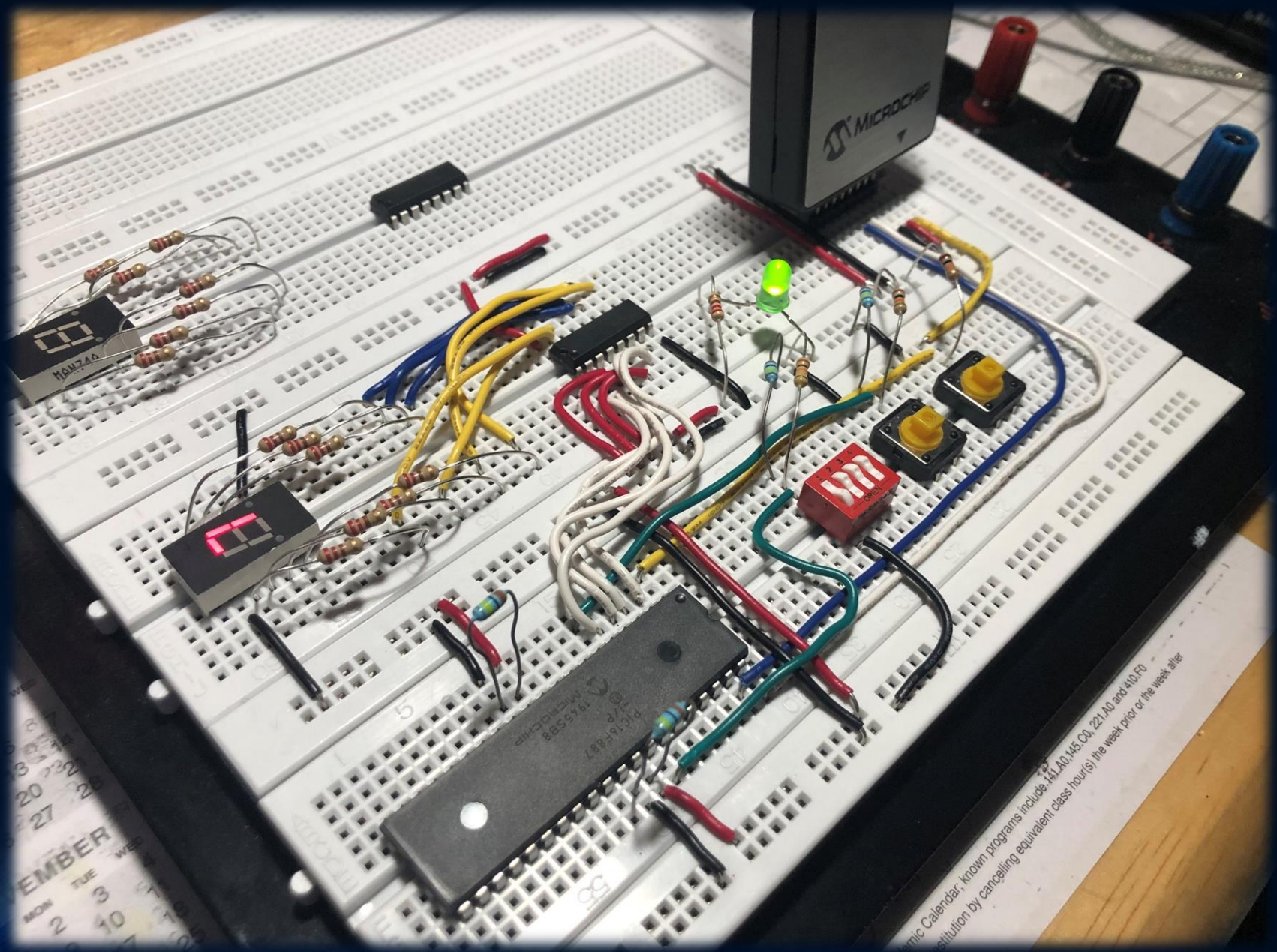
Two types of segmented displays



