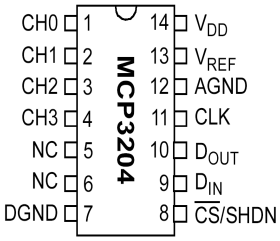
Cika Electrónica

Nota de Aplicación: CAN-010
Título: **Conexión de un conversor A/D MCP3204 a módulos Rabbit**
Autor: Sergio R. Caprile, Senior Engineer

Revisiones	Fecha	Comentarios
0	14/8/03	

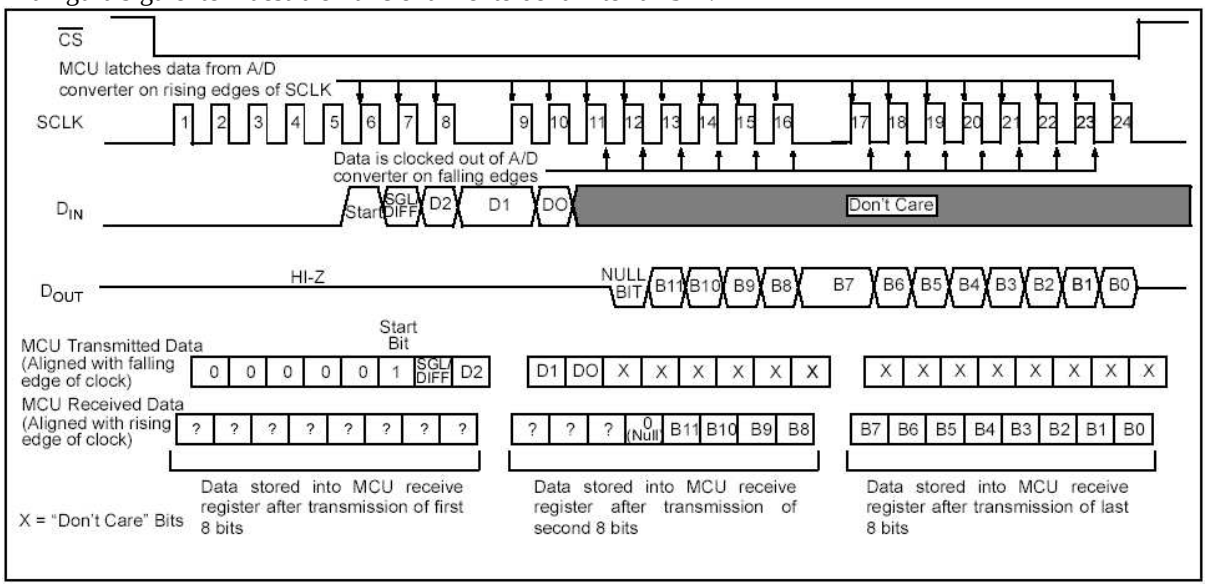
Con el fin de proporcionar entradas analógicas a los módulos Rabbit; introducimos el MCP3204 de Microchip (conversor analógico-digital de 12 bits), y desarrollamos su conexión con estos módulos mediante la interfaz SPI. Desarrollamos además un simple driver para obtener los datos del conversor, con un modesto ejemplo.

Descripción del MCP3204



El MCP3204 de Microchip es un conversor analógico-digital de 12 bits por aproximaciones sucesivas (SAR) con interfaz SPI. Dispone de cuatro entradas que puede configurar como 4 canales single-ended ó 2 canales pseudo-diferenciales (el potencial de la entrada IN- no debe alejarse más de unos 100mV del potencial de GND). La referencia de tensión debe ser externa, funciona a 3 ó 5V y su velocidad de conversión (12 pulsos de clock) ronda las 100Ksps a 5V. Siendo un conversor de 12 bits, su ecuación de funcionamiento es: $D = 2^{12} \times \frac{V_i}{V_{ref}}$, donde V_i es la tensión equivalente de entrada, V_{ref} la tensión de referencia, y D el número entregado por el conversor. Definimos como tensión equivalente de entrada a la tensión en el pin IN+ en modo single-ended y a la diferencia IN+ - IN- en modo diferencial.

