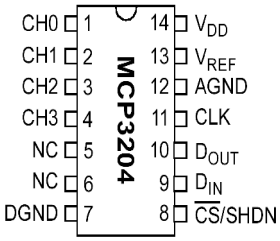
Cika Electrónica

Nota de Aplicación: CAN-061
Título: **Conexión de un conversor A/D MCP3204 a micros Ramtron VRS51L3074**
Autor: Sergio R. Caprile, Senior Engineer

Revisiones	Fecha	Comentarios
0	08/02/07	port de CAN-010

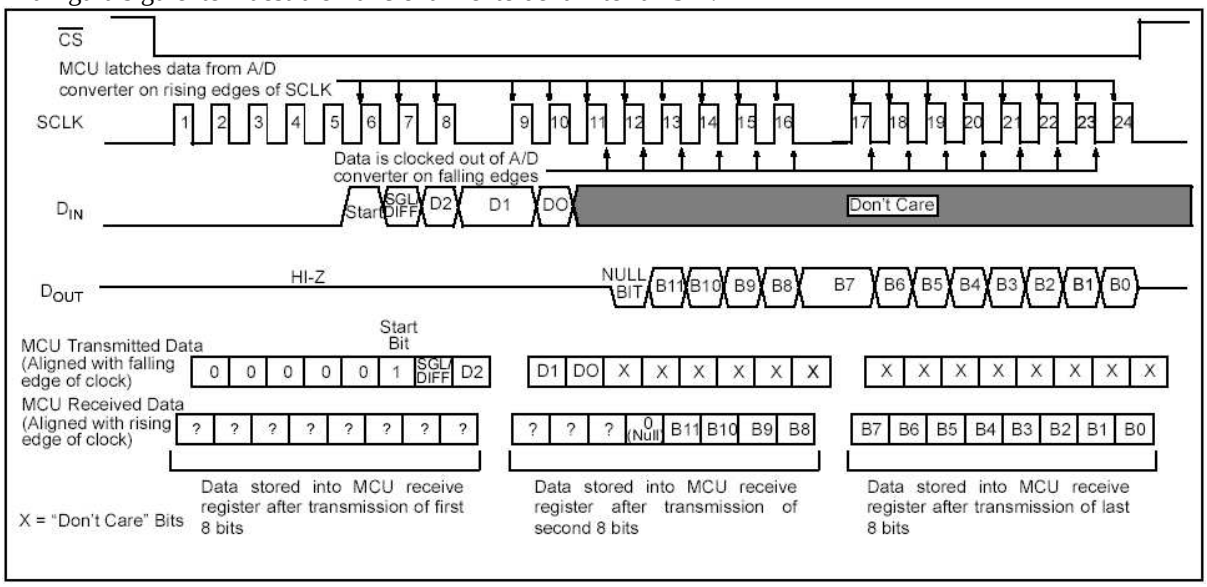
Con el fin de proporcionar entradas analógicas a los micros Ramtron VRS51L3074; introducimos el MCP3204 de Microchip (conversor analógico-digital de 12 bits), y desarrollamos su conexión con éstos mediante la interfaz SPI. Desarrollamos además un simple driver para obtener los datos del conversor, con 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.

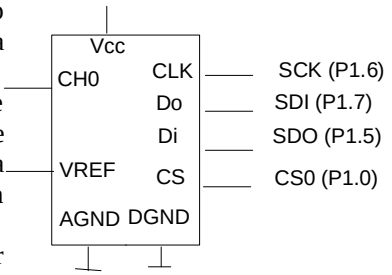
La figura siguiente muestra el funcionamiento de la interfaz SPI:



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 la interfaz SPI del micro. Como chip enable, utilizaremos el pin *CS0*, que será controlado de forma automática por el micro.



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 CH3; configurado como single-ended. El valor de salida corresponderá a 0x000 cuando el cursor esté a masa y 0xFFF cuando esté a 3,3V.

Software

Inicializamos el hardware para funcionar con esta nota, tomado de software ejemplo del fabricante, con menores modificaciones:

```
void initAD(void)
{
char readflag = 0x00;

        PERIPHEN1 |= 0xC0;                  // Habilita interfaz SPI

        while(!(SPISTATUS &= 0x08));        // espera inactividad

        SPICTRL = 0x85;                     // SPICLK = /32 (1.25MHz)
                                            // CS0 Activo
                                            // SPI Modo 0 Phase = 0, POL = 0
                                            // SPI Master

        SPICONFIG = 0x40;                   // SPI Chip select automático
                                            // Clear SPIUNDEFC Flag
                                            // SPILOAD = 0 -> Manual CS3
                                            // SPI Interrupt no habiltado

        SPISTATUS = 0x00;                   // MSB primero

        SPISIZE = 0x17;                     // Transaction Size = 24-bits

        readflag = SPIRXTX0;                // Lee SPI RX buffer para resetear RXAVF
}
```

Seguidamente, un simple driver para realizar la lectura del A/D. Hemos elegido C y un funcionamiento bloqueante para esta tarea como demostración, dada la velocidad de trabajo es suficiente para muchas aplicaciones. De ser necesaria máxima velocidad, es posible que el lector deba utilizar assembler e interrupciones, como se verá en otras notas de aplicación, más exigentes; por ejemplo CAN-063 muestra el modo de utilizar las interrupciones SPI con el ADC

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

```
#define START  (1<<2)
#define SINGLE (1<<1)

int ReadAD (unsigned char Channel)
{
int adcdata;

        SPIRXTX2 = 0x00;
        SPIRXTX1 = (Channel<<6);
        SPIRXTX0 = START | SINGLE | (Channel>>2);          // Escribir en SPIRXTX0 inicia
                                                          // la transacción

        // Espera SPIRXAVF
        while(!(SPISTATUS & 0x02));

        // Lee SPI
```

```
        adcdata= (SPIRXTX1 << 8);
        adcdata|= SPIRXTX0;
        adcdata&= 0x0FFF;                           // 12-bits justificados al LSb
        return(adcdata);
}
```

Para obtener una conversión, simplemente llamamos a la función descripta:

```
void main ()
{
int Value;

        initAD();

        while (1){
                Value = ReadAD ( 3 );
        }
}
```