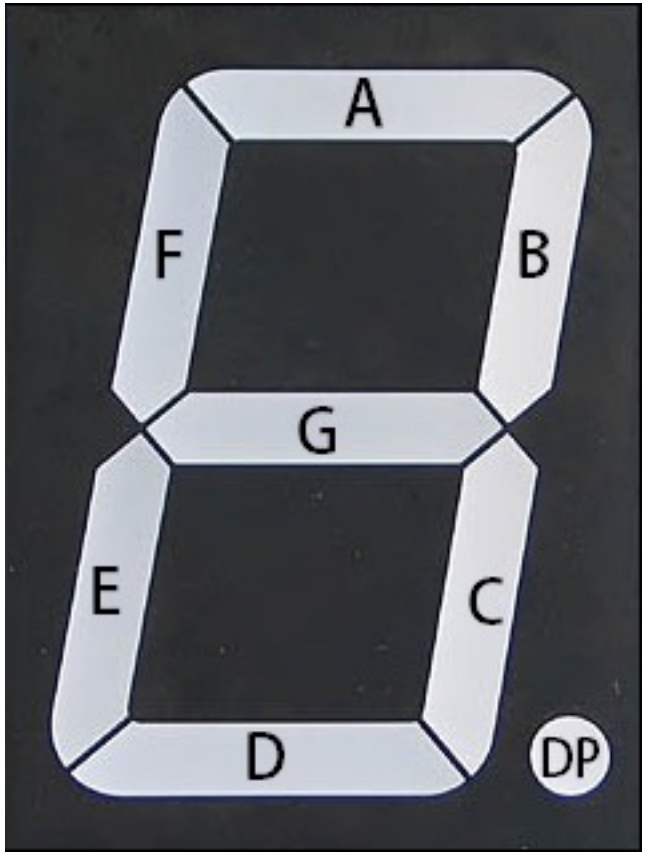
Microcontrollers and Industrial Applications 1
KOM3722
Week 7

Asst. Prof. Bahadır Çatalbaş

Common cathode configuration lookup table

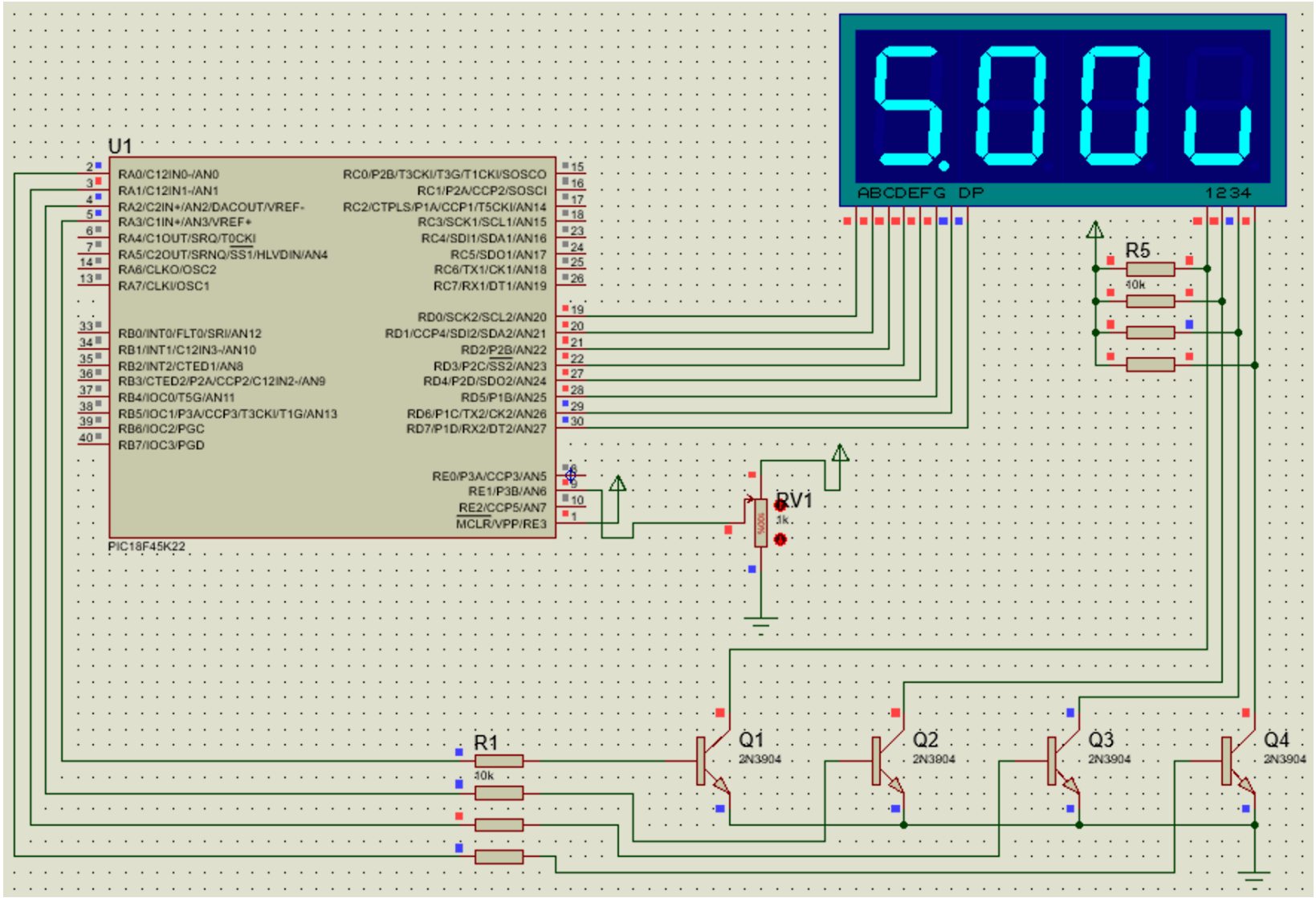
Number	DP	g	f	e	d	c	b	a	Hex
0	0	0	1	1	1	1	1	1	0x3F
1	0	0	0	0	0	1	1	0	0x06
2	0	1	0	1	1	0	1	1	0x5B
3	0	1	0	0	1	1	1	1	0x4F
4	0	1	1	0	0	1	1	0	0x66
5	0	1	1	0	1	1	0	1	0x6D
6	0	1	1	1	1	1	0	1	0x7D
7	0	0	0	0	0	1	1	1	0x07
8	0	1	1	1	1	1	1	1	0x7F
9	0	1	1	0	1	1	1	1	0x6F
u	0	0	0	1	1	1	0	0	0x1C



