Sistemas Embebidos para Tiempo Real

Laboratorio 3 - Round-Robin con Interrupciones

Introducción

En este laboratorio se implementará un módulo de comunicación serie asíncrona (UART, Universal Asynchronous Receiver Transmitter) y una aplicación basada en un sensor de temperatura. Se diseñará el sistema usando una arquitectura de software Round-Robin con interrupciones.

Objetivos

Objetivos generales

1. Escribir aplicaciones usando una arquitectura de software de tipo Round-Robin con interrupciones.
2. Escribir drivers de periféricos utilizando rutinas de atención a las interrupciones y modularizando con correcta abstracción.
3. Adquirir buenas prácticas de codificación, documentación, y control de versiones del código.

Objetivos particulares

1. Describir el funcionamiento de un puerto de comunicación serie asíncrono (UART).
2. Configurar y operar el periférico USCI del microcontrolador MSP430G2553 en modo UART.
3. Escribir rutinas de atención a las interrupciones del periférico UART.
4. Utilizar herramientas de depuración para reconocer el funcionamiento del protocolo UART con el Analog Discovery (AD2).
5. Verificar el funcionamiento en hardware de una aplicación basada en un módulo de comunicación serie.
6. Reconocer los beneficios de la programación modular y aplicarla adecuadamente en la integración con laboratorios anteriores.
7. Mantener un repositorio con diferentes versiones de una aplicación utilizando GIT.
8. Generar la documentación del código utilizando Doxygen.

Material

- PC con el IDE CCS (v8.3 o superior), Waveforms (AD2), Git y herramientas asociadas.
- Instrumento multifunción Analog Discovery 2.
- Repositorio grupal de los Laboratorios en Gitlab de Facultad de Ingeniería.
- Plataforma de desarrollo Launchpad MSP-EXP430G2ET o MSP-EXP430G2 basados en un MSP430G2553.
    - Usar ACLK proveniente del oscilador externo (32768 Hz).
    - Usar los módulos ya probados en laboratorios anteriores.

Descripción de la aplicación

La aplicación medirá periódicamente la temperatura y llevará un reloj de tiempo real. Se utilizará un sensor de temperatura interno al microcontrolador cuyo valor es adquirido por un conversor analógico-digital (ADC). Las lecturas de temperatura y parámetros de configuración serán enviadas a un PC a través de un puerto de comunicación serie asíncrono (UART). Se podrá configurar el período de muestreo para la adquisición de la temperatura y configurar la hora. Asimismo, se podrá solicitar el envío de la última medida del sensor, el período de muestreo y la hora actual.

La aplicación deberá enviar periódicamente una nueva medida de temperatura vía UART en texto plano (caracteres ASCII) y además recibirá comandos a través de la misma con el siguiente formato:

- WP xx Fija el período de adquisición (en ticks de 250 ms).
- WH Fija la hora del reloj (horas, minutos, segundos y milisegundos).

- RP Consulta el período de adquisición.
- RT Consulta la última medida de temperatura.
- RH Consulta la hora actual.

El sistema devolverá a través de UART los resultados de los comandos RP, RT y RH.

Se usará el módulo que implementa un reloj de tiempo real del laboratorio 2 (interrupción cada 250 ms con timer hardware), al que se le deberá modificar la ISR para cumplir con los requerimientos de la aplicación.

El módulo de adquisición de temperatura que deberá integrarse a la aplicación es el mismo que se utilizó en el laboratorio 2. También está disponible un módulo con funciones auxiliares para la conversión de enteros a arreglos de caracteres ASCII en la página web del curso. El módulo de comunicación UART deberá ser implementado por los estudiantes.

Observación: Inicialmente podrán asumir que el microcontrolador recibe el período de adquisición y la hora en el formato correcto.

Módulo UART

El módulo UART implementa la comunicación serie. La solución a implementar deberá usar interrupciones y utilizar algún mecanismo para evitar el problema de los datos compartidos.

El módulo de comunicación UART deberá configurar la misma en 9600 bps, 8 bits de datos, sin paridad y un bit de parada (8N1).

Descripción del funcionamiento utilizando interrupciones

Se dispondrá de un buffer de recepción y otro de transmisión. Sus tamaños se dimensionarán de acuerdo al tamaño máximo de los mensajes a enviar y recibir.

- Recepción, para recibir un mensaje se seguirá el siguiente mecanismo:
  1. Cada vez que se recibe por UART un carácter, la ISR asociada lo guarda en el buffer de recepción.
  2. Cuando se recibe el carácter de fin de mensaje, típicamente fin de línea ('\r'), se le indica con una bandera a la aplicación que el buffer de recepción contiene un mensaje completo.
- Transmisión, para transmitir un mensaje se utiliza una función que:
  1. Copia el mensaje a enviar en el buffer de transmisión.
  2. Envía el primer carácter y habilita las interrupciones de "registro vacío".
  3. En la ISR de “registro vacío” se obtiene del buffer de transmisión el carácter a ser transmitido y se copia el mismo en el registro de transmisión. Cuando se envía el último carácter, completando la transmisión de los datos, se deshabilita la interrupción correspondiente.

Parte 1) Módulo UART

1. Lectura obligatoria de las secciones que involucre la configuración del periférico UART del MSP430G2553 [2,3][1].
2. Implementación del código de los archivos del módulo UART y la aplicación de prueba descriptos a continuación[2].

Requerimientos:

- Módulo UART incluyendo:
  - Función de inicialización del módulo incluyendo la configuración del periférico UART con los parámetros indicados anteriormente, detallando el contenido de cada registro involucrado.
  - Función para transmitir una cadena de caracteres. La misma deberá copiar el arreglo a transmitir en el buffer de transmisión y luego realizar la misma según fue descripto anteriormente.
  - Función que copia el buffer de recepción a un buffer externo al módulo. Recibe como parámetro un puntero al buffer externo. Típicamente esta función es llamada por la aplicación cuando se recibe un mensaje completo por UART. La utilidad de esta función es liberar lo más rápido

1 Se recomienda consultar siempre, además de los manuales del microcontrolador (datasheet, guia de usuario, etc.), la hoja de erratas. En este caso MSP430G2553 Device Erratasheet [4].

2 Verificar que el Launchpad tiene los jumpers del bloque J101 en la posición adecuada para usar UART hardware (HW-UART). Se sugiere seguir las indicaciones de la inscripción en la placa (silkscreen) y el manual correspondiente [1].

posible el buffer de recepción por si llega otro mensaje.
    - Rutinas de atención a las interrupciones de transmisión y recepción tal cual se indicó anteriormente.
- test_uart.c con la función main
  Aplicación de prueba del módulo UART que espera a recibir un mensaje completo y luego lo retransmita por UART (echo).

Notas:

- Al iniciar las tareas previas de este laboratorio, crear un proyecto de nombre `lab3-parte1` en la carpeta `test/lab3-parte1`.
- Los archivos fuentes (`uart.h` y `uart.c`) deberán ser ubicados en las carpetas adecuadas (carpetas include, src y test).
- El archivo de encabezado del módulo deberá ser comentado utilizando Doxygen.
- Los pines de un microcontrolador pueden tener varios usos (input y output digitales) e incluso estar compartidos con un periférico (por ejemplo UART). Se deberá consultar la configuración de los registros P1SEL y P1SEL2 para que sean usados por el periférico UART [2,3].
- Verificar que los jumpers asociados al periférico UART estén colocados en el modo de HW UART.
- Para observar la comunicación serie se sugiere usar la terminal integrada de CCS. Se configura siguiendo los siguientes pasos:
  1. Menú View > Terminal
  2. Dentro de la ventana “Terminal”, clic en botón de "Open a Terminal".
  3. Seleccionar en "Choose Terminal”: "Serial Terminal"
  4. En “Serial Port” seleccionar el puerto correspondiente al puerto virtual (el otro corresponde a la interfaz para debug). El nombre depende del sistema operativo (por ejemplo en Windows será el correspondiente al MSP Application UART1 que puede verse en el administrador de dispositivos, en MacOS la opción a seleccionar es similar a /dev/cu.usbmodem1423 y en linux /dev/ttyACM).
  5. Seleccionar el baudrate adecuado (9600 bps) y luego clic en OK.
  6. En la ventana “Terminal” clic en "Toggle Command Input File” para poder enviar comandos.
- Para verificar la correcta configuración de los registros, el hardware y el uso del terminal de CCS, se recomienda que primero se centren en verificar que al transmitir un carácter desde la terminal es recibido correctamente por el microcontrolador (y viceversa).

Se deberá mostrar el código pedido, el cual deberá estar en el repositorio del grupo, identificado con el tag `lab3-parte1`.

Se deberá diseñar y realizar un demo de validación en que se pruebe del módulo UART utilizando la aplicación de prueba.

Parte 2) Depuración con AD2

En el desarrollo de aplicaciones que involucran protocolos de comunicación, es muy útil tener herramientas que permitan “espiar” el intercambio de mensajes para evaluar lo que está ocurriendo a nivel físico y lógico con las señales transmitidas. Esto mejora los tiempos de desarrollo y depuración.

A nivel físico: osciloscopio

1. Conectar el osciloscopio (Scope) del AD2 a los pines TXD y RXD del launchpad.
2. Enviar un mensaje conteniendo el carácter U (hexadecimal 0x55) usando test_uart.c desarrollado en la parte 1).
3. Observar el mensaje completo con el Scope de Waveforms en RXD y TXD.
4. Identificar el bit de inicio y de parada, decodificar el mensaje enviado. Deducir el baud-rate.
5. Guardar el Workspace y la Acquisition. En la defensa se pedirá que muestren estos archivos.

A nivel lógico: analizador lógico

6. Conectar el analizador lógico (Logic) del AD2 a los pines TXD y RXD del launchpad.
7. En el Logic de Waveforms seleccionar la opción “add channels” para configurar dos canales UART, uno de trasmisión y otro de recepción (seleccionar el baud-rate de forma manual).
8. Para poder visualizar el mensaje, configurar el trigger para que dispare por “Protocol->Idle”. Identificar el resto de las funcionalidades del trigger.
9. Realizar mismo análisis que en la parte 2.4 con el analizador lógico.
10. Guardar el Workspace y la Acquisition. En la defensa se pedirá que muestren estos archivos.

Notas:

• Recordar que el AD2 y el launchpad MSP-EXP430G2ET deben compartir una referencia de tierra (conectar GND del AD2 al GND del launchpad).

Parte 3) Integración con laboratorio 2

Crear un nuevo proyecto `lab3` en `test/lab3` integrando los módulos implementados en el laboratorio 2 y en la tarea previa de este laboratorio (archivos: `timer.h`, `timer.c`, `timer_hw.c`, `test_timer.c`, `temperatura.c` y `temperatura.h`) para implementar la aplicación pedida.

Prueba de modificaciones del timer

En esta parte se deberá modificar la ISR del timer del reloj de tiempo real del Laboratorio 2 para que cada cierta cantidad configurable de interrupciones, lance una medida de temperatura. Para verificar el correcto funcionamiento se puede integrar la prueba del módulo de temperatura haciendo un “loop infinito” que consulte la bandera que indica si hay una nueva medida. Mediante la observación de la variable (“watch”) que lleva la temperatura se podrá verificar si el módulo funciona correctamente.

Observación: Una forma de hacer variar la temperatura es tocar con el dedo el encapsulado del microcontrolador, que hará que se eleve la temperatura del mismo.

Parte 4) Aplicación final

Para la aplicación final se deberá:

- Enviar periódicamente (con una cierta frecuencia de adquisición inicial) una medida de temperatura por UART.
- Interpretar los comandos recibidos por UART y procesarlos correctamente. En particular:
  - Si se recibe el comando WP xx, se deberá cambiar el período de adquisición. xx indicará la cantidad de veces que tiene que interrumpir el timer para que se mande la medida de temperatura.
    Ejemplo: Si se recibe WP 20, se tomarán nuevas medidas cada 20 ticks, es decir cada 250 ms x 20 = 5 s.
  - Si se recibe el comando RP, se deberá enviar por UART el número de ticks actual.
  - Si se recibe el comando RT, se deberá enviar por UART la última medida de temperatura.
- Modificar la aplicación de forma tal que la presentación periódica de la temperatura incluya la hora actual con el formato:
  horas:minutos:segundos T=temperatura
- Agregar los siguientes comandos:
  - WH xx:yy:zz. Setea la hora en xx horas, yy minutos, zz segundos y configurando los milisegundos en cero.
  - RH. Envía la hora actual con el formato
    horas:minutos:segundos
- Para finalizar crear un tag en el repositorio de proyectos de nombre lab3.

Referencias

Documentos técnicos

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", Chapter 4: Interrupts
    1. Section 4.3: The Shared-Data Problem
    2. Section 4.4: Interrupt Latency
7. “MSP430 Microcontroller Basics”, Chapter 10: Communication, John Davies, 2008.

Enlaces

8. Launchpad MSP-EXP430G2: http://www.ti.com/tool/msp-exp430g2et
9. MSP430G2553: http://www.ti.com/product/MSP430G2553
10. Code Composer Studio: http://www.ti.com/tool/ccstudio
11. Code Composer Studio para MSP430 :http://www.ti.com/tool/ccstudio-msp
12. TI Clouds tools: http://dev.ti.com
13. TI Resource Explorer: http://dev.ti.com/tirex/
14. Test and Measurement Guide AD2 https://digilent.com/reference/test-and-measurement/guides/start

FACULTAD DE INGENIERÍA
UDELAR

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

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