Informe segunda práctica: A/D

Grupo SDMI 22 E

David Guerra

Roman Valls

Daniel Clemente

Se pide leer un dato (analógico) de una de las entradas del conversor A/D, y mostrar en binario su valor mediante los 8 leds de la placa. La entrada está simulada usando la placa externa, y oscila entre 0 y 5 V (8 bits son suficientes). La hemos conectado por el pin P0.0.

El botón externo (INT07) sirve para bloquear la visualización del valor o continuar viendo la conversión en tiempo real. El código es el de la práctica anterior, EXTINT.

Conexionado del potenciómetro:

Para simular la entrada Analógica se ha utilizado una resistencia variable; ésta se ha conectado a la patilla 1 del JP1 (Vcc) y a la patilla 19 del JP1 (GND), la salida de V variable se ha conectado a la patilla 25 del JP2, que corresponde a la entada ACH0/P0.0, o lo que es lo mismo, al primero de los ocho canales de entrada analógicos que tiene el 8X196KC/KD.

stad.a96:

Los únicos cambios consisten en colocar los vectores de interrupciones en memoria: el encargado de atender la interrupción externa (INT07) va en 0d0e0h, y el aviso generado al completar una conversión A/D (INT01) en 0d020h. Estos valores están sacados del documento "entorn2.pdf", página 9.

```
        ;; INT1 (AD Conversion complete)
cseg at 0d020h
        br AD_CONVERSION

cseg at 0d0e0h
        br EXTINT
```

AD.c:

Comentarios en el código.

```
int regtemp; // Para aplicar máscaras. 16 bits

void main()
{

/*Inicialitzacions*/

        wsr=0;
        ioport1=0xAA; //Empezamos mostrando 1010 1010 en los leds
```

```
        wConversio=0xAA;
        bADEnCurs=1; // Empezamos convirtiendo

/*  Inicialitzem el registre de màscara posant un 1 a les interrupcions que volem
    habilitar.*/

        asm { di; }
        wsr=0;  //Aunque podemos usar cualquier ventana para int_mask/int_mask1

        int_mask=int_mask | 0x82; //Habilitamos EXT_INT & AD Conversion complete
                                  //     1      x    x   x     x     1   x
                                  // ext_int|serial|swt|hsi0|hsidat|ad|t1ovf
                                                              //
        int_mask1=int_mask1 & 0xDF; // Desactivamos la INT13

        wsr=0xf; regtemp=ioc1;
        regtemp = regtemp & 0xFD; // Elegir pin p2.2
        wsr=0x0; ioc1=regtemp;

/*  Modifiquem el registre IOC2, seleccionem les opcions del conversor.*/
        wsr=0xf; regtemp=ioc2;
        regtemp = regtemp & 0xF7; // AD_TIME_ENA=0 para modo compatible con 196KB
        wsr=0x0; ioc2=regtemp;

        wsr=0xf; regtemp=ad_command;
        regtemp = regtemp | 0x10; // 8 bits
        regtemp = regtemp & 0xF8; // Elegir puerto 0
        wsr=0x0; ad_command=regtemp;

/*  Finalment cal habilitar les interrupcions per poder servir-les. */
        asm { ei;}

        wsr=0xf; regtemp=ad_command;
        regtemp = regtemp | 0x08; // GO=1
        wsr=0x0; ad_command=regtemp;

/*  Bucle del programa principal  */
/* Este bucle espera a que se haya acabado la interrupción que lee el contenido del
   registro de resultado del conversor AD (rutina RSIADDone). Una vez acabada se
   muestra el resultado por el array de leds de la placa placa */

        while(1)
        {
        wsr=0x0; ioport1=0xFF; // CHIVATO. QUITAR
        /* Estos chivatos se quitan en la versión de producción. Hemos descubierto
         técnicas de debugging más efectivas, tales como examinar los ficheros que
         se generan al compilar el proyecto (.lst & .m96) y usar el debug monitor
         con los valores que hemos obtenido de dichos ficheros.*/

        wsr=0x0; // Para leer ad_result
        while (ad_result&0x08); // Esperar a que haya acabado (encuesta)


        wsr=0x00; ioport1=0xB1; // CHIVATO. QUITAR
        if (bADEnCurs) ioport1=wConversio;
        ioport1=0xCA; // CHIVATO. QUITAR

        wsr=0xf; regtemp=ad_command;
```

```
        regtemp = regtemp | 0x08; // GO=1
        wsr=0x0; ad_command=regtemp;



        }
}



void RSIExtInt()
{
/* Si recibimos una interrupción de la RSI ExtInt, se bloquea el proceso de conversión
 * del AD, mostrando el último valor convertido por este. Si volvemos a pulsar el botón
 * (que activa la INT07), podremos continuar viendo la conversión analógica en la ristra
 * de leds en tiempo real.*/
        asm {di;}
        bADEnCurs=!bADEnCurs;
        asm {ei;}
}

/*****************************************************************************
;
;  Rutina de servei a la interrupció d'AD conversion complete.
;
;*****************************************************************************/

void RSIADDone()
{
/* Esta subrutina lee el resultado de la conversión del AD y la deja en la
 * variable wConversio, para ser procesada en el bucle principal. Tambíen se
 * actualiza el estado del conversor AD mediante la variable ad_result que se
 * usa en el bucle principal. */

        asm {di;}

        wsr=0x0; regtemp=ad_result;
        wConversio=regtemp>>8;
        regtempt = regtemp & 0xF7; // Desactivar el bit de aviso
        wsr=0xf; ad_result=regtemp;

        asm {ei;}
}
```