Prior to Le16

Analog-to-Digital Conversions

Analog-to-Digital Conversions (ADC)
The ADC hardware peripheral allows you to read an analog voltage.

Digital pins can be set as inputs or output using TRIS. Analog pins are ONLY inputs.

There are 13 pins that can be used for Analog inputs but really only one can be used at a time.

Syntax is completely different
2.2 A/D CONVERTER FUNCTIONS

The A/D peripheral is supported with the following functions:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BusyADC</td>
<td>Is A/D converter currently performing a conversion?</td>
</tr>
<tr>
<td>CloseADC</td>
<td>Disable the A/D converter.</td>
</tr>
<tr>
<td>ConvA DC</td>
<td>Start an A/D conversion.</td>
</tr>
<tr>
<td>OpenADC</td>
<td>Configure the A/D converter.</td>
</tr>
<tr>
<td>ReadADC</td>
<td>Read the results of an A/D conversion.</td>
</tr>
<tr>
<td>SetChAADC</td>
<td>Select A/D channel to be used.</td>
</tr>
</tbody>
</table>

Excellent, easy to use functions

Looks complex, but it isn’t...

// configure A/D converter
// config 1 = Setup the timing to a conservative value (you don’t need to ever change this)
// config 2 = Use channel 8, not interrupts off, use the power and ground as references
// portconfig = b0b1 setup only analog 0 as a possible analog input pin

OpenADC:ADC_F0C_8 = ADC_RIGHT_JUST = ADC_12_TAD,
ADC_CMD = ADC_INT_OFF = ADC_REF_VDB_V08,
00E6;

REGISTER 16.2: ADCON REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:6</td>
<td>Unimplemented. Read as '0'</td>
</tr>
<tr>
<td>5</td>
<td>VREFH (Voltage Reference Configuration bit (inter-source))</td>
</tr>
<tr>
<td>4</td>
<td>VREFL (Voltage Reference Configuration bit (inter-source))</td>
</tr>
<tr>
<td>3:0</td>
<td>VCO bit (4 bits)</td>
</tr>
</tbody>
</table>
Panel 9

**ADC is a four step program**

### 2.2 A/D CONVERTER FUNCTIONS

The A/D peripheral is supported with the following functions:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BusyADC</td>
<td>Is A/D converter currently performing a conversion?</td>
</tr>
<tr>
<td>ClrChADC</td>
<td>Disable the A/D converter.</td>
</tr>
<tr>
<td>StartADC</td>
<td>Start an A/D conversion.</td>
</tr>
<tr>
<td>OpenADC</td>
<td>Configure the A/D converter.</td>
</tr>
<tr>
<td>ReadADC</td>
<td>Read the results of an A/D conversion.</td>
</tr>
<tr>
<td>SetChADC</td>
<td>Select A/D channel to be used.</td>
</tr>
</tbody>
</table>

Panel 10

**Four step program:**

1. Select the Analog channel (the pin you want to read)
2. Start the ADC conversion
3. Wait for it to finish
4. Read the ADC result (and store it to a variable)

```c
SetChADC( ADC_CH0 ); // Select the pin
ConvertADC();         // Start conversion
while( BusyADC() );   // Wait for completion
RA0result = ReadADC(); // Read result
```

Panel 11

**Circle what will change and explain...**

```c
// configure A/D converter
// config = a list of a conservative raise you don't need to ever change this!
// preconfigure = configure only the enabling pin

OpenADC(ADC_CHANNEL0, ADC_INPUT_RA0, ADC_INTODULE, ADC_INT_DURATION, ADC_INPUT_LEVEL, ADC_INPUT_MODE);
```

Panel 12

**What are we gonna do in class?**

Much like the Timers example code is on-line

**Steps:**

- **Watch the program run**

  Modify the code to monitor RB0 as an analog input
  (even though it’s a button you can read it analog)

- You could use interrupts, but I rarely do so...

  We’re gonna skip interrupts with the ADC