Using the Analog to Digital Converter

INTRODUCTION

This application note is intended for PIC16C7X users with various degrees of familiarity with analog system design. The various sections discuss the following topics:

- Commonly used A/D terminology
- How to configure and use the PIC16C71 A/D
- Various ways to generate external reference voltage (VREF)
- Configuring RA0-RA3 pins

COMMONLY USED A/D TERMINOLOGY

The Ideal Transfer Function

In an A/D converter, an analog voltage is mapped into an N-bit digital value. This mapping function is defined as the transfer function. An ideal transfer is one in which there are no errors or non-linearity. It describes the "ideal" or intended behavior of the A/D. Figure 1 shows the ideal transfer function for the PIC16C7X A/D. Note that the digital output value is 00h for analog input voltage range of 0 to 1LSB. In some converters, the first transition point is at 0.5LSB and not at 1LSB as shown in Figure 2. Either way, knowing the transfer function the user can appropriately interpret the data.

Transition Point

It is the analog input voltage at which the digital output switches from one code to the next. The transition point is typically not a single threshold, rather a small region of uncertainty (see Figure 3) The transition point is therefore defined as the statistical average of many conversions. Stated differently, it is the voltage input at which the uncertainty of the conversion is 50%.

Code Width

It is the distance (voltage differential) between two transition points. Ideally the Code Width should be 1LSB. See Figure 1.

FIGURE 1 - PIC16C7X IDEAL TRANSFER FUNCTION

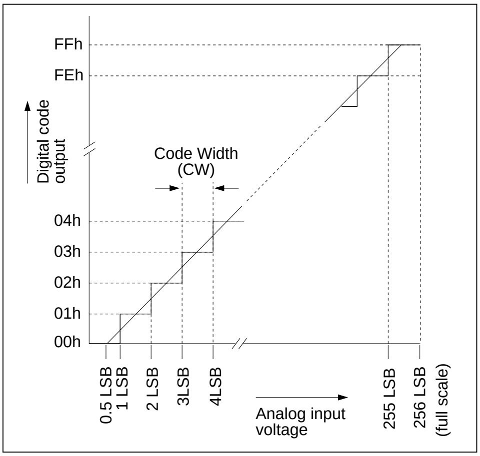
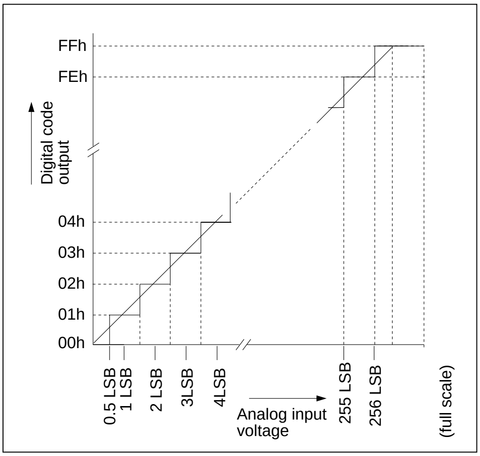


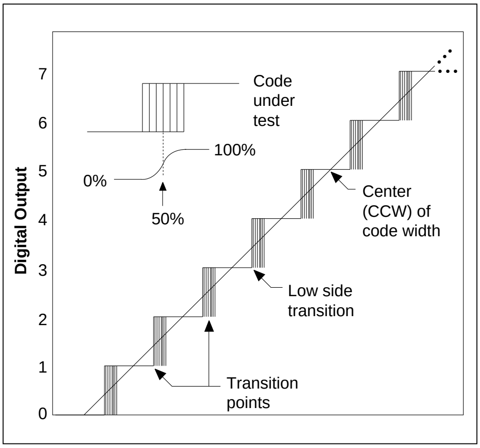
FIGURE 2 - ALTERNATE TRANSFER FUNCTION



Center of Code Width

It is the midpoint between two transition points. See Figure 3.

FIGURE 3 - TRANSITION POINTS



Differential Non-Linearity (DNL)

It is the deviation in code-width from 1LSB (Figure 7). The difference is calculated for each and every transition. The largest difference is reported as DNL.

It is important to note that the DNL is measured after the transfer function is normalized to match offset error and gain error.

Note that the DNL cannot be any less than -1LSB. In the other direction, DNL can be >1LSB.

Absolute Error

The maximum deviation between any transition point from the corresponding ideal transfer function is defined as the absolute error. This is how it is measured and reported in the PIC16C7X (Figure 8). The notable difference between absolute error and INL is that the measured data is not normalized for full scale and offset errors.

It is probably the first parameter the user will look at to evaluate an A/D. Sometimes absolute error is reported as the sum of offset, full-scale and integral non linearity errors.

Total Unadjusted Error

It is the same as absolute error. Again, sometimes it is reported as the sum of offset, full-scale and integral non-linearity errors.

No Missing Code

No missing code implies that as the analog input voltage is gradually increased from zero to full scale (or vice versa), all digital codes are produced. Stated otherwise, changing analog input voltage from one quantum of the analog range to the next adjacent range will not produce a change in the digital output by more than one code count.

Monotonic

Monotonicity guarantees that an increase (or decrease) in the analog input value will result in an equal or greater digital code (or less). Monotonicity does not guarantee that there are no missing codes. However, it is an important criterion for feedback control systems. Non-monotonicity may cause oscillations in such a system.

The first derivative of a monotonic function always has the same sign.

