Lab 5 – Analogue to digital converter and interfacing high power devices

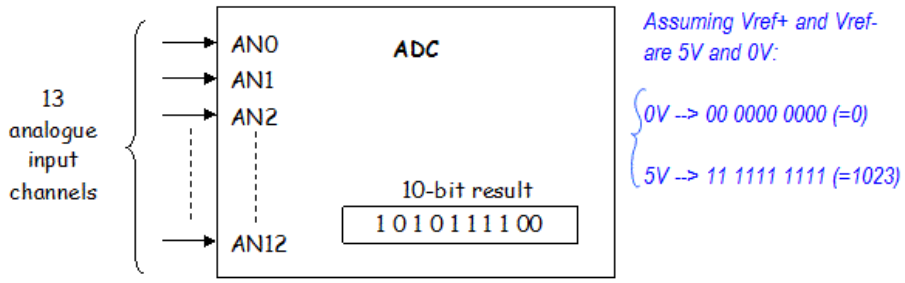
Objectives

- ☐ To learn to use the ADC (Analogue to Digital Converter) in the PIC18F4550 microcontroller.
- ☐ To learn to turn some high power devices (e.g. motor, relay) on and off.

Introduction / Briefing

PIC18F4550's ADC

- ☐ The PIC18F4550 microcontroller has a built-in ADC module capable of converting up to 13 analogue inputs to their corresponding 10-bit digital representations.



- ☐ The analogue input pins AN0 to AN12 could be connected to a variety of sensors for monitoring the environment.
- ☐ For instance, a LDR (Light Dependent Resistor) circuit can be used to sense the ambient brightness. The result is an analogue voltage, which can be converted to a digital equivalent, and then used by the PIC to make a decision – if it is too dark, switch on the lights. Other sensors include sensors for temperature, water level, humidity, PH value etc.
- ☐ In this experiment, a variable resistor is used to give a variable voltage input and the result of AD conversion will be shown on an L E D bar (- a low power output device). Later, you will work on a fish tank "water level monitoring" application – If the water level is too high, turn on the motor (- a high power output device) to pump away the excess water. If the water level is too low, turn on the alarm (- another high power output device) to alert the owner.

☐ First, the registers associated with PIC18F4550's ADC:

① ADCON0 – This register controls the **operation of the A/D module.**

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/$\overline{\text{DONE}}$	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 7-6 **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)

Select which input channel the ADC connects to.

bit 1 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit
When ADON = 1:
1 = A/D conversion in progress
0 = A/D Idle

Send a '1' to this bit starts the conversion.

While it is converting this bit remains as '1'.

This bit changes to '0' when the convesion has finished.

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

Turn this bit on if you need to use the ADC.

Q1: What binary pattern must be written to ADCON0 to select Channel 0 for conversion, and to power up the ADC module? (Don't start the conversion yet.)

Your answer: _____

☐ The C code required is ADCON0 = 0b00 0000 0 1;

☐ Note that to start the conversion, the GO bit must be set to 1. When conversion is finished, the same bit (read as DONE) will be set to 0.

(2)

ADCON1 - This register configures the **voltage references** and the **functions of the port pins.**

U-0	U-0	R/W-0	R/W-0	R/W-0(1)	R/W(1)	R/W(1)	R/W(1)
—	—	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 — 0V

bit 4 **VCFG0:** Voltage Reference Configuration bit (VREF+ source)
1 = VREF+ (AN3)
0 = VDD — 5V

(RA4 is a fixed digital input.)

Same pin as: RA5,RA3,RA2,RA1,RA0

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

PCFG3: PCFG0	AN12	AN11	AN10	AN9	AN8	AN7(2)	AN6(2)	AN5(2)	AN4	AN3	AN2	AN1	AN0
0000(1)	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(1)	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

Q2: What binary pattern must be written to ADCON1 to use VSS (0 Volt) and VDD (5 Volt) as reference voltages, and to make AN2 to AN0 analogue inputs (and the remaining inputs i.e. AN12 to AN3 digital)?

Your answer: *00 00 1100*

☐ The C code required is ADCON1 = 0b00 00 1100;

3

ADCON2 - This register configures the **A/D clock source**, **programmed acquisition time** and **justification**.

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

(See diagram next page)

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)

Must be long enough so that the input voltage can be sampled accurately enough.

TAD setup

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

For 20 MHz crystal:
Fosc = 20 x 2.4 = 48 MHz

How long should the TAD be?
- The shorter the better (faster conversion).
- But must > 0.7 us (i.e. 1/TAD < 1.4 MHz)

The total conversion time is 11 TAD (plus, say, 4 TAD for sampling).

Q3: What binary pattern must be written to ADCON2 to select left justified result (*), to set acquisition time of 4 T_{AD}, and to select $T_{OSC}/64$ as the clock source?

