Práctica 9

Programación del uC del periférico de comunicación serie utilizando interrupciones.

Objetivo: Mediante esta práctica el alumno aprenderá el uso básico para inicializar y operar, bajo un esquema de interrupciones, el puerto serie del microcontrolador.

Equipo:
- Computadora Personal
- Módulo T-Juino

Teoría:
- Manejo del Periférico de Comunicación Serie 0 (UART0) del microcontrolador ATmega1280/2560
- Secuencias de escape ANSI.

Actividades a realizar:

1. **Bajo los esquemas de la Figura 1 y Figura 2, implemente el manejador para recibir y enviar datos vía puerto serie 0 (UART0) utilizando interrupciones.**

Transmisión y Recepción de datos bajo el uso de interrupciones. Para hacer uso eficiente de la transmisión y recepción de datos bajo el esquema de interrupciones es necesario una cola circular para transmitir, y otra para recibir.

En la cola de transmisión se introducen los datos a transmitir para que después la rutina de servicio de interrupción (ISR) correspondiente los tome al momento adecuado para su transmisión. En la Figura 1 se observa que la función ***UART0_putchar( )*** es la que introduce cada dato a la cola circular **siempre y cuando exista lugar para hacerlo**; de lo contrario tendrá que esperar a que se libere un lugar.

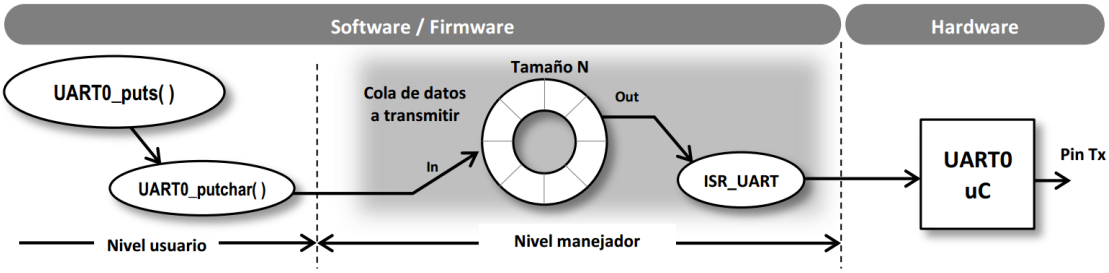

Figura 1. Esquema para el manejo de paquetes de transmisión mediante cola circular.

Listado 1. Tipo de dato **ring_buffer_t**.Y

```
#define BUFFER_SIZE   64

typedef struct {
  char buffer[BUFFER_SIZE];          /* espacio reservado          */
  volatile unsigned char in_idx;     /* indice entrada (Head)      */
  volatile unsigned char out_idx;    /* indice entrada (tail)      */
} ring_buffer_t;
```

La ISR correspondiente a la transmisión del UART0 es la que tiene la responsabilidad de tomar dato por dato de la cola circular e introducirlos al UART0, esto a cada momento que el registro UDR0 (de transmisión) esté libre para poder escribir en dicho registro. Si por algún motivo no existen datos en la cola circular la ISR debería de deshabilitar su interrupción correspondiente para evitar que continuamente se invoque la interrupción.

Considerando lo anterior, implementar la siguiente rutina:

a) **ISR(SIG_USART0_DATA)** : Rutina de servicio de interrupción para el evento de **registro de transmisión vació**. Esta rutina **extrae de la cola circular** el dato que será transmitido por el UART0.

Y modificar:

b) `void` **`UART0_putchar`**`(char data)`: Función que coloca el dato en la cola circular.

En la cola de recepción, la ISR es quien introduce los datos recibidos. En la Figura 2 se observa que UART0_getchar( ) es quien extrae los datos de la cola. implemente el manejador para recibir datos vía puerto serie 0 (UART0). Considera que se requiere la implementación de un mecanismo de recepción almacenar los datos en la cola circular por lo que se requieren las siguientes funciones.

