Control Digital en PC con MS-DOS

Abel Alberto Cuadrado Vega

19 de abril de 2006

1. Introducción

Un PC puede ser utilizado para realizar un sistema de control digital. Para ello necesita lo siguiente:

- tarjeta de conversión analógica-digital y digital-analógica (AD/DA)
- sistema operativo MS-DOS
- compilador de C (Borland Turbo C/C++)

MS-DOS es un sistema monousuario y monotarea. No cuenta con funciones relativas a hilos, sincronización entre tareas (mutex, semáforo,...), etc. Sin embargo admite control directo sobre el hardware de forma simple (interrupciones, entrada/salida, etc.), lo que permite programar directamente tarjetas AD/DA de bajo coste y programar uno de los temporizadores 8253/4 para generar interrupciones para la implementación del muestreo periódico.

2. El temporizador

Para conseguir una función donde implementar el regulador discreto que sea llamada periódicamente, hay que declarar una función de tipo *interrupt*, programar el temporizador 0 del PC con el periodo deseado y hacer que el vector de interrupción 0x1C apunte a nuestra función.

2.1. Función de interrupción

Nuestra función de ejecución periódica debe declararse de tipo *interrupt*. De esta forma el compilador hace uso de la instrucción máquina de retorno de interrupción en lugar de la de retorno de subrutina normal. También debe declararse de tipo *far* puesto que los vectores de interrupcción son punteros de este tipo, de la forma `segmento:desplazamiento`. El compilador no se preocupa de salvar el estado del coprocesador matemático a la entrada de la función ni de recuperarlo a la salida, así que es algo que debe incluirse en el código de la función.

```
void interrupt far FuncionPeriodica(void)
{
    /* Espacio para salvar estado del coprocesador */
    char *data87[94];
```

```
    /* Salvar estado del coprocesador y reiniciarlo */
    asm fsave data87
     _clear87();

    /* Aquí iría el código del algoritmo de control */

    /* Recuperar estado salvado */
    asm frstor data87
}
```

2.2. Programación del temporizador 8254

Un PC tiene tres temporizadores 8254, denominados respectivamente 0, 1 y 2. El primero se utiliza para mantener la hora del sistema operativo (no confundir con el reloj de tiempo real, que lleva la hora del PC), el segundo para el refresco de la RAM y el tercero para generar los pitidos del *speaker*. El más sencillo de usar es el temporizador 0, pero tiene el inconveniente de que al modificar su periodo, el reloj del sistema va a una velocidad distinta de la real.

En el PC, los temporizadores se encuentran ocupando cuatro direcciones consecutivas de los puertos de entrada/salida a partir del 0x40. Las direcciones de la 0x40 a la 0x42 corresponden respectivamente a los registros de cuenta de los temporizadores (que son de 16 bits, pero se accede a ellos con sucesivas lecturas o escrituras a través de un puerto de 8 bits). En la dirección 0x43 se encuentra el registro de control de los tres temporizadores.

De entre todos los modos disponibles, el temporizador 0 es usado por el sistema en modo 3 (generador de onda cuadrada) y así lo mantendremos. Sólo hay que cambiar el periodo del temporizador al valor deseado:

```
    float frec_base=1193180.0;    /* Frecuencia base: 1.193 MHz*/
    int contador, contador_hi, contador_lo;
    float Tm;                      /* Período en segundos */

    contador=ceil(frec_base*Tm);   /* valor del contador... */
    contador_hi=contador/256;      /* ... su byte alto ...*/
    contador_lo=contador-contador_hi*256; /* ... y el bajo */
    outportb(0x43,0x36);           /* escribe byte de control */
    outportb(0x40,contador_lo);    /* escribe el contador bajo */
    outportb(0x40,contador_hi);    /* y luego el alto */
```

Al acabar debe recuperarse el valor original del sistema de frecuencia 18.2 Hz:

```
    outportb(0x43, 0x36);
    outportb(0x40, 0xFF);
    outportb(0x40, 0xFF);
```

2.3. Modificación del vector de interrupción

El Turbo C cuenta con dos funciones para manejo de vectores de interrupción: una para leer el valor actual de un vector y otra para modificarlo. La forma de proceder correcta será hacer que el vector de interrupción correspondiente apunte a nuestra función, salvando previamente el valor original de dicho vector

```
    void interrupt (*viejoint1C) (void);

    disable();
    viejoint1C=getvect(0x1C);
    setvect(0x1C,FuncionPeriodica);
    enable();
```

para que cuando acabe el programa se pueda restaurar su valor:

```
    disable();
    setvect(0x1C,viejoint1C);
    enable();
```

Por supuesto, hay que deshabilitar las interrupciones para evitar que se produzca una durante el cambio, mientras el vector de interrupción está en un estado indefinido (la escritura del nuevo vector no es atómica). Aunque el vector de interrupción del temporizador 0 es el 0x08 (correspondiente a la IRQ 0), es mejor usar el 0x1C, que es llamado desde la 0x08, sobre todo porque ello evita tener que realizar ciertas operaciones necesarias en el controlador de interrupciones (PIC 8259).

