Universidad Autónoma de Baja California

Facultad de ciencias químicas e ingeniería

Materia.

Microprocesadores y Microcontroladores.

Maestro.

Garcia Lopez Jesus Adan.

Alumno

Gonzalez Cardiel Luis Enrique

Matricula:

1217258

Grupo:

561

Trabajo:

Practica No. 9

09/11/2018

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.

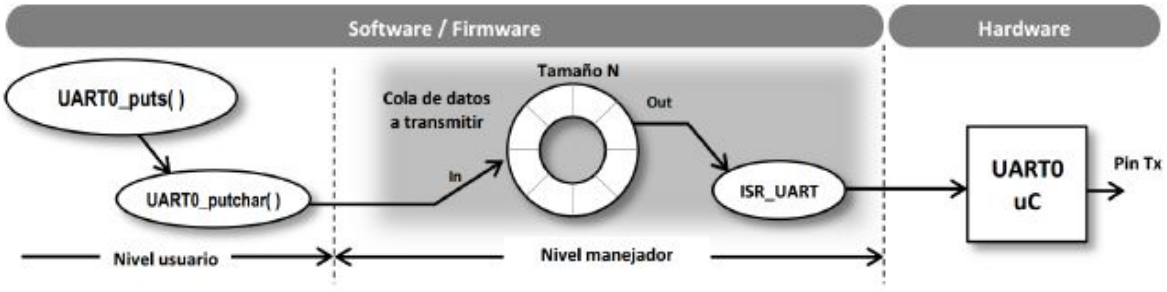

Figura 1. Esquema para el manejo de paquetes de transmisión mediante cola circular.

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.

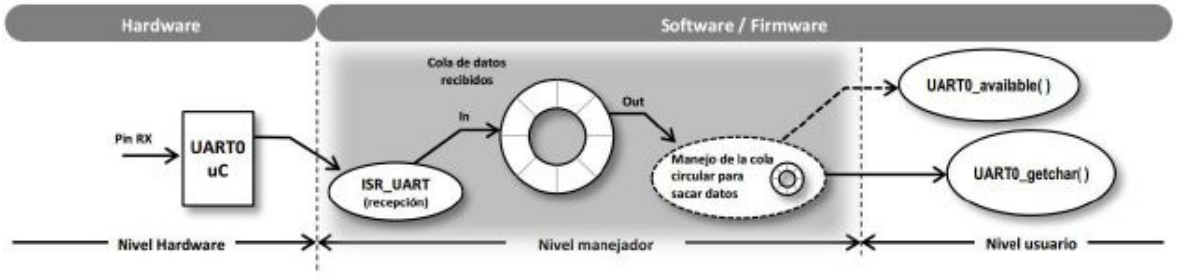

Figura 2. Esquema para el manejo de paquetes de recepción mediante cola circular.

