26 DE ABRIL DE 2017


# PRACTICA 8B

## PROGRAMACIÓN DEL UC DEL PERIFÉRICO DE COMUNICACIÓN SERIE UTILIZANDO INTERRUPCIONES

CARLOS OMAR CALDERON MEZA
MICROPROCESADORES Y MICROCONTROLDORES
Ingeniero en computación

**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:**

**-** Programación en lenguaje C en microcontroladores
- Manejo del Periférico de Comunicación Serie 0 (UART0) del microcontrolador ATmega1280/2560

El UART tiene que ser inicializado antes que cualquier comunicación tome lugar. El proceso de inicialización consiste normalmente de ajustar el baud rate, el formato de frame y habilitar el transmisor o el receptor dependiendo del uso. Para una operación de interrupción que conduzca el USART, las banderas de interrupción globales deberán ser limpiadas (y las interrupciones globales deshabilitadas) cuando se realiza la inicialización. Antes de la re inicialización cuando se cambie el baud rate u el formato del frame, tenga cuidado de que no se estén usando transmisiones de salida durante el periodo que los registros se cambien. La bandera TXC puede usarse para checar si el transmisor ha completado su transferencia, y la bandera de RCX puede usarse para checar que no exista un dato sin leerse en el buffer receptor. Note que la bandera TXC debe de limpiarse antes de cada transmisión (antes que el UDR se escriba) si se usa para este propósito.

El siguiente código simple de ejemplo de inicialización del USART muestra una función en ensamblador y una en C y son iguales en cuestión de funcionalidad. Los ejemplos asumen una operación asíncrona usando el "**polling**" (ninguna interrupción está habilitada) y un formato de frame fijo. El baud rate está dado como un parámetro en la función. Para el código ensamblador, el parámetro de baud rate se asume que se almacena en los registros r17:r16. Cuando la función escribe al registro UCSRC, el bit de URSEL (MSB) debe ser ajustado debido a que comparte la localidad de I/O de UBRRH y UCSRC.
Una rutina más avanzada de inicialización puede ser que incluya el formato del frame como parámetros, la deshabilitación de las interrupciones y así sucesivamente. Sin embargo, muchas aplicaciones usan un ajuste fijo de Baud rate y los registros de control, y para este tipo de aplicaciones el código de inicialización puede ser colocado directamente en la rutina main, o estar combinada con el código de inicialización para otros módulos de I/O.

Assembly Code Example[1]

```
USART_Init:
  ; Set baud rate
  out   UBRRH, r17
  out   UBRRL, r16
  ; Enable receiver and transmitter
  ldi   r16, (1<<RXEN)|(1<<TXEN)
  out   UCSRB,r16
  ; Set frame format: 8data, 2stop bit
  ldi   r16, (1<<URSEL)|(1<<USBS)|(3<<UCSZ0)
  out   UCSRC,r16
  ret
```

C Code Example[1]

```
void USART_Init( unsigned int baud )
{
  /* Set baud rate */
  UBRRH = (unsigned char)(baud>>8);
  UBRRL = (unsigned char)baud;
  /* Enable receiver and transmitter */
  UCSRB = (1<<RXEN)|(1<<TXEN);
  /* Set frame format: 8data, 2stop bit */
  UCSRC = (1<<URSEL)|(1<<USBS)|(3<<UCSZ0);
}
```

**Transmisión de Datos – El Transmisor del USART**

El transmisor del USART es habilitado ajustando el bit Habilitación de Transmisión (TXEN) en el registro UCSRB. Cuando el transmisor es habilitado, la operación normal del puerto del pin TxD es “overridden” por el USART dando la función de transmisor de salida serial. El baud rate, modo de operación y el formato del frame deben establecerse antes que inicie cualquier transmisión. Si la operación síncrona es utilizada, el reloj en el pin XCK será “overridden” y usada como transmisión del reloj.

**Enviando Frames de 5 a 8 bits de datos**

Una transmisión de datos es iniciada cargando al buffer del trasmisor con los datos a ser transmitidos. El CPU puede cargar al buffer de trasmisión con solo escribir en la localidad de I/O de UDR. Los datos almacenados en el buffer de transmisión serán movidos al registro de corrimiento cuando este registro este listo para enviar un nuevo frame. El registro de corrimiento es cargado con un nuevo dato si se encuentra en el estado libre (ninguna transmisión de salida) o inmediatamente después del último bit de stop del frame previo transmitido. Cuando el registro de corrimiento es cargado con un nuevo dato,

transferirá un frame completo a una razón dada por el registro del Baud, el bit U2X o por XCK dependiendo del modo de operación.