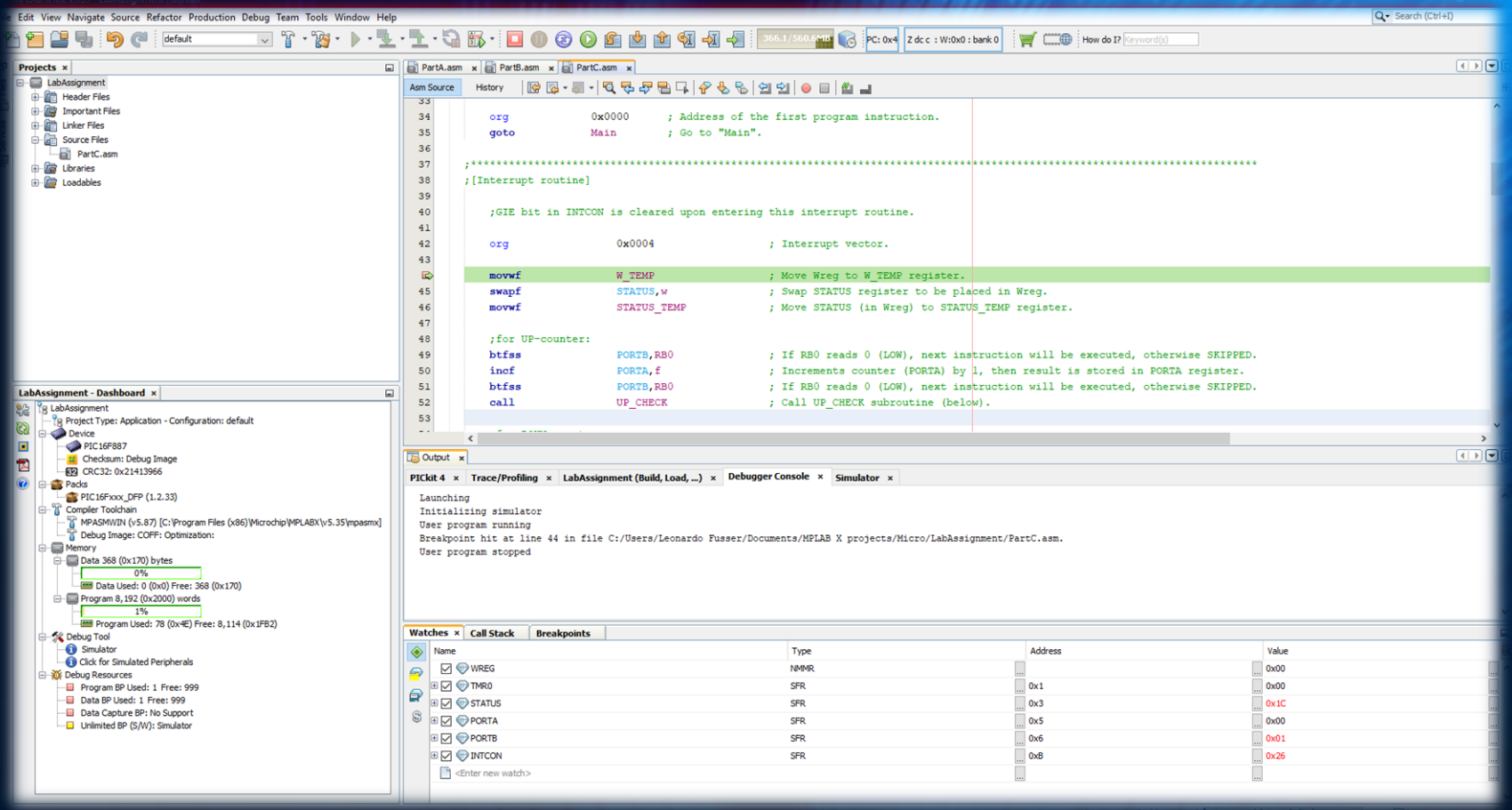
Typical hookup with common cathode 7-segment display