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);
	}
}
```

Teoria:

Manejo del Periférico de Comunicación Serie 0 (UART0) del microcontrolador ATmega1280/2560

(0x136)	UDR3	USART3 I/O Data Register								page 218
(0x135)	UBRR3H	-	-	-	-	USART3 Baud Rate Register High Byte				page 222
(0x134)	UBRR3L	USART3 Baud Rate Register Low Byte								page 222
(0x133)	Reserved	-	-	-	-	-	-	-	-	
(0x132)	UCSR3C	UMSEL31	UMSEL30	UPM31	UPM30	USBS3	UCSZ31	UCSZ30	UCPOL3	page 235
(0x131)	UCSR3B	RXCIE3	TXCIE3	UDRIE3	RXEN3	TXEN3	UCSZ32	RXB83	TXB83	page 234
(0x130)	UCSR3A	RXC3	TXC3	UDRE3	FE3	DOR3	UPE3	U2X3	MPCM3	page 233
(0xD6)	UDR2	USART2 I/O Data Register								page 218
(0xD5)	UBRR2H	-	-	-	-	USART2 Baud Rate Register High Byte				page 222
(0xD4)	UBRR2L	USART2 Baud Rate Register Low Byte								page 222
(0xD3)	Reserved	-	-	-	-	-	-	-	-	
(0xD2)	UCSR2C	UMSEL21	UMSEL20	UPM21	UPM20	USBS2	UCSZ21	UCSZ20	UCPOL2	page 235
(0xD1)	UCSR2B	RXCIE2	TXCIE2	UDRIE2	RXEN2	TXEN2	UCSZ22	RXB82	TXB82	page 234
(0xD0)	UCSR2A	RXC2	TXC2	UDRE2	FE2	DOR2	UPE2	U2X2	MPCM2	page 233
(0xCF)	Reserved	-	-	-	-	-	-	-	-	
(0xCE)	UDR1	USART1 I/O Data Register								page 218
(0xCD)	UBRR1H	-	-	-	-	USART1 Baud Rate Register High Byte				page 222
(0xCC)	UBRR1L	USART1 Baud Rate Register Low Byte								page 222
(0xCB)	Reserved	-	-	-	-	-	-	-	-	
(0xCA)	UCSR1C	UMSEL11	UMSEL10	UPM11	UPM10	USBS1	UCSZ11	UCSZ10	UCPOL1	page 235
(0xC9)	UCSR1B	RXCIE1	TXCIE1	UDRIE1	RXEN1	TXEN1	UCSZ12	RXB81	TXB81	page 234
(0xC8)	UCSR1A	RXC1	TXC1	UDRE1	FE1	DOR1	UPE1	U2X1	MPCM1	page 233
(0xC7)	Reserved	-	-	-	-	-	-	-	-	
(0xC6)	UDR0	USART0 I/O Data Register								page 218
(0xC5)	UBRR0H	-	-	-	-	USART0 Baud Rate Register High Byte				page 222
(0xC4)	UBRR0L	USART0 Baud Rate Register Low Byte								page 222
(0xC3)	Reserved	-	-	-	-	-	-	-	-	
(0xC2)	UCSR0C	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOL0	page 235
(0xC1)	UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80	page 234
(0xC0)	UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0	page 234

USART Block Diagram(1)

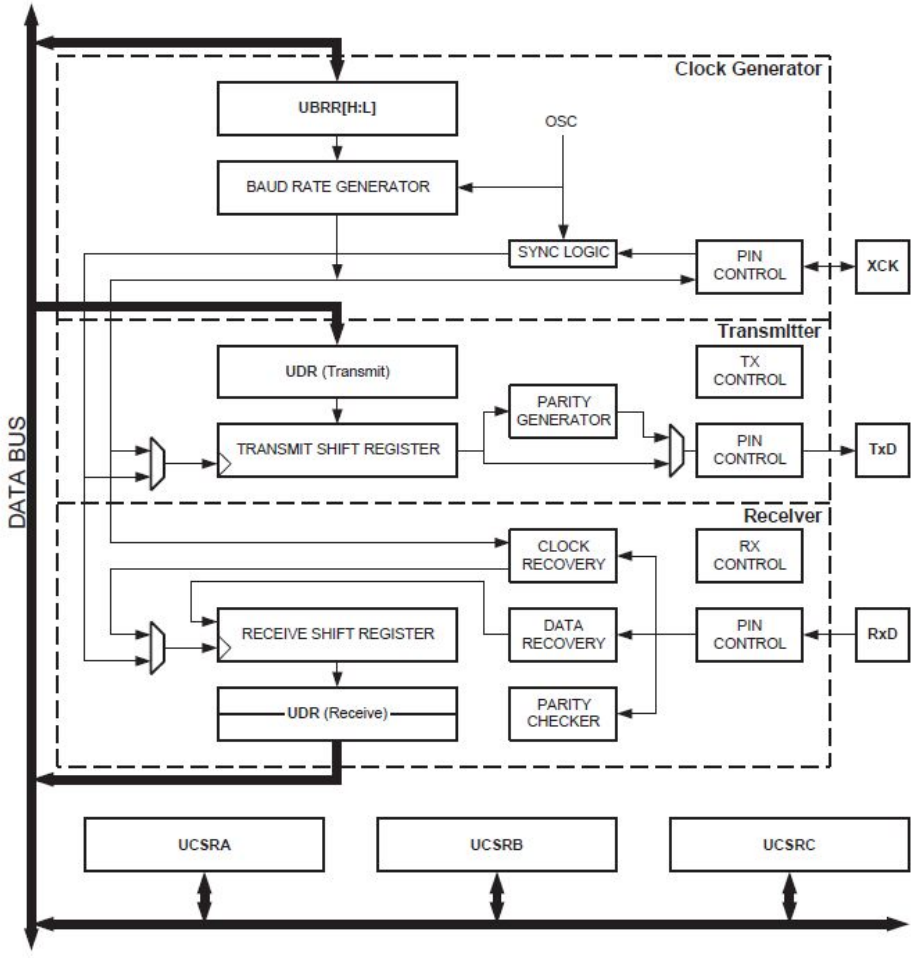

Table 22-1. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Asynchronous Normal mode (U2Xn = 0)	$BAUD = \frac{f_{OSC}}{16(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed mode (U2Xn = 1)	$BAUD = \frac{f_{OSC}}{8(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous Master mode	$BAUD = \frac{f_{OSC}}{2(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{2BAUD} - 1$

Nota: 1. La velocidad en baudios se define como la velocidad de transferencia en bits por segundo (bps).

BAUD Baud rate (en bits por segundo, bps).

fOSC Sistema de frecuencia de reloj del oscilador.

UBRRn Contenido de los registros UBRRHn y UBRRLn, (0-4095).

Operación de doble velocidad (U2Xn)

La velocidad de transferencia se puede duplicar estableciendo el bit U2Xn en UCSRnA. La configuración de este bit solo tiene efecto para la operación asíncrona. Establezca este bit a cero cuando use la operación síncrona.