https://embedjournal.com/interface-7-segment-display-with-microcontroller/

ADC Single Channel Voltmeter Example

```
#include <stdio.h>
#include <stdlib.h>
#include <pic18f45k22.h>
#include <htc.h>

#define _XTAL_FREQ 8000000

#pragma config FOSC = HSHP
#pragma config WDTEN = OFF

unsigned char seg_char[11] =
{0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F,0
x1C};
unsigned long value;
unsigned char digits[4]={0,0,0,0};
unsigned char digit_counter = 0;

void main(void) {
   ANSELA = 0x00;
   TRISA = 0xF0;
   PORTA = 0x00;

   ANSELD = 0x00;
   TRISD = 0x00;
   PORTD = 0x00;

   ANSELE = 0x02;
   TRISE = 0X07;

   ADCON2 = 0xA1;
   ADCON1 = 0x00;
   ADCON0 = 0x1B;

   T0CON = 0xC3;
   TMR0L = 6;

   TMR0IE = 1;
   TMR0IP = 1;
   GIE = 1;

   while(1){
      if(!GODONE){
         value =
((long)5000*(ADRESH*256+ADRESL))/1023;
         GODONE = 1;
      }
      digits[3] = value / 1000;
      digits[2] = (value % 1000) / 100;
      digits[1] = (value % 100) / 10;
      digits[0] = value % 10;
   }
   return;
}

void __interrupt(high_priority) isr() {
   if (TMR0IF) {
      TMR0IF = 0;
      TMR0L = 6;

      PORTA = 0x00;
      if (digit_counter == 0){
         PORTD = seg_char[10];
      }else{
         PORTD =
seg_char[digits[digit_counter]]+0x80*(digit_counter==
3);
      }
      PORTA = 0x01 << digit_counter;

      digit_counter++;
      if (digit_counter >= 4) {
         digit_counter = 0;
      }
   }
}
```


5.00v
ABCDEFG DP
1234
U1
RA0/C12IN0-/AN0
RA1/C12IN1-/AN1
RA2/C2IN+/AN2/DACOUT/VREF-
RA3/C1IN+/AN3/VREF+
RA4/C1OUT/SRQ/T0CKI
RA5/C2OUT/SRNQ/SS1/HLVDIN/AN4
RA6/CLKO/OSC2
RA7/CLKI/OSC1
RB0/INT0/FLT0/SRI/AN12
RB1/INT1/C12IN3-/AN10
RB2/INT2/CTED1/AN8
RB3/CTED2/P2A/CCP2/C12IN2-/AN9
RB4/IOC0/T5G/AN11
RB5/IOC1/P3A/CCP3/T3CKI/T1G/AN13
RB6/IOC2/PGC
RB7/IOC3/PGD
RC0/P2B/T3CKI/T3G/T1CKI/SOSCO
RC1/P2A/CCP2/SOSCI
RC2/CTPLS/P1A/CCP1/T5CKI/AN14
RC3/SCK1/SCL1/AN15
RC4/SDI1/SDA1/AN16
RC5/SDO1/AN17
RC6/TX1/CK1/AN18
RC7/RX1/DT1/AN19
RD0/SCK2/SCL2/AN20
RD1/CCP4/SDI2/SDA2/AN21
RD2/P2B/AN22
RD3/P2C/SS2/AN23
RD4/P2D/SDO2/AN24
RD5/P1B/AN25
RD6/P1C/TX2/CK2/AN26
RD7/P1D/RX2/DT2/AN27
RE0/P3A/CCP3/AN5
RE1/P3B/AN6
RE2/CCP5/AN7
MCLR/VPP/RE3
PIC18F45K22
RV1
1k
R5
10k
R1
10k
Q1
2N3904
Q2
2N3904
Q3
2N3904
Q4
2N3904

ADC Interrupt

The ADC module allows for the ability to generate an interrupt upon completion of an Analog-to-Digital Conversion. The ADC interrupt enable is the ADIE bit in the PIE1 register and the interrupt priority is the ADIP bit in the IPR1 register. The ADC interrupt flag is the ADIF bit in the PIR1 register. The ADIF bit must be cleared by software.

