UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

Facultad de Ciencias Químicas e Ingeniería

Materia: Microprocesadores y Microcontroladores

Practica 8 Programación del uC del periférico de comunicación serie utilizando interrupciones

Docente:
Garcia Lopez Jesus Adan

Alumno:

Morales Rosales Iván A. 1231098

Teoría

USART Initialization

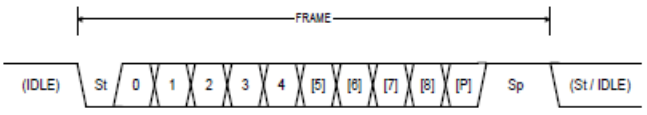
The USART has to be initialized before any communication can take place. The initialization process normally consists of setting the baud rate, setting frame format and enabling the Transmitter or the Receiver depending on the usage. For interrupt driven USART operation, the Global Interrupt Flag should be cleared (and interrupts globally disabled) when doing the initialization. Before doing a re-initialization with changed baud rate or frame format, be sure that there are no ongoing transmissions during the period the registers are changed. The TXC Flag (UCSRnA.TXC) can be used to check that the Transmitter has completed all transfers, and the RXC Flag can be used to check that there are no unread data in the receive buffer. The UCSRnA.TXC must be cleared before each transmission (before UDRn is written) if it is used for this purpose.

```
#define FOSC 1843200 // Clock Speed
#define BAUD 9600
#define MYUBRR FOSC/16/BAUD-1
void main( void )
{
   ...
   USART_Init(MYUBRR)
   ...
}
void USART_Init( unsigned int ubrr)
{
   /*Set baud rate */
   UBRR0H = (unsigned char)(ubrr>>8);
   UBRR0L = (unsigned char)ubrr;
   Enable receiver and transmitter */
   UCSR0B = (1<<RXEN0)|(1<<TXEN0);
   /* Set frame format: 8data, 2stop bit */
   UCSR0C = (1<<USBS0)|(3<<UCSZ00);
}
```

Equations for Baud Rate Register Calculation

Operating Mode	Equation for Calculating Baud Rate(1)	Equation for Calculating UBRRn Value
Asynchronous Normal mode (U2X = 0)	$BAUD = \frac{f_{OSC}}{16(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed mode (U2X = 1)	$BAUD = \frac{f_{OSC}}{8(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous Master mode	$BAUD = \frac{f_{OSC}}{2(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{2BAUD} - 1$

Frame Formats



St	Start bit, always low.
(n)	Data bits (0 to 8).
P	Parity bit. Can be odd or even.
Sp	Stop bit, always high.
IDLE	No transfers on the communication line (RxDn or TxDn). An IDLE line must be high.

Registers

Name: UDRn
Offset: 0xC6 + n*0x08 [n=0..1]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	TXB / RXB[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – TXB / RXB[7:0]: USART Transmit / Receive Data Buffer

Name: UCSR0A, UCSR1A
Offset: 0xC0 + n*0x08 [n=0..1]
Reset: 0x20
Property: -

Bit	7	6	5	4	3	2	1	0
	RXC	TXC	UDRE	FE	DOR	UPE	U2X	MPCM
Access	R	R/W	R	R	R	R	R/W	R/W
Reset	0	0	1	0	0	0	0	0

Name: UCSR0B, UCSR1B
Offset: 0xC1 + n*0x08 [n=0..1]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
Access	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Reset	0	0	0	0	0	0	0	0

Name: UCSR0C, UCSR1C
Offset: 0xC2 + n*0x08 [n=0..1]
Reset: 0x06
Property: -

Bit	7	6	5	4	3	2	1	0
	UMSEL[1:0]		UPM[1:0]		USBS	UCSZ1 / UDORD	UCSZ0 / UCPHA	UCPOL
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	1	1	0

Parity Mode Settings

UPM[1:0]	ParityMode
00	Disabled
01	Reserved
10	Enabled, Even Parity
11	Enabled, Odd Parity

Stop Bit Settings

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

Character Size Settings

UCSZ1[2:0]	Character Size
000	5-bit
001	6-bit
010	7-bit
011	8-bit
100	Reserved
101	Reserved
110	Reserved
111	9-bit

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.

Sintaxis:
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 deberíamos 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..., etc.)
- 4 -> Subrayado
- 5 -> Parpadeante
- 7 -> Inverso
- 8 -> Oculto (Pensado para pedir contraseñas al usuario)

Link de evidencia:
https://drive.google.com/drive/folders/1EizpNeRq-Edc4BGwnW7LnfT49J5QTsYS?usp=sharing

Conclusión
Es interesante saber como funciona el USART, como configurarlo para la recepción y trasmisión de datos.

Un UART es generalmente un circuito integrado (IC) individual (o parte de un) utilizado para comunicaciones en serie a través de una computadora o puerto serie de dispositivo periférico. Los módems que se conectan a la ranura de la placa base incluyen UART.