Sistemas Embebidos para Tiempo Real

Laboratorio 4 - Planificación por encolado de funciones

Introducción

En este laboratorio se implementará un registrador de temperatura involucrando comunicación serie asíncrona, el uso de un reloj de tiempo real y un sensor de temperatura. Para reducir el consumo se deberá configurar al microcontrolador en modo de bajo consumo (LPM3) con interrupciones habilitadas cuando no tenga ninguna tarea pendiente. Se deberán implementar dos versiones de la misma aplicación, usando: i) una arquitectura de software de Round-Robin con interrupciones, ii) una arquitectura de software de planificación por encolado de funciones con interrupciones.

Objetivos

Objetivos generales

1. Escribir aplicaciones usando las arquitecturas de software de Round-Robin con interrupciones y de planificación por encolado de funciones.
2. Crear aplicaciones modificando e integrando módulos escritos previamente, modularizando con correcta abstracción.
3. Desarrollar aplicaciones de bajo consumo.
4. Adquirir buenas prácticas de codificación, documentación, y control de versiones del código.

Objetivos particulares

1. Implementar una aplicación que utilice un reloj de tiempo real, un sensor de temperatura y se comunique vía UART.
2. Utilizar los modos de bajo consumo de un microcontrolador para bajar el consumo energético.
3. Verificar el funcionamiento en hardware de una aplicación típica básica.
4. Reconocer los beneficios de la programación modular y aplicarla adecuadamente en la integración con laboratorios anteriores
5. Utilizar la herramienta EnergyTrace para medir el consumo de corriente de la aplicación.
6. Analizar perfil de consumo de una aplicación embebida.
7. Optimizar el consumo considerando las sugerencias brindadas por la herramienta Ultra Low Power Advisor.
8. Reconocer los beneficios del “assert” y aplicarlo adecuadamente en el test de software embebido.
9. Mantener un repositorio con diferentes versiones de una aplicación utilizando GIT.
10. Generar la documentación del código utilizando Doxygen.

Material

- PC con el IDE CCS (v8.3 o superior), Git y herramientas asociadas.
- Repositorio grupal de los Laboratorios en Gitlab de Facultad de Ingeniería.
- Plataforma de desarrollo Launchpad MSP-EXP430G2ET o MSP-EXP430G2[1] basados en un MSP430G2553.

Descripción de la aplicación

La aplicación a desarrollar consiste en un registrador de datos que medirá periódicamente la temperatura, llevará un reloj de tiempo real y responderá a los comandos WP, RP, RT, WH y RH tal como se especificó en el Laboratorio 3. Para reducir el consumo, se agregará la funcionalidad para que el microcontrolador vaya a modo de operación LPM3 (con las interrupciones habilitadas) cuando no tenga ninguna tarea pendiente.

1 Se requiere que quienes cuentan con el launchpad MSP-EXP430G2 utilicen Windows 7, debido a problemas de compatibilidad reportados en Windows 10.

Se implementarán dos versiones de la aplicación, ambas basadas en interrupciones. La primera versión usará Round-Robin con interrupciones (como en el Laboratorio 3) y la segunda usará planificación por encolado de funciones.

Round-Robin con interrupciones

Esta versión de la aplicación se basará en el código implementado en el laboratorio anterior. Se agregará la funcionalidad de bajo consumo, llevando el microcontrolador a LPM3 con las interrupciones habilitadas cuando no haya procesamiento pendiente.

Se deberán modificar las ISR para que aquellas que “disparan” acciones posteriores (encendiendo banderas) dejen al microcontrolador en modo activo para continuar el procesamiento. Por lo tanto deben provocar que al retornar de la interrupción no se restaure el modo LPM3 cuando hay procesamiento pendiente.

Al retomar la ejecución en modo activo, en el “loop principal” se deben chequear las banderas y tomar la acción correspondiente. Cuando no haya procesamiento pendiente, se deberá volver a configurar al microcontrolador en LPM3.

Planificación por encolado de funciones

La versión con planificación por encolado de funciones se basará en un módulo a desarrollar que implemente una cola de punteros a funciones.

Una vez que se tenga implementado el módulo que maneja la cola de punteros a funciones se deberán realizar las siguientes modificaciones:

- La ISR del ADC encolará una función que implemente lo mismo que se realizaba en el ”loop principal” de la versión con Round-Robin al encender la bandera de temperatura.
- La ISR de recepción de la UART encolará una función que procese el comando recibido, una vez que llegue un mensaje completo.

Notas:

- Los archivos fuentes directamente relacionados con las aplicaciones de prueba (por ejemplo, para el encolado de funciones, o el procesado de comandos) se ubicarán en la carpeta `test`, mientras que los archivos de los módulos (.h y .c) que son reutilizados por diferentes aplicaciones (y probados con los test respectivos), se ubicarán en las carpetas `include` y `src`.
- Se recuerda que todos los archivos de encabezado deberán ser comentados utilizando la sintaxis Doxygen.

Parte 1) Round-Robin con interrupciones con LPM3

A) Lectura obligatoria de las sección 2.3 Operating Modes del MSP430x2xx Family User's Guide [3].

B) Revisión del código de los laboratorios anteriores implementando eventuales mejoras:
   - Módulo de reloj de tiempo real implementado en Laboratorios 1 y 2.
   - Módulo UART implementado en el Laboratorio 3.
   - Módulo de procesado de comandos basado en la aplicación del Laboratorio 3.
   - Módulo temperatura.

A) Modificar la aplicación desarrollada en el laboratorio 3 para ir a modo de bajo consumo LPM3:
   1. Crear un proyecto llamado `lab4-round-robin` en la carpeta `test/lab4-round-robin`.
   2. Hacer las modificaciones para que el microcontrolador vaya a modo de operación LPM3 tal como se explica en la descripción del sistema.
   3. Verificar el correcto funcionamiento del sistema probando todas sus funcionalidades (escribir y leer hora, escribir y leer período, leer temperatura).
   4. Crear en git un tag de nombre `lab4-parte1`

Parte 2) Planificación por encolado de funciones con LPM3

A) Desarrollo del módulo de software que implemente la cola de funciones (archivos .h y .c) y una aplicación de test.

   1. Se sugiere analizar los ejemplos de uso de punteros a funciones (puntero_funciones_v1.c y puntero_funciones_v2.c, disponibles en la página web del curso) y el módulo de cola de enteros del Ejercicio 3.1 y 3.2 de la sección Ejercicios complementarios disponible en EVA.
   2. Escribir un macro para realizar aserciones (assert) en la extracción de un elemento de una cola vacía y para el agregado de un elemento a una cola llena. Los “assert” se deben realizar al inicio de las funciones respectivas para verificar las condiciones anómalas.
   3. Para probar el módulo de planificación por encolado de funciones crear un proyecto de nombre `test-encolado-funciones` en la carpeta `test/lab4-test-encolado`. Al finalizar, para identificar la entrega en la historia del repositorio, crear en git un tag de nombre `lab4-`. La aplicación de test deberá encolar diferentes funciones para luego ir extrayéndolas para su ejecución. Para identificar la función que se ejecuta luego de la extracción de la cola, se sugiere ubicar *breakpoints* en las diferentes funciones que se encolan. Se deberá prever una forma de testear los “assert”.

B) Adaptar aplicación de la parte 1 para utilizar planificación por encolado de funciones:

   1. Crear un proyecto llamado `lab4-encolado-funciones` en la carpeta `test/lab4-encolado-funciones`.
   2. Hacer las modificaciones para utilizar planificación por encolado de funciones y para que el microcontrolador vaya a modo de operación LPM3, tal como se explica en la descripción del sistema. En particular, se modificará el módulo de temperatura para guardar un puntero a la función (‘callback function’) que luego encolará cuando se tenga una lectura pronta. Dicha función obtendrá una lectura del módulo de temperatura y se encargará de que se envíe el mensaje adecuado por UART.
   3. Verificar el correcto funcionamiento del sistema probando todas sus funcionalidades (escribir y leer hora, escribir y leer período, leer temperatura).
   4. Crear git un tag de nombre `lab4-parte2`.

Parte 3) Medida de consumo con Energy Trace

Para evaluar el consumo de corriente de la aplicación y un adecuado uso del modo de bajo consumo (LPM3) se utilizará el módulo Energy Trace del Launchpad y la funcionalidad asociada del CCS.

- Comenzar a debuggear cualquiera de las dos implementaciones y abrir el módulo Energy Trace (Menú Tools > Energy Trace)
- Verificar que la aplicación está entrando correctamente a LPM3 analizando los niveles de corriente. Observar los consumos máximos y mínimos de corriente. ¿Coinciden con lo reportado en la hoja de datos? ¿Por qué?
- Analizar las sugerencias del Ultra Low Power (ULP) Advisor que aparecen en la consola al compilar e implemente los consejos dados. Para la inicialización de los puertos se sugiere ver la conexión del puerto P1.3 en el Launchpad (esquemático de Figura 19 del MSP430G2553 LaunchPad™ Development Kit MSP-EXP430G2ET User's Guide [1]). Además se sugiere apagar el voltaje de referencia del ADC cuando no se está usando.
- Volver a observar los consumos máximos y mínimos de corriente. ¿Coinciden con lo reportado en la hoja de datos? ¿Por qué?
- Desconectar las líneas de Spy-Bi-Wire (SBW) del microcontrolador al launchpad, removiendo los jumpers SBWTCK y SBWTDIC.
- Volver a observar los consumos máximos y mínimos de corriente. ¿Coinciden con lo reportado en la hoja de datos? ¿Por qué?

Referencias

Manuales

1. MSP430G2553 LaunchPad™ Development Kit (MSP-EXP430G2ET) User's Guide
2. MSP430G2x53, MSP430G2x13 Mixed Signal Microcontroller datasheet (Rev. J)
3. MSP430x2xx Family User's Guide (Rev. J)
4. MSP430G2553 Device Erratasheet (Rev. I)
5. MSP430 Optimizing C/C++ Compiler v18.1.0 LTS User's Guide

Lecturas recomendadas

6. "An Embedded Software Primer", **Section 5.3:** Function-Queue-Scheduling Architecture.

Enlaces

7. Launchpad MSP-EXP430G2: http://www.ti.com/tool/msp-exp430g2et
8. MSP430G2553: http://www.ti.com/product/MSP430G2553
9. Code Composer Studio: http://www.ti.com/tool/ccstudio
10. Code Composer Studio para MSP430 :http://www.ti.com/tool/ccstudio-msp
11. TI Clouds tools: http://dev.ti.com
12. TI Resource Explorer: **http://dev.ti.com/tirex/**

FACULTAD DE INGENIERÍA UDELAR

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

NEON Network of Competence on Internet of Things

Co-funded by the Erasmus+ Programme of the European Union