PIE1 Register

REGISTER 9-9: PIE1: PERIPHERAL INTERRUPT ENABLE (FLAG) REGISTER 1

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **Unimplemented:** Read as '0'.

bit 6 **ADIE:** A/D Converter Interrupt Enable bit
1 = Enables the A/D interrupt
0 = Disables the A/D interrupt

bit 5 **RC1IE:** EUSART1 Receive Interrupt Enable bit
1 = Enables the EUSART1 receive interrupt
0 = Disables the EUSART1 receive interrupt

bit 4 **TX1IE:** EUSART1 Transmit Interrupt Enable bit
1 = Enables the EUSART1 transmit interrupt
0 = Disables the EUSART1 transmit interrupt

bit 3 **SSP1IE:** Host Synchronous Serial Port 1 Interrupt Enable bit
1 = Enables the MSSP1 interrupt
0 = Disables the MSSP1 interrupt

bit 2 **CCP1IE:** CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

bit 1 **TMR2IE:** TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit 0 **TMR1IE:** TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

IPR1 Register

REGISTER 9-14: IPR1: PERIPHERAL INTERRUPT PRIORITY REGISTER 1

U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **Unimplemented:** Read as '0'

bit 6 **ADIP:** A/D Converter Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **RC1IP:** EUSART1 Receive Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **TX1IP:** EUSART1 Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **SSP1IP:** Host Synchronous Serial Port 1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **CCP1IP:** CCP1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **TMR2IP:** TMR2 to PR2 Match Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **TMR1IP:** TMR1 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

PIR1 Register

REGISTER 9-4: PIR1: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 1

U-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
—	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **Unimplemented:** Read as '0'.

bit 6 **ADIF:** A/D Converter Interrupt Flag bit
1 = An A/D conversion completed (must be cleared by software)
0 = The A/D conversion is not complete or has not been started

bit 5 **RC1IF:** EUSART1 Receive Interrupt Flag bit
1 = The EUSART1 receive buffer, RCREG1, is full (cleared when RCREG1 is read)
0 = The EUSART1 receive buffer is empty

bit 4 **TX1IF:** EUSART1 Transmit Interrupt Flag bit
1 = The EUSART1 transmit buffer, TXREG1, is empty (cleared when TXREG1 is written)
0 = The EUSART1 transmit buffer is full

bit 3 **SSP1IF:** Host Synchronous Serial Port 1 Interrupt Flag bit
1 = The transmission/reception is complete (must be cleared by software)
0 = Waiting to transmit/receive

PIR1 Register

bit 2 **CCP1IF:** CCP1 Interrupt Flag bit

Capture mode:
1 = A TMR register capture occurred (must be cleared by software)
0 = No TMR register capture occurred

Compare mode:
1 = A TMR register compare match occurred (must be cleared by software)
0 = No TMR register compare match occurred

PWM mode:
Unused in this mode

bit 1 **TMR2IF:** TMR2 to PR2 Match Interrupt Flag bit

1 = TMR2 to PR2 match occurred (must be cleared by software)
0 = No TMR2 to PR2 match occurred

bit 0 **TMR1IF:** TMR1 Overflow Interrupt Flag bit

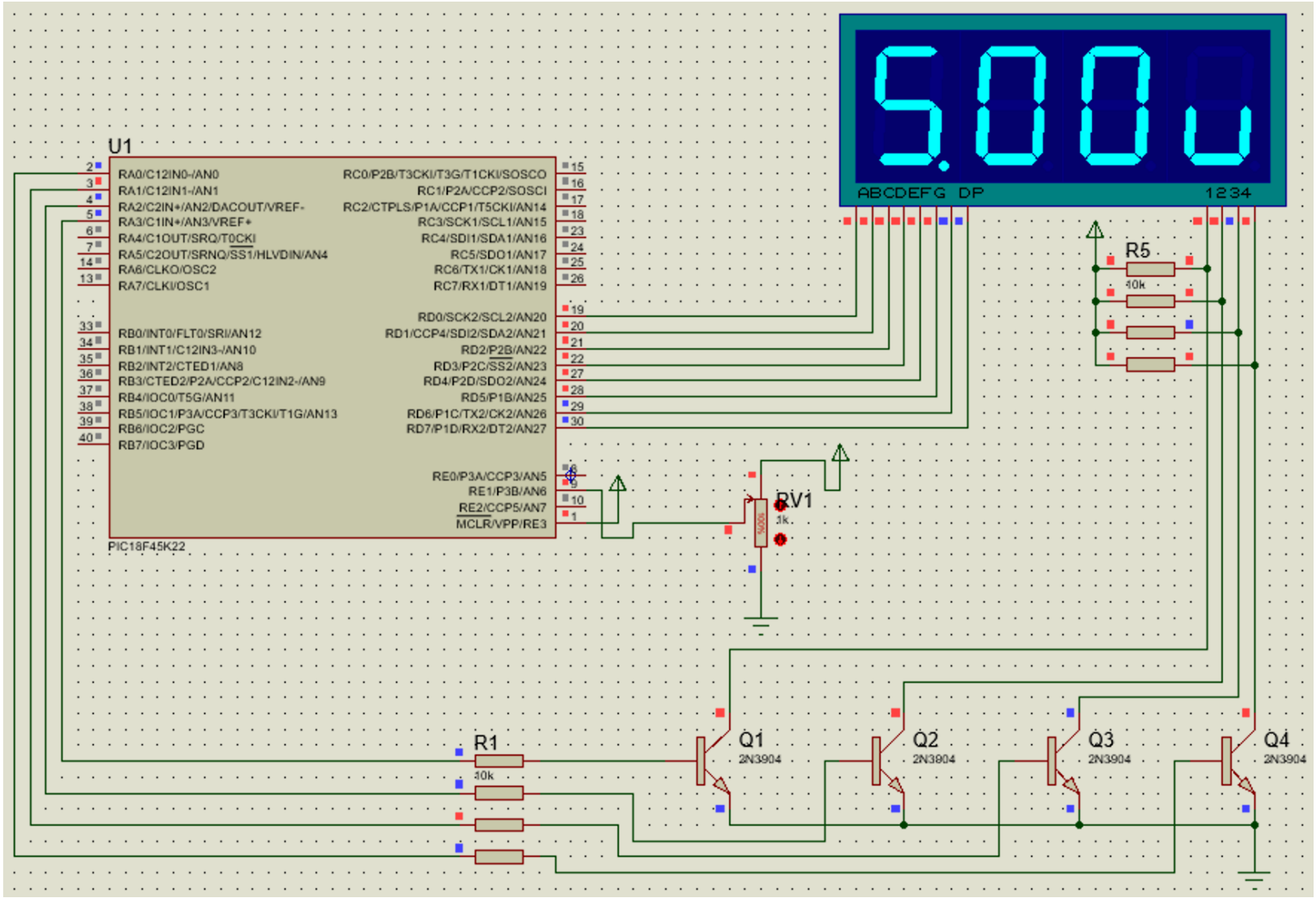
1 = TMR1 register overflowed (must be cleared by software)
0 = TMR1 register did not overflow