FIGURE 7 - DIFFERENTIAL NON-LINEARITY

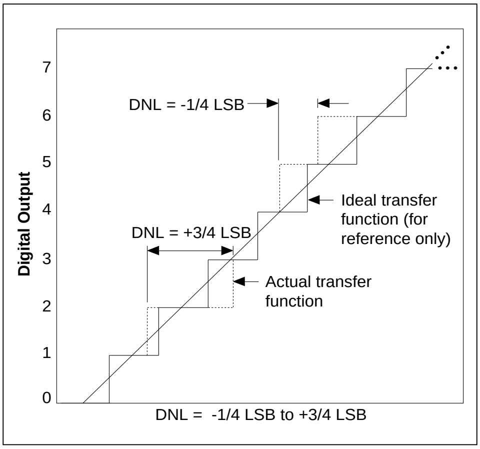
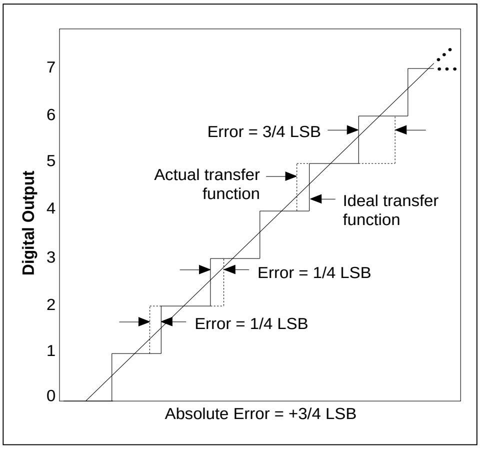


FIGURE 8 - ABSOLUTE ERROR



Ratiometric Conversion

It is the A/D conversion process where the binary result is a ratio of the supply voltage or reference voltage, the latter being equal to full-scale value by default. The PIC16C7X is a ratiometric A/D converter where the result depends on V_{DD} or V_{REF}.

In some A/D's, an absolute reference is provided resulting in "absolute conversion".

Sample and Hold

In sample and hold type A/D converters, the analog input has a switch (typically a FET switch in CMOS) which is opened for a short duration to capture the analog input voltage onto an on-chip capacitor. Conversion is typically started after the sampling switch is closed.

Track and Hold

It is basically the same as sample and hold, except the sampling switch is typically left on. Therefore the voltage on the on-chip holding capacitor "tracks" the analog input voltage. To begin a conversion, the sampling switch is shut off.

The PIC16C7X A/D falls in this category.

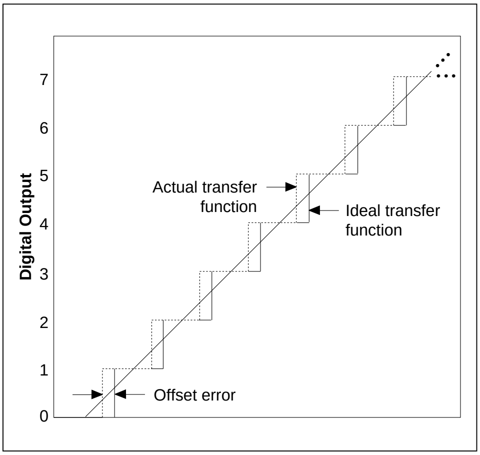
Sampling Time

It is the time required to charge the on-chip holding capacitor to the same value as on the analog input pin. The sampling time depends on the magnitude of the holding capacitor and the source impedance of the analog voltage input.

Offset Error (or Zero Error)

It is the difference between the first actual (measured) transition point and the first ideal transition point as shown in Figure 4. It can be corrected by the user by subtracting the offset error from each conversion result.

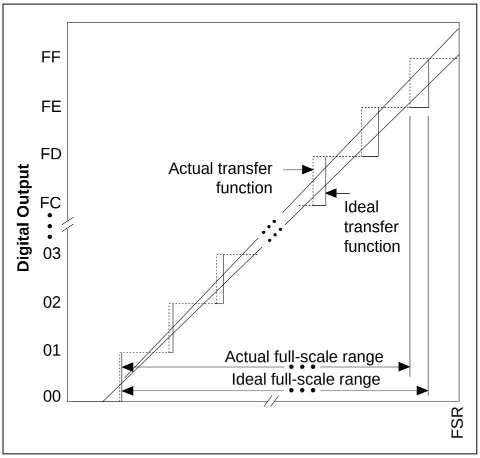
FIGURE 4 - OFFSET ERROR



Full Scale Error (or Gain Error)

It is the difference between the ideal Full Scale and the actual (measured) full scale range (see Figure 5). It is also called gain error, because the error changes the slope of the ideal transfer function creating a gain factor. It can be corrected by the user by multiplying each conversion result by the inverse of the gain.

FIGURE 5 - FULL SCALE ERROR

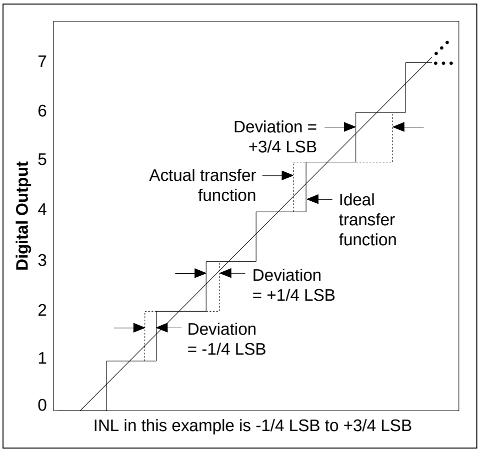


Integral Non-Linearity (INL), or Relative Error

It is the deviation of a transition point from its corresponding point on the ideal transfer curve (Figure 6). The maximum difference is reported as the INL of the converter.

It is important to note that Full Scale Error and the Offset Error are normalized to match end transition points before measuring the INL.

FIGURE 6 - INTEGRAL NON-LINEARITY



HOW TO USE THE PIC16C71 A/D

The A/D in the PIC16C71 is easy to set up and use. There are a few considerations:

1. Select either V_{DD} or V_{REF} as reference voltage. More on using V_{REF} input later.
2. Select A/D conversion clock (tad): 2 tosc, 8 tosc, 32 tosc or trc (internal RC clock). For the first three options, make sure that tad ≥ 2.0 µs. If deterministic conversion time is required, select tosc time base. If conversion during SLEEP is required, select trc.
3. Channel Selection : If only one A/D channel is required, program the ADCON1 register to 03h. This configures the A/D pins as digital I/O. If multiple channels are required, prior to each conversion the new channel must be selected.
4. Sampling and Conversion: After a new channel is selected, a minimum amount of sampling time must be allowed before GO bit in ADCON0 is set to begin conversion. Once conversion begins, it is OK to select the next channel, ***but sampling does not begin until current conversion is complete.*** Therefore, it is always necessary to provide minimum required sampling time

   i) after a conversion

   ii) after a new channel is selected

   iii) after A/D is turned on (ADON = 1).
5. Reading Result: Completion of conversion can be determined by either polling GO/$\overline{\text{DONE}}$ bit to cleared, polling the ADIF bit to be set, or waiting for an ADIF interrupt.