El siguiente ejemplo del código muestra una función simple de transmisión del USART en "**polling**" la bandera del registro de datos vacío (UDRE). Cuando se usan frames con menos de ocho bits, los bits más significantes escritos en el UDR son ignorados. El USART tiene que estar inicializado antes que la función se utilice. Para el código ensamblador, el dato a ser enviado se asume que está almacenado en el registro R16. La función simple espera que el buffer trasmita para que quede vacío checando la bandera UDRE, antes que se cargue un nuevo dato para ser transmitido. Si la interrupción de registro de datos vacío se utiliza, la rutina de interrupción escribe el dato en el buffer transmisor.

Assembly Code Example[1]

```
USART_Transmit:
  ; Wait for empty transmit buffer
  sbis  UCSRA,UDRE
  rjmp  USART_Transmit
  ; Put data (r16) into buffer, sends the data
  out   UDR,r16
  ret
```

C Code Example[1]

```
void USART_Transmit( unsigned char data )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSRA & (1<<UDRE)) )
        ;
  /* Put data into buffer, sends the data */
  UDR = data;
}
```

**El Receptor del USART – Recepción de Datos**

El receptor del USAR se habilita escribiendo en el bit de habilitación de recepción (RXEN) en el registro UCSRB a uno. Cuando el receptor se habilita, la operación normal del pin de RxD es "overridden" por el USART y se comporta como receptor de serial de entrada. El Baud Rate, el modo de operación y el formato del frame deben de ajustarse antes que cualquier recepción serial se lleve a cabo. Si se usa la operación síncrona, el reloj en el pin XCK se usara como transferencia de reloj.

**Recibiendo frames de 5 a 8 bits de datos**

El receptor inicia la recepción de datos cuando detecta un bit de inicio valido. Cada bit que siga al bit de inicio será muestreado a un baud rate o al reloj XCK, y se correrá dentro del

registro de corrimiento del receptor hasta que el primer bit de stop de un frame se reciba. Un segundo bit de stop será ignorado por el receptor. Cuando el primer bit de stop sea recibido, por ejemplo, un frame serial completo esté presente en el registro de corrimiento del receptor, el contenido de registro de corrimiento será movido dentro del buffer del receptor. El buffer del receptor puede entonces leerse por lectura de la localidad del registro de I/O UDR.
El siguiente código ejemplo muestra una función de recepción simple del USART basada en la técnica de "**polling"** la bandera de recepción completa (RXC). Cuando se usan frames con menos de ocho bits, los bits más significativos de datos leídos en el UDR serán enmascarados a cero. El USART tiene que ser inicializado antes que la función pueda usarse.

Assembly Code Example[1]

```
USART_Receive:
  ; Wait for data to be received
  sbis  UCSRA, RXC
  rjmp  USART_Receive
  ; Get and return received data from buffer
  in    r16, UDR
  ret
```

C Code Example[1]

```
unsigned char USART_Receive( void )
{
  /* Wait for data to be received */
  while ( !(UCSRA & (1<<RXC)) )
        ;
  /* Get and return received data from buffer */
  return UDR;
}
```

La función simplemente espera que el dato este presente en el buffer de recepción checando la bandera RXC antes que se de lectura al buffer, retornando el valor.

Anteriormente se mostró como se realiza la recepción y transmisión por "**polling**", a continuación se muestra como trabajar con **interrupciones**:

Para habilitar la interrupción USART AVR se utilizará el registro **UCSR0B**, además deben estar habilitadas las interrupciones globales, lo cual en el ATMEL STUDIO se hará con la instrucción **sei()**.

| 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 | |

Para habilitar la interrupción USART AVR al completar la recepción de un carácter**,** del registro **UCSR0B** se pone a 1 su bit7, lo que en el ATMEL STUDIO se hace así:

**UCSR0B |= (1<<7);** //habilita interrupción por recepción USART AVR.

El bit7 del registro **UCSR0A**, se pondrá a 1 automáticamente para indicar que se ha producido una interrupción USART AVR al completar la recepción de un caracter en el registro **UDR0**, cuando se lea el caracter recibido este bit se pondrá automáticamente a 0 para seguir detectando la interrupción USART AVR por la recepción de datos.

