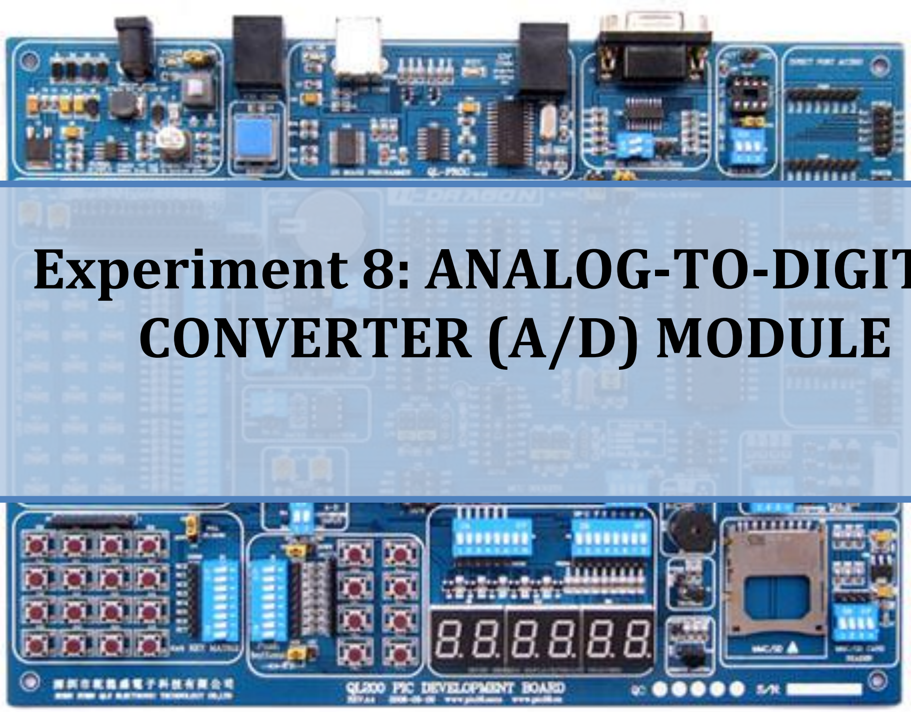
University of Jordan
Faculty of Engineering and Technology
Department of Computer Engineering
Embedded Systems Laboratory 0907334



8

Experiment 8: ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

Written by Eng. Enas Jaara

Objectives

- To familiarize you with the built-in A/D hardware module.

Pre-lab requirements

- Review the PIC16F877A datasheet section on the AD module.

Appendix A quickly reviews the AD module

Overview

An analog to digital converter converts analog voltages to digital information that can be used by a computer. In almost in all digital systems, there is a frequent need to convert analog signals generated by peripheral devices such as microphones, sensors, and etc. into digital values that can be stored and processed. As an example, temperature and brightness are changing continuously. This experiment will focus on A/D conversion by using the PIC16F877A Analog-To-Digital Converter.

The idea behind the code

Select RAO as input connected to potentiometer, get the result of a A/D conversion, convert the result into the BCD format and finally the result (the only low 8 bits) will be displayed on three 7-segment displays, The 7 segments display will use Time Division Multiplexing to display a 3-digit values.