ADDITIONAL TIPS:

1. The GO bit and the ADON bit may not be set at once. After the A/D is turned on by setting ADON, at least 5µs time must be allowed before conversion begins, longer if sampling time requirement is not met within 5µs.
2. Aborting a conversion: A conversion can be aborted by clearing GO bit. The A/D converter will stop conversion and revert back to sampling state.
3. Using ADRES register as a normal register: The A/D only writes to ADRES at the end of a conversion. Therefore, it is possible to use ADRES as a normal file register between conversions and when A/D is off.

The following are a few examples of using the A/D.

EXAMPLE 1: HOW TO DO A SIMPLE ADC CONVERSION

```
;
; InitializeAD, initializes and sets up the A/D hardware.
; Always ch2, internal RC OSC.
InitializeAD
          bsf       STATUS, 5      ; select pg1
          movlw     B'00000000'    ; select RA0-RA3…
          movwf     ADCON1         ; as analog inputs
          bcf       STATUS, 5      ; select pg0
          movlw     B'11010001'    ; select: RC osc, ch2...
          movwf     ADCON0         ; turn on A/D
Convert   call      sample-delay   ; provide necessary sampling time
;
          bsf       ADCON0, 2      ; start new A/D conversion
loop
          btfsc     ADCON0, 2      ; A/D over?
          goto      loop           ; no then loop
;
          movf      ADRES, w       ; yes then get A/D value
;
```

A detailed code listing is in Appendix A.

EXAMPLE 2: HOW TO DO SEQUENTIAL CHANNEL CONVERSIONS

```
;
; InitializeAD, initializes and sets up the A/D hardware.
; Select ch0 to ch3 in a round robin fashion, internal RC OSC.
; Load results in 4 consecutive addresses starting at ADTABLE (10h)
;
InitializeAD
          bsf       STATUS, 5    ; select pg1
          movlw     B'00000000'  ; select RA0-RA3…
          movwf     ADCON1       ; as analog inputs
          bcf       STATUS, 5    ; select pg0
          movlw     B'11000001'  ; select: RC osc, ch0...
          movwf     ADCON0       ; turn on A/D
          movlw     ADTABLE      ; point fsr to top of...
          movwf     FSR          ; table
;
new_ad    call      sample_delay ; provide necessary sampling time
          bsf       ADCON0, 2    ; start new A/D conversion
loop
          btfsc     ADCON0, 2    ; A/D over?
          goto      loop         ; no then loop
;
          movf      adres, w     ; yes then get A/D value
          movwf     0            ; load indirectly
          movlw     4            ; select next channel
          addwd     ADCON0       ;      /
          bcf       ADCON0, 5    ; reset carry over bit.
; increment pointer to correct table offset.
          clrf      temp         ; clear temp register
          btfsc     ADCON0, 3    ; test lsb of channel select
          bsf       temp, 0      ; set if ch1 selected
          btfsc     ADCON0, 4    ; test msb of channel select
          bsf       temp, 1      ;      /
          movlw     ADTABLE      ; get table address
          addwf     temp, w      ; add with temp
          movwf     FSR          ; move into indirect
          goto      new_ad
;
```

A detailed code listing is in Appendix B.

3

EXAMPLE 3: HOW TO WRITE THE INTERRUPT HANDLER FOR THE ADC

```
            org         0x00
            goto        start
            org         0x04
            goto        service_ad      ; interrupt vector
;
;           org         0x10
start
            movlw       B'00000000'     ;init I/O ports
            movwf       PORT_B
            tris        PORT_B
;
            call        InitializeAD
update
            bcf         flag,adover     ; reset software A/D flag
            call        SetupDelay      ; setup delay >= 10uS.
            bcf         ADCON0,adif     ; reset A/D int flag (ADIF
            bsf         ADCON0,adgo     ; start new A/D conversion
            bsf         INTCON,gie      ; enable global interrupt
loop
            btfsc       flag,adover     ; A/D over?
            goto        update          ; yes start new conv.
            goto        loop            ; no then keep checking
; InitializeAD, initializes and sets up the A/D hardware.
; select ch0 to ch3, RC OSC., a/d interrupt.
InitializeAD
            bsf         STATUS, 5       ; select pg1
            movlw       B'00000000'     ; select RA0-RA3...
            movwf       ADCON1          ; as analog inputs
            bcf         STATUS, 5       ; select pg0
            clrf        INTCON          ; clr all interrupts
            bsf         INTCON, 6       ; enable A/D int.
            movlw       B'11010001'     ; select: RC osc, ch2...
            movwf       ADCON0          ; turn on A/D
            return
;
service_ad
            btfss       ADCON0, 1       ; A/D interrupt?
            retfie                      ; no then ignore
            movf        ADRES, W        ; get A/D value
            return                      ; do not enable int
;
```

A detailed code listing is in Appendix C.

EXAMPLE 4: HOW TO DO CONVERSIONS DURING SLEEP MODE

```
;
; InitializeAD, initializes and sets up the A/D hardware.
; Select ch0 to ch3, internal RC OSC.
; While doing the conversion put unit to sleep.  This will
; minimize digital noise interference.
; Note that ad's RC osc. has to be selected in this instance.
;
InitializeAD
           bsf        STATUS, 5      ; select pg1
           movlw      B'00000000'    ; select RA0-RA3...
           movwf      ADCON1         ; as analog inputs
           bcf        STATUS, 5      ; select pg0
           movlw      B'11000001'    ; select: RC osc, ch0...
           movwf      ADCON0         ; turn on A/D & ADIE
           movlw      ADTABLE        ; point fsr to top of...
           movwf      FSR            ; table
;
new_ad
           bsf        ADCON0, 2      ; start new A/D conversion
           sleep                     ; goto sleep
; when A/D is over program will continue from here
;
           movf       ADRES, w       ; get A/D value
;
```

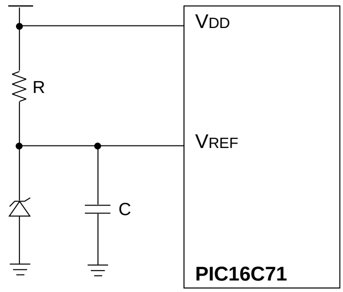
A detailed code listing is in Appendix D.

