Generate an analog signal with a µC

THOMAS SCHMIDT, MICROCHIP TECHNOLOGY, CHANDLER, AZ

Applications requiring D/A conversion abound, including dual-tone generation, motor-speed control, and offset-voltage generation for a sensor or for battery charging. Most designers believe the D/A converter must be either an integrated module in a µC or an external component; however, a simpler approach is possible. You can generate an analog signal by using a low-cost µC, thereby eliminating the need for external components and thus reducing board space and overall system cost. The RC network in **Figure 1** provides an easy way to convert a digital signal into an analog voltage. The RC network, a lowpass filter, connects to an I/O pin of the µC.

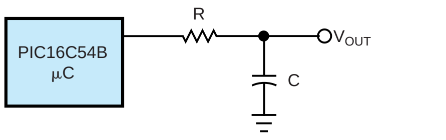
To generate an analog signal, the µC charges the capacitor via the resistor. The µC uses PWM to charge the capacitor. The voltage across the capacitor is the analog voltage. When the PWM signal is high, the capacitor charges. When the PWM signal is low, the capacitor discharges. By varying the duty cycle, you can generate a sine wave or any other analog signal. The µC in **Figure 1** is a low-cost, 8-bit RISC controller. The PWM signal, generated in software, drives the RC lowpass filter connected to one of the µC's I/O lines.

Listing 1 gives the software code. The PWM routine requires only three general-purpose registers. One register contains the value of the period; the other contains the duty cycle. The program starts by initializing the PWM output pin and the period and duty-cycle registers. The initial duty cycle is 50%. It's assumed in the initialization routine that the RC network connects to pin RA1 of the µC. After initialization, the main subroutine calls the routine in which the PWM signal is generated. In this example, the main routine calls only the PWM_Signal routine. You could easily implement other functions—a keypad or a seven-segment display, for example—just by adding call instructions for subroutines.

The PWM implementation requires a software counter. The register counter stores the software counter. The counter increments each time the program calls the PWM_Signal routine. Each time the counter increments, the program checks to see if the register's value is greater than or equal to the duty cycle. If this condition is true, the program sets the value at the port pin to logic 0. This action signifies that the time for the duty cycle has elapsed. After this time elapses, the routine checks to see if the time for the period is over. If the value of the counter is less than the value of the duty cycle, the PWM signal remains high.

If the value of the counter is greater than the value of the duty cycle, the program compares the counter with the value of the period register. If the value of the counter equals the value of the period register, the period for the PWM signal is over, and the next period starts. The performance of this PWM implementation depends on the number of times the program calls the PWM function from the main routine—the more calls, the higher the resolution. To generate a sine wave,

FIGURE 1



A simple RC network connected to a low-cost µC's output can generate analog signals of any desired resolution.

LISTING 1—ROUTINE FOR ANALOG-SIGNAL GENERATION

```
                          list p=pic16c54B, r=hex

                          #include <p16c5x.inc>
                          #define PWM_PORT    PORTA
                          #define PWM_PIN     0

DutyCycle      EQU        0x09
Counter        EQU        0x0A
Period         EQU        0x0B

                          ORG     0x00
Begin                     call    Initialize_PWM
Main                      call    PWM_Signal
                          goto    Main

Initialize_PWM            clrf    PORTA           ; reset PORTA
                          movlw   b'00001110'
                          tris    PORTA           ; Set PWM pin for output
movlw   80
                          movwf   DutyCycle       ; initialize duty cycle
                          movlw   FF
                          movwf   Period          ; initialize period register
                          clrf    Counter         ; reset counter
                          bsf     PORTA, PWM_PIN  ; set PWM signal to high
                          retlw   0

PWM_Signal                incf    Counter,f       ; Increment the counter
                          movf    DutyCycle,w
                          subwf   Counter,w       ; compare duty cycle against
                                                  ; period
                          btfss   STATUS,C        ; Is period < duty cycle
                          retlw   0               ; duty cycle is greater than
                                                  ;period, therefore PWM signal
                                                  ; remains high

                          bcf     PORTA,PWM_PIN           ; time of duty cycle
                                                          ;elapsed
                          movf    Period,w                ; compare the counter
                                                          ;against the period
                          subwf   Counter,w
                          btfss   STATUS,Z                ; if the counter ==
                                                          ; period,
                          retlw   0                       ; period is not over
                          clrf    Counter                 ; reset the counter
                          bsf     PORTA, PWM_PIN          ; Set PWM_Pin to high
                          retlw   0

                          ORG     0x1FF
Reset                     goto    Begin

                          END
```

for example, you can store the values for the duty cycle in a look-up table. The values in the look-up table depend on the values and tolerances of the resistor and capacitor and on the desired resolution of the sine wave. You can download the **listing** from *EDN*'s Web site www.ednmag.com. At the registered-user area, go into the Software Center to download the files from DI-SIG, #2268. (DI #2268). EDN

To Vote For This Design, Circle No. 351