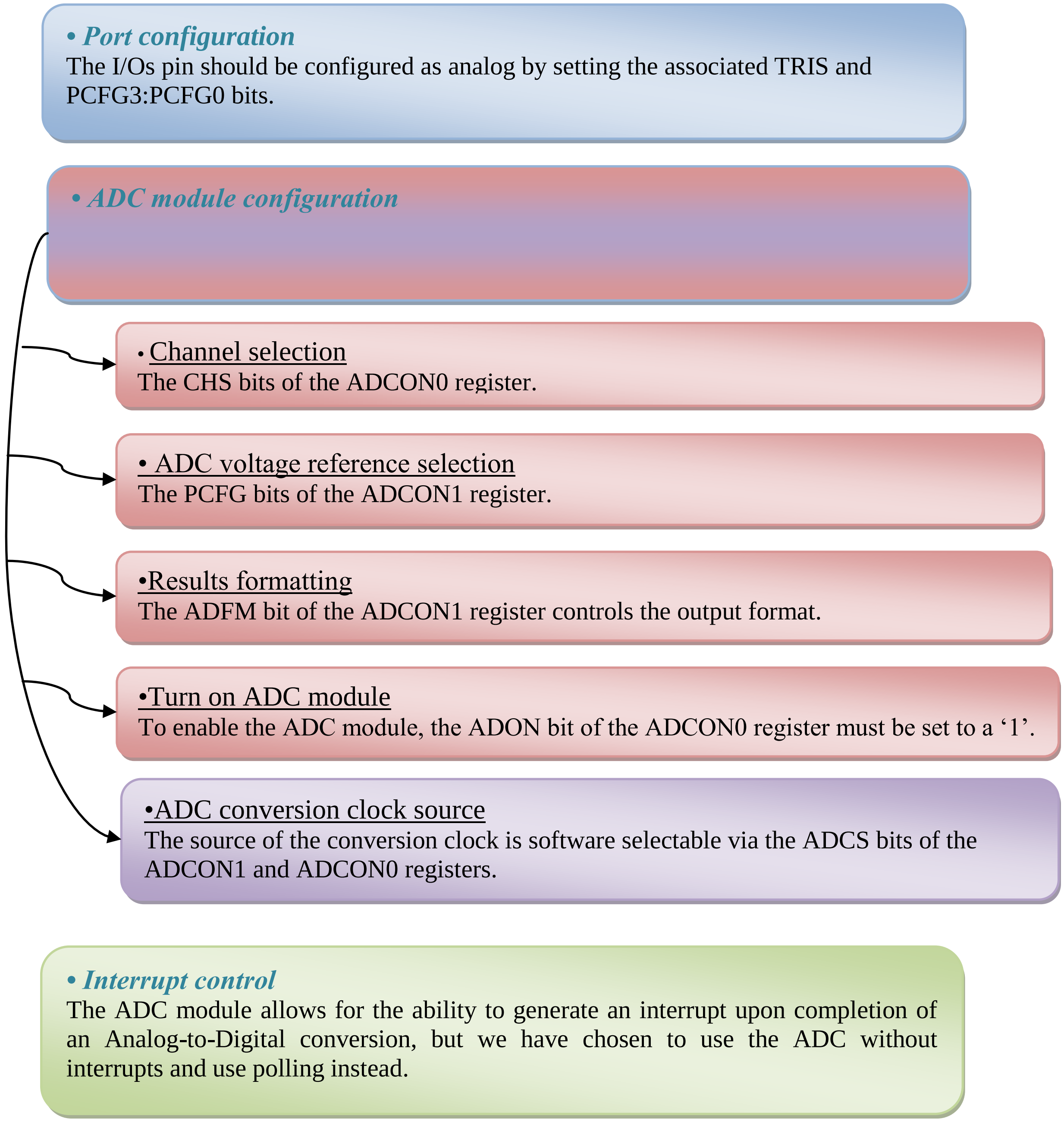
A Detailed View of the Interworking of the System

Based on the above discussion, we will further elaborate how this system works.

1. Initially, the system should be initialized as follows:
    - We need to connect an analogue signal to the PIC, we shall use either one of PORTA or PORTE, since both offer analogue input interfacing to the PIC. We will specify which PORT and which exact pin of the port to be used as analogue or digital through the use of the **ADCON1** register. In this experiment we chose RA0 as the analogue input (corresponding to channel 0 “AN0” of the AD module)
    - We will configure the AD module as follows, power on the module (set ADON), and choose the analogue channel 0 “AN0” as the analogue input of the AD module by setting CH2, CH1 and CH0 as zeros. We will set the voltage references to be between 0 and 5 volts (why?) and finally the result is to be right justified, that is the lower 8-bits will reside in ADRESL and the higher 2 bits will reside in ADRESH. In this program, we will choose to ignore ADRESL and only deal with the 8-bit digitized value to simplify program development.
    - We chose a conversion speed of Fosc/8, therefore ADCON1 will have the value of 0x8E
    - We implemented the code such that the main functionality is to convert analogue signals into digital ones and save them into ADRESL in a continuous fashion such that we will always have updated and recent values of the potentiometer, this is the code of the main subroutine will have all other actions: CHANGE _To_BCD ,this subroutine is used to convert the result of the conversion into BCD values (Units , Tens , Hundreds), then display the result on the 7 segment display , Time Division Multiplexing used to display a 3-digit values(Units , Tens , Hundreds).