Your answer: _____

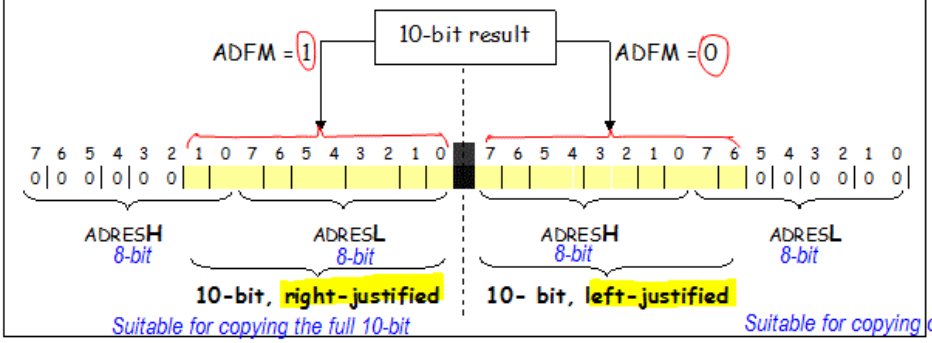
☐ The C code required is ADCON2 = 0b0 0 010 110;

Note: Refer to lecture notes for details of acquisition time & clock source selection.

AD Result (High & Low byte)

(*) Justification

- When an A/D conversion is completed, the 10-bit result is stored in the registers ADRESH and ADRESL, either **left-justified** or **right-justified**, depending on the value of the **ADFM** bit, as shown below. Note that the other bits are filled with 0's.

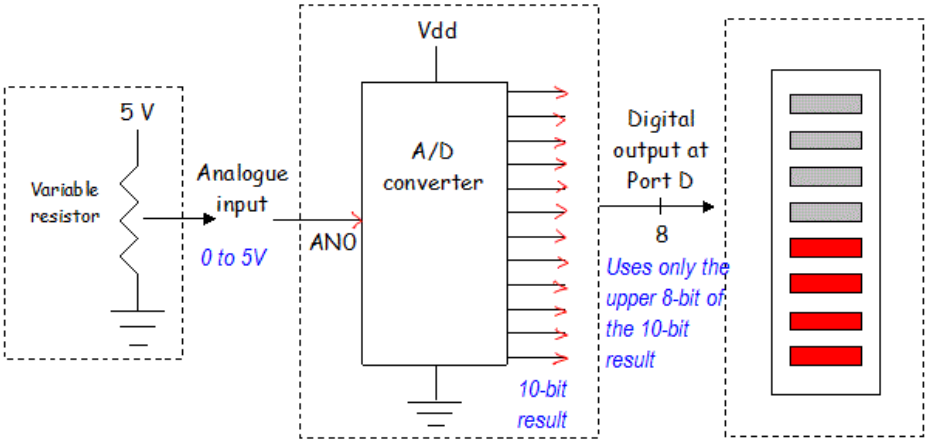


Suitable for copying the full 10-bit result to a 16-bit variable.

Suitable for copying only the most significant 8-bit of the result to an 8-bit variable.

Example 1 – Showing the result of conversion on an L E D bar

- In the circuit below, the variable resistor is used to produce an analogue input – by turning the knob on the variable resistor, the AN0 voltage can be varied. This voltage can then be AD converted into a 10-bit digital equivalent. The most significant 8 bits can then be displayed on the L E D bar.



- The complete C code is as follows:

```
main(void)
{
  TRISD = 0x00;   // Set PORTD to be output
  PORTD = 0x00;   // Initialise PORTD LEDs to zeros

  /* Configuring the ADC */
  ADCON0 = 0b00000001;   // bit5-2  0000 - select channel 0 for conversion
                         // bit1    A/D conversion status bit
                         //             1- GO to starts the conversion
                         //             0 - DONE when A/D is completed
                         // bit0    Set to 1 to power up A/D
  ADCON1 = 0b00001100;   // bit5    0 -reference is VSS
                         // bit4    0 - reference is VDD
                         // bit3-0  1100 - AN2 to AN0 Analog, the rest Digital
  ADCON2 = 0b00 010 110; // bit7    A/D Result Format:
                         //             0 - Left justified
                         //             1 - Right justified
                         // bit5-3  010 - acquisition time = 4 TAD
                         // bit2-0  110 - conversion clock = Fosc / 16
  for( ; ; )
  {
    ADCON0bits.GO = 1;           // This is bit2 of ADCON0, START CONVERSION NOW

    while(ADCON0bits.GO == 1);   // Waiting for DONE
    PORTD = ADRESH;              / Displaying only the upper 8-bits of the A/D result
.... // continue...
```

(Don't start yet)