USING EXTERNAL REFERENCE VOLTAGE

When using external reference voltage, keep in mind that any analog input voltage must not exceed VREF.

An inexpensive way to generate VREF is by employing zener diode (Figure 9). Most common zener diodes offer 5% accuracy. Reverse bias current may be as low as 10 µA. However, larger currents (1mA - 20mA) are recommended for stability, as well as lower impedance of the VREF source.

FIGURE 9 - LOW COST VOLTAGE REFERENCE



C = 0.01 to 0.1µF

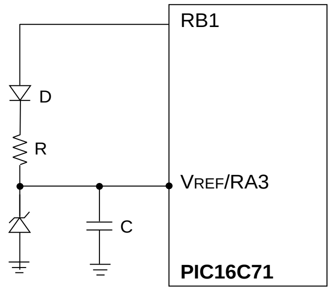
POWER MANAGEMENT IN USING VREF

In power sensitive applications, user may turn on VREF generator using another I/O pin as shown in Figure 10. Drive a “1” on RB1 pin in this example when using the A/D. Drive a “0” on RB1 pin when not using the A/D converter.

Note that this way RB1 is not floating. Even if VREF decays to some intermediate voltage, it will not cause the input buffer on RB1 to draw current.

Alternately, use RA0, RA1 or RA2 pin to supply the current instead of RB1. Configure the RA pin as analog (this will turn off its input buffer). Then use it as a digital output (Figure 11).

FIGURE 10 - POWER-SENSITIVE APPLICATIONS #1



ZENERS AND REFERENCE GENERATORS

Finally, various reference voltage generator chips (typically using on-chip band-gap reference) are available. These are more accurate.

TABLE 1 - ZENERS AND REFERENCE GENERATORS

Zeners	VZ	Tolerance
1N746	3.3V	±5%
1N747	3.6V	±5%
1N748	3.9V	±5%
1N749	4.3V	±5%
1N750	4.7V	±5%
1N751	5.1V	±5%
1N752	5.6V	±5%
Voltage References	**VREF**	**Tolerance**
AD580 (Maxim)	2.5V	±3% to ±0.4%
LM385	2.5V	±1.5%
LM1004	2.5V	±1.2%
LT1009 (LIN. Tech.)	2.5V	±0.2%
LT1019 (LIN. Tech.)	5.0V	±0.2%
LT1021 (LIN. Tech.)	5.0V	±0.05% to ±1%
LT1029 (LIN. Tech.)	5.0V	±0.2% to ±1%

VREF IMPEDANCE AND CURRENT SUPPLY REQUIREMENTS

Ideally, VREF should have as low a source impedance as possible. Referring to Figure 9, VREF source impedence ≈ R. However, smaller R increases current consumption. Since VREF is used to charge capacitor arrays inside the A/D converter and the holding capacitor CHOLD ≈ 51pF, the following guideline should be met:

tad = 6 (1K + R) 51.2 pF + 1.677µs

tad = conversion clock. For tad = 2µs and for CHOLD = 50 pF, RVREF ≈ 50Ω.

For VREF impedance higher than this, the conversion clock (tad) should be increased appropriately.

FIGURE 11 - POWER-SENSITIVE APPLICATIONS #2

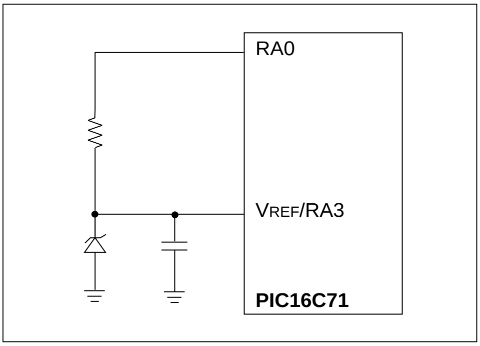


Table 2 gives examples of the maximum rate of conversion per bit, relating to the voltage reference impedance.

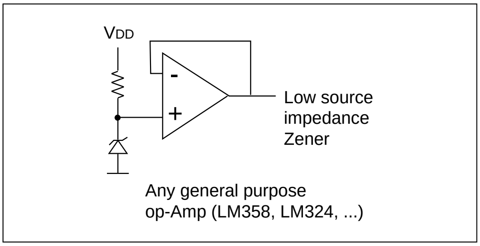
TABLE 2 - MAXIMUM RATE OF CONVERSION / BIT

RVREF	Tad (Max)
1K	2.29 µs
5K	3.52 µs
10K	5.056 µs
50K	16.66 µs
100K	32.70 µs

Assumes no external capacitors.

To achieve a low source impedance when using a Zener diode, a voltage follower circuit is recommended. This is shown in Figure 11A.

FIGURE 11A - VOLTAGE FOLLOWER CIRCUIT



CONFIGURING PORT A INPUTS AS ANALOG OR DIGITAL

Two bits in ADCON1 register PCFG1 and PCFG0 control how pins RA0–RA3 are configured. When any of these pins are selected as analog:

- The digital input buffer is turned off to save current (see Figure 12). Reading the port will read this pin as '0'.
- TRIS bit still controls the output buffer on this pin. So, normally the TRIS bit will be set (input).
- However, if the TRIS bit is cleared, then the pin will output whatever is in the data latch.

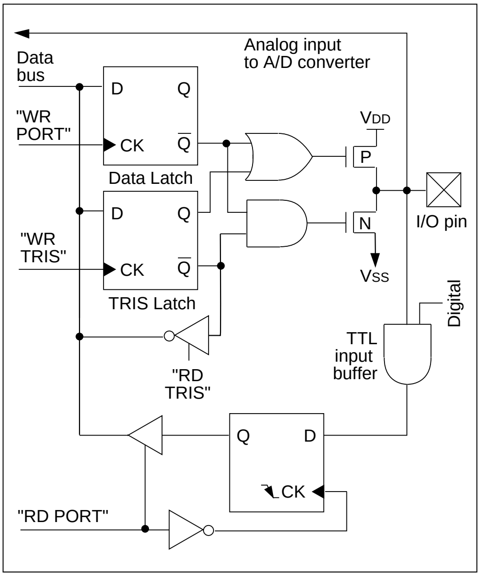
When any of these pins are selected as digital:

- The analog input still directly connects to the A/D and therefore the pin can be used as analog input.
- The digital input buffer is not disabled.

The user has, therefore, great flexibility in configuring these pins.

3

FIGURE 12 - BLOCK DIAGRAM OF RA0-RA3 PINS



CURRENT CONSUMPTION THROUGH INPUT BUFFER

A CMOS input buffer will draw current when the input voltage is around its threshold. (See Figure 13.)

In power-sensitive applications, the RA pins when used as analog inputs should be configured as "analog" to avoid unintended power drain.

Other considerations and tips:

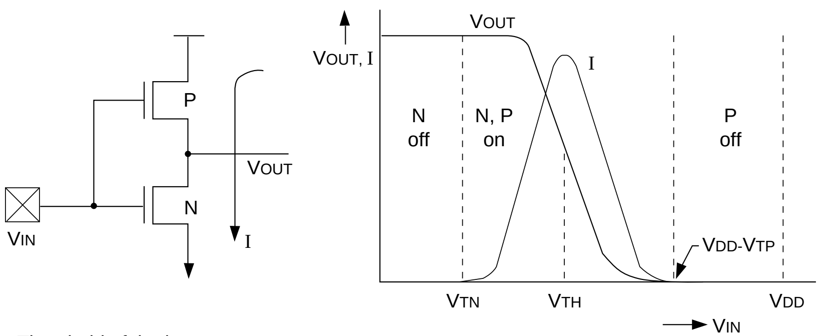
1. If possible, avoid any digital output next to analog inputs.
2. Avoid digital inputs that switch frequently (e.g., clocks) next to analog inputs.
3. If V_{REF} is used, then no analog pin being sampled should exceed V_{REF}.

SUMMARY

The PIC16C71 A/D converter is simple to use. It is versatile and low power.

Authors: Sumit Mitra, Stan D'Souza and Russ Cooper
Logic Products Division

FIGURE 13 - A SIMPLE CMOS INPUT BUFFER:



V_{TH} = Threshold of the inverter

V_{TN} = Device threshold of NMOS pull-down

$-V_{TP}$ = Device threshold of PMOS pull-up

I = On-current (or through current) of the inverter

I_{MAX} = Maximum on-current occurs when $V_{IN} = V_{TH}$. Value of I_{MAX} depends on the sizes of the devices. The larger the devices, the faster the input buffer, and the larger the value of I_{MAX}. Typically, $I_{MAX} \approx 0.2mA – 1mA$.

APPENDIX A - SINGLE CHANNEL A/D (SAD)

```
MPASM 1.00 Released       SAD.ASM   7-15-1994  13:27:21                 PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT

                     0001 ;TITLE   "Single channel A/D (SAD)"
                     0002 ;This program is a simple implementation of the PIC16C71's
                     0003 ;A/D. 1 Channel is selected (CH0).
                     0004 ;The A/D is configured as follows:
                     0005 ;       Vref = +5V internal.
                     0006 ;       A/D Osc. =  internal RC
                     0007 ;       A/D Channel = CH0
                     0008 ;Hardware for this program is the PICDEMO board.
                     0009 ;
                     0010 ;
                     0011        LIST P=16C71,F=INHX8M
                     0012 ;
                     0013    include "picreg.equ"
                     0083
                     0013
                     0014 ;
0010                 0015 TEMP    EQU     10h
0001                 0016 adif    equ     1
0002                 0017 adgo    equu    2
                     0018 ;
                     0019        ORG     0x00
                     0020 ;
                     0021 ;
0000 2810            0022        goto    start
                     0023 ;
                     0024        org     0x04
0004 281C            0025        goto    service_int      ;interrupt vector
                     0026 ;
                     0027 ;
                     0028        org     0x10
                     0029 start
0010 3000            0030        movlw   B'00000000'     ;set port b as
0011 0086            0031        movwf   PORT_B          ;all outputs
0012 0066            0032        tris    PORT_B          ;       /
                     0033 ;
0013 201D            0034        call    InitializeAD
                     0035 update
0014 0809            0036        movf    ADRES,W         ;get a/d value
0015 0086            0037        movwf   PORT_B          ;output to port b
0016 2025            0038        call    SetupDelay      ;setup time >= 10uS.
0017 1088            0039        bcf     ADCON0,adif     ;clear int flag
0018 1508            0040        bsf     ADCON0,adgo     ;start new conversion
                     0041 loop
0019 1888            0042        btfsc   ADCON0,adif     ;a/d done?
001A 2814            0043        goto    update          ;yes then update new value.
001B 2819            0044        goto    loop            ;no then keep checking
                     0045 ;
                     0046 ;no interrupts are enabled, so if the program ever reaches here,
                     0047 ;it should be returned with the global interrupts disabled.
                     0048 service_int
001C 0008            0049        return                  ;do not enable global.
                     0050 ;
                     0051 ;
                     0052 ;
                     0053 ;InitializeAD, initializes and sets up the A/D hardware.
                     0054 ;Select ch0 to ch3 as analog inputs, fosc/2 and read ch3.
                     0055 ;
                     0056 InitializeAD
001D 1683            0057        bsf     STATUS,5        ;select pg1
001E 3000            0058        movlw   B'00000000'     ;select ch0-ch3...
001F 0088            0059        movwf   ADCON1          ;as analog inputs
```

```
0020 1283               bcf     STATUS,5         ;select pg0
0021 30C1               movlw   B'11000001'      ;select:RC,ch0..
0022 0088               movwf   ADCON0           ;turn on A/D.
0023 0189               clrf    ADRES            ;clr result reg.
0024 0008               return
                ;
                ;This routine is a software delay of 10uS for the a/d setup.
                ;At 4Mhz clock, the loop takes 3uS, so initialize TEMp with
                ;a value of 3 to give 9uS, plus the move etc should result in
                ;a total time of > 10uS.
                SetupDelay
0025 3003               movlw   .3
0026 0090               movwf   TEMP
                SD
0027 0B90               decfsz  TEMP
0028 2827               goto    SD
0029 0008               return


                        END



MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : X-X------------- XXXXXXXXXXXXXXXX XXXXXXXXXX------ ----------------
0040 : ---------------- ---------------- ---------------- ----------------

All other memory blocks unused.


Errors   :    0
Warnings :    0
```

APPENDIX B

```
MPASM 1.00 Released     SLPAD.ASM   7-15-1994  13:27:15                 PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT

                     0001
                     0002 ;TITLE   "A/D in Sleep Mode"
                     0003 ;This program is a simple implementation of the PIC16C71's
                     0004 ;A/D feature. This program demonstrates
                     0005 ;how to do a a/d in sleep mode on the PIC16C71.
                     0006 ;The A/D is configured as follows:
                     0007 ;       Vref = +5V internal.
                     0008 ;       A/D Osc. = internal RC
                     0009 ;       A/D Interrupt = OFF
                     0010 ;       A/D Channels = ch 0
                     0011 ;
                     0012 ;The ch0 A/D result is displayed as a 8 bit binary value
                     0013 ;on 8 leds connected to port b. Hardware used is that of
                     0014 ;the PICDEMO board.
                     0015 ;
                     0016 ;
                     0017         LIST P=16C71,F=INHX8M
                     0018 ;
                     0019     include "picreg.equ"
                     0083
                     0019
                     0020 ;
0010                 0021 TEMP    EQU     10h
0001                 0022 adif    equ     1
0002                 0023 adgo    equ     2
                     0024 ;
                     0025 ;
                     0026         ORG     0x00
                     0027 ;
                     0028 ;
0000 2810            0029         goto    start
                     0030 ;
                     0031         org     0x04
0004 281B            0032         goto    service_int      ;interrupt vector
                     0033 ;
                     0034 ;
                     0035         org     0x10
                     0036 start
0010 3000            0037         movlw   B'00000000'     ;make port b all
0011 0086            0038         movwf   PORT_B          ;outputs.
0012 0066            0039         tris    PORT_B          ;       /
                     0040 ;
0013 201C            0041         call    InitializeAD
                     0042 update
0014 0809            0043         movf    ADRES,W
0015 0086            0044         movwf   PORT_B          ;save in table
0016 2025            0045         call    SetupDelay      ;
0017 1088            0046         bcf     ADCON0,adif     ;clr a/d flag
0018 1508            0047         bsf     ADCON0,adgo     ;start new a/d conversion
                     0048 ;
0019 0063            0049         sleep
001A 2814            0050         goto    update          ;wake up and update
                     0051 ;

                     0052 service_int
001B 0008            0053         return                  ;do not enable int
                     0054 ;
                     0055 ;InitializeAD, initializes and sets up the A/D hardware.
                     0056 InitializeAD
001C 1683            0057         bsf     STATUS,5        ;select pg1
001D 3000            0058         movlw   B'00000000'     ;select ch0-ch3...
```

```
001E 0088                   movwf   ADCON1          ;as analog inputs
001F 1283                   bcf     STATUS,5        ;select pg0
0020 30C1                   movlw   B’11000001'     ;select:internal RC, ch0.
0021 0088                   movwf   ADCON0          ;turn on a/d
0022 018B                   clrf    INTCON          ;clear all interrupts
0023 170B                   bsf     INTCON,ADIE     ;enable a/d
0024 0008                   return
                    ;
                    ;This routine is a software delay of 10uS for the a/d setup.
                    ;At 4Mhz clock, the loop takes 3uS, so initialize TEMp with
                    ;a value of 3 to give 9uS, plus the move etc should result in
                    ;a total time of > 10uS.
                    SetupDelay
0025 3003                   movlw   .3
0026 0090                   movwf   TEMP
                    SD
0027 0B90                   decfsz  TEMP
0028 2827                   goto    SD
0029 0008                   return

                    ;

                            END


MEMORY USAGE MAP (‘X’ = Used,  ‘-’ = Unused)

0000 : X—X—————— XXXXXXXXXXXXXXXX XXXXXXXXXX——— ————————
0040 : ———————— ———————— ———————— ————————

All other memory blocks unused.


Errors   :    0
Warnings :    0
```

