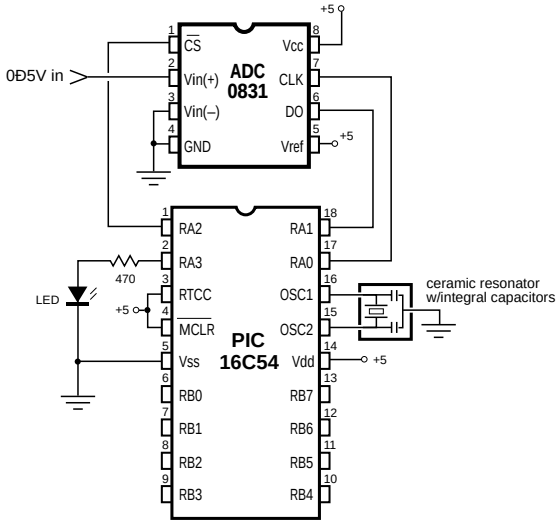
An External A/D Convertor

Introduction. This application note shows how to interface an 8-bit serial analog-to-digital converter to 16C5x-series PICs.

Background. Although the PIC 16C71 offers a built-in analog-to-digital converter (ADC), some users may prefer an external device. One that's particularly suited to use with the PIC is the National Semiconductor ADC0831, available from Digi-Key, among others.

Interfacing the 831 requires only three input/output lines, and of these, two can be multiplexed with other functions (or additional 831s). Only the chip-select (CS) pin requires a dedicated line. The ADC's range of input voltages is controlled by the V_{REF} and $V_{IN}(-)$ pins. V_{REF} sets the voltage at which the ADC will return a full-scale output of 255, while $V_{IN}(-)$ sets the voltage that will return 0.



In the example application, $V_{IN}(-)$ is at ground and V_{REF} is at +5; however, these values can be as close together as 1 volt without harming accuracy or linearity. You may use diode voltage references or trim pots to set these values.

How it works. The sample program reads the voltage at the 831's input pin and uses the eight-bit result to control a timing routine that flashes an LED. The LED flashes slowly when the input is 5 volts, and very rapidly as it approaches ground.

The subroutine *convert* handles the details of getting data out of the ADC. It enables the ADC by pulling the CS line low, then pulses the clock (CLK) line to signal the beginning of a conversion. The program then enters a loop in which it pulses CLK, gets the bit on pin *data*, and shifts it into the received byte using the rotate left (*rl*) instruction. Remember that *rl* affects not only the byte named in the instruction, but the carry bit as well.

When all bits have been shifted into the byte, the program turns off the ADC by returning CS high. The subroutine returns with the conversion result in the variable *ADresult*. The whole process takes 74 instruction cycles.

Modifications. You can add more 831s to the circuit as follows: Connect each additional ADC to the same clock and data lines, but assign separate CS pins. Modify *convert* to take the appropriate CS pin low when it needs to acquire data from a particular ADC. That's it.

Program listing. This program may be downloaded from our Internet ftp site at ftp.tech-tools.com. The ftp site may be accessed directly or through our web site at http://www.tech-tools.com.

; PROGRAM: AD831.SRC

```
; Program demonstrates the use of an external serial ADC (National ADC0831) with
; PIC16C5x-series controllers. A variable dc voltage (0-5V) at pin 2 of the ADC
; controls the blinking rate of an LED connected to PIC pin ra.3.

; Remember to change device info when programming part.
                device      pic16c54,xt_osc,wdt_off,protect_off
                reset       start

clk             =           ra.0
data            =           ra.1
CS              =           ra.2
LED             =           ra.3
```

```
; Put variable storage above special-purpose registers.
                org     8

counter1        ds      1
counter2        ds      1
ADresult        ds      1

; Set starting point in program ROM to zero
                org     0
start           mov     ra, #00000100b      ; Set CS to disable ADC for now.
                mov     !ra,#00000010b      ; Make ra.1 (data) input, rest
                                            ; outputs.
                mov     !rb,#00000000b      ; Make rb output.

blink           xor     ra, #8              ; Invert bit3 of ra (LED).
                call    wait                ; Delay depends on ADC data.
                goto    blink               ; Endless loop.

wait            call    convert             ; Get AD result and put into
                                            ; counter2.
                mov     counter2,ADresult
                add     counter2,#1         ; Add 1 to avoid underflow when
                                            ; AD=0.

; Time delay produced by loop depends on value of ADresult. Higher values
; produce longer delays and slower blinking.
:loop           djnz    counter1, :loop
                djnz    counter2, :loop
                ret

convert         clrb    CS                  ; Enable the ADC.
                mov     counter2,#8         ; Set up for eight data bits.
                setb    clk                 ; Pulse the clock line (insert
                nop                         ; nop if PIC clock >4 MHz).
                clrb    clk
                clr     ADresult            ; Clear byte to make way for new
                                            ; data.
:loop           setb    clk                 ; Pulse the clock line (insert nop
                nop                         ; if PIC clock >4 MHz).
                clrb    clk
                movb    c,data              ; Move data bit into carry.
                rl      ADresult            ; Rotate carry into byte.
                djnz    counter2,:loop      ; Eight bits collected? If not, loop.
                setb    CS                  ; End the conversion.
                ret
```