UNIVERSIDAD NACIONAL DE COLOMBIA
Facultad de Ingeniería
Departamento de Mecánica y Mecatrónica
Microcontroladores

Laboratorio 7: Comunicación Serial, Interfaz MCU-PC

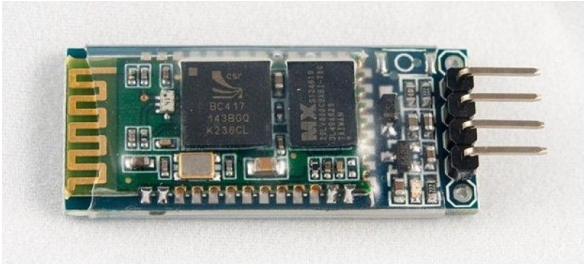
Fecha de publicación:
22/05/2019

Fecha de entrega:
29/05/2019

El objetivo de esta práctica es trabajar con el módulo de comunicaciones serial del QE16, este módulo nos permite enviar y recibir datos (1 byte por *transacción* en el modo más simple) entre dispositivos que empleen el mismo tipo de comunicación, nosotros tendremos el caso específico de implementación de comunicación serial entre MCU-PC de manera bilateral por medio de una **interfaz gráfica**, además de que será una conexión wireless a través del uso de un módulo Bluetooth.

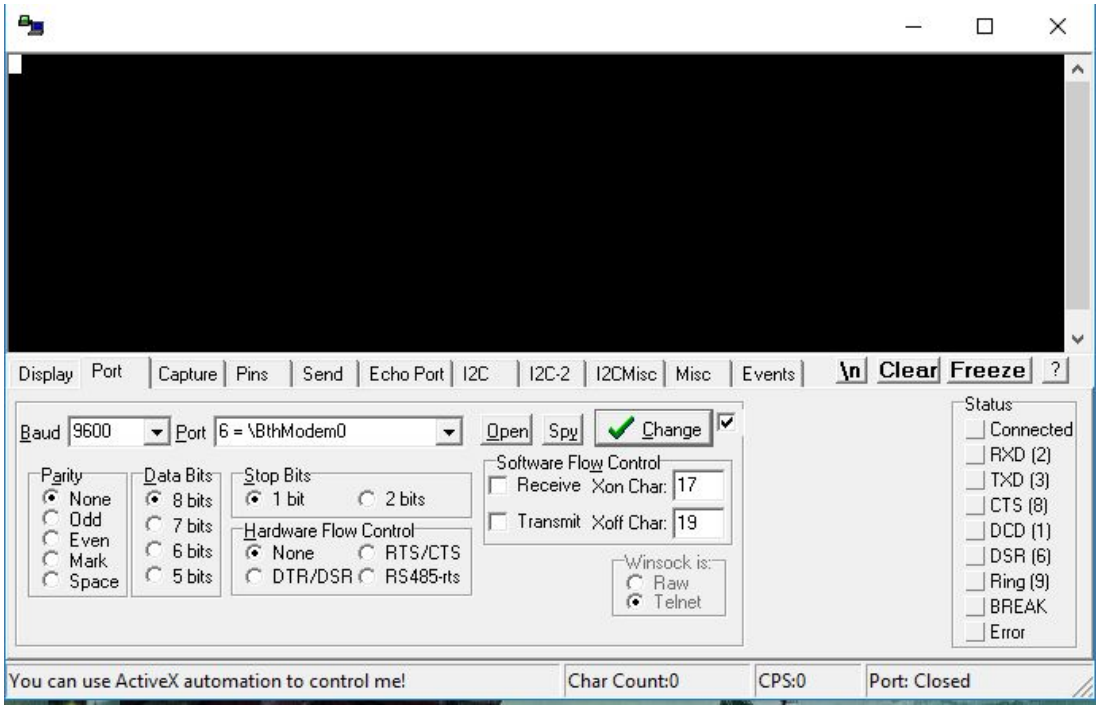
Módulo Bluetooth

Existen muchos dispositivos que implementan comunicación inalámbrica, del mecanismo del Bluetooth tenemos el **módulo HC-06** el cual es de los más usuales en proyectos con microcontroladores.



-Info Recomendada: Using the HC-06 Bluetooth Module, se trata de cubrir todos los aspectos referentes al módulo, desde la configuración de este por medio de comandos AT hasta la instalación como dispositivo en el PC, sin embargo es bueno que busquen videos donde puedan explicar mejor el uso del módulo. (Configuración nombre y velocidad).
Importante: Este módulo funciona a 5V, este tiene un regulador ya integrado para alimentar su sistema con 3.3V, pero el regulador debe recibir una entrada de al menos 5V.

Después de su configuración y emparejamiento solo basta con conectar las salidas RX y TX al MCU y escribir datos al módulo SCI, el Bluetooth se encargará de enviarlas al PC. A modo de prueba pueden usar **RealTerm** como terminal para ver los datos que se reciben del puerto COM al que está conectado el Bluetooth, desde la terminal pueden configurar varios parámetros como el puerto a utilizar, la velocidad a la que se realiza la transacción (baudios) y el tipo de dato que se espera recibir para su correcta visualización.



