6- Qüestionari Pràctica L6B ADC

Nom i Cognoms: Sergio Shmyhelskyy Yaskevych && Songhe Wang Grup: 31
1) Consulteu el resultat de la conversió de l'ADC per polling o per interrupcions?

El resultado de la conversión por ADC se realiza por polling. En este caso, el programa utiliza un bucle (while (GODONE);) para esperar a que la conversión A/D se complete antes de continuar con la ejecución del programa.

Cuando GODONE está en curso (es decir, la conversión A/D está en progreso), el programa se queda en el bucle, evitando que avance hasta que la conversión se complete (GODONE se establezca en 0). Esto se hace para garantizar que se obtenga un valor de voltaje ADC válido antes de proceder con los cálculos de temperatura y la actualización del ciclo de trabajo del PWM.

2) Heu configurat el timer 2 perquè generi interrupcions? Tant si és que sí com si no, quin registre cal configurar perquè es generi una interrupció del timer 2 associada al PWM?

```
PIR1bits.TMR2IF = 0;       // Clear timer 2 interrupt flag bit
IPR1bits.TMR2IP = 1;       // Timer 2 interrupt priority high (does not really matter)
PIE1bits.TMR2IE = 1;       // Enable interrupts for timer 2
PR2 = 249;                 // Configurar registro de TIMER2 (Wave length en PWM)
T2CONbits.T2CKPS = 0b10; establece un prescaler de 16 para el Timer 2.
T2CONbits.TMR2ON = 1; enciende el Timer 2.
T2CONbits.T2OUTPS = 0b0000; configura un postescalador de 1:1.
```

Por lo tanto, cuando el Timer 2 desborda, se genera una interrupción que ejecuta la interrupción. Entonces, para que el timer funcione correctamente, tendremos que configurar los registros PIR1, IPR1, PIE1, PR2, T2CON.

3) Quins pins heu configurat com entrades analògiques i quins com a digitals?

El único pin que hemos configurado como entrada analógica es el pin RE1 (AN6), dado que es por donde leemos valores de entrada del "conversor". Los pins restantes se han configurado como digital (para el motor y el osciloscopio).

4) Amb quin valor (en binari) heu programat els següents registres?

```
ADCON0bits.ADON = 1;
ADCON0bits.CHS = 0b00110;

ADCON1bits.NVCFG = 0;          // Vref+ = VDD connected to internal signal
ADCON1bits.PVCFG = 0;          // Vref+ = GND connected to internal signal

ADCON1bits.TRIGSEL = X         // Indiferente dado que está asociado a CCP3
ADCON2bits.ACQT = 0b100;       // 8*Tad
ADCON2bits.ADCS = 0b001;       // Tad = 8/Fosc Base conversion clock selection for A/D
ADCON2bits.ADFM = 1;           // Right justified to easy conversion
```

5) Què és el TAD? - TAD es el período del reloj de A/D.

Es la unidad que se suele usar la data sheet para describir el tiempo de adquisición. Este valor se calcula a partir del reloj que hemos escogido. En nuestro caso hemos escogido un reloj de A/D de Fosc/8.

6) A quina duració has configurat un TAD?
ADCON2bits.ADCS = 0b001; // Tad = 8/Fosc Base conversion clock selection for A/D

Esta configuración establece que el tiempo de adquisición (TAD) es igual a 8 veces el período del reloj de la base de conversión del convertidor analógico a digital (ADC), donde Fosc es la frecuencia del PIC (8MHz). Por tanto obtenemos una duración de **1 microsegundo.**

7) Què és el temps d'adquisició?

Es el tiempo que transcurre desde que se activa el bit GO/DONE hasta que se completa la adquisición de la muestra de voltaje y se tiene un valor estable en el capacitor del circuito sample-and-hold. Durante este tiempo, el ADC adquiere y mantiene la señal analógica estable que se usa para su conversión a un valor digital.

En detalle, cuando se inicia una conversión ADC, el bit GO/DONE se activa, y el ADC comienza a cargar el capacitor del circuito sample-and-hold con el voltaje de la señal analógica de entrada. En nuestro caso hemos cogido un tiempo de adquisición de 8 TAD.