La configuración de este bit reducirá el divisor del divisor de la velocidad de transmisión de 16 a 8, duplicando efectivamente la velocidad de transferencia para la comunicación asíncrona. Sin embargo, tenga en cuenta que, en este caso, el Receptor solo usará la mitad del número de muestras (reducido de 16 a 8) para el muestreo de datos y la recuperación del reloj, y por lo tanto, se requiere una configuración de la velocidad en baudios y un reloj del sistema más precisos cuando se usa este modo. El transmisor, no hay desventajas.

UCSRnA – USART Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

• Bit 7 - RXCn: USART Receive Complete

Este bit de indicador se establece cuando hay datos no leídos en el búfer de recepción y se borra cuando el búfer de recepción está vacío (es decir, no contiene ningún dato no leído). Si el receptor está deshabilitado, el búfer de recepción se vaciará y, en consecuencia, el bit RXCn se pondrá a cero. El indicador RXCn se puede utilizar para generar una interrupción de recepción completa (consulte la descripción del bit RXCIEn).

• Bit 6 - TXCn: USART Transmit Complete

Este bit de indicador se establece cuando todo el cuadro en el Registro de cambio de transmisión se ha desplazado y no hay datos nuevos presentes en el búfer de transmisión (UDRn). El bit del indicador TXCn se borra automáticamente cuando se ejecuta una interrupción de transmisión completa, o se puede borrar escribiendo un uno en su ubicación de bit. El indicador TXCn puede generar una interrupción de transmisión completa (consulte la descripción del bit TXCIEn).

• Bit 5 - UDREn: Registro de datos USART vacío

El indicador UDREn indica si el búfer de transmisión (UDRn) está listo para recibir nuevos datos. Si UDREn es uno, el búfer está vacío y, por lo tanto, está listo para ser escrito. El indicador UDREn puede generar una interrupción de registro de datos vacío (consulte la descripción del bit UDRIEn).

UDREn se configura después de un reinicio para indicar que el transmisor está listo.

• Bit 4 - FEn: Error de trama

Este bit se establece si el siguiente carácter en el búfer de recepción tuvo un error de trama cuando se recibió, es decir, cuando el primer bit de parada del siguiente carácter en el búfer de recepción es cero. Este bit es válido hasta que se lee el búfer de recepción (UDRn). El bit FEn es cero cuando el bit de parada de los datos recibidos es uno. Siempre establezca este bit a cero al escribir en UCSRnA.

• Bit 3 - DORn: Data OverRun

Este bit se establece si se detecta una condición de OverRun de datos. Un Data OverRun se produce cuando el búfer de recepción está lleno (dos caracteres), es un nuevo carácter en espera en el registro de cambio de recepción y se detecta un nuevo bit de inicio. Este bit es válido hasta que se lee el búfer de recepción (UDRn). Siempre establezca este bit a cero al escribir en UCSRnA.

• Bit 2 - UPEn: Error de paridad USART

Este bit se establece si el siguiente carácter en el búfer de recepción tuvo un Error de paridad cuando se recibió y la Comprobación de paridad se habilitó en ese punto (UPMn1 = 1). Este bit es válido hasta que se lee el búfer de recepción (UDRn). Siempre establezca este bit a cero al escribir en UCSRnA.

• Bit 1 - U2Xn: duplica la velocidad de transmisión de USART