Processing

Processing es un entorno de programación basado en JAVA orientado a aplicaciones visuales y temas relacionados con el arte y la tecnología, si bien no es una plataforma muy "*ingenieril*" nos da desde un punto de vista académico varias herramientas que facilitan la creación de interfaces gráficas, entre las muchas otras posibilidades que ofrece este software.

El componente principal del que haremos uso en processing es la librería Serial, librería nativa que controla las transacciones del puerto Serial implementando varios métodos (*myPort.read();*) y una función **Event** que sería el equivalente a una interrupción en un MCU.

```
int[] arr = new int[4];

void serialEvent (Serial port){
  arr[cont]=myPort.read();
  cont++;
  if(cont==4){
    listo();
  }
}
```

(Ejemplo en el que se leen 4 bytes enviados desde el MCU y se guardan en un arreglo, **serialEvent** ocurre cada que se recibe un byte en el puerto serial).

Módulo SCI: Serial Communications Interface (S08SCIV4)

El módulo SCI es el responsable de implementar el Hardware para realizar las transacciones de comunicación serial. Este es un protocolo que ya tiene un buen tiempo, por lo que tiene muchas funcionalidades y opciones de configuración.

La primera y la más fundamental es la configuración de la velocidad o el BAUD RATE, esta debe ser la misma que el dispositivo receptor de la comunicación. Según la hoja de datos tenemos la siguiente relación para configurar el BAUD RATE:

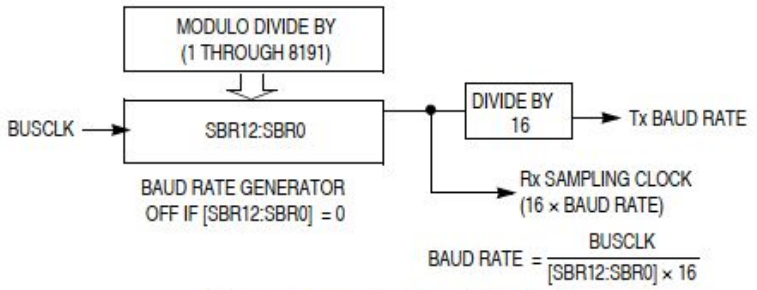


Figure 14-13. SCI Baud Rate Generation

Para nuestro caso, si queremos una velocidad de 9600 usando la velocidad de BUSCLK del QE16 de 4MHz tendríamos que:

$$[SBR12 : SBR0] = \frac{4MHz}{9600*16} \approx 26$$

Dado que el reloj no es muy preciso, el monitor obtiene los 9600 configurando los bits con un 27, a veces toca hacer pequeños ajustes de manera manual.

Enviar datos (Tx): Para el envío de datos basta con escribir al registro SCIxD, para esto existen dos métodos.

Enviar por Polling (TDRE): En este modo ponemos a el MCU a que espere a que el registro esté vacío y listo para escribirle el dato a enviar, esto lo hacemos con un *while* de la siguiente manera:

```
while(SCI2S1_TDRE == 0);
SCI2D = dato;
```

Al ser utilizada esta estructura nos limitamos a la velocidad con la que se despachan los bits del byte a enviar y perdemos tiempo de procesamiento ya que mientras el bit TDRE sea 0 el MCU se quedará dentro del *while*.

Enviar por Interrupción (TIE): Este es el método más adecuado ya que no ponemos a el MCU a que espere a que se realice la transacción, sino que cuando el

registro haya terminado de enviar el último bit se genere una interrupción, si son varios datos la idea es ponerlos en un *"buffer"* para que la interrupción los vaya enviando uno por uno de forma *automática*.

Recibir datos (Rx): Los datos que se reciben llegan al mismo registro SCIxD, para obtenerlos solo debe leerse el registro y guardar el dato en una variable.

Recibir por Polling (RDRF): De la misma manera que el envío por polling, se utiliza un while para esperar que el registro se llene, cuando está lleno es porque se ha recibido un byte. Este método jamás debe ser implementado ya que nunca se sabe cuando se va a recibir un dato, a menos que sea una situación muy específica, dejariamos el MCU en un while *eterno* hasta que se reciba un dato.

Recibir por interrupción (RIE): Este es el método idóneo para la tarea, al recibir datos por medio de un mecanismo de interrupción podemos usar el MCU para otras tareas mientras no se están recibiendo bytes. Se implementa de la siguiente manera:

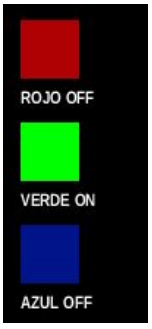
```
21 interrupt VectorNumber_Vsci2rx void sci_rx(){
22     flag_rx=SCI2S1_RDRF;
23     if(SCI2D == 101){
24             PTED_PTED0 = !PTED_PTED0;
25         }
26 }
```

Como siempre se debe realizar el proceso de Acknowledge de una interrupción, para el SCI_rx se debe leer el bit RDRF usando una variable dummy para la acción. Luego ya se procede a el manejo de datos leyendo el SCIxD.