Note 1: Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the Global Interrupt Enable bit, GIE/GIEH of the INTCON register.

Note: User software may ensure the appropriate interrupt flag bits are cleared prior to enabling an interrupt and after servicing that interrupt.

ADC Single Channel Voltmeter Example With ADC Interrupt

```
#include <stdio.h>
#include <stdlib.h>
#include <pic18f45k22.h>
#include <htc.h>

#define _XTAL_FREQ 8000000

#pragma config FOSC = HSHP
#pragma config WDTEN = OFF

unsigned char seg_char[11] =
{0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F,0x1C};
unsigned long value;
unsigned char digits[4]={0,0,0,0};
unsigned char digit_counter = 0;
unsigned char conversion_done = 0;

void main(void) {
   ANSELA = 0x00;
   TRISA = 0xF0;
   PORTA = 0x00;

   ANSELD = 0x00;
   TRISD = 0x00;
   PORTD = 0x00;

   ANSELE = 0x02;
   TRISE = 0X07;

   ADCON2 = 0xA1;
   ADCON1 = 0x00;
   ADCON0 = 0x1B;

   T0CON = 0xC3;
   TMR0L = 6;

   PIE1bits.ADIE = 1;
   IPR1bits.ADIP = 1;

   TMR0IE = 1;
   TMR0IP = 1;
   GIE = 1;

   while(1){
      //if(!GODONE){
      //    value = ((long)5000*(ADRESH*256+ADRESL))/1023;
      //    GODONE = 1;
      //}

      if (conversion_done == 1) {
         conversion_done = 0;
         digits[3] = value / 1000;
         digits[2] = (value % 1000) / 100;
         digits[1] = (value % 100) / 10;
         digits[0] = value % 10;
         GODONE = 1;
      }
   }
   return;
}

void __interrupt(high_priority) isr() {
   if (TMR0IF) {
      TMR0IF = 0;
      TMR0L = 6;

      PORTA = 0x00;
      if (digit_counter == 0){
         PORTD = seg_char[10];
      }else{
         PORTD =
seg_char[digits[digit_counter]]+0x80*(digit_counter==3);
      }
      PORTA = 0x01 << digit_counter;

      digit_counter++;
      if (digit_counter >= 4) {
         digit_counter = 0;
      }
   }
   if (ADIF) {
      ADIF = 0;
      value = ((long) 5000 * (ADRESH * 256 + ADRESL)) / 1023;
      conversion_done = 1;
   }
}
```