Complete Schematic





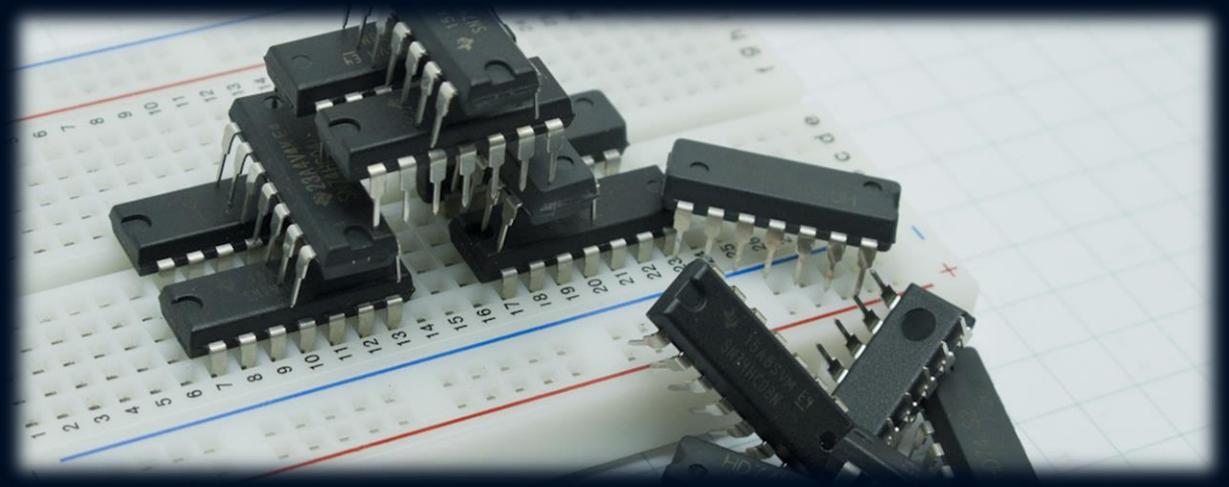
Final circuit prototype



Software Approach

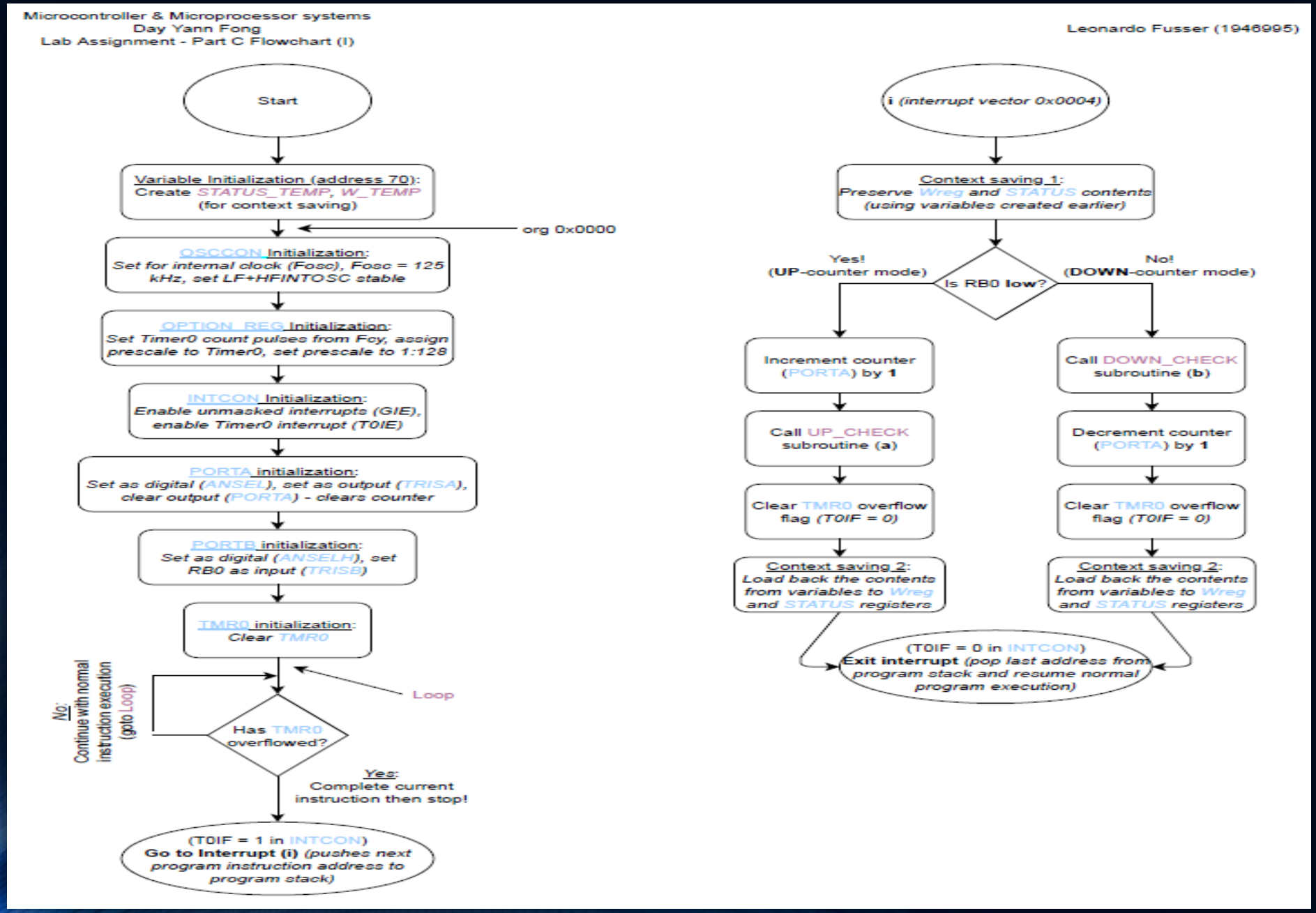
LOGIC, FLOWCHART & IMPLEMENTATION

The Logic



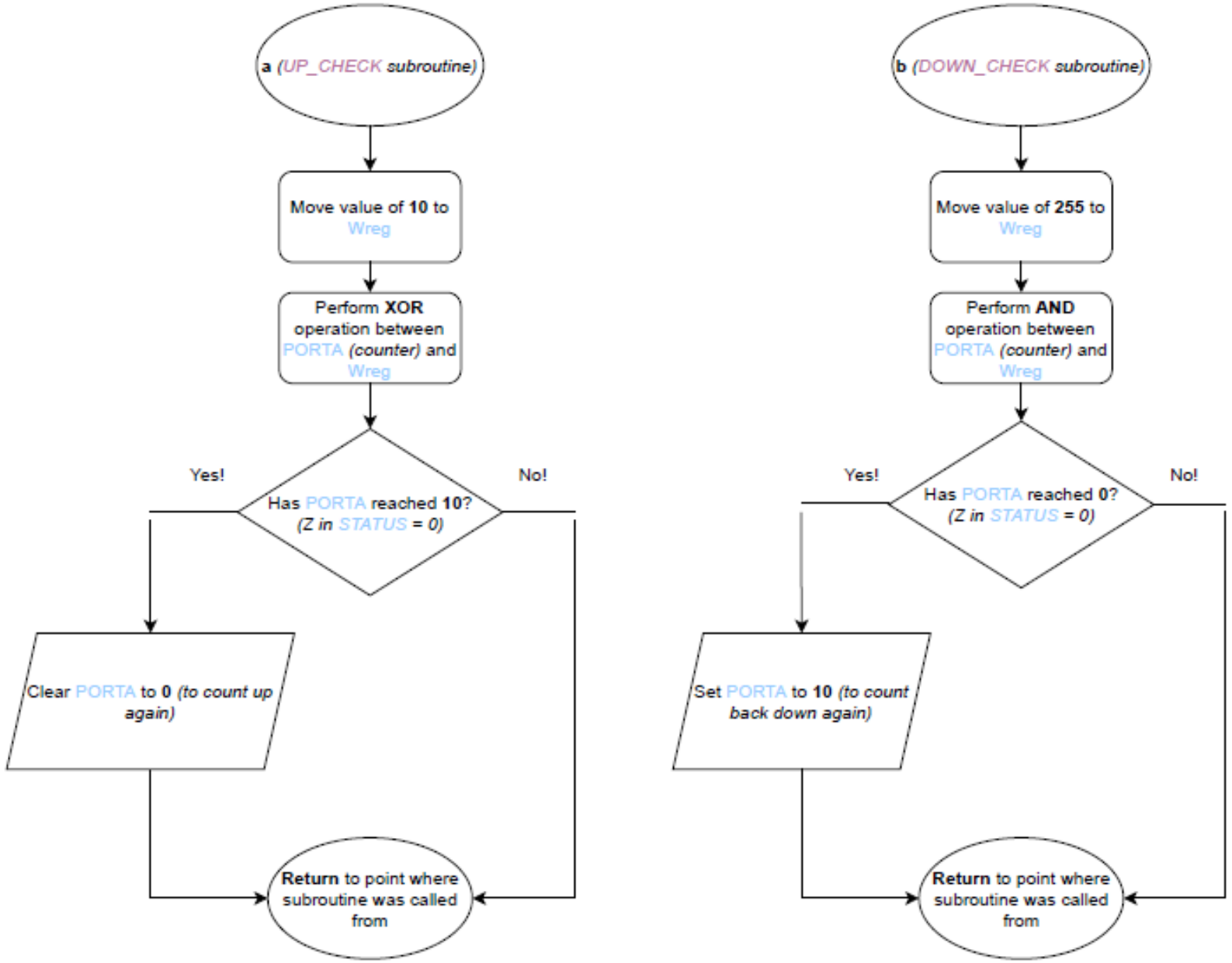
- For UP-counter -> ***0*** *to* ***9***
- For DOWN-counter -> ***9*** *to* ***0***
- Implement condition control for UP/DOWN-counter
- Implement condition evaluation in Assembly language (for threshold check)
- Use interrupts for UP/DOWN-count functionality.
- Keep **1-second delay** for UP/DOWN-counter!

Flowchart


Microcontroller & Microprocessor systems
Day Yann Fong
Lab Assignment - Part C Flowchart (I)
Leonardo Fusser (1946995)
Start
Variable Initialization (address 70):
Create STATUS_TEMP, W_TEMP
(for context saving)
org 0x0000
OSCCON Initialization:
Set for internal clock (Fosc), Fosc = 125 kHz, set LF+HFINTOSC stable
OPTION_REG Initialization:
Set Timer0 count pulses from Fcy, assign prescale to Timer0, set prescale to 1:128
INTCON Initialization:
Enable unmasked interrupts (GIE), enable Timer0 interrupt (T0IE)
PORTA initialization:
Set as digital (ANSEL), set as output (TRISA), clear output (PORTA) - clears counter
PORTB initialization:
Set as digital (ANSELH), set RB0 as input (TRISB)
TMR0 initialization:
Clear TMR0
Loop
No:
Continue with nomal instruction execution (goto Loop)
Has TMR0 overflowed?
Yes:
Complete current instruction then stop!
(T0IF = 1 in INTCON)
Go to Interrupt (i) (pushes next program instruction address to program stack)
i (interrupt vector 0x0004)
Context saving 1:
Preserve Wreg and STATUS contents (using variables created earlier)
Yes!
(UP-counter mode)
Is RB0 low?
No!
(DOWN-counter mode)
Increment counter (PORTA) by 1
Call UP_CHECK subroutine (a)
Clear TMR0 overflow flag (T0IF = 0)
Context saving 2:
Load back the contents from variables to Wreg and STATUS registers
Call DOWN_CHECK subroutine (b)
Decrement counter (PORTA) by 1
Clear TMR0 overflow flag (T0IF = 0)
Context saving 2:
Load back the contents from variables to Wreg and STATUS registers
(T0IF = 0 in INTCON)
Exit interrupt (pop last address from program stack and resume normal program execution)

Microcontroller & Microprocessor systems
Day Yann Fong
Lab Assignment - Part C Flowchart (II)

Leonardo Fusser (1946995)



Implementation

- Must read pin status using bit-oriented file register operations *(BTFSS, BTFSC)*
- Must use byte-oriented file register operation for condition evaluation:
- *Use **XOR** operation for UP-counter*
- *Use **AND** operation for DOWN-counter*
- Check **Z** in **STATUS register** to determine if condition yields true!