APPENDIX C

```
MPASM 1.00 Released     INTAD.ASM   7-15-1994  13:27:32                 PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT

                     0001
                     0002 ;TITLE      "Single channel A/D with interrupts"
                     0003 ;This program is a simple implementation of the PIC16C71's
                     0004 ;A/D. 1 Channel is selected (CH0). A/D interrupt is turned on,
                     0005 ;hence on completion of a/d conversion, an interrupt is generated.
                     0006 ;The A/D is configured as follows:
                     0007 ;       Vref = +5V internal.
                     0008 ;       A/D Osc. = internal RC Osc.
                     0009 ;       A/D Interrupt = On
                     0010 ;       A/D Channel = CH0
                     0011 ;
                     0012 ;The A/D result is displayed as a 8 bit value on 8 leds connected
                     0013 ;to port b. Hardware setup is the PICDEMO board.
                     0014 ;
                     0015 ;
                     0016         LIST P=16C71,F=INHX8M
                     0017 ;
                     0018     include "picreg.equ"
                     0083
                     0018
                     0019 ;
0010                 0020 flag    equ     10
0011                 0021 TEMP    equ     11
0000                 0022 adover  equ     0
0001                 0023 adif    equ     1
0002                 0024 adgo    equ     2
0006                 0025 adie    equ     6
0007                 0026 gie     equ     7
0005                 0027 rp0     equ     5
                     0028 ;
                     0029         ORG     0x00
                     0030 ;
                     0031 ;
0000 2810            0032         goto    start
                     0033 ;
                     0034         org     0x04
0004 281C            0035         goto    service_ad      ;interrupt vector
                     0036 ;
                     0037 ;
                     0038         org     0x10
                     0039 start
0010 3000            0040         movlw   B'00000000'     ;init i/o ports
0011 0086            0041         movwf   PORT_B
0012 0066            0042         tris    PORT_B
                     0043 ;
0013 2022            0044         call    InitializeAD
                     0045 update
0014 1010            0046         bcf     flag,adover     ;reset software a/d flag
0015 202B            0047         call    SetupDelay      ;setup delay >= 10uS.
0016 1088            0048         bcf     ADCON0,adif     ;reset a/d int flag (ADIF)
0017 1508            0049         bsf     ADCON0,adgo     ;start new a/d conversion
0018 178B            0050         bsf     INTCON,gie      ;enable global interrupt
                     0051 loop

0019 1810            0052         btfsc   flag,adover     ;a/d over?
001A 2814            0053         goto    update          ;yes start new conv.
001B 2819            0054         goto    loop            ;no then keep checking
                     0055 ;
                     0056 service_ad
001C 1C88            0057         btfss   ADCON0,adif     ;ad interrupt?
001D 0009            0058         retfie                  ;no then ignore
001E 0809            0059         movf    ADRES,W         ;get a/d value
```

```
001F 0086                   movwf   PORT_B          ;output to port b
0020 1410                   bsf     flag,adover     ;a/d done set
0021 0008                   return                  ;do not enable int
                     ;
                     ;
                     ;InitializeAD, initializes and sets up the A/D hardware.
                     ;select ch0 to ch3, RC OSC., a/d interrupt.
                     InitializeAD
0022 1683                   bsf     STATUS,rp0      ;select pg1
0023 3000                   movlw   B'00000000'     ;select ch0-ch3...
0024 0088                   movwf   ADCON1          ;as analog inputs
0025 1283                   bcf     STATUS,rp0      ;select pg0
0026 018B                   clrf    INTCON          ;clr all interrupts
0027 170B                   bsf     INTCON,adie     ;enable a/d int.
0028 30C1                   movlw   B'11000001'     ;select:RC osc,ch0...
0029 0088                   movwf   ADCON0          ;turn on a/d
002A 0008                   return
                     ;
                     ;This routine is a software delay of 10uS for the a/d setup.
                     ;At 4Mhz clock, the loop takes 3uS, so initialize TEMp with
                     ;a value of 3 to give 9uS, plus the move etc should result in
                     ;a total time of > 10uS.
                     SetupDelay
002B 3003                   movlw   .3
002C 0091                   movwf   TEMP
                     SD
002D 0B91                   decfsz  TEMP
002E 282D                   goto    SD
002F 0008                   return
                     ;
                     ;
                            END


MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : X-X------------- XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX ----------------
0040 : ---------------- ---------------- ---------------- ----------------

All other memory blocks unused.


Errors   :    0
Warnings :    0
```