0V

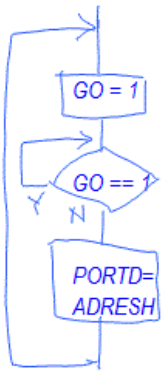
5V

Fosc / 64

Same as: while(1)

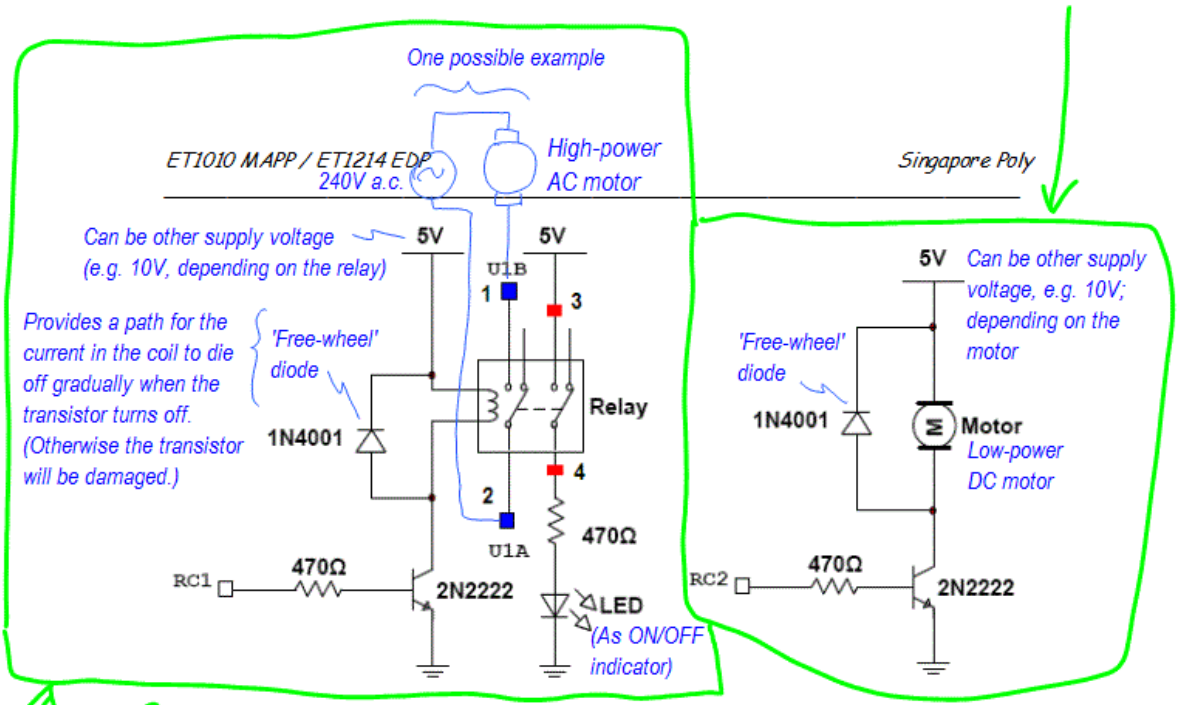
bit 1

i.e. wait for GO == 0



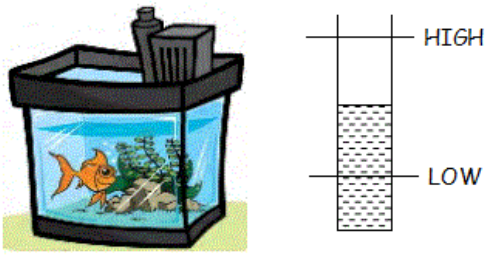
Using transistor as a switch – motor & relay

- Low power devices such as LED's can be driven directly by the microcontroller pins. To control a higher power device such as motor / relay, a transistor can be used as a switch as shown below.
- For instance, in the motor circuit on the right below, the microcontroller pin RC2 is used to turn the motor on and off.
- When RC2 = 1, the transistor (2N2222) is switched on, allowing a current to flow through & turn the motor.
- When the transistor is switched off, the diode (1N4001) allows current to continue flowing through the motor, avoiding its damage.



- In the relay circuit on the left above, the microcontroller pin RC1 is used to energise the relay.
- When RC1 = 1, the transistor is switched on, allowing a current to flow through the coil & the relay is energised. An energised relay closes the contact between points 1 and 2, as well as 3 and 4.
- Since 3 & 4 are now in contact, a current can flow through and turn on the LED. Likewise, points 1 & 2 can be used for connecting an alarm or other high power device.
- Note that the high power device connected to points 1 & 2 could be powered by a different power source. The relay can thus provide power isolation.

Example 2 – Turning on the pump/motor when water level is high, turning on the alarm when water level is low



☐ With the above discussion, it will not be difficult for you to build a fish tank "water level monitoring" application. If the water level is too high, the pump motor will be turned on to drain away the excess water. If the water level is too low, an alarm will be activated to alert the owner.

