

SISTEMAS EMPOTRADOS

3º Grado en Ingeniería Informática

PRÁCTICA 8: Conversor Analógico/Digital

8.1. Introducción

El conversor analógico/digital del LPC2378 es un conversor de aproximación sucesiva de 10 *bits* con un tiempo de conversión de 2,44 μ Sec o por debajo de 410 Kbps. Dispone de 8 canales de conversión.

Los registros, desde el punto de vista del programador, son los siguientes:

- **ADxCR:** Registro de control del conversor A/D del canal $x(0..7)$
- **ADxGDR:** Registro de datos global del conversor A/D.
- **ADxSTAT:** Registro de estado del conversor A/D.
- **ADxINTEN:** Registro de habilitación de las interrupciones para el ADC.
- **ADxDRy:** Registro de datos y del canal x .

El registro de control del conversor (**ADxCR**) es el de la siguiente tabla, cuyos campos se describen a continuación:

				ED GE	START					PDN		CLKS			BURST	CLKDIV								SEL									
31				28	27	26	25	24			21		19	18	17	16	15								8	7							0

- ✓ **SEL:** Selecciona los pines que van a ser convertidos. Para el pin AD0.0 activaremos (a uno) el *bit* 0 del registro AD0CR
- ✓ **CLKDIV:** El reloj del bus APB (PCLK) se divide por este valor más uno para obtener el reloj del ADC, que debe ser menor o igual a 4.5 MHz.
- ✓ **BURST:** Cuando este *bit* vale cero, la conversión es controlada por *software* y requiere 11 ciclos de reloj.
- ✓ **CLKS:** Número de ciclos utilizados para cada conversión cuando BURST=1. Si BURST=0 debe estar a 000, utilizando 10 *bits* para la conversión.
- ✓ **PDN:** Si está a uno se habilita el ADC, si está a cero está en modo *power-down*.

- ✓ **START:** Cuando BURST=0, estos *bits* controlan si se produce la conversión o bajo qué condiciones. 000 (no hay conversión) 001 (comienza la conversión).
- ✓ **EDGE:** Este *bit* sólo se utiliza cuando START es 010 ó 111.
- El registro de habilitación de las interrupciones (**ADxINTEN**) es el siguiente:

No usado																ADGINTEN	ADINTEN 7:0							
31															9	8	7							0

- ✓ **ADINTEN 7:0:** Estos *bits* permiten controlar qué canales A/D generan una interrupción al completar la conversión. Cuando el *bit* 0 es uno, la finalización de una conversión en el canal A/D 0 generará una interrupción, cuando el *bit* 1 es uno, la finalización de una conversión en el canal A/D 1 generará una interrupción, etc.
- ✓ **ADGINTEN:** Cuando vale 1, habilita el indicador DONE del ADGDR para generar una interrupción. Cuando vale 0, sólo los canales A/D individuales habilitados por ADINTEN 7:0 generarán una interrupción.
- **ADxDRy:** Registro de datos y del canal *x*.

DONE	OVERRUN	No usado																V/V _{REF}								No usado							
31	30	29													16	15									6	5							0

- ✓ **DONE:** Este *bit* se pone a “1” cuando se completa una conversión A/D. Al leer este registro se coloca a “0”.
- ✓ **OVERRUN:** Este *bit* es 1 en modo ráfaga si los resultados de una o más conversiones se ha perdido y se sobrescribió antes de la conversión que produjo el resultado en los *bits* menos significativos. Este *bit* se borra al leer este registro.
- ✓ **V/VREF:** Cuando DONE es 1, este campo contiene la división del voltaje en el pin A_{in} entre el voltaje en el pin V_{REF}. Cuando este campo es 0x0000 indica que el voltaje en el pin A_{in} es menor o igual o cercano a V_{REF}, mientras que 0x03FF indica que el voltaje en A_{in} era cercano, mayor o igual a V_{REF}.

8.2. Objetivos

El objetivo de esta práctica será configurar correctamente el canal 0 del conversor AD del LPC2378 para visualizar por un terminal (*Hyperterminal*) del puerto serie (COM1) la conversión que se realiza en el puerto P0.23 configurado como conversor AD. Al ser un conversor de 10 *bits* el rango de la visualización debe de ser de 0x0000 hasta 0x03FF.

El potenciómetro de la placa MCB2300 está conectado al puerto 0.23 (AD0.0) cuando el *jumper* AD0.0 está habilitado. El rango de tensión es 0.0–3.3 V.

8.3. Material necesario:

- Ordenador personal con *Windows*.
- Placa de desarrollo MCB2300 de *Keil*.
- Adaptador USB–JTAG de la familia ULINK para depurar programas.
- Dos cables USB A–B.
- Cable serie RS232 o adaptador USB–UART.