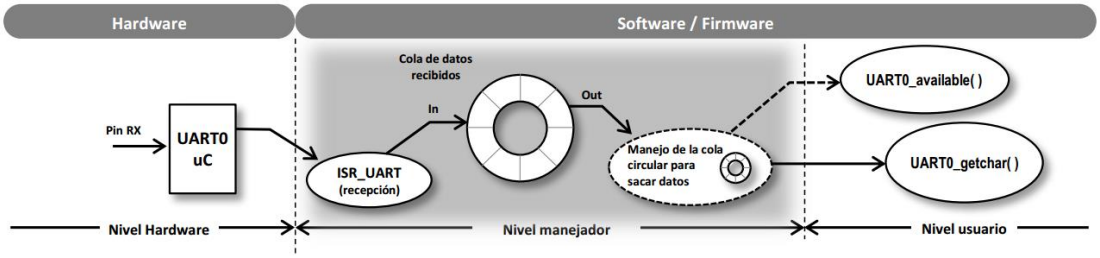

Figura 2. Esquema para el manejo de paquetes de recepción mediante cola circular.

Bajo el esquema de la Figura 2 implemente el manejador para recibir datos vía puerto serie 0 (UART0). Considera que se requiere la implementación de un mecanismo de recepción almacenar los datos en la cola circular por lo que se requieren las siguientes funciones:

c) ISR(SIG_USART0_RECV): Rutina de servicio de interrupción para el evento de **recepción completa**. Esta rutina **inserta a la cola circular** el dato que fue recibido por el UART0.

d) uint8_t **UART0_available**(void): Función que retorna 1 si existe(n) dato(s) en la cola circular.

e) `char` **`UART0_getchar`**`(void)`: Función toma el **dato correspondiente a salir de la cola circular**. Esto si existe alguno dato en la cola circular, de lo contrario espera a que exista uno para tormarlo y retornarlo.

2. Implementar la siguiente función:

```
UART_Ini(uint8_t com, uint16_t baudrate, uint8_t size,
         uint8_t parity, uint8_t stop)
```

Función que inicializa el periférico del UART en un esquema de interrupciones. Y la configuración es dada por los parámetros, donde:

- **com**: representa el número de UART a configurar. Considerar 0 y 1.
- **baudrate**: representa la velocidad en Baud de configuración, puede ser no estándar.
- **size**: representa el número de bits de los datos con los que operará el UARTx. Considerar de 5 a 8 bits.
- **parity**: representa el tipo de paridad con los que operará el UARTx. Considerar 0: No paridad, 1: impar, 2: par.
- **stop**: representa el número de bits de paro con los que operará el UARTx. Considerar 1 ó 2.

3. Reutilizar las funciones de la Práctica 2:

a) `void gets(char *str)` → `void UART0_gets(char *str)`
Función que retorna una cadena haciendo uso de *UART0_getchar()*, la cadena se retorna en el apuntador *str*.

b) `void puts(char *str)` → `void UART0_puts(char *str)`
Función que imprime una cadena mediante *UART0_putchar()*.

c) `void itoa(char* str, uint16_t number, uint8_t base)`

d) `uint16_t atoi(char *str)`

4. En base a las secuencias de escape, implementar las siguientes funciones:

a) `void clrscr( void )`
Funcion que limpia la terminal mediante la secuencia de escape.

b) `void setColor(uint8_t color)`
Función que envía la secuencia de escape para configurar el color del texto que se desplegará en la terminal.

c) `void gotoxy(uint8_t x, uint8_t y)`
Función que posiciona el cursor en la terminal en la coordenada *x,y* que lleguen como parámetro, utilizando la secuencia de escape.

Utilizar el Listado 2 para comprobar el funcionamiento de las funciones.

Listado 2.

```
#include <avr/io.h>
#include "UART.h"

int main( void )
{
      char cad[20];
      uint16_t num;
      UART_Ini(0,12345,8,1,2);
      while(1) {
            UART0_getchar();
            clrscr();
            gotoxy(2,2);
            setColor(YELLOW);  //Definirlo en UART.h
            UART0_puts("Introduce un número:");
            gotoxy(22,2);
            setColor(GREEN);   //Definirlo en UART.h
            UART0_gets(cad);
            num = atoi(cad);
            itoa(cad,num,16);
            gotoxy(5,3);
            setColor(BLUE);    //Definirlo en UART.h
            UART0_puts("Hex: ");
            UART0_puts(cad);
            itoa(cad,num,2);
            gotoxy(5,4);
            UART0_puts("Bin: ");
            UART0_puts(cad);
      }
}
```

Comentarios y Conclusiones.

Bibliografía.