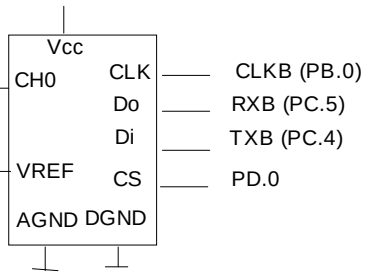
La figura siguiente muestra el funcionamiento de la interfaz SPI:



Como podemos observar, formateando adecuadamente el comando, obtendremos 24 bits de los cuales nuestro resultado estará justificado a la derecha, es decir, en los 12 bits menos significativos

Hardware

Conectamos el MCP3204 a un port serie sincrónico del módulo Rabbit. En este caso, dado que disponemos de un RCM2100, utilizamos el port serie B. Necesitamos además una salida para oficiar de chip enable, utilizaremos PD.0 para tal fin.



De más está decir que la estabilidad de la fuente de VREF influye directamente sobre la estabilidad de la medición, y que la conexión de las masas analógica y digital es fundamental, junto con una buena disposición de las pistas. Se recomienda particularmente la utilización de planos de tierra.

Para el ejemplo, como simplemente nos interesa desarrollar el driver y observar que funciona, usaremos Vcc como referencia y conectaremos un potenciómetro entre Vcc y GND, con su cursor a la entrada del canal CH0; configurado como single-ended. El valor de salida corresponderá a 0x000 cuando el cursor esté a masa y 0xFFF cuando esté a 5V.

Software

Utilizar la interfaz SPI en el port B significa definir el port e incluir la biblioteca de funciones:

```
#define SPI_SER_B
#define SPI_CLK_DIVISOR       5
#use SPI.LIB
```

Ahora inicializamos el hardware de Rabbit para funcionar con el hardware descripto en esta nota:

```
void initAD()
{
       BitWrPortI ( PDDR, &PDDRShadow, 1, 0 );      // CS = 1 (off)
       BitWrPortI ( PDDCR, &PDDCRShadow, 0, 0 );    // PD.0 "normal" (sin open drain)
       WrPortI ( PDCR, &PDCRShadow, 0 );            // salida controlada por instrucción
       BitWrPortI ( PDDDR, &PDDDRShadow, 1, 0 );    // PD.0 = output
       SPIinit();                                   // inicializa interfaz SPI
}
```

Seguidamente, un simple driver para realizar la lectura del A/D. Hemos elegido C para esta tarea como demostración. De ser necesaria máxima velocidad, es posible que el lector deba utilizar assembler.

La variable *Command* es formateada acorde al comando del MCP3204: bit de start, single-ended o diferencial, y 3 bits que definen el canal, aunque sólo se utilizan los dos menos significativos.

La rutina se invoca con el valor del canal como parámetro y devuelve el resultado del A/D en un entero.

```
#define START  0x80
#define SINGLE 0x40

int ReadAD ( int Channel)
{
       int Command, j;
       struct  {                                    // 24 bits
              char    b;
              int     i;
       } Data;

       Command = START|SINGLE | ((Channel/4)<<5) | ((Channel/2)<<4) | ((Channel&1)<<3);
       Command <<= 3;                       // posición para obtener LSB justif a la derecha
       Command = SwapBytes ( Command );     // pone el MSB primero (Z80 es LSB primero)

       BitWrPortI ( PDDR, &PDDRShadow, 0, 0 );      // baja CS
       SPIWrRd ( &Command, &Data, 3 );              // transmite y recibe 24 bits
       BitWrPortI ( PDDR, &PDDRShadow, 1, 0 );      // sube CS
       j = Data.i;                                  // toma 16 bits útiles
       j = SwapBytes ( j ) & 0x0FFF;                // pone LSB primero, considera 12 bits
       return(j);
}
```

```
// invierte (swap) los bytes de un entero
// parámetro y resultado en HL
#asm
SwapBytes::
        ld              a, L            ; salva LSB
        ld              L, H            ; MSB -> LSB
        ld              H, A            ; recupera LSB en MSB
        ret
#endasm
```

Un simple programa de ejemplo: leemos el valor del conversor, y sabiendo que su entrada está entre 0 y 5V mostramos en una ventana de Dynamic C el valor de dicha tensión.

```
void main ()
{
        int Value;
        float volts, ScaleFactor;

        ScaleFactor = 5.0/4096.0;
        initAD();

        while (1) {
                Value = ReadAD ( 0 );
                volts = (float) Value * ScaleFactor;
                printf ( "Value = %5.3f  \r", volts );
        }
}
```