TEMA 9: COMUNICACIÓN SERIE ASÍNCRONA (USART)

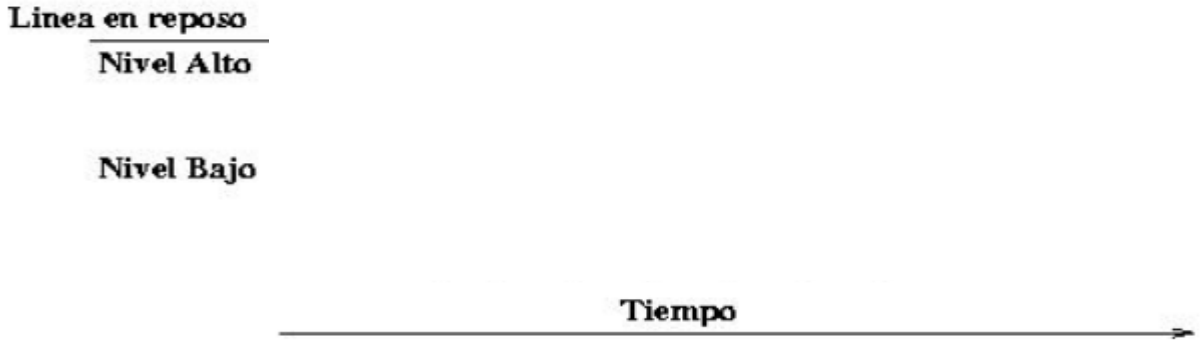
© Raúl Sánchez Reíllo

Comunicaciones Serie

- En este tipo de comunicación se trata de transmitir la información bit a bit:
  - Se divide la información en palabras
  - Cada palabra se transmite bit a bit
- Hay que buscar métodos y puntos de sincronismo
  - Para indicar la existencia de cada bit
  - Para indicar el inicio de una palabra
- Atendiendo al sincronismo de bit, se tendrá:
  - **Comunicación Síncrona:** cuando una señal de reloj indique la validez del bit transmitido
  - **Comunicación Asíncrona:** cuando no existe dicha señal de reloj, por lo que la temporización de emisor y receptor se realiza por medios independientes

Comunicación Serie Asíncrona

- La comunicación serie asíncrona se realiza de la siguiente manera:
  - En estado de reposo, la línea de transferencia de datos tiene que estar a nivel alto
  - El inicio de la transmisión de un carácter se indica poniendo, ***durante el tiempo de 1 bit***, la línea a 0V (**bit de arranque**)
  - Se transmiten uno por uno cada uno de los bits de la palabra. Se mantiene el valor de cada bit durante el ***tiempo de 1 bit***.
    - Se transmite primero el bit menos significativo del carácter
    - Si el bit es un '1' se colocan 5V en la línea, si es un '0' se pone la línea a 0V
  - Después del último bit del carácter, se pone la línea a 5V, para indicar el final del carácter (**bit de parada**) durante el tiempo de 1 bit

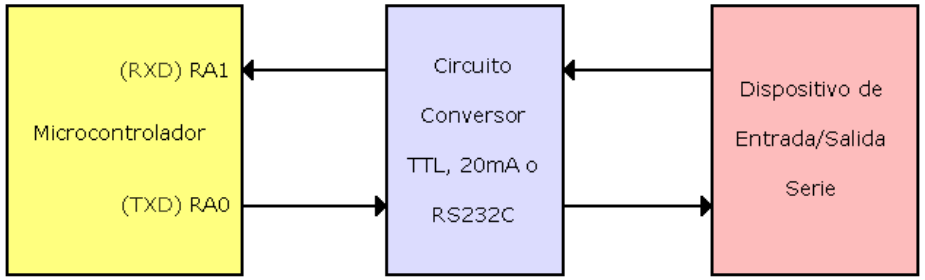


Comunicación Serie Asíncrona

- Esta forma de comunicación está regida por la selección previa y común (entre receptor y transmisor) de una serie de parámetros:
  - Velocidad de transmisión
    - Los baudios a los que se transmite (9600, 19200, 33600, etc.)
    - Esto determina el tiempo de bit: 9600bps ➔ t_{bit} = 0.1ms
  - Longitud del carácter
    - Número de bits por carácter
    - Típicamente suelen ser 7 u 8 (últimamente siempre 8)
  - Tipo de Paridad (par / impar / ninguna)
    - Se trata de un bit que se añade al final del carácter para que se cumpla que en todo carácter transmitido, el número de 1s es un número par o impar (dependiendo de la paridad escogida)
    - Si la paridad es “ninguna” entonces no se transmite ese bit
  - Número de bits de parada (típicamente 1)