APPENDIX D

```
MPASM 1.00 Released    MULTAD.ASM   7-15-1994  13:27:26                 PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT

                     0001 ;TITLE   “A/D using Multiple Channels”
                     0002 ;This program is a simple implementation of the PIC16C71’s
                     0003 ;A/D feature. This program demonstrates
                     0004 ;how to select multiple channels on the PIC16C71.
                     0005 ;The A/D is configured as follows:
                     0006 ;       Vref = +5V internal.
                     0007 ;       A/D Osc. = internal RC osc.
                     0008 ;       A/D Interrupt = Off
                     0009 ;       A/D Channels = all in a “Round Robin” format.
                     0010 ;       A/D reuslts are stored in ram locations as follows:
                     0011 ;       ch0 –> ADTABLE + 0
                     0012 ;       ch1 –> ADTABLE + 1
                     0013 ;       ch2 –> ADTABLE + 2
                     0014 ;       ch3 –> ADTABLE + 3
                     0015 ;
                     0016 ;The ch0 A/D result is displayed as a 8 bit value on 8 leds
                     0017 ;connected to port b.
                     0018 ;Hardware: PICDEMO board.
                     0019 ;               Stan D’Souza 7/6/93.
                     0020 ;
                     0021         LIST P=16C71,F=INHX8M
                     0022 ;
                     0023     include “picreg.equ”
                     0083
                     0023
                     0024 ;
0010                 0025 TEMP    EQU     10h
0001                 0026 adif    equ     1
0002                 0027 adgo    equ     2
                     0028 ;
0006                 0029 ch2     equ     6
0007                 0030 ch3     equ     7
000C                 0031 flag    equ     0C
0020                 0032 ADTABLE equ     20
                     0033 ;
                     0034         ORG     0x00
                     0035 ;
                     0036 ;
0000 2810            0037         goto    start
                     0038 ;
                     0039         org     0x04
0004 2823            0040         goto    service_int      ;interrupt vector
                     0041 ;
                     0042 ;
                     0043         org     0x10
                     0044 start
0010 3000            0045         movlw   B’00000000'     ;make port b
0011 0086            0046         movwf   PORT_B          ;as all outputs
0012 0066            0047         tris    PORT_B          ;       /
                     0048 ;
0013 2024            0049         call    InitializeAD
                     0050 update
0014 0809            0051         movf    ADRES,W

0015 0080            0052         movwf   0               ;save in table
0016 3020            0053         movlw   ADTABLE         ;chk if ch0
0017 0204            0054         subwf   FSR,W           ;     /
0018 1D03            0055         btfss   STATUS,Z        ;yes then skip
0019 281C            0056         goto    NextAd          ;else do next channel
001A 0809            0057         movf    ADRES,W         ;get a/d value
001B 0086            0058         movwf   PORT_B          ;output to port b
                     0059 NextAd
```

```
001C 202E              call    NextChannel     ;select next channel
001D 203A              call    SetupDelay      ;set up > = 10uS
001E 1088              bcf     ADCON0,adif     ;clear flag
001F 1508              bsf     ADCON0,adgo     ;start new a/d conversion
                  loop
0020 1888              btfsc   ADCON0,adif     ;a/d done?
0021 2814              goto    update          ;yes then update
0022 2820              goto    loop            ;wait till done
                  ;
                  service_int
0023 0008              return                  ;do not enable int
                  ;
                  ;
                  ;InitializeAD, initializes and sets up the A/D hardware.
                  InitializeAD
0024 1683              bsf     STATUS,5        ;select pg1
0025 3000              movlw   B’00000000'     ;select ch0-ch3...
0026 0088              movwf   ADCON1          ;as analog inputs
0027 1283              bcf     STATUS,5        ;select pg0
0028 30C1              movlw   B’11000001'     ;select:fosc/2, ch0.
0029 0088              movwf   ADCON0          ;turn on a/d
002A 3020              movlw   ADTABLE         ;get top of table address
002B 0084              movwf   FSR             ;load into indirect reg
002C 0189              clrf    ADRES           ;clr result reg.
002D 0008              return
                  ;
                  ;NextChannel, selects the next channel to be sampled in a
                  ;”round-robin” format.
                  NextChannel
002E 3008              movlw   0x08            ;get channel offset
002F 0788              addwf   ADCON0          ;add to conf. reg.
0030 1288              bcf     ADCON0,5        ;clear any carry over
                  ;increment pointer to correct a/d result register
0031 0190              clrf    TEMP
0032 1988              btfsc   ADCON0,3        ;test lsb of chnl select
0033 1410              bsf     TEMP,0          ;set if ch1 or ch3
0034 1A08              btfsc   ADCON0,4        ;test msb of chnl select
0035 1490              bsf     TEMP,1          ;set if ch0 or ch2
0036 3020              movlw   ADTABLE         ;get top of table
0037 0710              addwf   TEMP,W          ;add with temp
0038 0084              movwf   FSR             ;allocate new address
0039 0008              return
                  ;
                  ;This routine is a software delay of 10uS for the a/d setup.
                  ;At 4Mhz clock, the loop takes 3uS, so initialize TEMp with
                  ;a value of 3 to give 9uS, plus the move etc should result in
                  ;a total time of > 10uS.
                  SetupDelay
003A 3003              movlw   .3
003B 0090              movwf   TEMP
                  SD
003C 0B90              decfsz  TEMP
003D 283C              goto    SD
003E 0008              return

                  ;

                       END


MEMORY USAGE MAP (‘X’ = Used,  ‘-’ = Unused)

0000 : X—X————— XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXX-
0040 : ———————— ———————— ———————— ————————

All other memory blocks unused.

Errors   :    0
Warnings :    0
```

WORLDWIDE SALES & SERVICE

AMERICAS

Corporate Office
Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 602 786-7200 Fax: 602 786-7277
Technical Support: 602 786-7627
Web: http://www.mchip.com/microhip

Atlanta
Microchip Technology Inc.
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770 640-0034 Fax: 770 640-0307

Boston
Microchip Technology Inc.
5 Mount Royal Avenue
Marlborough, MA 01752
Tel: 508 480-9990 Fax: 508 480-8575

Chicago
Microchip Technology Inc.
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 708 285-0071 Fax: 708 285-0075

Dallas
Microchip Technology Inc.
14651 Dallas Parkway, Suite 816
Dallas, TX 75240-8809
Tel: 214 991-7177 Fax: 214 991-8588

Dayton
Microchip Technology Inc.
35 Rockridge Road
Englewood, OH 45322
Tel: 513 832-2543 Fax: 513 832-2841

Los Angeles
Microchip Technology Inc.
18201 Von Karman, Suite 455
Irvine, CA 92715
Tel: 714 263-1888 Fax: 714 263-1338

New York
Microchip Technology Inc.
150 Motor Parkway, Suite 416
Hauppauge, NY 11788
Tel: 516 273-5305 Fax: 516 273-5335

AMERICAS (continued)

San Jose
Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408 436-7950 Fax: 408 436-7955

ASIA/PACIFIC

Hong Kong
Microchip Technology
Unit No. 3002-3004, Tower 1
Metroplaza
223 Hing Fong Road
Kwai Fong, N.T. Hong Kong
Tel: 852 2 401 1200 Fax: 852 2 401 3431

Korea
Microchip Technology
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku,
Seoul, Korea
Tel: 82 2 554 7200 Fax: 82 2 558 5934

Singapore
Microchip Technology
200 Middle Road
#10-03 Prime Centre
Singapore 188980
Tel: 65 334 8870 Fax: 65 334 8850

Taiwan
Microchip Technology
10F-1C 207
Tung Hua North Road
Taipei, Taiwan, ROC
Tel: 886 2 717 7175 Fax: 886 2 545 0139

EUROPE

United Kingdom
Arizona Microchip Technology Ltd.
Unit 6, The Courtyard
Meadow Bank, Furlong Road
Bourne End, Buckinghamshire SL8 5AJ
Tel: 44 0 1628 851077 Fax: 44 0 1628 850259

France
Arizona Microchip Technology SARL
2 Rue du Buisson aux Fraises
91300 Massy - France
Tel: 33 1 69 53 63 20 Fax: 33 1 69 30 90 79

Germany
Arizona Microchip Technology GmbH
Gustav-Heinemann-Ring 125
D-81739 Muenchen, Germany
Tel: 49 89 627 144 0 Fax: 49 89 627 144 44

Italy
Arizona Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Pegaso Ingresso No. 2
Via Paracelso 23, 20041
Agrate Brianza (MI) Italy
Tel: 39 039 689 9939 Fax: 39 039 689 9883

JAPAN

Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shin Yokohama
Kohoku-Ku, Yokohama
Kanagawa 222 Japan
Tel: 81 45 471 6166 Fax: 81 45 471 6122

9/22/95