Lab 4: UART - Introducción

1. Objetivo

Este laboratorio introduce a una de las características más importantes de los microcontroladores: la Comunicación Serial. Esta funcionalidad permite transmitir y recibir distintos tipos de contenido entre diferentes dipositivos, permitiendo un mayor nivel de control que usando simplemente GPIOs o señales analógicas. En esta oportunidad se trabajará con el protocolo de comunicación UART o *Universal Synchronous and Asynchronous Receiver and Transmitter*, más específicamente, con la transmisión de datos desde el microcontrolador al computador.

Al finalizar esta experiencia se espera que usted sea capaz de gestionar la transmisión de caracteres individuales y agrupados (*strings*) desde el microcontrolador a su computadora, familiarizándose con el protocolo UART y entendiendo sus ventajas y desventajas.

2. Descripción de la actividad

Esta experiencia estará separada en 2 partes, **cada una de ellas con un microcontrolador distinto.**

2.1. *Task* 1: Envío de caracteres individuales.

En esta primera tarea, deberá transmitir un caracter individual a la vez, desde su microcontrolador hacia alguna interfaz serial en su computador. Los caracteres o *char's* son elementos con un valor arbitrario de 8bits, cuyo valor se asigna de acuerdo al estándar dado por el código ASCII (v.g. en ASCII el valor 0x44 corresponde a la letra D).

En esta *Task* se le proveerá de un archivo de ejemplo preescrito, el cual corresponde a una librería para transmisión UART. Su labor consistirá en completar con el código faltante e integrar la librería al código principal. Una vez realizada esta labor, deberá configurar un programa terminal en su computador y comunicarlo con el microcontrolador a utilizar. De este modo, el resultado final debiese ser el envío de caracteres individuales desde el MCU hacia la interfaz del programa terminal.

Nota: Para llevar a cabo esta labor, necesitará de un programa terminal como minicom, gtkterm, CuteCom o RealTerm. Es de su responsabilidad la elección, instalación y familiarización con el programa que usted escoja.

2.2. *Task* 2: Punteros, *strings* y estructuras.

En esta *Task* se le proveerá de un segundo archivo de ejemplo preescrito. Nuevamente su labor consistirá en completar con el código faltante e integrar la librería al código principal. De este modo, al finalizar la experiencia contará con una nueva función integrada a su librería, la cual permite enviar un *string* completo al programa terminal.

En primer lugar, deberá completar el código de la nueva función, agregando las rutinas necesarias para permitir la transmisión efectiva de *strings*. En segundo lugar, deberá modificar su función de inicialización, de modo que esta sea capaz de recibir una variable de tipo *struct* y evaluarla. Este *struct* o estructura deberá contener los siguientes parámetros:

- *Baud Rate*: cualquier valor estándar entre 1200 y 57600 debiese de funcionar.
- Numero de *data bits*: de 5 a 8 según corresponda, el uso de 9 bits no es necesario.
- Numero de *stop bits*: 1 o 2.
- Paridad: ninguna, par o impar.

Se recomienda encarecidamente que la evaluación del *struct* sea hecha por medio de declaraciones *switch-case*. Además, si se entregan parámetros no válidos, retorne un mensaje de error.

Cuando logre implementar el código correctamente, el programa de prueba principal debería entregarle algunos mensajes al terminal. En este sentido, concéntrese en responder a la pregunta:

- ¿Por qué algunos mensajes se transmiten de forma correcta y otros no? (Identifique claramente cuáles).

Durante la ejecución de su programa puede que los ayudantes le soliciten que haga cambios a los parámetros de su protocolo, esté preparado para cambiarlos en caso de que así sea. Recuerde que su código debe ser lo suficientemente genérico como para aceptar cualquier combinación de parámetros válida.

3. Lectura recomendada

- ATmega328/P Complete Datasheet.
- MSP430x5xx and MSP430x6xx Family User's Guide.
- MSP430F552x, MSP430F551x Mixed-Signal Microcontrollers datasheet.
- Tutorial de C: material de utilidad para iniciarse en el mundo de la programación en C.

4. Pauta de Evaluación

4.1. Consideraciones generales

- El laboratorio será evaluado exclusivamente con nota 1.0 (**R**eprobado), 4.0 (**S**uficiente), 5.5 (**A**probado) y 7.0 (**D**istinguido). En ningún caso habrán notas intermedias.
- No se reciben trabajos después del módulo de presentación. Trabajos no entregados son calificados con nota 1.0 y son considerados dentro del criterio de aprobación del curso. La hora límite para inscribir a revisión es a las 10:10 hrs, posterior a esto se asignará una posición aleatoria.
- La nota **S**uficiente se otorgará en el caso de falla de una de las tareas de este laboratorio, quedando a criterio del ayudante. En caso de que un alumno haya decidido solamente hacer un 50 % del trabajo, se evaluará con un 1.0.
- Respecto a los puntos de aprobación acumulados, si un alumno obtiene una nota **S**uficiente, una mitad del puntaje queda asignada a Aprobación, el restante a Reprobación.

  A modo de ejemplo, este Laboratorio es nivel 3, si un alumno tiene Suficiente, 15 pts se acumularán a Aprobación y 15 pts será para Reprobación.
- Cualquier consulta sobre los criterios de evaluación de cada laboratorio debe ser realizada en las issues, donde estará disponible para que sea revisada por todos los alumnos.

4.2. Criterios de Aprobación

Se requiere cumplir con todos los puntos mencionados a continuación para poder aprobar. No existen casos excepcionales.

Funcionamiento de los requerimientos

El alumno realiza una presentación de su trabajo y se responsabiliza de exponer que este satisfaga todos los requerimientos mínimos solicitados en la *Descripción de la actividad*, los cuales incluyen en este laboratorio:

Requerimientos *Task* 1

- Programa compilado y ejecutándose.
- Implementación adecuada de las librerías de acuerdo a las especificaciones.
- Valor correcto del registro de configuración.
- Comprobación *in situ* del funcionamiento del código, revisado en terminal serial.
- Utilización de *headers*.

Requerimientos *Task* 2

- Programa compilado y ejecutándose.
- Implementación adecuada de las librerías de acuerdo a las especificaciones.
- Todas las configuraciones deben lograr transmisión adecuadamente, salvo dos. Dos de las configuraciones no funcionan. Debe especificar correctamente cuáles.
- Todos los registros de configuración de USART deben estar completos para ser susceptibles a prueba. **Estos serán usados prácticamente en la totalidad de experiencias restantes, razón por la cual se recomienda implementarlos, de ser posible, en ambos microcontroladores.**
- Utiliza un microcontrolador distinto al *task* anterior.

Preguntas

Se responde satisfactoriamente a 2 de 3 preguntas aleatorias al momento de la presentación final, las cuales abarcan los siguientes temas:

- Qué representa cada línea de código y en qué se traducen en el funcionamiento del programa.
- Funcionamiento de la macro #include. ¿Cómo se utilizan los archivos .h y los archivos .c? ¿Qué es el proceso de *linking* y cómo se realiza?
- Teoría general de funcionamiento del protocolo de comunicación USART **(¡nada específico del μC!)**: número de bits, paridad, estructura del mensaje, *baud rates*, etc.
- Funcionamiento de la implementación UART/USART en ambos μC de elección. Registros de configuración, *flags* de estado, registro de datos, número de módulos, etc.
- ¿Qué es un *struct*? ¿Cómo se representan? ¿Cómo se inicializa uno? ¿Qué ventajas entregan en la programación?
- Punteros: ¿Qué son? ¿Qué representan los operadores & y $*$? ¿Cómo se des-referencia?¿Cómo se recibe un puntero en una función? ¿Qué utilidad puede prestar esto? ¿Por qué en la función de envío se recibe un puntero y no un arreglo? ¿Por qué el uso de punteros representa un concepto fundamental en programación en C?
- ¿Qué es un *buffer* en transmisión de datos? ¿Dónde se ve implementado uno en el código de la experiencia? ¿En qué momento podría necesitarlo en su experiencia con microcontroladores?

Solo se dispone de una oportunidad para responder estas preguntas. Fallar en este requisito se traduce en la reprobación inmediata de la experiencia de forma inapelable.

4.3. Criterios de Distinción

La distinción representa un trabajo adicional que sobresale a los requerimientos mínimos para la aprobación. Agregados adicionales no constituyen por sí mismo una distinción si no representan un verdadero trabajo adicional de comprensión y/o análisis.

Los trabajos distinguidos pueden caer (no exclusivamente) en algunas de las siguientes líneas generales:

- Funcionalidades creativas :D
- Funcionalidades sobresalientes, en la línea de recepción, transmisión de texto de largo indefinido (mucho texto) o interrupciones.
- Implementación de otros protocolos de comunicación.
- **Portabilidad de código**: Un código es portable si el código fuente en C del laboratorio puede ser compilado y cargado en cualquiera de los microcontroladores del curso de forma indistinta, **sin hacer ninguna modificación a dicho código**[1].

Para optar a distinguido en este laboratorio debe realizar dos ideas distintas entre si y respecto a laboratorios anteriores.

Las Distinciones son discutidas caso a caso por la totalidad del equipo de ayudantes al finalizar la corrección del laboratorio. Serán notificadas públicamente después del módulo de evaluación.

[1] Hint