```c
volatile char CONVERT=0;

unsigned char character[8] = {
  0b00000,
  0b00000,
  0b01010,
  0b11111,
  0b11111,
  0b01110,
  0b00100,
  0b00000,
};

void __interrupt(high_priority) FNC()
{
    if(INTCONbits.INT0IF)
    {
        //LCDStr("INT");
        CONVERT = 1;
        INTCONbits.INT0IF = 0;
    }
}

void main(void) {
  initADC();
  InitLCD();
  LCDStr("Hellooo ");

  LCDAddSpecialCharacter(0, character);
  LCDGoto(8, 1);
  LCDDat(0);

  char values[10] = {0};

  unsigned short convertion = 0;

  TRISBbits.RB0 = 1;
  INTCONbits.INT0IE = 1; //Enable INT0 pin interrupt
  INTCONbits.INT0IF = 0;
  INTCONbits.GIE = 1;
  INTCONbits.PEIE = 1;


  while(1)
  {
      if(CONVERT == 1)
      {
          convertion = readADCChannel(0);
          sprintf(values, "%d", convertion);
          LCDCmd(LCD_CLEAR);
          LCDGoto(5, 2);
          LCDStr(values);
          CONVERT = 0;
      }

  }


  return;
}
```

ADCON1

REGISTER 19-2: ADCON1: A/D CONTROL REGISTER 1

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
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 7-6 **Unimplemented:** Read as '0'

bit 5 **VCFG1:** Voltage Reference Configuration bit (VREF- source)
1 = VREF- (AN2)
0 = VSS

bit 4 **VCFG0:** Voltage Reference Configuration bit (VREF+ source)
1 = VREF+ (AN3)
0 = VDD

bit 3-0 **PCFG3:PCFG0:** A/D Port Configuration Control bits:

PCFG3:PCFG0	AN12	AN11	AN10	AN9	AN8	AN7(1)	AN6(1)	AN5	AN4	AN3	AN2	AN1	AN0
0000(2)	A	A	A	A	A	A	A	A	A	A	A	A	A
0001	A	A	A	A	A	A	A	A	A	A	A	A	A
0010	A	A	A	A	A	A	A	A	A	A	A	A	A
0011	D	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	D	A	A	A	A	A	A	A	A	A
0111(3)	D	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D	D

A = Analog Input — D = Digital I/O

Note 1: The POR value of the PCFG bits depends on the value of the PBADEN Configuration bit. When PBADEN = 1, PCFG=2:0> = 000; when PBADEN = 0, PCFG=2:0> = 111.
2: ANS through AN7 are available only on 40/44-pin devices.

ADCON2

REGISTER 19-3: ADCON2: A/D CONTROL REGISTER 2

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
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 **ADFM:** A/D Result Format Select bit
1 = Right justified
0 = Left justified

bit 6 **Unimplemented:** Read as '0'

bit 5-3 **ACQT2:ACQT0:** A/D Acquisition Time Select bits
111 = 20 TAD
110 = 16 TAD
101 = 12 TAD
100 = 8 TAD
011 = 6 TAD
010 = 4 TAD
001 = 2 TAD
000 = 0 TAD(1)

bit 2-0 **ADCS2:ADCS0:** A/D Conversion Clock Select bits
111 = FRC (clock derived from A/D RC oscillator)(1)
110 = FOSC/64
101 = FOSC/16
100 = FOSC/4
011 = FRC (clock derived from A/D RC oscillator)(1)
010 = FOSC/32
001 = FOSC/8
000 = FOSC/2

Note 1: If the A/D FRC clock source is selected, a delay of one TCY (instruction cycle) is added before the A/D clock starts. This allows the SLEEP instruction to be executed before starting a conversion.