2. As stated above, the main subroutine is to continuously update ADRESL register with a recent digitized value of the potentiometer. The routine starts by starting the conversion process (bsf ADCON0, GO), the value of ADRESL is not read until we are sure that the conversion process has truly finished. This is done through polling the ADIF flag (remember that we have not enabled the interrupt for AD, yet the flags of interrupts are set and cleared no matter whether they were enabled or not, this is why polling is possible). When the conversion is finished, the value of ADRESL is copied into TEMP register in order to display it on the 7 segment display!

The steps should be followed for doing an A/D Conversion:



- ***Port configuration***

  The I/Os pin should be configured as analog by setting the associated TRIS and PCFG3:PCFG0 bits.

- ***ADC module configuration***

  - Channel selection

    The CHS bits of the ADCON0 register.

  - ADC voltage reference selection

    The PCFG bits of the ADCON1 register.

  - Results formatting

    The ADFM bit of the ADCON1 register controls the output format.

  - Turn on ADC module

    To enable the ADC module, the ADON bit of the ADCON0 register must be set to a ‘1’.

  - ADC conversion clock source

    The source of the conversion clock is software selectable via the ADCS bits of the ADCON1 and ADCON0 registers.

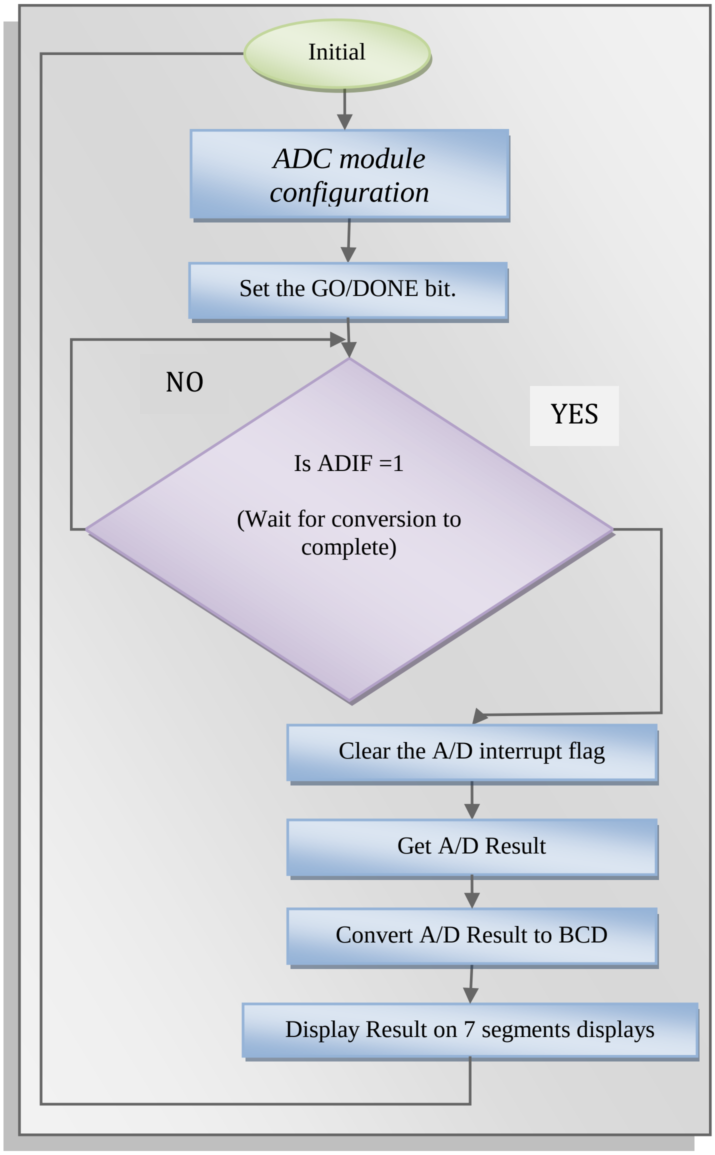
- ***Interrupt control***

  The ADC module allows for the ability to generate an interrupt upon completion of an Analog-to-Digital conversion, but we have chosen to use the ADC without interrupts and use polling instead.

;;;Main Subroutine;;;

This subroutine shows theA/D Conversion Procedure.

Start conversion by setting the GO/DONE bit. Poll the AD interrupt flag ADIF (interrupts disabled) to check whether conversion has finished or not. Clear the ADC interrupts flag (required). Finally Read ADC Result found in ADRESH and/or ADRESL. Convert Result into BCD Format and display it on the 7 segments displays.