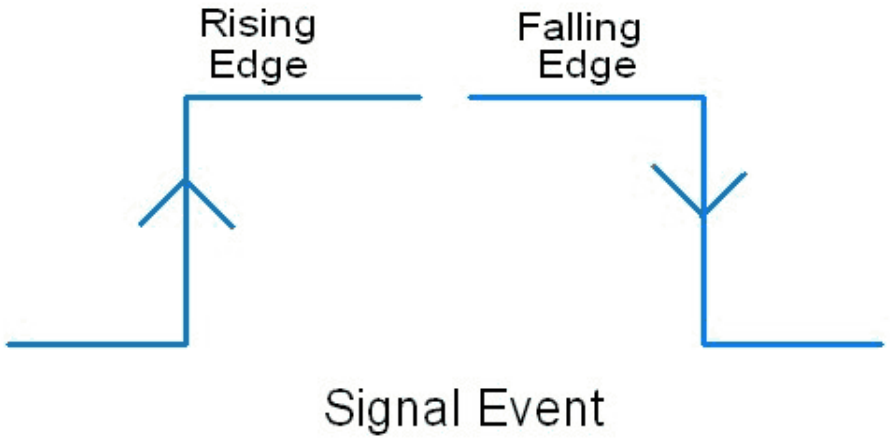

5.00v
ABCDEFG DP
1234
U1
RA0/C12IN0-/AN0
RA1/C12IN1-/AN1
RA2/C2IN+/AN2/DACOUT/VREF-
RA3/C1IN+/AN3/VREF+
RA4/C1OUT/SRQ/T0CKI
RA5/C2OUT/SRNQ/SS1/HLVDIN/AN4
RA6/CLKO/OSC2
RA7/CLKI/OSC1
RB0/INT0/FLT0/SRI/AN12
RB1/INT1/C12IN3-/AN10
RB2/INT2/CTED1/AN8
RB3/CTED2/P2A/CCP2/C12IN2-/AN9
RB4/IOC0/T5G/AN11
RB5/IOC1/P3A/CCP3/T3CKI/T1G/AN13
RB6/IOC2/PGC
RB7/IOC3/PGD
RC0/P2B/T3CKI/T3G/T1CKI/SOSCO
RC1/P2A/CCP2/SOSCI
RC2/CTPLS/P1A/CCP1/T5CKI/AN14
RC3/SCK1/SCL1/AN15
RC4/SDI1/SDA1/AN16
RC5/SDO1/AN17
RC6/TX1/CK1/AN18
RC7/RX1/DT1/AN19
RD0/SCK2/SCL2/AN20
RD1/CCP4/SDI2/SDA2/AN21
RD2/P2B/AN22
RD3/P2C/SS2/AN23
RD4/P2D/SDO2/AN24
RD5/P1B/AN25
RD6/P1C/TX2/CK2/AN26
RD7/P1D/RX2/DT2/AN27
RE0/P3A/CCP3/AN5
RE1/P3B/AN6
RE2/CCP5/AN7
MCLR/VPP/RE3
PIC18F45K22
RV1
1k
R5
10k
R1
10k
Q1
2N3904
Q2
2N3904
Q3
2N3904
Q4
2N3904

INTn Pin Interrupts

External interrupts on the RB0/INT0, RB1/INT1 and RB2/INT2 pins are edge-triggered. If the corresponding INTEDGx bit in the INTCON2 register is set (= 1), the interrupt is triggered by a rising edge; if the bit is clear, the trigger is on the falling edge. When a valid edge appears on the RBx/INTx pin, the corresponding flag bit, INTxF, is set. This interrupt can be disabled by clearing the corresponding enable bit, INTxE. Flag bit, INTxF, must be cleared by software in the Interrupt Service Routine before re-enabling the interrupt.

All external interrupts (INT0, INT1 and INT2) can wake-up the processor from Idle or Sleep modes if bit INTxE was set prior to going into those modes. If the Global Interrupt Enable bit, GIE/GIEH, is set, the processor will branch to the interrupt vector following wake-up.

Interrupt priority for INT1 and INT2 is determined by the value contained in the interrupt priority bits, INT1IP and INT2IP of the INTCON3 register. There is no priority bit associated with INT0. It is always a high priority interrupt source.



INTCON2

REGISTER 9-2: INTCON2: INTERRUPT CONTROL 2 REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	U-0	R/W-1	U-0	R/W-1
$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **$\overline{\text{RBPU}}$:** PORTB Pull-up Enable bit
1 = All PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled provided that the pin is an input and the corresponding WPUB bit is set.

bit 6 **INTEDG0:** External Interrupt 0 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 5 **INTEDG1:** External Interrupt 1 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 4 **INTEDG2:** External Interrupt 2 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 3 **Unimplemented:** Read as '0'

bit 2 **TMR0IP:** TMR0 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **Unimplemented:** Read as '0'

bit 0 **RBIP:** RB Port Change Interrupt Priority bit
1 = High priority
0 = Low priority

Note: Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global enable bit. User software may ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows for software polling.

INTCON

REGISTER 9-1: INTCON: INTERRUPT CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **GIE/GIEH:** Global Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked interrupts
0 = Disables all interrupts including peripherals
When IPEN = 1:
1 = Enables all high priority interrupts
0 = Disables all interrupts including low priority

bit 6 **PEIE/GIEL:** Peripheral Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts
When IPEN = 1:
1 = Enables all low priority interrupts
0 = Disables all low priority interrupts

bit 5 **TMR0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 overflow interrupt
0 = Disables the TMR0 overflow interrupt

bit 4 **INT0IE:** INT0 External Interrupt Enable bit
1 = Enables the INT0 external interrupt
0 = Disables the INT0 external interrupt

bit 3 **RBIE:** Port B Interrupt-On-Change (IOCx) Interrupt Enable bit(2)
1 = Enables the IOCx port change interrupt
0 = Disables the IOCx port change interrupt

bit 2 **TMR0IF:** TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared by software)
0 = TMR0 register did not overflow

bit 1 **INT0IF:** INT0 External Interrupt Flag bit
1 = The INT0 external interrupt occurred (must be cleared by software)
0 = The INT0 external interrupt did not occur