8) Quina és la duració del teu temps d'adquisició?

ADCON2bits.ACQT = 0b100; // 8*Tad

En nuestro caso, obtenemos un tiempo de adquisición de **8 microsegundos**.

9) Quin és el temps de còmput i el tamany de programa pel càlcul exacte de la temperatura?

El tiempo de cómputo para realizar el cálculo es de **6.9275 milisegundos**. El tamaño de la función que realiza el cálculo es de 374 bytes.

Para poder saber el tiempo de cómputo hemos usado la funcionalidad del debug de proteus. Marcamos en el código el inicio y el final de la función que calcula la temperatura por fórmula. El proteus, al hacer un step, nos indica el "elapsed time". Para el tamaño de la función, en modo debug, damos click derecho al código y seleccionamos mostrar "Display address". Ahora que tenemos las posiciones de memoria podemos obtener el tamaño total del programa haciendo simplemente sumas y restas. En nuestro caso:

```
00007826    float ByFormula (int n) {
00007828       float vout = (n/1023.0)*5.0;
00007876       return (BETA/log(-((R2 - ((R2*VCC)/vout))/A))) - ABS_Z;
00007946    }

00007054    float ByRegresion(int n) {
00007056       return 100*(n/1024.0) - 24;
000070C0    }

00006F44    float getTemp (char mode) {
00006F46       ADCON0bits.GODONE = 1;
00006F48       while (ADCON0bits.GODONE);
00006F4C       int n = (ADRESH << 8) + ADRESL;
00006F5E       if (mode == 'F') return ByFormula(n);
00006F80       return ByRegresion(n);
00006F98    }
```

(0x6F98 - 0x6F44 + 0x1) + (0x7946 - 0x7826 + 0x1) = 0x174 bytes = 0d374 bytes

10) Quin és el temps de còmput i el tamany de programa pel càlcul aproximat de la temperatura?

El tiempo de cómputo para realizar el cálculo es de **0.906 milisegundos**. El tamaño de la función que realiza el cálculo es de 194 bytes.

Idem al caso anterior

```
00007826    float ByFormula (int n) {
00007828       float vout = (n/1023.0)*5.0;
00007876       return (BETA/log(-((R2 - ((R2*VCC)/vout))/A))) - ABS_Z;
00007946    }

00007054    float ByRegresion(int n) {
00007056       return 100*(n/1024.0) - 24;
000070C0    }

00006F44    float getTemp (char mode) {
00006F46       ADCON0bits.GODONE = 1;
00006F48       while (ADCON0bits.GODONE);
00006F4C       int n = (ADRESH << 8) + ADRESL;
00006F5E       if (mode == 'F') return ByFormula(n);
00006F80       return ByRegresion(n);
00006F98    }
```

(0x6F98 - 0x6F44 + 0x1) + (0x70C0 - 0x7054 + 0x1) = 0xC2 bytes = 0d194 bytes

11) Explica com has linealitzat la NTC, el procediment que has seguit i els càlculs necessaris.

Hemos realizado una aproximación lineal, que básicamente se trata de hacer una regla de tres.

Primero de todo vamos a calcular el voltaje que aumenta por cada grado que se sube.

V50ºC = 3.72
V0ºC = 1.15

$$\frac{3.72-1.15}{50} = 0.0514V/ºC$$

Con esto hacemos una regla de tres con la n, que sabemos que 5V es n=1023 y 0V es n=0.

$$\frac{0.0514V}{1ºC} = \frac{1023}{5V} = 10.51\,n/ºC = 0.095089ºC/n$$

Sin embargo, con solo la pendiente no es necesario, ya que la n también incluye valores negativos.

Sabemos que cuando la temperatura es 0ºC, el voltaje es de 1.15V. Entonces con la relación que hemos encontrado anteriormente:

$$\frac{1.15}{0.0514} = 22.37354$$

Por lo tanto, tendremos que la relación lineal entre n y temperatura es TºC = 0.095089*n - 22.37354