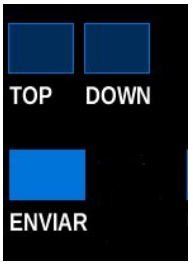
Esta es como la información básica en cuanto a el módulo, tiene muchas más opciones y funcionalidades que deben explorar en la hoja de datos.

1. **Procedimiento:**
   Se debe implementar una interfaz gráfica que permita hacer las siguientes tareas (demostración de lo aquí requerido en el siguiente video):
   a. Controlar la conexión o el emparejamiento a el MCU (a través del Bluetooth) con dos botones, uno para iniciar las comunicaciones y otro para detener las comunicaciones.

   b. Controlar un LED RGB: Con tres "botones" de la interfaz se debe controlar el encendido y apagado individual de cada color de un LED RGB.



c. Se debe controlar un **cronómetro descendente** por medio de dos campos de texto donde se indique el tiempo desde donde iniciará este, se implementará un botón para enviar el dato a el MCU.

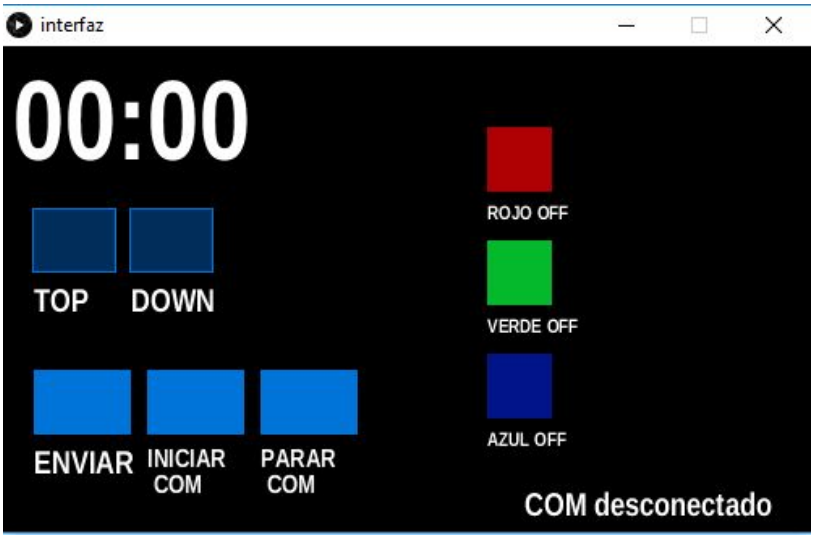


d. Con el valor de partida ya configurado se debe poder controlar el cronómetro descendente con dos botones de KBI para inicio y parada tal como se hizo en la práctica anterior.
e. Después de dejar correr el cronómetro y detenerlo en algún tiempo arbitrario, se deberá enviar el valor en el que se encuentre el cronómetro a el PC y este debe visualizarse en la interfaz.

f. Finalmente un botón que resetee el cronómetro por interrupción IRQ.
g. Como siempre el código debe ir debidamente documentado además de que deben emplearse técnicas de programación como el uso de etiquetas.
h. Diagramas de flujo y esquemas de conexiones.

Todo esto esto debe hacerse utilizando un módulo **Bluetooth (HC-06)** como intermediario entre MCU-PC.

(Este es un ejemplo de cómo se vería la interfaz completa, no es obligatorio que lo hagan igual en el aspecto visual, se promueve que lo hagan de forma más refinada y original).



2. **Preguntas:**
    a. ¿Que significa comunicación unidireccional?, ¿que sería comunicación bidireccional?
    b. ¿Que es un protocolo de comunicación sincrono y asincrono?
    c. ¿Que es *baud rate*?
    d. ¿Qué es *bit rate*?
    e. ¿Qué es código ASCII?
    f. Realice un diagrama de flujo para un algoritmo que envíe el nombre en código ASCII de uno de los integrantes del grupo por medio del módulo serial a un receptor como un PC.

3. **Bibliografía:**
    a. Processing y Serial COM
    b. Datasheet MCU QG8
    c. Datasheet MCU QE16

4. **Tabla requisitos**

Requisito	**Cumple**	**No cumple**
Control RGB por interfaz		
Configuración valor inicial del cronómetro desde la interfaz		
Envio de dato del cronómetro hacia la interfaz		

Uso Módulo Bluetooth.		
Control del cronómetro con botones KBI y IRQ		

5. **Código ejemplo para el laboratorio**
   Como código ejemplo el material disponible en el Moodle se encuentra un código para controlar un LED desde una interfaz gráfica en **Processing** y recibir la entrada de un pulsador desde el MCU en la misma interfaz.
   Está el código tanto de Processing como del MCU.

   Deben descargar la libreria **ControlP5** de processing ya que esta se utiliza para implementar los botones y lo métodos que controlan estos.