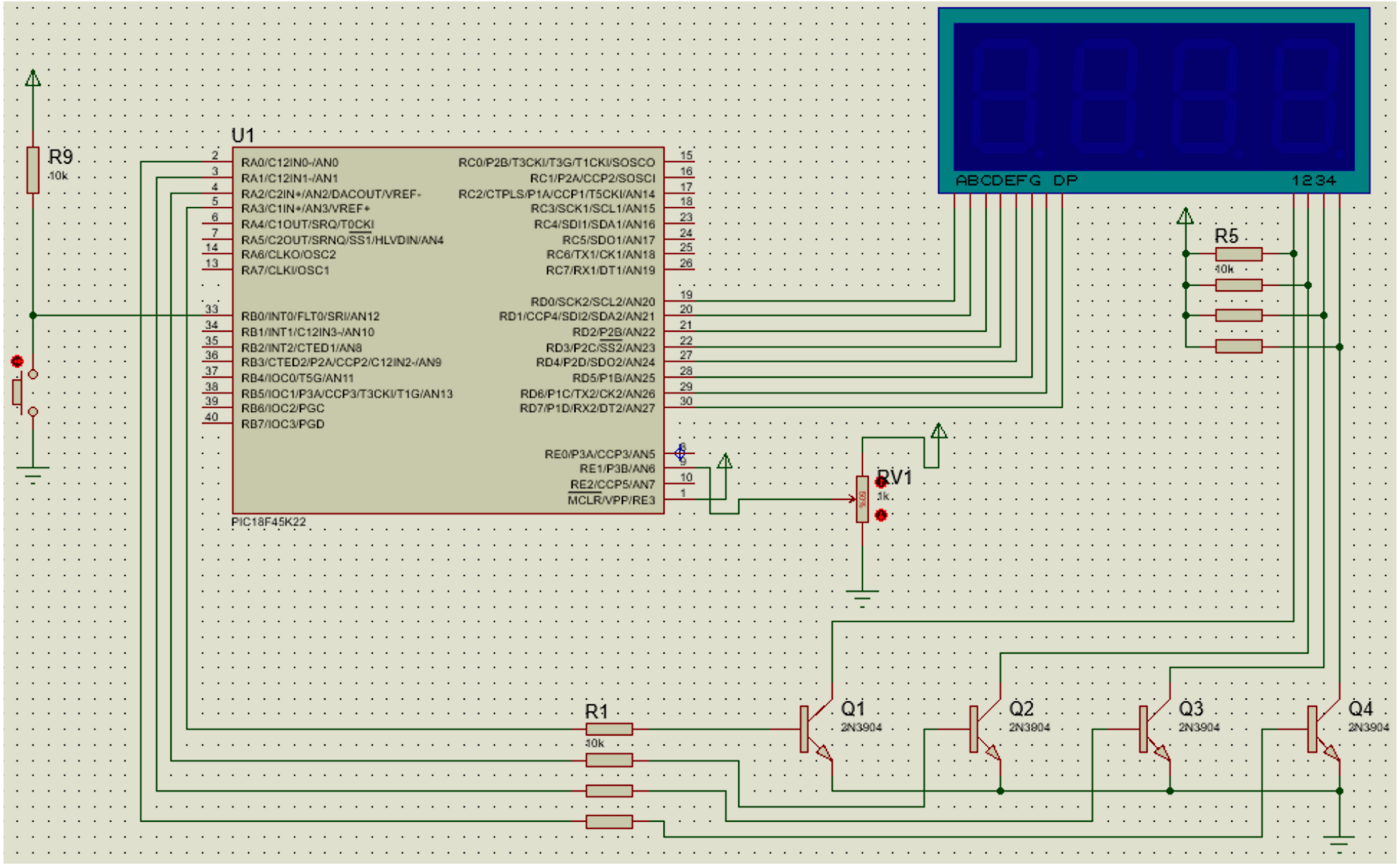
bit 0 **RBIF:** Port B Interrupt-On-Change (IOCx) Interrupt Flag bit(1)
1 = At least one of the IOC<3:0> (RB<7:4>) pins changed state (must be cleared by software)
0 = None of the IOC<3:0> (RB<7:4>) pins have changed state

Note 1: A mismatch condition will continue to set the RBIF bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared.
2: RB port change interrupts also require the individual pin IOCB enables.

Note: Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global enable bit. User software may ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows for software polling.

ADC Single Channel Voltmeter Example With Interrupts

```
#include <stdio.h>
#include <stdlib.h>
#include <pic18f45k22.h>
#include <htc.h>

#define _XTAL_FREQ 8000000

#pragma config FOSC = HSHP
#pragma config WDTEN = OFF

unsigned char seg_char[11] =
{0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F,0x1C};
unsigned long value;
unsigned char digits[4]={0,0,0,0};
unsigned char digit_counter = 0;
unsigned char conversion_done = 0;
unsigned char power = 1;

void main(void) {
  ANSELA = 0x00;
  TRISA = 0xF0;
  PORTA = 0x00;

  ANSELB = 0x00;
  TRISB = 0xFF;

  ANSELD = 0x00;
  TRISD = 0x00;
  PORTD = 0x00;

  ANSELE = 0x02;
  TRISE = 0X07;

  ADCON2 = 0xA1;
  ADCON1 = 0x00;
  ADCON0 = 0x1B;

  T0CON = 0xC3;
  TMR0L = 6;

  PIE1bits.ADIE = 1;
  IPR1bits.ADIP = 1;

  INTCON2bits.INTEDG0 = 0;
  INTCONbits.INT0IE = 1;

  TMR0IE = 1;
  TMR0IP = 1;
  GIE = 1;

  while(1){
    //if(!GODONE){
    //    value = ((long)5000*(ADRESH*256+ADRESL))/1023;
    //    GODONE = 1;
    //}

    if (conversion_done == 1) {
      conversion_done = 0;
      digits[3] = value / 1000;
      digits[2] = (value % 1000) / 100;
      digits[1] = (value % 100) / 10;
      digits[0] = value % 10;
      GODONE = 1;
    }
  }
  return;
}

void __interrupt(high_priority) isr() {
  if (TMR0IF) {
    TMR0IF = 0;
    TMR0L = 6;

    PORTA = 0x00;
    if (digit_counter == 0){
      PORTD = seg_char[10];
    }else{
      PORTD = seg_char[digits[digit_counter]]+0x80*(digit_counter==3);
    }
    PORTA = (0x01*power) << digit_counter;

    digit_counter++;
    if (digit_counter >= 4) {
      digit_counter = 0;
    }
  }
  if (ADIF) {
    ADIF = 0;
    value = ((long) 5000 * (ADRESH * 256 + ADRESL)) / 1023;
    conversion_done = 1;
  }
  if (INT0IF) {
    INT0IF = 0;
    if (power == 1){
      power = 0;
    }else{
      power = 1;
    }
  }
}
```