2.4. Listado del programa base completo

A continuación se presenta el programa básico completo, donde se indican los lugares a colocar el código para el algoritmo de control y la interfaz de usuario y en el que hay que dar un valor al periodo de muestreo `Tm`:

```
#pragma inline     /* hay instrucciones en ensamblador */

#include<dos.h>    /* enable(), disable(), getvec(), setvec() */
#include<math.h>   /* ceil(), fmod() */
#include<float.h>  /* _clear87() */

/*****************************************************************
   Plantilla para interrupciones temporizadas
   En este fichero se ofrece un cascaron de rutina de
   interrupcion, se programa el timer del PC para seleccionar
   una frecuencia de interrupcion entre 18.2 Hz y 1.193 MHz.
   y se instala la rutina de interrupción en el vector 0x1C
   (que es llamado por la interrupcion del timer, la 0x08.)

   Antonio Robles Alvarez, 2004
   Area de Ingenieria de sistemas y automatica (ISA)
   Universidad de Oviedo

   Ligeras modificaciones:
   Abel Alberto Cuadrado Vega, 2006
*****************************************************************/

void interrupt far FuncionPeriodica(void)
{
```

```
    /* Espacio para salvar estado del coprocesador */
    char *data87[94];

    /* Salvar estado del coprocesador y reiniciarlo */
    asm fsave data87
     _clear87();

    /* Lugar para el código del algoritmo de control */

    /* Recuperar estado salvado */
    asm frstor data87
}

void main(void)
{
    float frec_base=1193180.0;    /* Frecuencia base: 1.193 MHz*/
    int contador, contador_hi, contador_lo;
    void interrupt (*viejoint1C) (void);
    float Tm;                     /* Periodo en segundos */

    /* Programacion del temporizador */
    contador=ceil(frec_base*Tm);  /* valor del contador... */
    contador_hi=contador/256;     /* ... su byte alto ...*/
    contador_lo=contador-contador_hi*256; /* ... y el bajo */
    disable();                    /* desactiva interrupciones */
    outportb(0x43,0x36);          /* escribe byte de control */
    outportb(0x40,contador_lo);   /* escribe el contador bajo */
    outportb(0x40,contador_hi);   /* y luego el alto */
    /* Sustitucion de vector de interrupcion */
    viejoint1C=getvect(0x1C);     /* salva la dirección vieja */
    setvect(0x1C,FuncionPeriodica);/* coloca la nueva */
    enable();                     /* habilita interrupciones */

    /*************************************************************
       Lugar para la interfaz de usuario:
    * Lectura de teclado
    * Salida por pantalla
    * Etc.
    *************************************************************/

    /* Antes de salir, restaurar las condiciones originales */
    disable();                    /* desactiva interrupciones */
    setvect(0x1C,viejoint1C);     /* restaura la vieja ISR */
    outportb(0x43, 0x36);         /* restaura timer */
    outportb(0x40, 0xFF);         /* frecuencia 18.2 Hz */
    outportb(0x40, 0xFF);
    enable();                     /* habilita interrupciones */
}
```

3. La tarjeta AD/DA

La tarjeta de AD/DA usada es la CIO-DAS08/JR. Entre otras características, es una tarjeta de 12 bits con 8 canales analógicos de entrada y 2 de salida, todos ellos con un rango de ±5V. No genera interrupciones hardware; por ello las temporizaciones y controles de las conversiones deben realizarse en el software del usuario. En los equipos de prácticas la dirección BASE de sus registros de E/S se encuentra en 0x300.

3.1. Conversión A/D

Para la lectura de un canal de entrada analógico hay que realizar los siguientes pasos:

1. Seleccionar el canal analógico a leer, de 0 a 7, escribiendo su número como un byte en el puerto BASE+2.

2. Comenzar la conversión A/D escribiendo cualquier valor de byte en el puerto BASE+1.

3. La conversión termina cuando el bit 7 (mayor peso) del byte del puerto BASE+2 vale 0.

4. Los 12 bits del valor convertido se encuentran distribuidos en los bytes de los puertos BASE y BASE+1 de la siguiente manera[1]:

	bit 7	*bit 6*	*bit 5*	*bit 4*	*bit 3*	*bit 2*	*bit 1*	*bit 0*
BASE	A/D 3	A/D 2	A/D 1	A/D 0	X	X	X	X
BASE+1	A/D 11	A/D 10	A/D 9	A/D 8	A/D 7	A/D 6	A/D 5	A/D 4

5. Conversión del valor binario de la tarjeta a voltios teniendo en cuenta que es una relación lineal donde 0 corresponde a -5V y 4095 corresponde a $+5$V.

3.2. Conversión D/A

Para la escritura de un canal de salida analógico hay que realizar los siguientes pasos:

1. Conversión del valor en voltios v a valor binario sin signo de 12 bits de la tarjeta n, dado por la siguiente relación:

$$v = \frac{n}{4096} \cdot 10 - 5 \tag{1}$$

[1]Para mas claridad se ha denominado a los bits del valor convertido, A/D x, como suele ser usual, es decir, siendo 11 el de más peso y 0 el de menos peso, al contrario que en el manual de la tarjeta.

asegurándose previamente que el valor de v es tal que se puede representar con un valor de n entre 0 y 4095 inclusive.

2. Escribir el byte bajo del valor binario en BASE+4 y el alto en BASE+5 si se trata del canal 0, o bien en BASE+6 y BASE+7 si se trata del canal 1. También es posible escribir la palabra de 16 bits entera al puerto BASE+4 para el canal 0 o BASE+6 para el canal 1 con un único comando de salida por puerto de 16 bits[2].

3. Leer el byte del puerto BASE+3 para que se actualicen las salidas analógicas con los valores dados. Esto permite actualizar los valores eléctricos de los dos canales simultáneamente.

[2]En todos los casos se está suponiendo una extensión de la palabra de 12 bits a una de 16 bits rellenado con 0 los bits de mayor peso (de 12 a 15, aunque en la práctica la tarjeta simplemente ignora esos bits, valgan lo que valgan).