Activites:

Before you begin, ensure that the Micro-controller Board is connected to the General IO Board.

Showing the result of AD conversion on an L E D bar

1. Launch *MPLAB IDE*. Open Lab1 workspace by clicking *Project -> Open...* and selecting *ProjetA.mcp* from the *D:\PICProject* folder.

2. Replace *CountLeds.c* with *ADC.c*. If you have forgotten the steps, you will need to refer to one of the previous lab sheets.

3. Study the code and describe what this program will do:

   _____

4. Build, download and execute the program.

5. Turn the variable resistor R1 (on the General IO Board) to each of the 5 positions marked below.

Observe & record the 10-bit digital result on the LED bar (also on the General IO Board) – actually only the 8 most significant bits are shown on the LED bar, so the 2 least significant bits can be taken as 0 0.

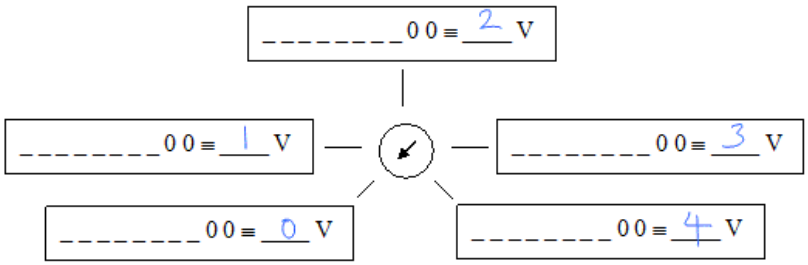
The 10-bit result is proportional to Vin:

$$\frac{Vin}{5V} = \frac{\text{10-bit result}}{2^{10}}$$

The equivalent input voltage can be "back computed" using the formula:

Voltage = (10-bit digital result x 5 V) / 2^N (N = 10 for the 10-bit converter)

[Note: some said it should be divided by $2^N - 1$. It really depends on the coding scheme used. See Wikipedia for a discussion.]



6. Are the results as expected i.e. the 10-bit digital result as well as the equivalent input voltage should increase as the knob is turned clockwise?

Your answer: _____

Fish tank "water level monitoring"

7. Modify the code above so that if the water level is above a certain level (for instance when the 10-bit digital result exceeds 0xD0), the pump motor at RC2 is turned on to pump away the excess water. On the other hand, if the water level is below a certain level (for instance 0x10), the relay at RC1 is turned on to sound an alarm to alert the owner.

   Note that you need to define LOW in your code. The if-else statement must also be expanded to control the relay.

8. Build, download and execute the program to verify your coding. Debug until the program can work. When your program is working, show it to your lecturer.

   Lecturer's signature _____

// ADC.c

```
// Program to use ADC to read variable resistor input and display on LEDs

#include <p18F4550.h>
#include <delays.h>

// other lines not shown...

main(void)
{
  #define HIGH 0xD0       // HIGH water level indicator

  TRISD = 0x00;           // Set PORTD to be output
  PORTD = 0x00;           // Initialise PORTD LEDs to zeros

  TRISCbits.TRISC1 = 0;  // RC1 as output pin
   PORTCbits.RC1    = 0;   // RC1 is connected to Relay

  TRISCbits.TRISC2 = 0;  // RC2 as output pin
   PORTCbits.RC2    = 0;   // RC2 is connected to Motor

  /* Initialise analog to digital conversion setting */

  ADCON0 = 0b00000001;        // bit5-2  0000 select channel 0 conversion
                              // bit1    A/D conversion status bit
                              //             1- GO to starts the conversion
                              //             0 - DONE when A/D is completed
                              // bit0    Set to 1 to power up A/D

  ADCON1 = 0b00001100;        // bit5    reference is VSS
                              // bit4    reference is VDD
                              // bit3-0  AN2 to AN0 Analog, the rest Digital

  ADCON2 = 0b00 010 110;      // bit7    A/D Result Format:
                              //             0 - Left justified
                              //             1 - Right justified
                              // bit5-3  010 acquisition time = 4 TAD
                              // bit2-0  110 conversion clock = Tosc / 16

  for( ; ; )
  {
    ADCON0bits.GO = 1;        // This is bit2 of ADCON0, START CONVERSION NOW

    while(ADCON0bits.GO == 1); // Waiting for DONE

    PORTD = ADRESH;           // Displaying only the upper 8-bits of the A/D result

    if(ADRESH > HIGH)
    {
      PORTCbits.RC2 = 1;      // Turn on Motor
    }
    else
    {
      PORTCbits.RC2   = 0;     // Turn off Motor
    }
  }
}
```

Initialisation:
1. PORTD
2. PORTC
3. ADC

Forever loop: