FACULTAD DE INGENIERÍA
UNIVERSIDAD DE LA REPÚBLICA URUGUAY
NEON Network of Competence on Internet of Things
Co-funded by the Erasmus+ Programme of the European Union

Práctica 4: Consumo en microcontroladores

En está práctica se estudiarán los distintos modos de operación disponibles en un microcontrolador msp430 de Texas Instruments y se evaluarán diferentes técnicas para la reducción del consumo del mismo. Para verificar la efectividad de las técnicas se realizarán mediciones de consumo utilizando la herramienta EnergyTrace y un medidor externo. Para las pruebas se trabajará con una aplicación simple que mide la temperatura interna del procesador y lo transmite por el puerto serie.

Objetivos

Familiarizarse con las técnicas y herramientas de desarrollo de aplicaciones de bajo consumo y ultra-bajo consumo en microcontroladores.

Materiales

IDE para microcontroladores msp430 de Texas: Code Composer Studio (versión 8.3 o superior).

Placa de evaluación EXP-MSP430G2ET (launchpad) la cual posee un micro msp430g2553 y un deug probe eZ-FET con EnergyTrace (ET).

Fuente de tensión variable e instrumento para medir consumo con buena precisión (p.e. Otii o NGU201).

Aplicación

La aplicación a desarrollar consiste en un registrador de datos que mide periódicamente la temperatura interna del microcontrolador y la transmite por un puerto serie.

Para facilitar la implementación se proveerá parte del código de la aplicación resuelto. A continuación se describen mínimamente los archivos entregados, para entender su funcionamiento referirse al código y sus comentarios:

timer_hw.c/.h: Configuración y rutina de atención a la interrupción de un timer del microcontrolador.
temperatura.c/.h: Configuración y rutina de atención a interrupción del ADC conectado al sensor de temperatura. Función para leer el valor de temperatura medido.
uart.c/.h: Configuración de la UART y funciones para transmitir mensajes.
main.c: Programa principal que implementa la aplicación haciendo uso de las librerías anteriores.

Pasos previos

En primer lugar se deberá instalar el Code Composer Studio (v8.3 o superior) con soporte para la familia de microcontroladores msp430. El software puede descargarse de forma gratuita de: https://software-dl.ti.com/ccs/esd/documents/ccs_downloads.html

Una vez instalado, abrir el programa, seleccionar un workspace e importar el proyecto `ddlowp_lab4`. Sin modificar el código, compilar () e iniciar una sesión de debug () para probar su funcionamiento en el launchpad. Al ejecutar el código se debería observar que el led parpadea, y si se inicia una conexión al puerto serie de la placa se deberían recibir las medidas de temperatura junto a un timestamp. La configuración de la UART es:

> `9600 bps, sin paridad, 8 bits de datos, 1 bit de parada, sin control de flujo`.

Verificar que la posición de los jumpers entre el debug probe y el microcontrolador esté configurado para usar HW-UART (ver leyenda en la placa). Desde el CCS se puede iniciar una conexión a puerto serie en View>Terminal y luego presionar el botón “Open a Terminal” ().

Para medir el consumo, mientras se esté ejecutando el código abrir el EnergyTrace (), configurar un tiempo de captura () de 10s y comenzar la medida ().

Si comparan el resultado obtenido con los rangos de consumo de corriente de la hoja de datos (I_{AM} at 1MHz, 3V) observarán que están muy por encima. Antes de comenzar con las medidas de consumo realicen las siguientes modificaciones:

Consideraciones para reducir el consumo

- Desactivar los LEDs: Los LEDs son muy útiles a la hora de dar feedback al usuario (o para debugging), pero dado que tienen un consumo del orden de mA, en aplicaciones de ultra-bajo consumo su uso es prohibitivo. (Opcional: Desactivar el LED y comprobar con el ET la reducción del consumo).
- Configurar todos los puertos como salidas: Las pines se configuran por defecto como entradas y las entradas flotando consumen corriente. Para evitar esto configurar todos los pines como salidas en 0 (a excepción del P1.3 que en el launchpad posee un pull-up, por lo que debe ser configurado en nivel alto).

Mantenga estas modificaciones para el resto de la práctica.

Parte A: Frecuencia de Operación

En primer lugar se analizará cómo cambia el consumo al variar la frecuencia de operación del microcontrolador. La aplicación entregada tiene configurado como reloj del sistema el DCO (Digitally Controlled Oscillator) interno, a una frecuencia de 1MHz. Utilizando el EnergyTrace, medir el consumo del microcontrolador para el DCO configurado en 1MHz, 8MHz y 12MHz y 16Mhz.

Para configurar la frecuencia del DCO se deben escribir los registros: BCSCTL1 y DCOCTL.

Obtener la relación entre la potencia del microcontrolador en modo activo y la frecuencia del reloj (curva que ajuste los valores medidos en función de la frecuencia). Verificar que las medidas se encuentren en el orden de los valores provistos por el fabricante.

Parte B: Modos de bajo consumo (LPM)

Por la relación entre la frecuencia del reloj del sistema, el período de medida y la cantidad de ciclos que toma realizar la operación, la aplicación tiene un alto porcentaje de tiempo ocioso (consulta una bandera que está apagada en el loop infinito del archivo main.c). Para reducir el consumo se puede poner el microcontrolador en un modo de bajo consumo (LPM) mientras no tenga tareas pendientes. Al expirar la cuenta del timer, este interrumpe al microcontrolador y lo pone en modo activo (AM).

Para DCO configurado en 1 MHz, ponga el procesador al LPM0 mientras no tenga tareas pendientes. Para ello puede utilizar el siguiente macro dentro del loop principal:

```
__bis_SR_register(LPM0_bits + GIE);   // Voy a LPMx hasta que llegue la
próxima interrupción
```

Si bien las interrupciones despiertan al procesador, al finalizar las mismas se restaura el estado del sistema (guardado previamente en el stack). Esto significa que si el procesador se encontraba en el LPM0, luego de la ISR va a retornar a dicho modo. Por lo tanto, es necesario agregar la siguiente línea al final de la ISR (del ADC) a fin de que el procesador se mantenga despierto luego de la interrupción:

```
__bic_SR_register_on_exit(LPM0_bits);   // Borro LPMx bits del SR
```

Luego de las modificaciones ejecute la aplicación y mida el consumo usando el EnergyTrace. Observando la gráfica del ET verifique que el microcontrolador alterna entre modo activo (consumo alto) y LPM (consumo más bajo).

Compare el consumo en LPM0 con el de la hoja de datos[1]. ¿El consumo medido es acorde a los datos provistos por el fabricante? En caso de que no, aplique la siguiente sugerencia y vuelva a medir:

- Desconectar el debugger: El uso del debugger incrementa el consumo. Desconectar las líneas de Spy-Bi-Wire (SBW) del launchpad que conectan el microcontrolador con la parte del *emulator* (debugger), removiendo los jumpers SBWTCK y SBWTDIC. Es posible medir el consumo con ET sin tener activa una debug session. Simplemente presionar el ícono del ET en el modo CCS Edit con la placa conectada (y el programa previamente cargado).

Analizar: relación entre la potencia en LPM0 y curva obtenida en parte A.

[1] En la datasheet se da el valor para vcc=2.2V, la idea es comparar si da en el órden.

Parte C: Reducir tensión de alimentación

Para esta parte se debe alimentar el microcontrolador desde una fuente externa variable que permita configurar voltajes menores a 3.3V. Las medidas de consumo se deberán realizar utilizando un multímetro o un osciloscopio, ya que el ET no puede medir una fuente externa. Para ello desconectar todos los jumpers entre el microcontrolador y el debugger y conectar la alimentación entre los pines 3V3 y GND.

Para las siguientes tensiones de alimentación: 2.2V, 2.7V y 3.3V, determinar la máxima frecuencia de reloj a la que puede funcionar el microcontrolador (ver hoja de datos). Medir el consumo de la aplicación en los tres casos.

Parte D: Duty Cycle

En esta parte se generará un ciclo de trabajo artificial utilizando una interrupción periódica del timer (período **T**) y un bucle for en el loop principal que genera una demora (**d**) en función a la cuenta. Para ello se deberá configurar el timer en modo 'cuenta hasta CCR0' y utilizando el ACLK (sin divisor). Luego se deberá manejar el sleep de modo que el microcontrolador esté en modo activo un tiempo **d** y en LPM3 un tiempo **T-d**. A continuación se presentan las modificaciones a realizar con mayor detalle:

- Configurar el DC0 en 16MHz.
- Eliminar todo lo relacionado con la configuración y el uso de la uart y el adc en main.c.
- Configurar el timer para que interrumpa cada **T = 1s**:
  - en timer_hw.h, poner MAX_COUNT = 32767.
- Configurar el timer para que utilice el ACLK, sin divisor y en modo cuenta hasta CCR0:
  - TACTL = TASSEL_1 + ID_0 + MC_1;
- El loop principal de main.c debe quedar de la siguiente forma:

```
while(1) {
        for(i=0; i<DELAY_COUNT; ++i); //delay
        __bis_SR_register(LPM3_bits + GIE);
}
```

Encontrar el valor de **DELAY_COUNT** para que el ciclo de trabajo sea **d/T = 0.001**. Esto se puede lograr midiendo el ancho del pulso correspondiente al consumo en modo activo. Para ello conectar un shunt en serie en la entrada de 3.3V de la placa y medir la caída usando el AD2. Alternativamente se puede configurar un puerto del microcontrolador para que sea 1 mientras dure el modo activo y 0 el resto del tiempo. En este caso se mide el pulso generado por el puerto.

Una vez obtenido el ciclo de trabajo deseado, medir el consumo usando el ET. Repetir para ciclos de trabajo de 0.002 y 0.005.

En lugar de medir el consumo promedio usando el ET, también se puede calcular el consumo promedio usando el valor de **d** y los valores de consumo en modo activo y LPM3 medidos individualmente: Pavg = P_{AM}*d + P_{LPM3}*(T-d)

Se deberá entregar un reporte conteniendo lo siguiente:

1. Resultados de las medidas de todos los experimentos realizados. Aclarar en cada caso lo que se está probando (que parte de la práctica) y el escenario en que se tomó la medida. Puede ser útil generar tablas como las de la práctica anterior.

NEON
Network of Competence on Internet of Things
Co-funded by the Erasmus+ Programme of the European Union

Disclaimer: The European Commission support for the production of this website does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.