8.4. Desarrollo de la práctica:

Crearemos una carpeta nueva para la realización de la práctica donde copiaremos los ficheros `serial.c`, `retarget.c`, `misTipos.h`, y `LPC2300.s` de cualquiera de las prácticas anteriores. Los ficheros en que realizaremos modificaciones son los siguientes:

- `Practica8.c`: En este fichero se realizará el programa principal de la aplicación, desde donde llamaremos al resto de las funciones.
- `HAL.c`: Fichero para definir el *hardware* que utilizaremos en la aplicación, en esta práctica el conversor A/D y el *timer* 0.

Al crear un nuevo proyecto deberemos de incluir estos archivos de manera que el manejador de proyectos quede, por ejemplo, como se indica en la figura 8-1.

Para configurar el canal 0 del conversor A/D necesitaremos buscar en el manual del fabricante del microcontrolador los siguientes registros:

- `PCONP`: Registro para controlar la potencia de cada periférico.
- `PINSEL`: Registro donde deberemos buscar el periférico AD0.0 que es el que utilizaremos por estar conectado al potenciómetro de la placa MCB2300.
- `AD0INTEN`: Registro para habilitar las interrupciones en el canal ADC0.
- `AD0CR`: Registro de control del ADC0 descrito en la introducción.

- AD0DR0: Registro donde se almacena el resultado de la conversión que será utilizado en la programación.

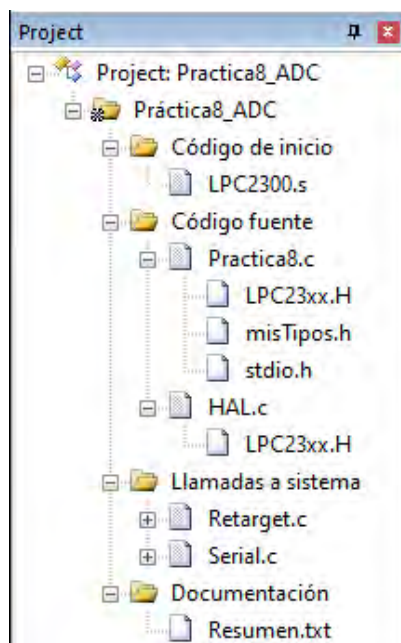


Figura 8-1: Vista previa del manejador de proyectos (*Project Manager*).

Hay que tener en cuenta que hay que configurar el temporizador (*timer 0*) en modo *hardware* al igual que en las primeras prácticas por sondeo (*polling*).

8.5. Ficheros a realizar:

```

Practica8.c  HAL.c  misTipos.h  Serial.c  Resumen.txt
1  /*****
2  /* Practica 8: Conversor Analógico-Digital
3  /*****
4  /* Apellidos y nombre del alumno/s:
5  /*****
6  /* Sistemas Empotrados. 3º de Graduado en Ingeniería Informática
7  /* Universidad de Córdoba
8  /*****
9  #include <LPC23xx.H>
10 #include <stdio.h>
11 #include "misTipos.h"
12 extern void hardwareInit(void);
13 extern void init_serial(void);
14 extern void delayT0Unlocked(unsigned int);
15 int main() {
16     UINT32 AD_Value;
17     AD_Value=0;
18     hardwareInit();
19     init_serial();
20     while(1)
21     {
22         ADCR |= 0x01000000; /* Start A/D Conversion */
23         AD_Value=AD0DR0;
24         delayT0Unlocked(10000);
25         while(!(TOIR & 0x00000001));
26         TOIR = 1;
27         while ((AD_Value & 0x80000000) != 0x80000000) (AD_Value=AD0DR0);
28         AD_Value= (AD0DR0 >> 6) & 0x3FF; /* Read Conversion Result */
29         printf ("Valor de la conversion = 0x%03x\n\r", AD_Value);
30     }
31 }
32