Initial
ADC module configuration
Set the GO/DONE bit.
NO
YES
Is ADIF =1
(Wait for conversion to complete)
Clear the A/D interrupt flag
Get A/D Result
Convert A/D Result to BCD
Display Result on 7 segments displays

```
Main
 MOVLW    8EH             ;A/D data right justified
 MOVWF    ADCON1          ; RA0  is analogue input
 Banksel  PORTA           ;BANK 0
 MOVLW    41H             ;A/D enabled
 MOVWF    ADCON0         ;select CLOCK is fosc/,
 CALL     DELAY
 BSF      ADCON0,GO        ;startup ADC divert
WAIT
 BTFSS    PIR1,ADIF      ;Is the convert have finished?
 GOTO     WAIT            ; wait for the convert finished
 bcf      PIR1, ADIF     ; Clear the A/D flag
Banksel   TRISA
MOVF      ADRESL,W        ;read the result of convert
Banksel   PORTA
MOVWF     TEMP             ; keep in temporary register
CALL      CHANGE_To_BCD    ; call result convert subr.
CALL      DELAY
CALL      DISPLAY            ; call display subroutine
CALL      DELAY
GOTO      Initial            ; Do it again
```

```
 ;****************************************************
;Code Function:Select RAO as input connected to potentiometer,
;get the result of a A/D conversion ,convert the result into the BCD format
; and finally the result (the only low 8 bits) will be displayed on 7-segment displays.

#INCLUDE<P16F877a.INC>

TEMP          EQU   20H     ;temporary register
hundreds      EQU   21H     ;the hundred bit of convert result
tens          EQU   22H     ;the ten bit of convert result
units         EQU   23H     ;the ones bit of convert result
;****************************************************
          ORG     00H
           NOP
           GOTO     Initial

;******************Initial subroutine****************************
Initial
                   CLRF     hundreds
                    CLRF     tens
                    CLRF     units
                    Banksel   TRISA                         ;select bank 1
                    MOVLW    01H                            ;PORTA bit Number0 is INPUT
                    MOVWF    TRISA
                    CLRF     TRISD                        ;All of the PORTD bits are outputs
;**********************MAIN program**********************
Main
                    MOVLW    8EH             ;A/D data right justified
                    MOVWF    ADCON1          ;only select RA0 as ADC PORT,the rest are data PORT
                    Banksel   PORTA           ;BANK 0
                    MOVLW    41H
                    MOVWF    ADCON0          ;select CLOCK is fosc/8,A/D enabled
                    CALL     DELAY           ;call delay program,ensure enough time to sampling
                    BSF      ADCON0,GO       ;startup ADC divert
WAIT
                    BTFSS     PIR1,ADIF        ;is the convert have finished?
                    GOTO      WAIT             ;wait for the convert finished
                    Bcf       PIR1, ADIF       ; Clear the A/D flag
                    Banksel    TRISA
                    MOVF      ADRESL,W         ;read the result of convert
                    Banksel    PORTA
                    MOVWF    TEMP                ;keep Result in temporary register
                    CALL        CHANGE_To_BCD   ;call result convert subroutine
                    CALL        DELAY
                    CALL        DISPLAY          ;call display subroutine
                    CALL        DELAY
                    GOTO        Initial          ;Do it again
;***********************Convert subroutine*******************
CHANGE_To_BCD
gen_hunds