U1
PIC18F45K22
RA0/C12IN0-/AN0
RA1/C12IN1-/AN1
RA2/C2IN+/AN2/DACOUT/VREF-
RA3/C1IN+/AN3/VREF+
RA4/C1OUT/SRQ/T0CKI
RA5/C2OUT/SRNQ/SS1/HLVDIN/AN4
RA6/CLKO/OSC2
RA7/CLKI/OSC1
RB0/INT0/FLT0/SRI/AN12
RB1/INT1/C12IN3-/AN10
RB2/INT2/CTED1/AN8
RB3/CTED2/P2A/CCP2/C12IN2-/AN9
RB4/IOC0/T5G/AN11
RB5/IOC1/P3A/CCP3/T3CKI/T1G/AN13
RB6/IOC2/PGC
RB7/IOC3/PGD
RC0/P2B/T3CKI/T3G/T1CKI/SOSCO
RC1/P2A/CCP2/SOSCI
RC2/CTPLS/P1A/CCP1/T5CKI/AN14
RC3/SCK1/SCL1/AN15
RC4/SDI1/SDA1/AN16
RC5/SDO1/AN17
RC6/TX1/CK1/AN18
RC7/RX1/DT1/AN19
RD0/SCK2/SCL2/AN20
RD1/CCP4/SDI2/SDA2/AN21
RD2/P2B/AN22
RD3/P2C/SS2/AN23
RD4/P2D/SDO2/AN24
RD5/P1B/AN25
RD6/P1C/TX2/CK2/AN26
RD7/P1D/RX2/DT2/AN27
RE0/P3A/CCP3/AN5
RE1/P3B/AN6
RE2/CCP5/AN7
MCLR/VPP/RE3
ABCDEFG DP
1234
R9
10k
R5
10k
RV1
1k
R1
10k
Q1
2N3904
Q2
2N3904
Q3
2N3904
Q4
2N3904

Register Definitions: ADC Control

ADCON0 = 0xxx xx11 for different channels

REGISTER 17-1: ADCON0: A/D CONTROL REGISTER 0

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	CHS<4:0>					GO/$\overline{\text{DONE}}$	ADON
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **Unimplemented:** Read as '0'

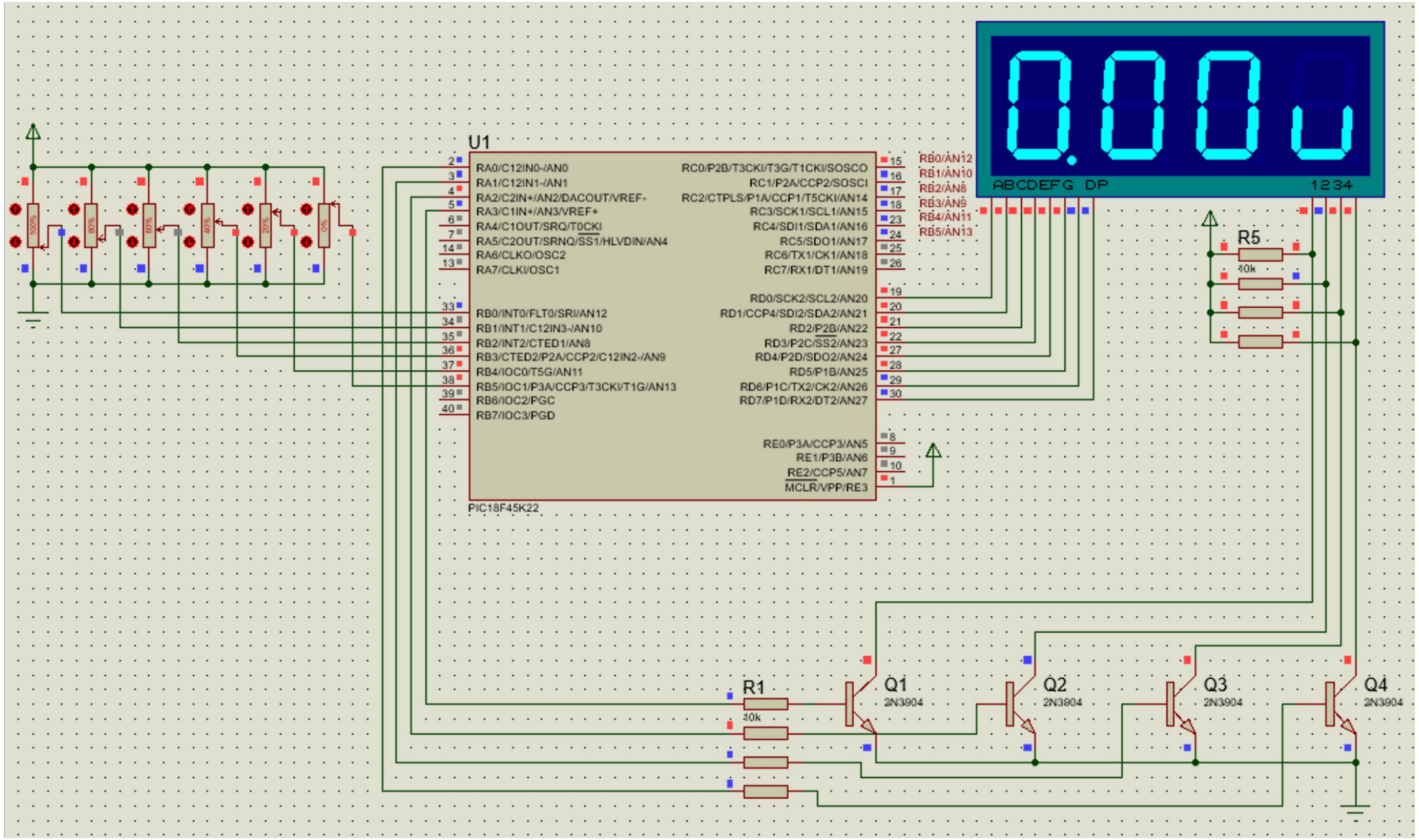
bit 6-2 **CHS<4:0>: Analog Channel Select bits**
00000 = AN0
00001 = AN1
00010 = AN2
00011 = AN3
00100 = AN4
00101 = AN5(1)
00110 = AN6(1)
00111 = AN7(1)
01000 = AN8
01001 = AN9
01010 = AN10
01011 = AN11
01100 = AN12
01101 = AN13