```

Figura 8-2: Programa principal de la aplicación (*main*).

```

Practica8.c  HAL.c  misTipos.h  Serial.c  Resumen.txt
1  /******
2  /* HAL.C: Capa de abstracción del hardware                                */
3  /* Sistemas Empotrados. Universidad de Córdoba                            */
4  /******
5  #include <LPC23xx.H>                /* LPC23xx definitions                */
6  /******
7  /* ADC_Init                          */
8  /******
9  /* Esta función configura el convertor analógico/digital 0 del LPC2378    */
10 /******
11 void ADC_Init(void)
12 {
13     /* Power enable, Setup pin, enable and setup AD converter interrupt    */
14     PCONP   |= (1 << 12);           /* Habilita el ADC                */
15     PINSEL1  |= (1 << 14);           /* Configura P0.23 como AD0.0    */
16     AD0INTEN = (1 << 0);             /* Habilita interrupción en CH0  */
17     AD0CR    = (1 << 0)              /* Selecciona canal AD0.0        */
18             | (3 << 8)               /* CLKDIV = 3 (para 3MHz)        */
19             | (1 << 21);             /* Habilita ADC                  */
20     //AD0CR  = 0x00200301;          /* Power up, PCLK/4, sel AD0.0  */
21 }
22 /******
23 /* timer0Init                        */
24 /******
25 /* Esta función configura el timer 0 con los parámetros que no cambian    */
26 /* durante la aplicación                                                    */
27 /******
28 void timer0Init(void)
29 {
30     TOPR = 0x00;                    /* activa el preescalador a cero */
31 }
32 /******
33 /* delayT0Unlocked                    */
34 /******
35 /* Esta función arranca el timer 0 y programa el registro match0          */
36 /******
37 void delayT0Unlocked(unsigned int delayInDecimaMiliseg)
38 {
39     TOTCR = 0x02;                    /* reset timer                    */
40     TOMRO = delayInDecimaMiliseg * 12000000 / 10000;
41     TOMCR = 0x07;                    /* timer on match                 */
42     TOTCR = 0x01;                    /* inicia timer y para cuando se llegue al final de cuenta*/
43 }
44 /******
45 /* hardwareInit                      */
46 /******
47 /* Esta función se llama al comienzo del programa para inicializar el Hardware*/
48 /******
49 void hardwareInit(void)
50 {
51     ADC_Init();                      /* Configura el convertor analógico/digital */
52     timer0Init();                   /* Inicializa el timer 0            */
53 }

```

Figura 8-3: Capa de abstracción del hardware (HAL.c).

Estudiar principalmente la configuración del registro de control del AD0 (AD0CR).

8.6. Resultados de la simulación:

Los periféricos que nos interesan visualizar en esta práctica son: A/D Converter 0, Timer 0 y UART1.

Además tendremos que visualizar en el menú *View* → *Serial Windows* → UART2 para ver los resultados de la simulación.

En el periférico del conversor analógico/digital 0 simularemos una entrada analógica en la parte inferior del periférico (*Analog Inputs*, AD0.0) unos valores entre 0 V y 3'3 V).

A/D Converter 0

A/D Control

AD0CR: 0x01200301 SEL: 0x01 ☒ PDN

CLKS: 11clk/10bit CLKDIV: 0x03 ☐ BURST ☐ EDGE

START: Now A/D Clock: 250000

A/D Global Data & Status

AD0GDR: 0x80000000 RESULT: 0x0000 ☒ DONE ☐ OVERUN

AD0STAT: 0x00010001 CHN: 0x00 ☒ ADINT

A/D Channel Data

AD0DR	RESULT	DONE	OVERUN
AD0DR0: 0x80000000	RESULT0: 0x0000	<input checked="" type="checkbox"/> DONE0	<input type="checkbox"/> OVERUN0
AD0DR1: 0x00000000	RESULT1: 0x0000	<input type="checkbox"/> DONE1	<input type="checkbox"/> OVERUN1
AD0DR2: 0x00000000	RESULT2: 0x0000	<input type="checkbox"/> DONE2	<input type="checkbox"/> OVERUN2
AD0DR3: 0x00000000	RESULT3: 0x0000	<input type="checkbox"/> DONE3	<input type="checkbox"/> OVERUN3
AD0DR4: 0x00000000	RESULT4: 0x0000	<input type="checkbox"/> DONE4	<input type="checkbox"/> OVERUN4
AD0DR5: 0x00000000	RESULT5: 0x0000	<input type="checkbox"/> DONE5	<input type="checkbox"/> OVERUN5
AD0DR6: 0x00000000	RESULT6: 0x0000	<input type="checkbox"/> DONE6	<input type="checkbox"/> OVERUN6
AD0DR7: 0x00000000	RESULT7: 0x0000	<input type="checkbox"/> DONE7	<input type="checkbox"/> OVERUN7

A/D Interrupt Enable

AD0INTEN: 0x00000001 ☒ ADINTEN0 ☐ ADINTEN4

☐ ADGINTEN ☐ ADINTEN1 ☐ ADINTEN5

☐ ADINTEN2 ☐ ADINTEN6

☐ ADINTEN3 ☐ ADINTEN7

Analog Inputs

AD00	AD01	AD02	AD03
0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000

Reference

VREF: 3.3000

UART #2

```
AD value = 0x000
AD value = 0x000
AD value = 0x000
AD value = 0x000
AD value = 0x000
AD value = 0x000
AD value = 0x000
AD value = 0x000
AD value = 0x000
AD value = 0x000
```

Figura 8-4: Valor leído del conversor para 0 V.