                    MOVLW    .100                ;sub 100,result keep in W
                   SUBWF    TEMP,0
                    BTFSS    STATUS,C            ;judge if the result biger than 100
                    GOTO     gen_tens            ;no,get the ten bit result
                    MOVWF    TEMP                ;yes,result keep in TEMP
                    INCF     hundreds,1          ;hundred bit add 1
                    GOTO     gen_hunds           ;continue to get hundred bit result
gen_tens
                    MOVLW    .10                ;sub 10,result keep in W
                    SUBWF    TEMP,0
                    BTFSS    STATUS,C           ;judge if the result biger than 10
                    GOTO     gen_ones           ;no,get the Entries bit result
                    MOVWF    TEMP               ;yes,result keep in TEMP
                    INCF     tens,1             ;ten bit add 1
                    GOTO     gen_tens          ;turn  to continue get ten bit
```

```
gen_ones
                 MOVF     TEMP,W
                 MOVWF    units              ;the value of Entries bit
                 RETURN

;************************Display subroutine*******************
        DISPLAY
         MOVF    hundreds,W              ;display Hundreds bit
         CALL    TABLE
         MOVWF   PORTD
         BCF     PORTA,3
         CALL    DELAY
         CALL    DELAY
         BSF     PORTA,3

         MOVF    tens,W                  ;display Tens bit
         CALL    TABLE
         MOVWF   PORTD
         BCF     PORTA,4
         CALL    DELAY
         CALL    DELAY
         BSF     PORTA,4

         MOVF    units,W                 ;display Units bit
         CALL    TABLE
         MOVWF   PORTD
         BCF     PORTA,5
         CALL    DELAY
         CALL    DELAY
         BSF     PORTA,5
         RETURN

;*******************************************************
TABLE
                 ADDWF   PCL,      1
                 RETLW   B'11000000'            ;'0'
                 RETLW   B'11111001'            ;'1'
                 RETLW   B'10100100'            ;'2'
                 RETLW   B'10110000'            ;'3'
                 RETLW   B'10011001'            ;'4'
                 RETLW   B'10010010'            ;'5'
                 RETLW   B'10000010'     ;'6'
                 RETLW   B'11111000'            ;'7'
                 RETLW   B'10000000'            ;'8'
                 RETLW   B'10010000'            ;'9'

;*************************Delay subroutine*********************
DELAY
        MOVLW   0xFF
        MOVWF   TEMP
L1      DECFSZ  TEMP,1
        GOTO    L1
        RETURN

;*******************************************************
 END              ;program end
```

Appendix A

Analog-to-Digital Conversion (ADC)

An analog-to-digital converter, or simply ADC, is a module that is used to convert an analog signal into a digital code. In the real world, most of the signals sensed and processed by humans are analog signals. Analog-to-digital conversion is the primary means by which analog signals are converted into digital data that can be processed by Microcontroller for various purposes.

Sensors signals is an analog quantity, and digital systems often use signals to implement measurement, control, and protection functions so it is the necessary to convert the analog signal to digital information.

There's generally a lot of confusion about using the A/D inputs, but it's actually really very simple - it's just a question of Extraction the information you need out of the datasheets.