The seven possible options for TAD are: $2T_{OSC}$, $4T_{OSC}$, $8T_{OSC}$, $16T_{OSC}$, $32T_{OSC}$, $64T_{OSC}$

TABLE 19-1: Tad vs. DEVICE OPERATING FREQUENCIES

A/D Clock Source (TAD)		Maximum Device Frequency	
Operation	ADCS2:ADCS0	PIC18FXXXX/44XX	PIC18LFXXXX/44XX(4)
2 TOSC	000	2.86 MHz	1.43 MHz
4 TOSC	100	5.71 MHz	2.86 MHz
8 TOSC	001	11.43 MHz	5.71 MHz
16 TOSC	101	22.86 MHz	11.43 MHz
32 TOSC	010	40.0 MHz	22.86 MHz
64 TOSC	110	40.0 MHz	22.86 MHz
RC(3)	011	1.00 MHz(2)	1.00 MHz(2)

Note 1: The RC source has a typical TAD time of 1.2 µs.
2: The RC source has a typical TAD time of 2.5 µs.
3: For device frequencies above 1 MHz, the device must be in Sleep for the entire conversion or the A/D accuracy may be out of specification.
4: Low-power (PIC18LFXXXX) devices only.

see parameter 136, Table 26-28, page 360 for more information

For correct A/D conversions, the A/D conversion clock (T_{AD}) must be as short as possible (e.g. hold capacitor may discharge more), but greater than the minimum TAD (min 0.7 µs, max 25.0 µs (TOSC based, V REF ≥ 3.0V)).

ADCON0

REGISTER 19-1: ADCON0: A/D CONTROL REGISTER 0

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CHS3	CHS2	CHS1	CHS0	GO/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 7-6 **Unimplemented:** Read as '0'

bit 5-2 **CHS3:CHS0:** Analog Channel Select bits
0000 = Channel 0 (AN0)
0001 = Channel 1 (AN1)
0010 = Channel 2 (AN2)
0011 = Channel 3 (AN3)
0100 = Channel 4 (AN4)
0101 = Channel 5 (AN5)(1,2)
0110 = Channel 6 (AN6)(1,2)
0111 = Channel 7 (AN7)(1,2)
1000 = Channel 8 (AN8)
1001 = Channel 9 (AN9)
1010 = Channel 10 (AN10)
1011 = Channel 11 (AN11)
1100 = Channel 12 (AN12)
1101 = Unimplemented(2)
1110 = Unimplemented(2)
1111 = Unimplemented(2)

bit 1 **GO/DONE:** A/D Conversion Status bit
When ADON = 1:
1 = A/D conversion in progress
0 = A/D idle

bit 0 **ADON:** A/D On bit
1 = A/D Converter module is enabled
0 = A/D Converter module is disabled

Note 1: These channels are not implemented on 28-pin devices.
2: Performing a conversion on unimplemented channels will return a floating input measurement.

```c
void initADC()
{
    ADCON1bits.PCFG3 = 1; // RA0 = Analog, RA1=1 Analog, RA2 = Analog
    ADCON1bits.PCFG2 = 1;
    ADCON1bits.PCFG1 = 0;
    ADCON1bits.PCFG0 = 0;

    ADCON1bits.VCFG0 = 0; // Vref+=5.0, Vref-=0
    ADCON1bits.VCFG1 = 0;

    TRISAbits.RA0 = 1;
    TRISAbits.RA1 = 1;
    TRISAbits.RA2 = 0;

    // For 40MHZ -> Tosc = 1/40 us
    // Tad options (2xTosc, 4xTosc, 8xTosc, 16xTosc, 32xTosc, 64xTosc)
    // min Tad 0.7 us -  max Tad 25 us (keep as short as possible)
    // The closest one to min Tad (32xTosc) hence Tad = 32xTosc = 0.8 us (ADCS2:ADCS0=010)
    // Acquisition time options (0xTad, 2xTad, 4xTad, 6xTad, 8xTad, 12xTad, 16xTad, 20xTad)
    // Min acquisition time = 2.4 us (the closest acquisition time we can set 4xTad = 3.2us)
    //(ACQT2:ACQT0=010)

    ADCON2bits.ADCS2 = 0; // Tad (32xTOSC) -> 0.8us
    ADCON2bits.ADCS1 = 1;
    ADCON2bits.ADCS0 = 0;

    ADCON2bits.ACQT2 = 0; // Acquisition time (4xTad) = 3.2 us
    ADCON2bits.ACQT1 = 1;
    ADCON2bits.ACQT0 = 0;

    ADCON2bits.ADFM = 1; // Right justified...

    ADCON0bits.ADON = 1; // ADC module is enabled....
}
```