Para habilitar la interrupción USART AVR al completar la transmisión de un caracter**,** del registro **UCSR0B** se pone a 1 su bit6, lo que en el ATMEL STUDIO se hace así:

**UCSR0B |= (1<<6);** //habilita interrupción por transmisión USART AVR.

El bit6 del registro **UCSR0A**, se pondrá a 1 automáticamente para indicar que se ha producido una interrupción USART AVR por la transmisión del caracter que se encontraba en el registro **UDR0**, cuando se cargue un nuevo caracter en el registro UDR0 este bit se pondrá automáticamente a 0 para seguir detectando la interrupción USART AVR por la transmisión de datos.

Para habilitar la interrupción USART AVR al quedarse vacío el registro UDR0, del registro **UCSR0B** se pone a 1 su bit5 ,lo que en el ATMEL STUDIO se hace así:

**UCSR0B |= (1<<5);** //habilita interrupción por vaciado del registro UDR0 USART AVR.

El bit5 del registro **UCSR0A**, se pondrá a 1 automáticamente para indicar que se ha producido una interrupción USART AVR por vaciado del registro **UDR0**, cuando se cargue un nuevo caracter en el registro UDR0 este bit se pondrá automáticamente a 0 para seguir detectando la interrupción USART AVR por vaciado del registro UDR0.

**Rutinas de interrupción USART AVR en C/C++ en el ATMEL STUDIO**

Si se utiliza la interrupción por recepción USART AVR, la rutina de interrupción se realizará dentro de la siguiente función:

```
ISR(USART_RX_vect){

        dato = UDR0;     //al dato de tipo char se le asigna el valor del registro UDR0

                        //otras tareas a realizar

}
```

Si se utiliza la interrupción por transmisión USART AVR, la rutina de interrupción se realizará dentro de la siguiente función:

```
ISR(USART_TX_vect){
```

```
UDR0 = caracter;   //en el registro UDR0 se carga el caracter a transmitir

//otras tareas a realizar
```

**}**

Si se utiliza la interrupción por vaciado del registro UDR0 USART AVR, la rutina de interrupción se realizará dentro de la siguiente función:

```
ISR(USART_UDRE_vect){

    //tareas a realizar dentro de la rutina de interrupción por vaciado del registro UDR0
```

**}**

## Conclusiones y comentarios

El uso de interrupciones para los microcontroladores de Atmega es muy diferente a lo que conocía en x86, que solo invocaba al verter de interrupción y colocaba un servicio en un registro, pero no es muy complicado utilizar tampoco aquí en esta arquitectura las interrupciones para el caso de puesto UART veo ventajas trabajar con interrupciones que por "**polling**", ya que se gasta tiempo de ejecución esperando hasta que termine de transmitir para mandar otro byte por ejemplo, y con interrupciones se puede trabajar en otras cosas mientras el puerto saca los bytes de la cola como se hizo en esta práctica.

A simple vista al correr el programa no note ninguna diferencia en consideración con la practica 8a pero esta manera de hacerlo si es mejor dependiendo de la aplicación o sistema en que este embebido el microcontrolador.

## Bibliografía


[1] Microcontroladores. (2010). Interrupcion USART AVR. Abril 27, 2017, de Programación de Microcontroladores PIC, AVR, ARDUIN Sitio web: http://microcontroladores-mrelberni.com/interrupcion-usart-avr/

[2] ITPS. (2010). capitulo 8 USART del ATmega . Abril 27, 2017, de ITPS Sitio web: https://www.google.com.mx/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0ahUKEwikiszNncXTAhWEiFQKHZKUBW0QFggIMAA&url=https%3A%2F%2Fuvirtual.unet.edu.ve%2Fpluginfile.php%2F199032%2Fmod_folder%2Fcontent%2F0%2FTutorial%2520ATMEGA%2F7843052-Capitulo8-USART-del-ATmega32-espanol.pdf%3Fforcedownload%3D1&usg=AFQjCNFeMQpfCzJQwy-Rb7ITxznDvNcfyQ&sig2=2M7PHtxikt7d36-OfeGkhQ&cad=rja