There are four main registers associated with using the analogue inputs; these are summarized in the following table:

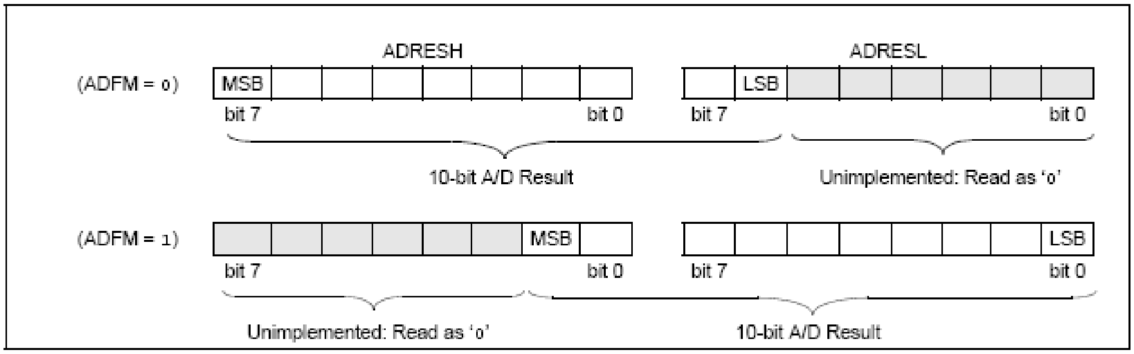
Main registers used for Analog-to-Digital Conversion.

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADRESH	A/D Result Register - High Byte							
ADRESL	A/D Result Register - Low Byte							
ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	-	ADON
ADCON1	ADFM	ADCS2	-	-	PCFG3	PCFG2	PCFG1	PCFG0

- **ADCON0** and **ADCON1** are the registers that control the A/D conversation process.
- **ADRESH** and **ADRESL** are the registers that return the 10-bit result of the analogue to digital conversion, the only slightly tricky thing about them is that they are in different memory banks.

RESULT FORMATTING:

The 10-bit A/D conversion result can be supplied in two formats, left justified or right justified. The desired formatting is chosen by sitting the ADFM bit in the ADCON0 register.



ADCON0 Details

ADON (bit 0), turns the A/D On (when = 1) or off (when = 0), thus saving the power it consumes.

GO/DONE (bit 2), this bit has a dual function, the first is that by setting the bit it initiates the start of the analogue to digital conversion process, the second is that when the bit is automatically cleared when the conversion is complete, it can be polled to check if conversion has ended before initiating a subsequent conversion.

CHS2, **CHS1** and **CHS0 (bits 3 - 5)**, the channel selection bits, choose one channel among the available eight AD analogue channels and specify which one is to be used as an input for the AD module for digitization. Be careful that the first five channels AN0-AN4 map to pins (RA0-RA3, RA5). Further notice that AN4 uses digital pin RA5, not RA4 as you would expect. And the remaining three channels AN5-AN7 map to pins (RE0-RE2). See adjacent figure.

ADCS1 and ADCS0 (bits 6 - 7): A/D Conversion Clock Select bits (see **ADCS2)**

CHS2	CHS1	CHS0	Channel	Pin
0	0	0	Channel0	RA0/AN0
0	0	1	Channel1	RA1/AN1
0	1	0	Channel2	RA2/AN2
0	1	1	Channel3	RA3/AN3
1	0	0	Channel4	RA5/AN4
1	0	1	Channel5	RE0/AN5
1	1	0	Channel6	RE1/AN6
1	1	1	Channel7	RE2/AN7

ADCON1 Details

ADFM (bit 7), the **Result Format Selection Bit**, selects if the output is Right Justified (bit set) or Left Justified (bit cleared). For full digitization precision, the whole 10 bits are to be used.