Este bit solo tiene efecto para la operación asíncrona. Escriba este bit a cero cuando use la operación síncrona.

Escribir este bit en uno reducirá el divisor del divisor de velocidad en baudios de 16 a 8, duplicando efectivamente la velocidad de transferencia para la comunicación asíncrona.

• Bit 0 - MPCMn: Modo de comunicación multiprocesador

Este bit habilita el modo de comunicación multiprocesador. Cuando el bit MPCMn se escribe en uno, se ignorarán todas las tramas entrantes recibidas por el receptor USART que no contienen información de dirección. El transmisor no se ve afectado por la configuración de MPCMn.

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Bit 7 - RXCIEn: RX Complete Interrupt Enable n

Escribir este bit en uno habilita la interrupción en el indicador RXCn. Una interrupción de recepción completa de USART se generará solo si el bit RXCIEn se escribe en uno, el Indicador de interrupción global en SREG se escribe en uno y el bit RXCn en UCSRnA se establece.

• Bit 6 - TXCIEn: habilitación de interrupción completa de TX n

Escribir este bit en uno habilita la interrupción en el indicador TXCn. Se generará una interrupción de transmisión completa de USART solo si el bit TXCIEn se escribe en uno, el Indicador de interrupción global en SREG se escribe en uno y el bit TXCn en UCSRnA se establece.

• Bit 5 - UDRIEn: Registro de datos USART Vacío Interrupción Habilitar n

Escribir este bit en uno habilita la interrupción en el indicador UDREn. Se generará una interrupción de Registro de Datos Vacíos solo si el bit UDRIEn se escribe en uno, el Indicador de Interrupción Global en SREG se escribe en uno y el bit UDREn en UCSRnA se establece.

• Bit 4 - RXENn: receptor habilitado n

Escribir este bit en uno habilita el receptor USART. El receptor anulará la operación normal del puerto para el pin RxDn cuando esté habilitado. Al deshabilitar el receptor, se vaciará el búfer de recepción y se invalidarán los indicadores FEn, DORn y UPEn.

• Bit 3 - TXENn: Transmisor habilitado n

Escribir este bit en uno habilita el transmisor USART. El transmisor anulará la operación normal del puerto para el pin TxDn cuando esté habilitado. La desactivación del transmisor (escribir TXENn en cero) no entrará en vigencia hasta que se completen las transmisiones en curso y pendientes, es decir, cuando el Registro de cambios de transmisión y el Registro de almacenamiento intermedio de transmisión no contengan datos para transmitir. Cuando está deshabilitado, el transmisor ya no anulará el puerto TxDn.

• Bit 2 - UCSZn2: Tamaño de carácter n

Los bits UCSZn2 combinados con el bit UCSZn1: 0 en UCSRnC establecen el número de bits de datos (tamaño de caracteres) en una trama que utilizan el receptor y el transmisor.

• Bit 1 - RXB8n: Bit de recepción de datos 8 n

RXB8n es el noveno bit de datos del carácter recibido cuando se opera con tramas en serie con nueve bits de datos. Debe leerse antes de leer los bits bajos de UDRn.

• Bit 0 - TXB8n: Bit de transmisión de datos 8 n

TXB8n es el noveno bit de datos en el carácter que se transmitirá cuando se opera con tramas en serie con nueve bits de datos. Debe escribirse antes de escribir los bits bajos en UDRn.