XOR Truth Table

Inputs		Output
A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Same logic used for UP-counter conditional check

AND Truth Table

Inputs		Output
A	B	Y = A.B
0	0	0
0	1	0
1	0	0
1	1	1

Same logic used for DOWN-counter conditional check

TABLE 15-2: PIC16F882/883/884/886/887 INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS						
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff	C, DC, Z	1, 2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff	Z	1, 2
CLRF	f	Clear f	1	00 0001 lfff ffff	Z	2
CLRW	–	Clear W	1	00 0001 0xxx xxxx	Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff	Z	1, 2
DECF	f, d	Decrement f	1	00 0011 dfff ffff	Z	1, 2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00 1011 dfff ffff		1, 2, 3
INCF	f, d	Increment f	1	00 1010 dfff ffff	Z	1, 2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00 1111 dfff ffff		1, 2, 3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff	Z	1, 2
MOVF	f, d	Move f	1	00 1000 dfff ffff	Z	1, 2
MOVWF	f	Move W to f	1	00 0000 lfff ffff		
NOP	–	No Operation	1	00 0000 0xx0 0000		
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff	C	1, 2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff	C	1, 2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff	C, DC, Z	1, 2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff		1, 2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff	Z	1, 2

Excerpt taken from pg. 227 of PIC 16F887 datasheet

TABLE 15-2: PIC16F882/883/884/886/887 INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS						
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff	C, DC, Z	1, 2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff	Z	1, 2
CLRF	f	Clear f	1	00 0001 lfff ffff	Z	2
CLRW	–	Clear W	1	00 0001 0xxx xxxx	Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff	Z	1, 2
DECF	f, d	Decrement f	1	00 0011 dfff ffff	Z	1, 2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00 1011 dfff ffff		1, 2, 3
INCF	f, d	Increment f	1	00 1010 dfff ffff	Z	1, 2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00 1111 dfff ffff		1, 2, 3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff	Z	1, 2
MOVF	f, d	Move f	1	00 1000 dfff ffff	Z	1, 2
MOVWF	f	Move W to f	1	00 0000 lfff ffff		
NOP	–	No Operation	1	00 0000 0xx0 0000		
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff	C	1, 2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff	C	1, 2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff	C, DC, Z	1, 2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff		1, 2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff	Z	1, 2

Excerpt taken from pg. 227 of PIC 16F887 datasheet

```
;for UP-counter:
btfss           PORTB,RB0           ; If RB0 reads 0 (LOW), next instruction will be executed, otherwise SKIPPED.
incf            PORTA,f             ; Increments counter (PORTA) by 1, then result is stored in PORTA register.
btfss           PORTB,RB0           ; If RB0 reads 0 (LOW), next instruction will be executed, otherwise SKIPPED.
call            UP_CHECK            ; Call UP_CHECK subroutine (below).

;for DOWN-counter:
btfsc           PORTB,RB0           ; If RB0 reads 1 (HIGH), next instruction will be executed, otherwise SKIPPED.
call            DOWN_CHECK          ; Call subroutine DOWN_CHECK (below).
btfsc           PORTB,RB0           ; If RB0 reads 1 (HIGH), next instruction will be executed, otherwise SKIPPED.
decf            PORTA,f             ; Decrements counter (PORTA) by 1, then result is stored in PORTA register.
```

Implementation done in ISR and through subroutines

```
;Subroutine for checking if counter (PORTA) has reached 10.
UP_CHECK

    movlw       b'00001010'         ; 10 in binary (in Wreg).
    xorwf       PORTA,w             ; Performs XOR operation between Wreg (value of 10) and value stored in PORTA (value of counter).
    btfsc       STATUS,Z            ; Checks to see if zero flag is set (from XOR operation above), otherwise next instruction is SKIPPED.
    clrf        PORTA               ; Clears counter back down to zero if the previous instruction yields TRUE.
    clrw                            ; Clears Wreg.

    return                          ; Return to point where subroutine was called from.
```

Subroutine to check if UP-counter reached threshold

```
;Subroutine for checking if counter (PORTA) has reached 255.
DOWN_CHECK

    movlw       b'11111111'         ; 255 in binary.
    andwf       PORTA,w             ; Performs AND operation between Wreg (value of 255) and value stored in PORTA (value of counter).
    btfsc       STATUS,Z            ; Checks to see if zero flag is set (from AND operation above), otherwise next instruction is SKIPPED.
    movlw       b'00001010'         ; 10 in binary (in Wreg).
    btfsc       STATUS,Z            ; Checks to see if zero flag is set (from AND operation above), otherwise next instruction is SKIPPED.
    movwf       PORTA               ; Moves value of 10 (in Wreg) to counter if previous instruction yields TRUE.
    clrw                            ; Clears Wreg.

    return                          ; Return to point where subroutine was called from.
```

Subroutine to check if DOWN-counter reached threshold

Observation table for 4-bit binary UP counter (0 – 9)				
Decimal value	*(Value of 00001010 is moved to Wreg when subroutine is called)*		Z flag in STATUS	PORTA goes back to 0?
	Value of counter (PORTA)	Result after xorwf PORTA,w		
0	00000000	00001010	0	No
1	00000001	00001011	0	No
2	00000010	00001000	0	No
3	00000011	00001001	0	No
4	00000100	00001110	0	No
5	00000101	00001111	0	No
6	00000110	00001100	0	No
7	00000111	00001101	0	No
8	00001000	00000010	0	No
-> 9	00001001	00000011	0	No
10**	00001010	00000000	1	Yes!