Comunicación Serie Asíncrona

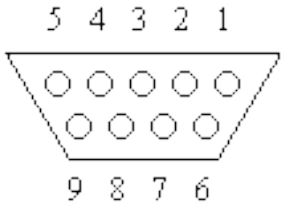
- La comunicación serie asíncrona se suele especificar con expresiones del tipo: 9600,8,N,1
  - 9600bps
  - 8 bits de datos
  - N = sin paridad (E – paridad par / O – paridad impar)
  - 1 bit de parada
- Además hay que tener en cuenta que la comunicación serie está pensada para transmitir información a distancia
  - Si los bits se envían en TTL (0V – 3V) se atenuarán en la línea y no llegarán al destino si el cable es largo
  - Por tanto se utilizan transductores que convierten a otra señal eléctrica
    - RS-232: los 1s se convierten a una tensión negativa entre -5 y -15V, mientras que los 0s se convierten a una tensión positiva entre +5 y +15V. La conversión la hace un chip (p. ej. MAX232)
    - RS-485: se transmite mediante tensiones diferenciales, sobrepasando una distancia máxima de 1km
    - Etc.



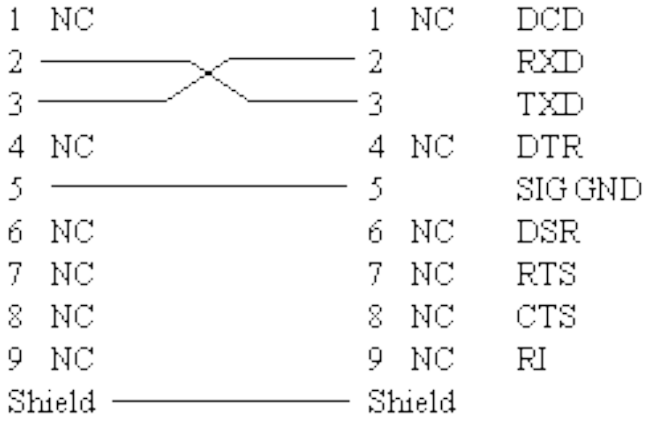
Circuito de interfaz para el metodo de conversion serie/paralelo por software.

Comunicación Serie Asíncrona

- La comunicación serie asíncrona suele ser ***full-duplex***, ya que se contempla una línea de Tx y otra de Rx
  - En concreto, se utiliza frecuentemente la conexión **null-modem** (o módem nulo):
    - Tx del dispositivo 1 conectado al Rx del dispositivo 2
    - Rx del dispositivo 1 conectado al Tx del dispositivo 2
    - Una señal de GND común a los dos dispositivos
  - Aunque se pueden hacer conexiones más complejas, con control de flujo hardware
    - Hay un conjunto de señales (DCE, RTS, DTE, etc.) que se utilizaban antiguamente para poner en contacto los dos dispositivos, al estilo de una comunicación telefónica
    - A día de hoy casi nunca se utiliza. Se plantean o controles de flujo software (Xon – Xoff), o directamente (lo más habitual) sin **ningún control de flujo** (salvo en la capa de aplicación)



Conector DB-9
Hembra
Vista Frontal



TEMA 9: COMUNICACIÓN SERIE ASÍNCRONA (USART)

ARQUITECTURA

ARQUITECTURA

- La USART del STM32L152RB tiene las siguientes características principales:
  - Comunicación síncrona (que no se va a utilizar)
  - Comunicación asíncrona full duplex
  - Formato de codificación estándar NRZ
  - Generador programable de tasas de comunicación (baudrate)
    - Común a recepción y transmisión
    - Puede llegar a 4Mbps
  - Tamaño de datos programable (8 o 9 bits)
  - Número de bits de parada configurables (1 o 2)
  - Codificador IrDA
  - Capacidad de emulación de comunicación con tarjetas inteligentes (ISO/IEC 7816-3)
  - Habilitación independiente para recepción y transmisión
  - Control de estado de buffers de Rx y de Tx
  - Control de paridad