Bit	7	6	5	4	3	2	1	0	
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

• Bits 7: 6 - UMSELn1: 0 USART Mode Select

Estos bits seleccionan el modo de operación del USARTn

Table 22-4. UMSELn Bits Settings

UMSELn1	UMSELn0	Mode
0	0	Asynchronous USART
0	1	Synchronous USART
1	0	(Reserved)
1	1	Master SPI (MSPIM)[1]

• Bits 5: 4 - UPMn1: 0: Modo de paridad

Estos bits habilitan y establecen el tipo de generación de paridad y verificación. Si está habilitado, el Transmisor generará y enviará automáticamente la paridad de los bits de datos transmitidos dentro de cada trama. El receptor generará un valor de paridad para los datos entrantes y lo comparará con la configuración de UPMn. Si se detecta una falta de coincidencia, se establecerá el indicador UPEn en UCSRnA.

Table 22-5. UPMn Bits Settings

UPMn1	UPMn0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

• Bit 3 - USBSn: Stop Bit Select

Este bit selecciona el número de bits de parada a ser insertados por el transmisor. El receptor ignora esta configuración.

Table 22-6. USBS Bit Settings

USBSn	Stop Bit(s)
0	1-bit
1	2-bit

• Bit 2: 1 - UCSZn1: 0: Tamaño del carácter

Los bits UCSZn1: 0 combinados con el bit UCSZn2 en UCSRnB establecen el número de bits de datos (Tamaño de Caracteres) en una trama que utilizan el Receptor y el Transmisor.

Table 22-7. UCSZn Bits Settings

UCSZn2	UCSZn1	UCSZn0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

• Bit 0 - UCPOLn: polaridad del reloj

Este bit se usa solo para el modo síncrono. Escriba este bit a cero cuando se utilice el modo asíncrono. El bit UCPOLn establece la relación entre el cambio de salida de datos y la muestra de entrada de datos, y el reloj síncrono (XCKn).

Table 22-8. UCPOLn Bit Settings

UCPOLn	Transmitted Data Changed (Output of TxDn Pin)	Received Data Sampled (Input on RxDn Pin)
0	Rising XCKn Edge	Falling XCKn Edge
1	Falling XCKn Edge	Rising XCKn Edge

Secuencias de escape ANSI.

Las secuencias de escape ANSI permiten enviar información de control a la consola para cambiar los atributos del texto representado.

Así es posible seleccionar:

1. El estilo del texto: normal, claro, subrayado, parpadeante, inverso y oculto
2. El color del fondo
3. El color del texto

El funcionamiento es muy sencillo, entre la información que se envía de salida a la consola, se incluyen las secuencias de escape dando instrucciones de cómo se ha de representar el texto a continuación.

Por ejemplo, si con `printf` se vuelca `"Palabra %sresaltada%s"` insertando en el primer `%s` la cadena de control para hacer que el texto sea verde, el texto que se escribe a continuación ('resaltada') tendrá color verde. En el último `%s` se debería introducir la secuencia de escape para volver a la normalidad. De no ser así, toda salida posterior sería de color verde !!!

Sintáxis:

Las secuencias de escape se pueden utilizar desde cualquier script o programa que envíe información a la consola. Por ejemplo, en C podríamos definir las siguientes secuencias:

1. `const char *const normal = "\033[0m";`
2. `const char *const verde = "\033[0;40;32m";`
3. `const char *const subrayado_fazul_verde = "\033[4;44;32m";`

