The Analog-to-Digital converter (A/D) is the primary tool that allows analog signals to be quantized into the world of digital electronics. Once the signal is digitally represented, it can be stored, analyzed and manipulated by a variety of logic devices. The PIC16C7X microcontrollers have an A/D integrated onto the PIC16CXX core processor. Utilizing Microchip's A/D requires only a basic level of understanding to get a result. However, maximizing the effectiveness of the A/D for each specific application requires a higher level of thought and understanding. Typically, a thorough comprehension of a device is obtained through experience, studying data sheets and studying application notes for a reasonable amount of time. This article addresses the main technical considerations for an effective design to reduce your design time.

When detailing how to maximize the effectiveness of any design, we look at the weakest (and strongest) elements. Please DO NOT misinterpret the statements detailing weaknesses to be below our specification. Our part specifications are based upon the worst case. All suggestions should improve on the worst case specifications.

We'll start by covering the A/D Basics, then dive into the other three general technical categories, which are essential for an effective design:

- Speed
- Accuracy
- Power Usage

**BASICS**

**Specifications**

Microchip’s A/D is a successive approximation A/D which uses a bank of internal capacitors totalling 51.2 pF. The maximum resolution of the A/D is 8-bits. The converter accuracy specification is ±1 bit, but that can be made better or worse by your design. The analog channels are multiplexed to the converter which means that only one analog channel can be sampled at a time. The conversion time and maximum sampling frequency is application specific.

**General**

Even though we are converting to the ‘digital world’, we must remember that certain analog ‘laws’ still hold true. Specifically, the A/D is basically a capacitor which has to have time to charge/discharge to the analog level on the I/O pin before a conversion can be started. Source impedance and internal impedances add up to give an effective resistance in series with the capacitor(s). This RC time constant determines the minimum amount of charge/discharge time to achieve a desired accuracy. This time is the minimum tracking time (referred to as the minimum sampling time in our data book). Once the minimum time has been met, the voltage on the capacitor will track the voltage on the I/O pin until a conversion is started. Once the conversion has begun, the I/O pin is internally disconnected from the capacitor. The voltage on the capacitor is held constant for the entire conversion process. This type of sampling circuit is referred to as a "track and hold".

**Step by Step**

1. In order to perform a conversion, you must enable the A/D by setting the ADON bit (ADCON0 register).
2. Select the channel to be sampled by setting/clearing the CHS2, CHS1, and CHS0 bits (ADCON0 register). See the ADCON0 register’s bit descriptions in the device’s data sheet for the actual bit combinations for the selected channel(s).
3. The default input setup of all analog I/O pins is analog, not digital. For I/O pins set up as analog, the digital input buffers are internally disconnected. This is to keep analog voltages off of the CMOS input buffer. If the input voltage on a digital I/O was 0.5 VDD for example, then both the PMOS and NMOS cells which make up the CMOS input buffer would be turned on. This causes the input buffer to draw around 100 to 150 µA.

If you are using an analog I/O as a digital input, load the register ADCON1 with the correct value to change the default from analog to digital. The I/O pin’s bit will always read as a 0 when configured as an analog input (since it is disconnected from the pin). See the ADCON1 register’s bit descriptions in the device’s data sheet for the actual bit combinations for the selected channel(s).
Step by Step (cont'd)

4. Wait for at least the minimum tracking time. The tracking time is the time required to charge a 51.2 pF capacitor (located internal of the PIC16C7X) to the voltage level of the selected channel. The tracking time is made up of three components:
   - Amplifier Settling Time
   - Holding Capacitor Charging Time
   - Temperature Coefficient

\[ T_{\text{TRACKING}} = T_{\text{AMP}} + T_{\text{CHARGE}} + T_{\text{TEMP}} \]

Given: \( T_{\text{AMP}} = 5 \mu s \)

Given: \( T_{\text{TEMP}} = (T_{\text{Temp}} - 25\,\text{°C})(0.05 \mu s/\text{°C}) \), if \( T_{\text{Temp}} > 25\,\text{°C} \)
\( T_{\text{TEMP}} = 0 \), if \( \leq 25\,\text{°C} \)

**T\text{CHARGE} Equation Derivation:**

**Basic Capacitor Charge Equation**

\[ V_{\text{CAP}} = V_{\text{FINAL}} - (V_{\text{FINAL}} - V_{\text{INITIAL}}) e^{-\frac{T}{RC}} \]

or as it applies

\[ V_{\text{HOLDCAP}} = V_{\text{REF}} - (V_{\text{REF}} - 0) e^{-\frac{T}{RCHOLD}} \]

Where:
- \( R = R_{\text{IC}} + R_{\text{SS}} + R_{s} \)
- \( R_{s} = \text{Source Impedance} \)
- \( R_{\text{IC}} = \text{Internal Interconnect Impedance} \)
- \( R_{\text{SS}} = \text{Internal Sampling Switch Impedance} \)

(see PIC16/17 Data Book, Section 13.1 for values)

\( \text{CHOLD} = 51.2 \text{ pF} \)

\[ V_{\text{HOLDCAP}} = V_{\text{REF}} - \frac{V_{\text{REF}}}{e^{\text{ERROR}}} \]

Therefore:

\[ \left( V_{\text{REF}} - \frac{V_{\text{REF}}}{e^{\text{ERROR}}} \right) = V_{\text{REF}} - V_{\text{REF}} e^{\left( R_{\text{IC}} + R_{\text{SS}} + R_{s} \right) \text{CHOLD}} \]

\[ V_{\text{REF}} - \frac{1}{\text{ERROR}} = V_{\text{REF}} e^{\left( R_{\text{IC}} + R_{\text{SS}} + R_{s} \right) \text{CHOLD}} \]

\[ \frac{1}{\text{ERROR}} = e^{\left( R_{\text{IC}} + R_{\text{SS}} + R_{s} \right) \text{CHOLD}} \]

\[ T_{\text{CHARGE}} = -(R_{\text{IC}} + R_{\text{SS}} + R_{s}) \text{CHOLD} \ln \left( \frac{1}{\text{ERROR}} \right) \]

Reducing this time can significantly reduce the accuracy of the result.

**EXAMPLE 1:**

\[
\begin{align*}
V_{\text{DD}} &= V_{\text{REF}} = 5V \\
R_{\text{IC}} &= 1\,k\Omega \\
R_{\text{SS}} &= 7\,k\Omega \\
R_{s} &= 10\,k\Omega \\
\text{Error} &= 8\text{ bits } \pm 1/2 = 2^{9} = 512 \\
T_{\text{CHARGE}} &= -(1\,k\Omega + 7\,k\Omega + 10\,k\Omega)(51.2\text{ pF}) \ln 1/512 = 5.75\,\mu s
\end{align*}
\]

**EXAMPLE 2:**

Same values as Example 1, except we put a .01 \( \mu \)F decoupling capacitor on the I/O pin. This effectively reduces \( R_{s} \) to the impedance of the capacitor:

\( R_{s} \approx 50 \Omega \)

\[ T_{\text{CHARGE}} = -(1\,k\Omega + 7\,k\Omega + 50\,\Omega)(51.2\text{ pF}) \ln 1/512 = 2.57\,\mu s \]

5. Choosing the A/D clock source (\( T_{\text{AD}} \)) determines the time required for the conversion of each bit. There are four choices for the clock source: \( F_{\text{OSC}}/2 \), \( F_{\text{OSC}}/8 \), \( F_{\text{OSC}}/32 \) and \( \text{FRC} \). Bits \( \text{ADCS1} \) and \( \text{ADCS0} \) (\( \text{ADCON0} \) register) determine the clock source.

Table 1 will help you to choose the clock source according to your oscillator frequency.

**TABLE 1: DEVICE OSCILLATOR FREQUENCY BY CLOCK SOURCE**

<table>
<thead>
<tr>
<th></th>
<th>( F_{\text{osc/2}} )</th>
<th>( F_{\text{osc/8}} )</th>
<th>( F_{\text{osc/32}} )</th>
<th>( \text{Frc} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{PIC16C71} )</td>
<td>( \leq 1 \text{ MHz} )</td>
<td>( \leq 4 \text{ MHz} )</td>
<td>( \leq 16 \text{ MHz} )</td>
<td>any ( F )</td>
</tr>
<tr>
<td>( \text{PIC16C70/71A/72/73/74} )</td>
<td>( \leq 1.25 \text{ MHz} )</td>
<td>( \leq 5 \text{ MHz} )</td>
<td>( \leq 20 \text{ MHz} )</td>
<td>any ( F )</td>
</tr>
</tbody>
</table>

**Note:** \( \text{FRC} \) assigns \( T_{\text{AD}} \) to an internal RC oscillator which is typically 4 \( \mu \)s (min = 2 \( \mu \)s, max = 6 \( \mu \)s).

To calculate the precise \( T_{\text{AD}} \) period you have chosen:

\[ T_{\text{AD}} = \frac{x}{F_{\text{OSC}}} \]

where \( x = \text{either 2, 8 or 32} \)

6. After setting up the A/D and waiting the minimum tracking time, set the GO/DONE bit (\( \text{ADCON0} \) register) to start the conversion.

7. Either poll for the GO/DONE bit to be cleared, poll for the ADIF to be set, or enable interrupts and wait for the interrupt to happen. When any one of these events occurs, the conversion is finished and the result is located in the ADRES register.

8. The A/D will begin tracking the analog signal again 2 \( T_{\text{AD}} \) (\( T_{\text{RESET}} \)) after the conversion has completed.

9. The time necessary to take an analog-to-digital sample is the total of tracking time plus conversion time. For repetitive samples, \( T_{\text{RESET}} \) is added to the total time. Use this equation to calculate the max. repetitive sampling rate (\( F_{\text{SAMPLING}} \)).

\[
\frac{1}{F_{\text{SAMPLING}}} = \frac{T_{\text{SAMPLE}}}{T_{\text{RESET}}} = T_{\text{TRACKING}} + T_{\text{CONVERSION}} + T_{\text{RESET}}
\]
SPEED

If your design will require an A/D sample time less than 60 µs, then you will want to consider the following factors: tracking time, conversion time, and minimum acceptable precision.

When choosing the clock source to get the fastest conversion time, we want to get a TAD as close to (but not less than) 2 µs for a PIC16C71 and 1.6 µs for a PIC16C70, PIC16C71A, PIC16C72, PIC16C73 or PIC16C74.

1. What is the minimum precision your design requires (i.e., 4-bit, 6-bit or 8-bit)? This is an important question, since we can accelerate the conversion after the desired accuracy has been achieved. We cannot interrupt or abort the conversion after achieving the desired accuracy, since the result is not written to ADRES until the entire 8 bits have been converted (regardless of the accuracy of each bit). Figure 1 illustrates the timing required to determine each bit. Since the A/D operates synchronously with respect to the internal instruction cycle (4/Fosc), we can accurately determine when to select a faster TAD (Fosc/2). The Least Significant bits determined after the conversion has been accelerated will be unreliable and most likely inaccurate. The following calculation determines the minimum amount of time before you can change the TAD clock select bits for the resolution required.

\[
T_{CONVERSION} = 1.5 \cdot TAD + N \cdot TAD + (8-N)(2 \cdot TOSC)
\]

(\text{where } N = \text{Number of bits of resolution})

FIGURE 1: A/D CONVERSION TIMING

Note 1: If the A/D clock source is selected as RC, a time of TCY is added before the A/D clock starts. This allows the SLEEP instruction to be executed.
TABLE 2: A/D CONVERSION REQUIREMENTS*

<table>
<thead>
<tr>
<th>Param. #</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ. †</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>130</td>
<td>TAD</td>
<td>A/D Clock Period</td>
<td>1.6</td>
<td></td>
<td></td>
<td>µs</td>
<td>VREF ≥ 3.0V</td>
</tr>
<tr>
<td>130</td>
<td>TAD</td>
<td>A/D Internal RC Oscillator source</td>
<td>3.0</td>
<td>6.0</td>
<td>9.0</td>
<td>µs</td>
<td>ADCl1 = 11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2.0</td>
<td>4.0</td>
<td>6.0</td>
<td>µs</td>
<td>(RC oscillator source)</td>
</tr>
<tr>
<td>131</td>
<td>Tcnv</td>
<td>Conversion Time (not including S/H time) (Note 1)</td>
<td>—</td>
<td>9.5 TAD</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>132††</td>
<td>TSMP</td>
<td>Sampling Time</td>
<td>5</td>
<td>—</td>
<td>—</td>
<td>µs</td>
<td>The minimum time is the amplifier settling time. This may be used if the “new” input voltage has not changed by more than 1 LSb (i.e., 20 mV @ 5.12V) from the last sampled voltage (as stated on CHOLD).</td>
</tr>
</tbody>
</table>

* These parameters are characterized but not tested.
† Data in “Typ” column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.
†† TTRACK = TSMP. Parameter #132 is TSMP Sampling Time.

Note 1: ADRES register may be read on the following TCY cycle.

EXAMPLE 3: A STANDARD 8-BIT (±1 LSb) FAST CONVERSION

Processor = PIC16C74
Oscillator = 20 MHz
Max possible TAD = 1.6 µs
TAD clock source = Fosc/32 ∴ TAD = 1.6 µs
TTRACKING = (5 µs + 0 µs + 2.57 µs) = 7.65 µs [from Example 2]
TCONVERSION = 1.5 • 1.6 µs + 8 • 1.6 µs + (8 - 8)(2 • 50 ns) = 15.2 µs
TAD_SAMPLE = 15.2 µs + 7.6 µs + 2 • 1.6 µs = 26 µs
FMAX_A/D_SAMPLING_RATE = 38.4 kHz

EXAMPLE 4: A 4-BIT (±1/32 LSb) FAST CONVERSION

Processor = PIC16C71
Oscillator = 16 MHz
Max possible TAD = 2.0 µs
TAD clock source = Fosc/32 ∴ TAD = 2.0 µs
TTRACKING = (5 µs + 0 µs + 2.57 µs) = 7.65 µs [from Example 2]
TCONVERSION = 1.5 • 2.0 µs + 4 • 2.0 µs + (8 - 4)(2 • 62.5 ns) = 11.5 µs
TAD_SAMPLE = 11.5 µs + 7.6 µs + 2 • 2.0 µs = 23.1 µs
FMAX_A/D_SAMPLING_RATE = 43.3 kHz

2. Executing a few instructions while the previous conversion is still running will allow ‘overhead’ code to be executed during the tracking time and conversion time. Don’t worry about using the data from the A/D immediately after the GO bit has been cleared. The ADRES register will contain the previous A/D result until it is updated with the new result. (The GO bit is cleared at that time.) In order to ensure that the sampling time is started as quickly as possible, set up the A/D channel for the next conversion (ADCON1) after the current tracking has stopped and before the current conversion is done (this will not affect the current conversion).
ACCURACY

Every designer should understand what effects the accuracy of the A/D. The following techniques may improve your designs. A meticulous and careful design can achieve approximately ±1 LSB.

1. A critical factor in our A/D’s accuracy is the power supply to the PIC16C7X. Noise or ripple on VDD can adversely affect the conversion’s accuracy (the degree to which the accuracy is affected is dependent upon the amplitude of the noise). To improve the supply, use common filtering techniques, such as decoupling capacitors for various frequencies.

2. VREF can pull up to 1 mA for 8 to 20 ns when a conversion is started. The VREF source (either RA3 or VDD) doesn’t change the current draw. Make sure your design can source this current (and be able to do it very fast). Capacitors on VREF will act as a source for this current draw.

3. VDD is a better VREF source than RA3 because if noise is present on VDD, it is absolutely coupled, NOT linearly coupled.

EXAMPLE 5:

20 mV of noise when VDD = 5.0V = VREF is ±1 LSB
20 mV of noise when RA3 = 3.0V = VREF is ±1 LSB

4. Noise on the A/D input channel during tracking will add error to the desired result. A properly sized filter capacitor on the input pin will help correct this.

Note: Remember that capacitors are not ideal and are really band-pass, not low-pass, filters (i.e., more than one of different values may be necessary).

5. Increasing the tracking time (TTRACKING) > 5 µs produces a measurable increase in accuracy. 20 µs is generally a good length of tracking time before starting a conversion.

6. Increase TAD to about 4 µs. This allows the internal comparator response time to increase with less overdrive error. Notice that Trc is typically 4 µs.

7. Operate at room temperature (25°C), since a high temperature increases leakage from the sample capacitor and low temperatures shift threshold levels. If you know that high temperature operation is possible, keep TAD as close to the minimum as possible.

8. Do not change the value on any I/O pins which are configured as output while a conversion is in progress (regardless of the port). The high source/sink capability of the pins could directly inject noise onto VDD.

9. The A/D is capable of finishing a conversion while the part is sleeping. Putting the part to SLEEP helps accuracy, since this mode eliminates the internal switching noise of the processor core.

Note: You can do this with any Oscillator mode, but you will have to allow for the osc wake-up time when using LP, XT or HS modes. RC mode has no start-up timer and starts up instantly.

EXAMPLE 6: (PIC16C71 AND PIC16C70)

If you put the processor to SLEEP, consider clearing all interrupt enable bits (including the GIE bit), except for the ADIE. This will cause the processor to wake-up from SLEEP when the conversion is complete and resume executing code with the instruction following the SLEEP instruction. This method helps simplify your Interrupt Service Routine for other peripherals.

Note: On the PIC16C71 and PIC16C70, RA0 is right next to OSC1 and cannot help but pick up some noise. If possible, make this pin an output to help isolate the noise from getting to RA1.
POWER USAGE

The following simple, straightforward pointers can help any application meet its power requirements.

1. Keeping the A/D turned off until you are ready to start tracking will reduce approximately 180 µA of current draw from your system. The ADON (ADCON0) bit turns the A/D on or off.

2. Configure all analog inputs to be analog since this will disconnect the CMOS digital input buffer. If you do not, and the signal on the pin is 0.5 VDD, both transistors in the CMOS input buffer will be turned on, causing the maximum current draw of the input buffer.

3. You may want to put the controller to SLEEP during the conversion. There are several factors to consider for this decision, here are a few:
   - The FRC move must be selected for A/D conversion clock.
   - Do you have any timers running that cannot be stopped?
   - Is the wake-up from SLEEP time longer than the conversion time?
   - Can you afford to lose the processing time while sleeping and waking up from SLEEP?

Keeping the A/D Usage Simple

- Put a decoupling capacitor on VDD.
- Put a .01 µF capacitor on the analog input source.
- Select which pins are digital and which pins are analog (register ADCON1).
- Using the internal RC oscillator to determine the TAD provides a relatively fast and accurate conversion for most applications (bits ADCS1, ADCS0 = 1,1 in register ADCON1). If you are using RC mode for the core’s oscillator, then you should put the part to SLEEP during the conversion, or choose an appropriate TAD source other than RC.
- Select which channel is to be sampled (bits CHS2, CHS1 and CHS0 in register ADCON0).
- Turn on the A/D by setting the ADON bit (register ADCON0).
- Wait 20 µs (tracking time).
- Set the GO bit (register ADCON0) to start the conversion.
- Poll either the ADIF or the GO bit to determine when the conversion has completed.
- Utilize the result located in the ADRES register.
- Clear the ADIF bit.

EXAMPLE 7:

```assembly
BSF STATUS,RP0
MOVLW 0xFE ;MAKE RA0 AN OUTPUT
         ;AND RA1 AN INPUT
         MOVLW 0xFF
         MOVWF TRISA
         CLRF ADCON1 ;MAKE ANALOG INPUTS
         MOVLW 0C9h
         MOVWF ADCON0,W
         CALL 20US_DELAY
         GIE_OFF
         BCF INTCON,GIE ;MAKE SURE BCF WAS NOT INTERRUPTED
         GOTO GIE_OFF
         AD_GOING
         BTFSC ADCON0,GO
         GOTO AD_GOING
         ;Result is located in ADRES
```
Note the following details of the code protection feature on PICmicro® MCUs.

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable”.
- Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip’s products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, FilterLab, KEELOQ, microD, MPLAB, PIC, PICmicro, PICMASTER, PICSTART, PRO MATE, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

dsPIC, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, MXDEV, PICC, PICDEM, PICDEM.net, rIPIC, Select Mode and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A.

Serialized Quick Term Programming (SQTP) is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2002, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

Printed on recycled paper.