*Note: for Z flag in STATUS register, 0 -> not set (no zero) & 1 -> is set (is zero)!

**Although not shown on 7-segment, it does count up to 10 in background.

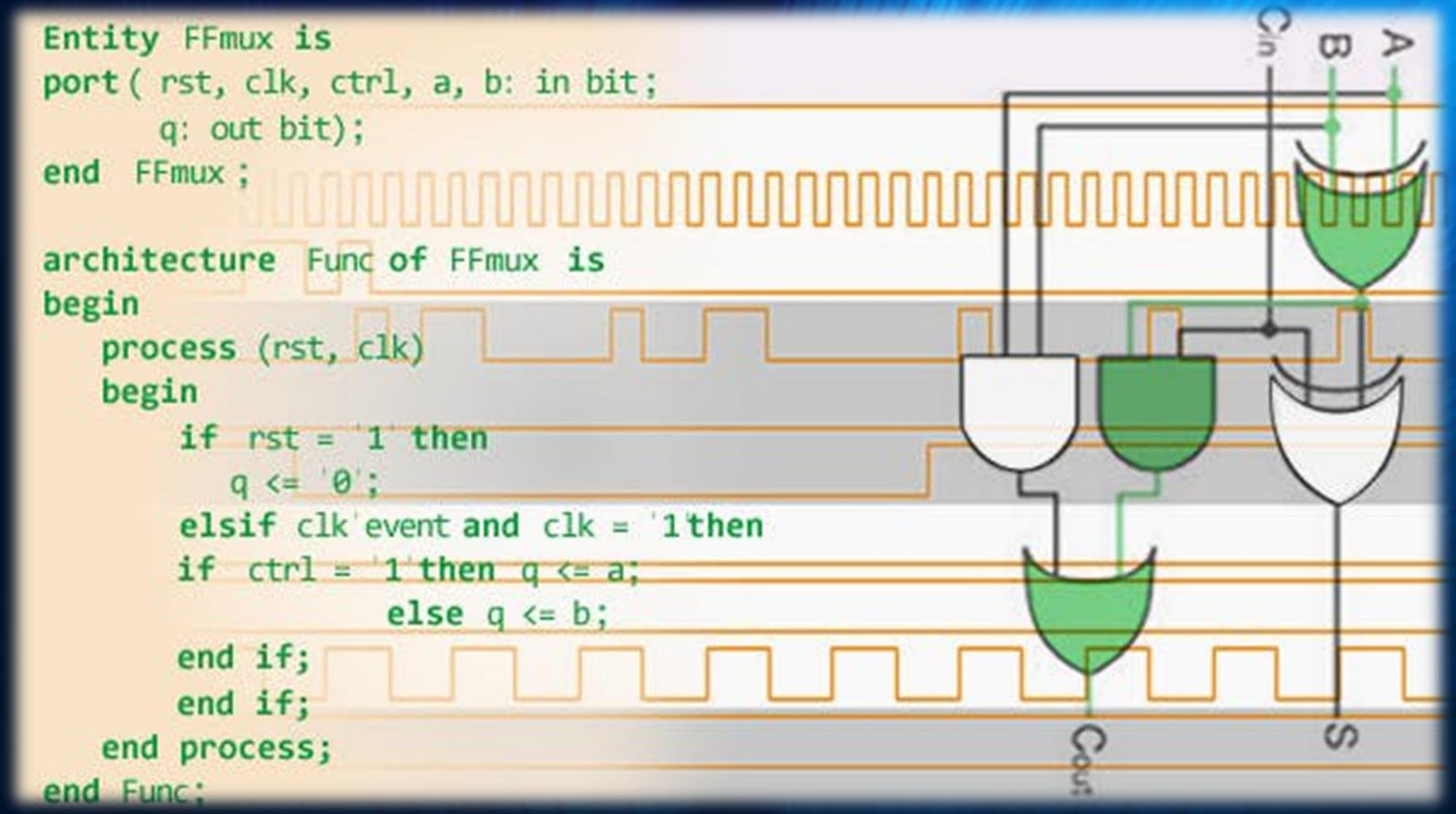
Observation done through MPLAB X IDE simulator/debugger

Observation table for 4-bit binary DOWN counter (9 - 0)				
Decimal value	*(Value of 11111111 is moved to Wreg when subroutine is called)*		Z flag in STATUS	PORTA goes back to 9?
	Value of counter (PORTA)	Result after andwf PORTA,w		
10**	00001010	00001010	0	No
-> 9	00001001	00001001	0	No
8	00001000	00001000	0	No
7	00000111	00000111	0	No
6	00000110	00000110	0	No
5	00000101	00000101	0	No
4	00000100	00000100	0	No
3	00000011	00000011	0	No
2	00000010	00000010	0	No
1	00000001	00000001	0	No
0	00000000	00000000	1	Yes!