La primera selecciona el estilo, fondo y color de texto normal. Es la cadena que deberiamos utilizar para terminar los efectos anteriormente aplicados. La segunda cadena determina que el color de texto sea verde. La tercera cadena determina que el texto ha de estar subrayado, ser de color verde y estar sobre fondo azul.

La sintáxis sería: `"\033[x;xx;xxm"` dónde cada 'x' representa un dígito.

El primer dígito especifica el estilo:

- 0 -> Normal
- 1 -> Claro (el color se diluye, permite hacer dos tonos de cada color: azul/azulclaro, rojo/rojoclaro...)
- 4 -> Subrayado
- 5 -> Parpadeante
- 7 -> Inverso
- 8 -> Oculto (Pensado para pedir contraseñas al usuario)

Siendo ESC = \033

Algunos códigos de escape

Secuencia	Acción
ESC [*n* A	Desplaza el cursor hacia arriba *n* filas. Si el cursor se encontraba en la parte superior de la pantalla, no tenía ningún efecto. Si no se especificaba *n* el cursor subía una fila.
ESC [*n* B	Desplaza el cursor *n* filas hacia abajo. Al igual que con el desplazamiento hacia arriba, si el cursor se encontraba en la parte inferior de la pantalla el cursor no se movía, y si no se especificaba *n* bajaba una fila.
ESC [*n* C	Mueve el curosr *n* columnas hacia la derecha. Si el cursor se encuentra en la última columna este comando no tiene efecto. Si no se especifica *n* el desplazamiento es de una columna.
ESC [*n* D	Mueve el cursor *n* columnas a la izquierda, salvo que se encuentre en la primera columna, en cuyo caso no tiene efecto. Si *n* no se especifica toma el valor de 1.
ESC [*n*; *m* f	Mueve el cursor a la fila *n* y columna *m*. Si *n* no se especifica el cursor se mueve a la primera fila.
ESC [*n* J	Borra parte de la pantalla. Si *n* vale 0 se borra desde el cursor hasta el final de la pantalla. En caso de que *n* valga 1 se borra hasta el principio de la pantalla desde la posición del cursor. Si *n* vale 2 se borra toda la pantalla (y utilizando ANSI. SYS de MS-DOS el cursor va al principio de la pantalla)
ESC [*n* K	Borra parte de la línea. Si *n* es cero, desde el cursor al final de la línea, en caso de que valga 1 se borra hasta el principio. Si *n* vale 2 se borra toda la línea.
ESC [*código*; *parámetro* [; *parámetro*] p	Con este comando se redefine el comportamiento de una tecla.
ESC [s	Guarda la posición actual del cursor.
ESC [u	Coloca el cursor en la posición guardada anteriormente.
ESC [*b* ; *fg* ; *bg* m	Establece la intensidad, el color del primer plano y el color de fondo del texto. Más ejemplos de códigos y colores en http://softwarelivre.org/terceiro/blog/a-visual-cheat-sheet-for-ansi-color-codes

Conclusión:

El aprender a usar un microcontrolador con la configuración USART es muy importante ya que a través de este nos podemos comunicar con módulos wifi,bluethooth una computadora entre otros.Me parecio interesante como el micro se comunica por el UART0 así que me puse a investigar y mire que hay otro microcontrolador el Atmega16u2 el que es el que se encarga de hacer la comunicación de la computadora al Atmega2560/1280. Investigando tambien mire que existen los FTDI y con estos nos podríamos comunicar con los demás USART's sin necesidad de usar el 0.