which pins are analog

reference voltage

inputs

output

$T_{AD} = 0.8\mu s$

acquisition time: $4 \cdot T_{AD} = 3.2\mu s$

right justified

enable A/D

$T_{Analog\ to\ digital} = [0.7\mu s, 25.0\ \mu s]$

our oscillator period $= \frac{1}{40\ MHz} = \frac{1}{40}\ \mu s$

$32 \cdot \frac{1}{40}\ \mu s = 0.8\mu s$ → closest to minimum

Min Acquisition time = 2.4

$T_{AD} = 0.8\mu s$ x4 → minimum

$8 \cdot \frac{1}{10}\ \mu s$ → 0.8 TAD

```c
// Initializes the LCD
// First sets the TRIS as required then configures LCD
void InitLCD(void) {
  TRISEbits.RE1 = 0; → LCD enable
  TRISEbits.RE2 = 0; → LCD reset
  TRISD = 0;         → LCD output is PORTD

  LCD_EN = 0;
  LCD_RS = 0;
  LCDCmd(LCD_FS_2LINE);
  LCDCmd(LCD_OFF);
  LCDCmd(LCD_ON);
  LCDCmd(LCD_ENTRY_FORWARD);
  LCDCmd(LCD_CLEAR);
  LCDCmd(ROW1);
}

void LCDCmd(unsigned char cmd) {
  LCD_EN = 0;
  LCD_RS = 0;
  LCD_PORT = cmd;
  LCD_EN = 1;
  __delay_us(LCD_PULSE_TIME);
  LCD_EN = 0;
  __delay_us(LCD_PULSE_TIME);
}

// Prints the given null terminated string `str` at the current cursor position
// It auto-wraps the given string if it doesn't fit the display.
void LCDStr(const char* str) {
  for (unsigned char i = 0; str[i] != 0; i++) {
    LCDDat(str[i]);
    lcd_x ++;
    if(lcd_x == 17) {
      lcd_x = 1;
      lcd_y++;
      if (lcd_y == 5) {
        lcd_y = 1;
      }
      //LCDGoto(lcd_x, lcd_y);
    }
  }
}

void LCDDat(unsigned char dat) {
  LCD_EN = 0;
  LCD_RS = 1;
  LCD_PORT = dat;
  LCD_EN = 1;
  __delay_us(LCD_PULSE_TIME);
  LCD_EN = 0;
  __delay_us(LCD_PULSE_TIME);
  LCD_RS = 0;
}

// Stores the custom character provided in the CGRAM of LCD character_index:
// specifies the index of the character in CGRAM, [0, 7] max 8 characters can be
// designed (for 16x4 display) data: is a array of 8 bytes where each entry
// specifies one of the character (5x8 dot)
void LCDAddSpecialCharacter(byte character_index, byte * data) {
  //Each custom character occupies 8 byte location
  // First custom character address 0x40-0x47
  // Second custom character address 0x48-0x4f
  // And so on...

    LCDCmd(0b01000000+(character_index*8));
    for(byte i=0; i < 8; i++) {
        LCDDat(data[i]);
    }

    //LCDGoto(lcd_x, lcd_y);
}

//Sets the current display cursor of the LCD
// p_1 : the row at which the text will be displayed, a value from [1, 4]
// p_2 : the column  at which the text will be displayed, a value from [1, 16]
void LCDGoto(byte p_2, byte p_1) {
  if(p_1==1) {
    LCDCmd(ROW1+((p_2-1)%16));
  }
  else if (p_1==2){
    LCDCmd(ROW2+((p_2-1)%16));
  }
  else if (p_1==3){
    LCDCmd(ROW3+((p_2-1)%16));
  }
  else {
    LCDCmd(ROW4+((p_2-1)%16));
  }
  lcd_x = p_2;
  lcd_y = p_1;
}

unsigned short readADCChannel(unsigned char channel)
{
    // 0b 0101 -> 5th chanel
    ADCON0bits.CHS0 =  channel & 0x1; // Select channel..
    ADCON0bits.CHS1 = (channel >> 1) & 0x1;
    ADCON0bits.CHS2 = (channel >> 2) & 0x1;
    ADCON0bits.CHS3 = (channel >> 3) & 0x1;


    ADCON0bits.GODONE = 1; //Start convertion
    while(ADCON0bits.GODONE == 1); //Wait the conversion to finish


    PIR1bits.ADIF = 0; // Clear interrupt flag...

    return (unsigned short)((ADRESH << 8)+ADRESL);

}
```