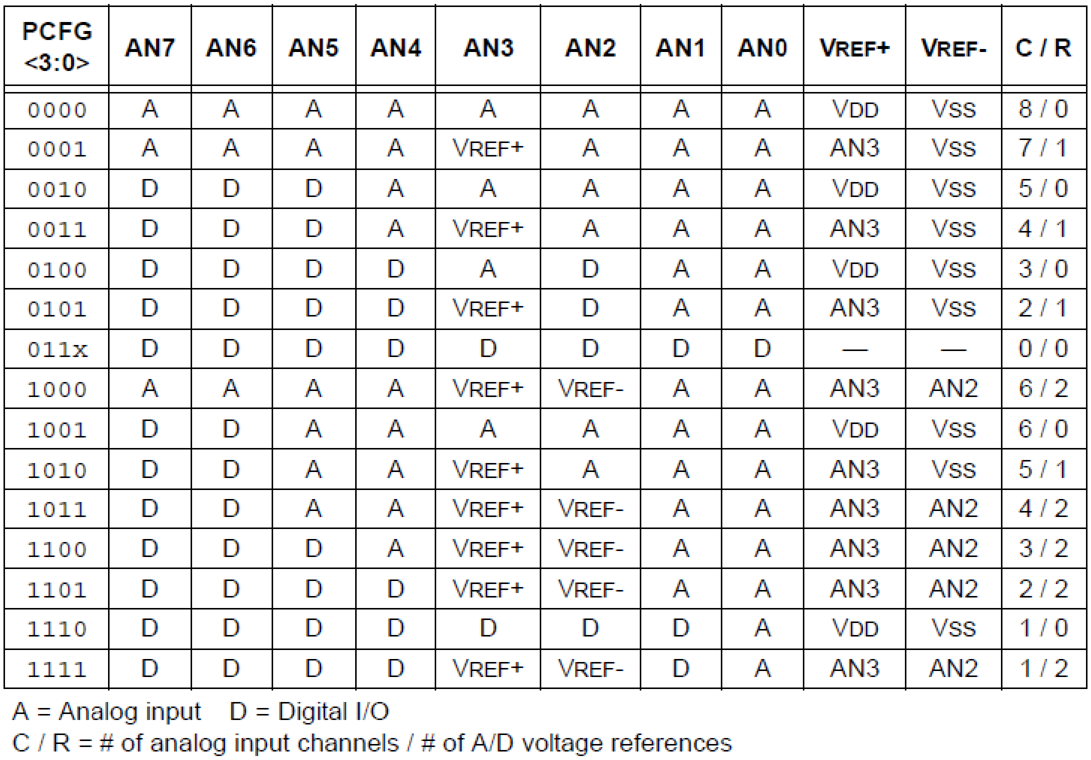
ADCS2 (bit 6), which set the clock frequency used for the analogue to digital conversion, this clock is divided down from the system clock (or can use an internal oscillator), bit 4 and bit 5 Unimplemented: Read as '0'.

ADCON1	**ADCON0**		**A/D Conversion Clock Select bits.**
ADCS2	**<ADCS1:ADCS0>**		
0	0	0	Fosc/2
0	0	1	Fosc/8
0	1	0	FOsc/32
X	1	1	FRC (clock derived from a dedicated Internal oscillator = 500 kHz max.)
1	0	0	Fosc/4
1	0	1	Fosc/16
1	1	0	Fosc/64

PCFG3:PCFG0 (bit 3:0): A/D Port Configuration Control bits

Example

If we make ADCON1 = 0x80, then we have 8 analog channels, and Vref+ = VDD, and Vref- = Vss.



PCFG <3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C / R
0000	A	A	A	A	A	A	A	A	VDD	VSS	8 / 0
0001	A	A	A	A	VREF+	A	A	A	AN3	VSS	7 / 1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5 / 0
0011	D	D	D	A	VREF+	A	A	A	AN3	VSS	4 / 1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3 / 0
0101	D	D	D	D	VREF+	D	A	A	AN3	VSS	2 / 1
011x	D	D	D	D	D	D	D	D	—	—	0 / 0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6 / 2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6 / 0
1010	D	D	A	A	VREF+	A	A	A	AN3	VSS	5 / 1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4 / 2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3 / 2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2 / 2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1 / 0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1 / 2

A = Analog input   D = Digital I/O
C / R = # of analog input channels / # of A/D voltage references