*Note: for Z flag in STATUS register, 0 -> not set (no zero) & 1 -> is set (is zero)!

**Although not shown on 7-segment, it does count down from 10 in background.

Observation done through MPLAB X IDE simulator/debugger

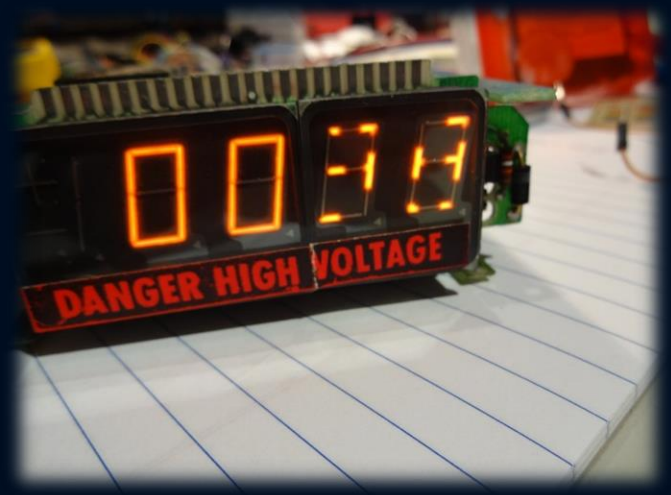


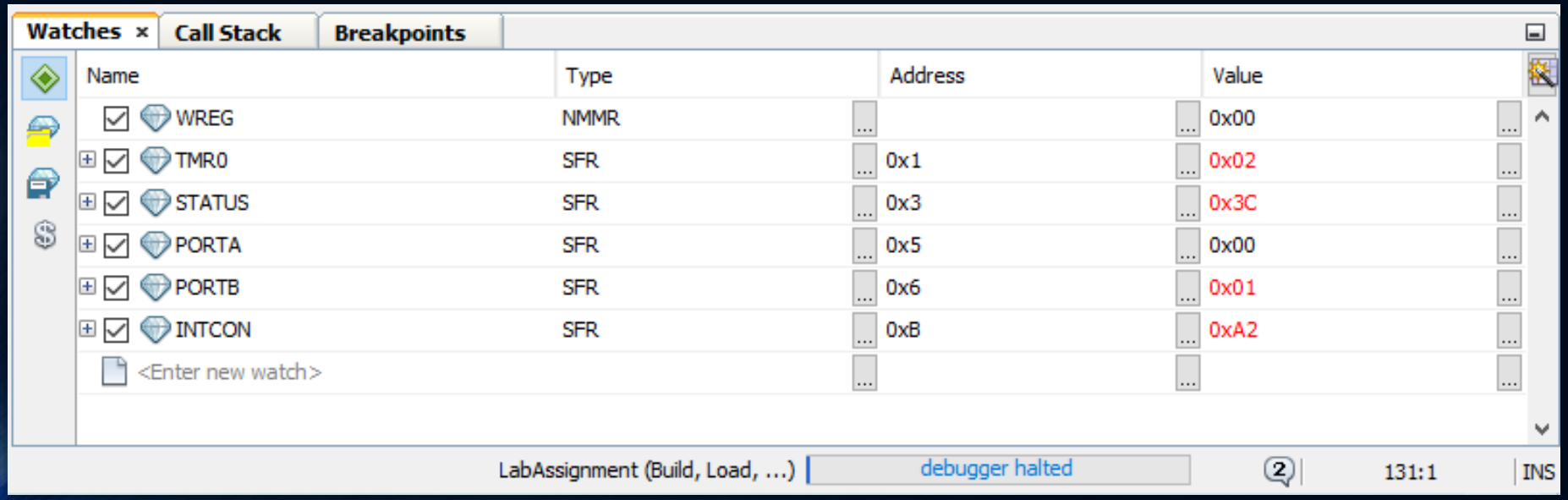
Problems Encountered & Solutions

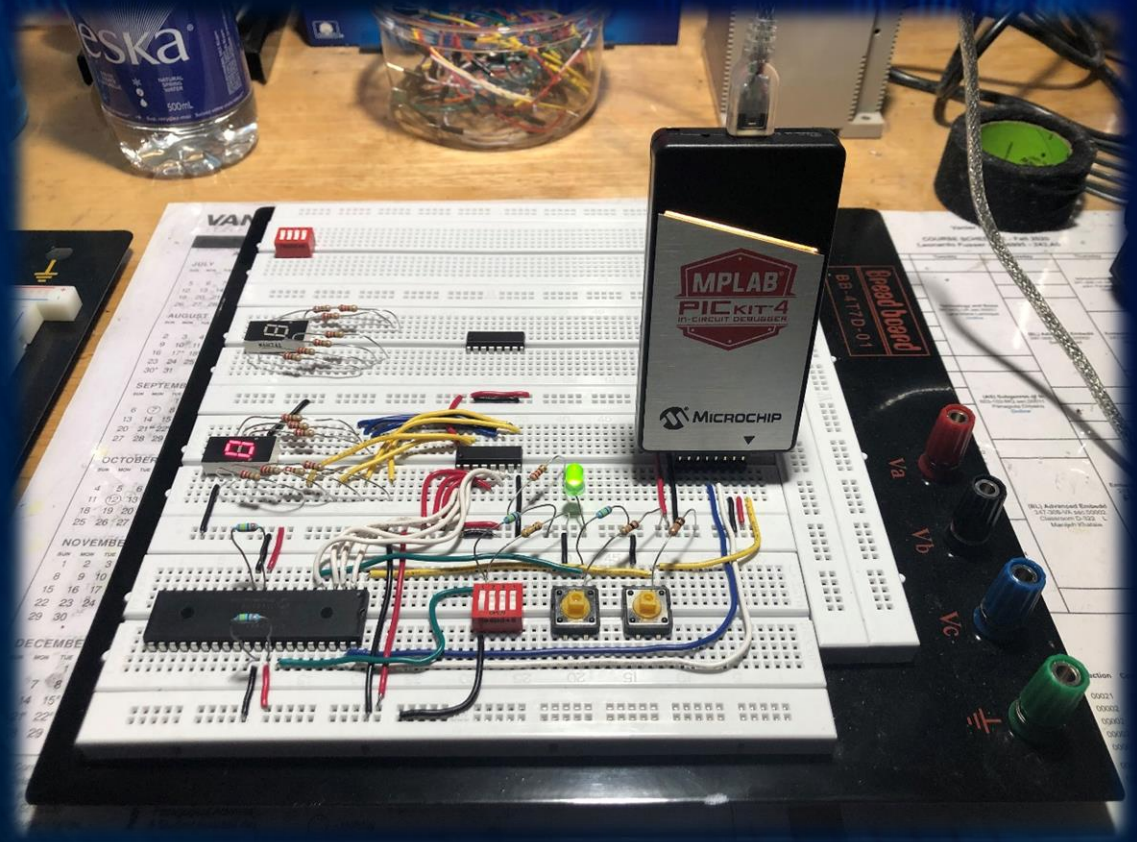
PROBLEMS & MPLAB X IDE SIMULATOR

The Problem

- For UP-counter -> ***0 - 8*** instead of ***0 - 9***
- For DOWN-counter -> ***8 - 0*** instead of ***9 - 0***
- Solution: debugging with MPLAB X IDE simulator!

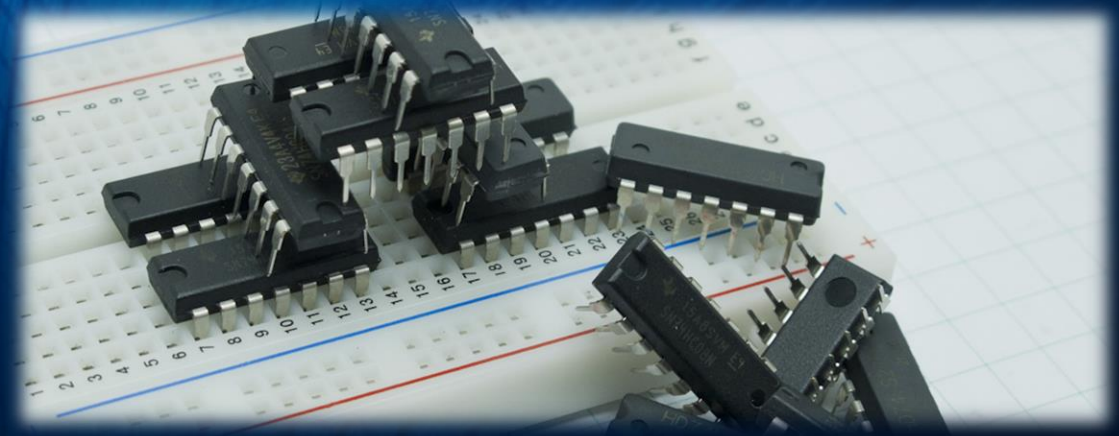







```
Entity FFmux is
port( rst, clk, ctrl, a, b: in bit;
      q: out bit);
end  FFmux ;

architecture  Func of FFmux  is
begin
   process (rst, clk)
   begin
        if  rst = '1' then
          q <= '0';
        elsif clk'event and clk = '1'then
        if  ctrl = '1' then q <= a;
                  else q <= b;
        end if;
        end if;
   end process;
```



Take away from this assignment

LESSONS LEARNED & WHAT I CAN IMPROVE ON…

Take away…

- *Learned* how to use a 7-segment display
- *Learned* how to use a 7448 decoder/7-segment display driver
- *Learned* how to implement the display with PIC 16F887
- *Learned* how to implement conditional statement in Assembly
- To improve: be more comfortable with the simulator/debugger!