How to use the CCS compiler with the ADC port

Setting up and initialization

1. Tell the compiler to use the ADC port as either 8- or 10-bit; e.g. `#device ADC=8`

   Hint: Do not leave spaces around the = sign!

2. Inside the `main()` function, after any variables have been declared:

   1. Use the `setup_adc()` function to set up the ADC clock. For instance to use the system clock frequency divided by eight: `setup_adc(ADC_CLOCK_DIV_8);`

      Hint: The magic incantations for the various functions are listed in the PIC header file; e.g. 16f877.h which you can view in the folder **`Program files\picc\devices`**. These incantations are usually in capital lettering. See also the appendix in Slide 5

2. Set which pins are to be analog and which are to be digital using the **`setup_adc_ports()`** function. For instance if you want pins **RA0**, **RA1** and **RA3** to be analog (Channels **AN0**, **AN1** and **AN3**) and all other pins in Ports A & E to be digital (and to use the power supply for the reference voltages) then:
   `setup_adc_ports(AN0_AN1_AN3);`
   Hint: All the options from the header file are listed in Slide 5.
3. Set up which channel you want to do the conversion. For instance, to convert an analog voltage at pin **RA1** (i.e. Channel 1):
   `set_adc_channel(1);`
   HINT: If you are only ever going to use one channel then put this function along with the other setup functions near the start of the **`main()`** function, otherwise put it before starting a conversion.

Using the ADC module

Say you want to read an analog value into a variable. If we have a variable called, say, **`value`** then simply use the function **`read_adc();`**

e.g.:

```
int value;
 /* Various setting up functions etc                */


 value = read_adc();
```

Hint: If the ADC has been set up for 8-bit then **`value`** should be **`int`**, if for 10-bit then it should be **`long int`**.

Hint: Somewhere the channel number should be set up using the **`set_adc_channel(x)`** function.

Appendix: Extract from header file `16f877.h`

```
//////////////////////////////////////////////////////////////////////// ADC
// ADC Functions: SETUP_ADC(), SETUP_ADC_PORTS() (aka SETUP_PORT_A),
//                SET_ADC_CHANNEL(), READ_ADC()
// Constants used for SETUP_ADC() are:
#define ADC_OFF              0                                  // ADC Off
#define ADC_CLOCK_DIV_2    0x100
#define ADC_CLOCK_DIV_8     0x40
#define ADC_CLOCK_DIV_32    0x80
#define ADC_CLOCK_INTERNAL  0xc0                                // Internal 2-6us

// Constants used in SETUP_ADC_PORTS() are:
#define NO_ANALOGS                      7                       // None
#define ALL_ANALOG                      0                       // A0 A1 A2 A3 A5 E0 E1 E2
#define AN0_AN1_AN2_AN4_AN5_AN6_AN7_VSS_VREF 1                  // A0 A1 A2 A5 E0 E1 E2 VRefh=A3
#define AN0_AN1_AN2_AN3_AN4             2                       // A0 A1 A2 A3 A5
#define AN0_AN1_AN2_AN4_VSS_VREF        3                                  // A0 A1 A2 A5 VRefh=A3
#define AN0_AN1_AN3                     4                       // A0 A1 A3
#define AN0_AN1_VSS_VREF                5                       // A0 A1 VRefh=A3
#define AN0_AN1_AN4_AN5_AN6_AN7_VREF_VREF 0x08                  // A0 A1 A5 E0 E1 E2 VRefh=A3 VRefl=A2
#define AN0_AN1_AN2_AN3_AN4_AN5         0x09                    // A0 A1 A2 A3 A5 E0
#define AN0_AN1_AN2_AN4_AN5_VSS_VREF    0x0A                    // A0 A1 A2 A5 E0 VRefh=A3
#define AN0_AN1_AN4_AN5_VREF_VREF       0x0B                    // A0 A1 A5 E0 VRefh=A3 VRefl=A2
#define AN0_AN1_AN4_VREF_VREF           0x0C                    // A0 A1 A5 VRefh=A3 VRefl=A2
#define AN0_AN1_VREF_VREF               0x0D                    // A0 A1 VRefh=A3 VRefl=A2
#define AN0                     0x0E                            // A0
#define AN0_VREF_VREF                   0x0F                    // A0 VRefh=A3 VRefl=A2
```