Cika Electrónica

Nota de Aplicación: CAN-067
Título: **Conversor AD por software con Holtek**
Autor: Sergio R. Caprile, Senior Engineer

Revisiones	Fecha	Comentarios
0	12/02/07	

La presente nota de aplicación muestra una forma rápida, sencilla y económica de agregar conversión AD a las familias HT48 de Holtek.

Introducción

Vamos a desarrollar un conversor analógico-digital del tipo integrador. Sumamente simplificado, intentaremos generar una señal digital cuyo valor medio sea igual al de la señal analógica a convertir. Para ello, luego de cargar el capacitor de un circuito integrador al valor de la señal a medir, aplicamos sobre el mismo una serie de pulsos iguales de carga y descarga dentro de un intervalo de tiempo; la relación entre la cantidad de pulsos de carga aplicados y el total de pulsos posibles nos da el valor medio de la señal generada, que debe ser igual al de la señal a medir:

$$V_x = \overline{V_g} = \frac{1}{T}\int_0^T V(t) = \frac{Vcc}{N}\sum_{i=1}^{N} p(i)$$

$$p(i) = \left\{ \begin{array}{c} pulso\,de\,carga: 1 \\ pulso\,de\,descarga: 0 \end{array} \right\}$$

El máximo valor medible corresponde a todos los pulsos de carga (ninguno de descarga), es decir: Vcc[1]

Lo interesante de esto es que podemos determinar la precisión y resolución del conversor jugando con el término $\frac{Vcc}{N}$, por ejemplo:

- ➔ Si el número de cuentas es igual al valor de Vcc multiplicado por 10, cada cuenta tiene una resolución de 100mV
- ➔ Si el número de cuentas es 256, tenemos un conversor de 8-bits (con referencia=Vcc)

Hardware

Necesitamos un comparador, y una red RC. El tiempo de medición es corto y la resolución baja, por lo que no nos preocuparemos por las corrientes de fuga y otros problemas. Las únicas consideraciones que vamos a tener en cuenta son:

- Dado que el comparador debe detectar el valor de 0V a la entrada, el mismo deberá ser capaz de funcionar con este valor o alimentarse con fuente partida.
- Dado que el comparador debe detectar el valor de Vcc a la entrada, el mismo deberá ser capaz de funcionar con este rango de entrada (rail-to-rail input), o alimentarse con una tensión mayor, lo cual suele ser más simple con un comparador como el LM393, cuya salida es open-colector.

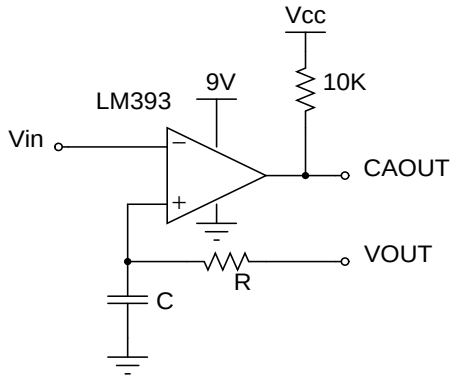
El cálculo de la red RC puede hacerse tomando algunas consideraciones:

- Si lo consideramos como un filtro integrador, su constante de tiempo debe ser mucho mayor que el período de la señal a integrar.

1 En realidad será levemente menor, dado que el algoritmo espera a pasarse para detectar la carga del capacitor, por lo que Vcc quedaría una cuenta fuera, de igual modo que en muchos conversores

- Para tener una precisión suficiente, el capacitor no debe descargarse más del valor correspondiente a una cuenta durante el intervalo de un pulso. Si la constante de tiempo es muy baja, se descarga demasiado rápido.
- Si la constante de tiempo es muy alta, el capacitor no llega a descargarse durante un intervalo de descarga.

Analizado esto, determinamos un valor teórico de acuerdo a la resolución utilizada y la frecuencia de operación. La experiencia práctica indica que a la precisión de este circuito (+- 0,1V), podemos elegir un valor de compromiso que nos permita funcionar de forma aceptable, sin cambios, con R=100K y C=22n.



Software

El algoritmo es muy simple:

- ✔ Cargamos el capacitor hasta exceder la tensión a medir
- ✔ Realizamos el proceso de integración
  - ➔ Pulso de descarga, se descargó el capacitor por debajo del valor a medir ? (repite si no es así)
  - ➔ Pulso de carga, se cargó por encima del valor a medir ? (repite si no es así)

```
CAOUT   equ     PB.0
VOUT    equ     PB.1

; setear según la resolución deseada
;ADCFSCALE     equ     0               ; 0 => 256 cuentas (8-bit ADC)
ADCFSCALE      equ     50              ; 50 => 5V Vcc, resolución 0.1 V
;ADCFSCALE     equ     33              ; 33 => 3.3V Vcc, resolución 0.1 V

        public ADCdata,ADC_convert

adcvars .section 'data'
ADCdata db ?

adcc    .section 'code'

ADC_convert:
        mov a,ADCFSCALE                 ; Full scale
        clr ADCdata                     ; Resetea cuenta
        set VOUT                        ; Carga capacitor
prechg: snz CAOUT                       ; Comparador ?
        jmp prechg                      ; LO, loop
        clr VOUT                        ; Descarga capacitor (tiempo de ciclo constante)
        jmp loop
DACpulse:
        snz CAOUT                       ; Comparador ?
        jmp charge                      ; LO: sige cargando
        clr VOUT                        ; HI: descarga capacitor (tiempo de ciclo constante)
        jmp loop
charge: set VOUT                        ; Carga capacitor (tiempo de ciclo constante)
        inc ADCdata                     ; incrementa ADCData
loop:   sdz ACC                         ; Chequea la cuenta, terminó ?
        jmp DACpulse                    ; No, loop
        clr VOUT                        ; Sí, descarga el capacitor
        ret
```

El programa principal inicializa los pines y llama a la subrutina.

```
vars    .section 'DATA'

        extern ADCdata:byte,ADC_convert:near


reset   .section at 0 'CODE'
        ORG 0
        jmp start

        ORG 00Ch
start:  clr PC
        clr PB
        set PAC                         ; PA0-7 = inputs
        mov a,0F1h                      ; PB1-3 = outputs, PB0,4-7 = inputs
        mov PBC,a
        clr PCC                         ; PC0-7 = outputs
        set PGC                         ; PG0 = input

main:   call ADC_convert
        mov a,ADCdata
        jmp main

        end
```