bit 1 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit
1 = A/D conversion cycle in progress. Setting this bit starts an A/D conversion cycle.
This bit is automatically cleared by hardware when the A/D conversion has completed.
0 = A/D conversion completed/not in progress

bit 0 **ADON:** ADC Enable bit
1 = ADC is enabled
0 = ADC is disabled and consumes no operating current

ADC Multi Channel Voltmeter Example

```
#include <stdio.h>
#include <stdlib.h>
#include <pic18f45k22.h>
#include <htc.h>

#define _XTAL_FREQ 8000000

#pragma config FOSC = HSHP
#pragma config WDTEN = OFF

unsigned char seg_char[11] =
{0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F,0x1C};
unsigned long value[6] = {0,0,0,0,0,0};
unsigned char digits[4] = {0,0,0,0};
unsigned char digit_counter = 0;
unsigned char an_cnt = 0;
unsigned char an_list[6] = {12,10,8,9,11,13};
unsigned char disp_cnt = 0;
unsigned char disp_num = 0;

void main(void) {
  ANSELA = 0x00;
  TRISA = 0xF0;
  PORTA = 0x00;

  ANSELB = 0x3F;
  TRISB = 0xFF;

  ANSELC = 0x00;
  TRISC = 0xC0;
  PORTC = 0x00;

  ANSELD = 0x00;
  TRISD = 0x00;
  PORTD = 0x00;

  ADCON2 = 0xA1;
  ADCON1 = 0x00;

  //ADCON0 = 0x1B;
  ADCON0bits.CHS = an_list[0];
  ADCON0bits.ADON = 1;
  GODONE = 1;

  T0CON = 0xC3;
  TMR0L = 6;

  TMR0IE = 1;
  TMR0IP = 1;
  GIE = 1;

  while(1){
    if(!GODONE){
      value[an_cnt] = ((long)5000*(ADRESH*256+ADRESL))/1023;
      an_cnt++;
      if(an_cnt >= 6){
        an_cnt=0;
      }
      ADCON0bits.CHS = an_list[an_cnt];
      GODONE = 1;
    }
    digits[3] = value[disp_num] / 1000;
    digits[2] = (value[disp_num] % 1000) / 100;
    digits[1] = (value[disp_num] % 100) / 10;
    digits[0] = value[disp_num] % 10;
  }
  return;
}

void __interrupt(high_priority) isr() {
  if (TMR0IF) {
    TMR0IF = 0;
    TMR0L = 6;

    PORTA = 0x00;
    if(digit_counter == 0){
      PORTD = seg_char[10];
    }else{
      PORTD = seg_char[digits[digit_counter]]+0x80*(digit_counter==3);
    }
    PORTA = 0x01 << digit_counter;

    digit_counter++;
    if (digit_counter >= 4) {
      digit_counter = 0;
      disp_cnt++;
      if(disp_cnt >= 125){
        disp_cnt = 0;
        disp_num++;
        if(disp_num >= 6){
          disp_num = 0;
        }
        PORTC = 0x01<<disp_num;
      }
    }
  }
}
```


U1
PIC18F45K22
RA0/C12IN0-/AN0
RA1/C12IN1-/AN1
RA2/C2IN+/AN2/DACOUT/VREF-
RA3/C1IN+/AN3/VREF+
RA4/C1OUT/SRQ/T0CKI
RA5/C2OUT/SRNQ/SS1/HLVDIN/AN4
RA6/CLKO/OSC2
RA7/CLKI/OSC1
RB0/INT0/FLT0/SRI/AN12
RB1/INT1/C12IN3-/AN10
RB2/INT2/CTED1/AN8
RB3/CTED2/P2A/CCP2/C12IN2-/AN9
RB4/IOC0/T5G/AN11
RB5/IOC1/P3A/CCP3/T3CKI/T1G/AN13
RB6/IOC2/PGC
RB7/IOC3/PGD
RC0/P2B/T3CKI/T3G/T1CKI/SOSCO
RC1/P2A/CCP2/SOSCI
RC2/CTPLS/P1A/CCP1/T5CKI/AN14
RC3/SCK1/SCL1/AN15
RC4/SDI1/SDA1/AN16
RC5/SDO1/AN17
RC6/TX1/CK1/AN18
RC7/RX1/DT1/AN19
RD0/SCK2/SCL2/AN20
RD1/CCP4/SDI2/SDA2/AN21
RD2/P2B/AN22
RD3/P2C/SS2/AN23
RD4/P2D/SDO2/AN24
RD5/P1B/AN25
RD6/P1C/TX2/CK2/AN26
RD7/P1D/RX2/DT2/AN27
RE0/P3A/CCP3/AN5
RE1/P3B/AN6
RE2/CCP5/AN7
MCLR/VPP/RE3
RB0/AN12
RB1/AN10
RB2/AN8
RB3/AN9
RB4/AN11
RB5/AN13
ABCDEFG DP
1234
R5
10k
R1
10k
Q1
2N3904
Q2
2N3904
Q3
2N3904
Q4
2N3904