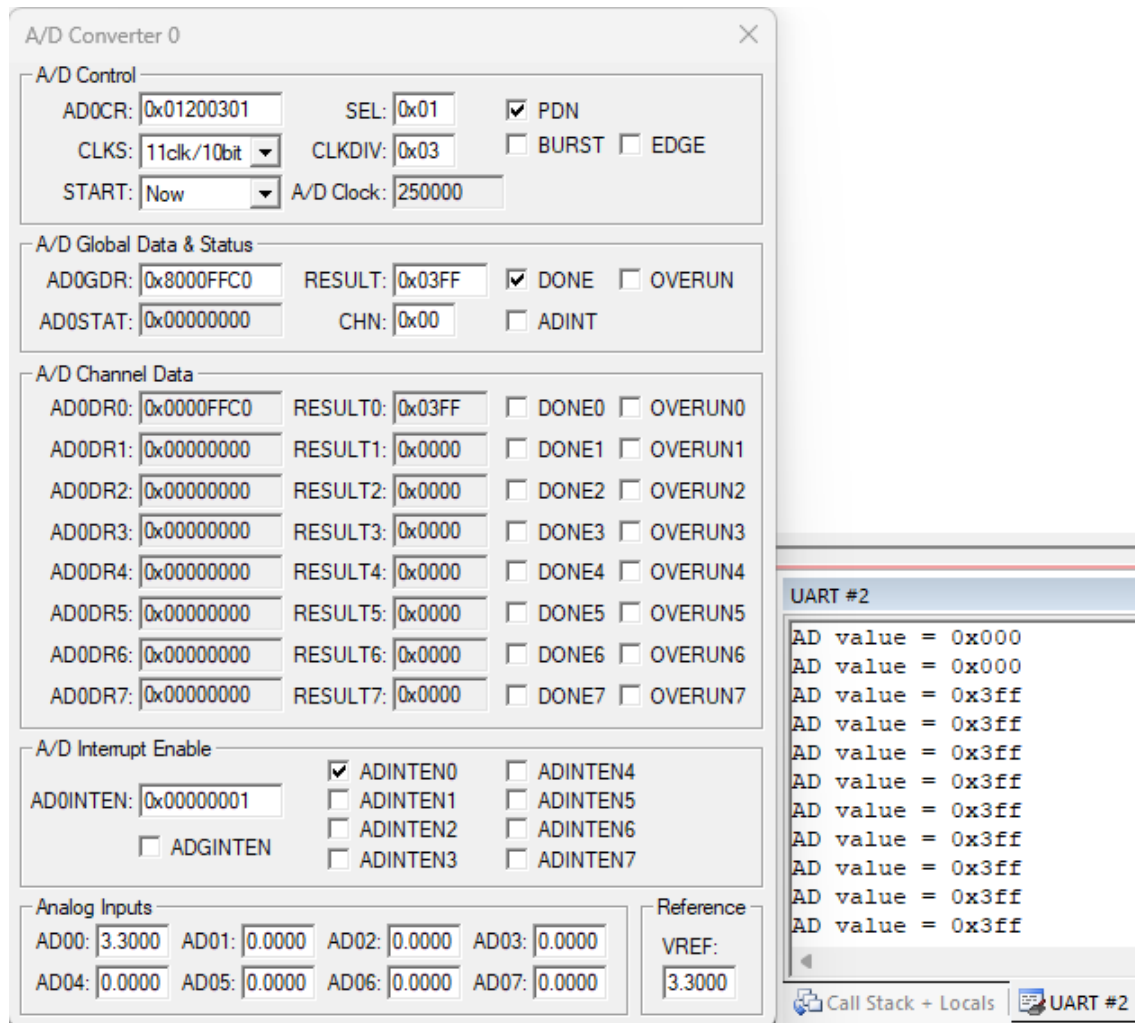


Figura 8-5: Valor leído del conversor para 3'3 V.

8.7. Montaje de la práctica en la placa MCB2300:

Después de simulada y depurada, tendremos que programar la *flash* EPROM de nuestra placa y comprobar el funcionamiento correcto de nuestra aplicación a través del *hyperterminal* utilizado en las prácticas anteriores con la UART.

Como ya se ha indicado, el potenciómetro de la placa MCB2300 está conectado al puerto 0.23 (AD0.0) cuando el *jumper* AD0.0 está habilitado. Por tanto iremos variando el potenciómetro y comprobando que en el *hyperterminal* nos van saliendo resultados correctos con los diez *bits* de que dispone el conversor, es decir; de 0x000 hasta 0x3FF.