ARQUITECTURA: TRANSMISIÓN

- Para poder transmitir, el bit TE debe estar a 1.
  - En ese caso, cada vez que se envíe una palabra, se irán poniendo los valores correspondientes de cada bit en el pin Tx, atendiendo a la configuración de reloj realizada.
- Procedimiento:
  1. Escribe el valor del bit M y del bit PCE para definir el tamaño de palabra y el uso de paridad, en USARTx→CR1
  2. Decide el número de bits de parada en USARTx→CR2
  3. Selecciona el baudrate en USARTx→BRR
  4. Pon a 1 el bit TE en USARTx→CR1
  5. Habilita la USART poniendo UE=1 en USARTx→CR1
  6. Espera a que el buffer no esté lleno (TXE=1)
  7. Escribe el dato a enviar en USARTx→DR
  8. Repite los pasos 6 y 7 hasta haber terminado de enviar todos los caracteres.
  9. Espera hasta que TC=1, para asegurarse que toda la transmisión ha finalizado.
- Si se habilitan interrupciones por Tx, la RAI saltará siempre que TXE=1.

Arquitectura: Recepción

- La USART, una vez habilitada la recepción, detecta la llegada de palabras de 8 o 9 bits (dependiendo del valor de M), con o sin paridad (bit PCE) de forma totalmente automática.
- Procedimiento:
  1. Programa el valor deseado de M y PCE en USARTx→CR1
  2. Programa el número de bits de parada deseado en USARTx→CR2
  3. Selecciona el baudrate deseado en USARTx→BRR
  4. Habilita la recepción poniendo RE=1 en USARTx→CR1
  5. Habilita la USART poniendo UE=1 en USARTx→CR1
  6. Cuando se recibe un carácter:
     1. La USART pone RXNE = 1 indicando que ha llegado algo
        - Si las IRQs están habilitadas, saltará la RAI
        - Si se ha detectado algún tipo de error, se activarán los flags correspondientes
     2. Lee el dato recibido en USARTx→DR
        - Esto pone RXNE=0 de forma automática
        - Si no se lee el dato antes de que llegue el siguiente carácter, se detectará un error de overrun

Arquitectura: Generador de Baudrates

- El baudrate es común para la recepción que para la transmisión
- La ecuación que relaciona el baudrate es:

$$\text{Tx/Rx baud} = \frac{f_{CK}}{8 \times (2 - \text{OVER8}) \times \text{USARTDIV}}$$

  - Si OVER8=0, la parte fraccional se codifica con 4 bits y programado por DIV_fraction[3:0] en el USARTx→BRR
  - Si OVER8=1, entonces se hace con 3 bits: DIV_fraction[2:0] en USARTx→BRR
- Ejemplo:
  - Con OVER8=0 y USARTDIV = 25.62
    - Parte fraccionaria:
      - 0.62 * 16 = 9.92
      - El entero más cercano = 10
      - DIV_Fraction = 0x0A
    - Parte entera (mantisa):
      - DIV_Mantissa = 25 = 0x19
    - USARTx→BRR = 0x19A
  - Con OVER8=1 es igual pero multiplicando por 8

ARQUITECTURA: GENERADOR DE BAUDRATES

Table 90. Error calculation for programmed baud rates at f_{PCLK} = 16 MHz or f_{PCLK} = 32 MHz), oversampling by 16(1)

Oversampling by 16 (OVER8 = 0)							
Baud rate		f_{PCLK} = 16 MHz			f_{PCLK} = 32 MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired)B.Rate / Desired B.Rate	Actual	Value programmed in the baud rate register	% Error
1	1.2 KBps	1.2 KBps	833.3125	0	1.2 KBps	1666.6875	0
2	2.4 KBps	2.4 KBps	416.6875	0	2.4 KBps	833.3125	0
3	9.6 KBps	9.598 KBps	104.1875	0.02	9.601 KBps	208.3125	0.01
4	19.2 KBps	19.208 KBps	52.0625	0.04	19.196 KBps	104.1875	0.02
5	38.4 KBps	38.369 KBps	26.0625	0.08	38.415 KBps	52.0625	0.04
6	57.6 KBps	57.554 KBps	17.375	0.08	57.554 KBps	34.75	0.08
7	115.2 KBps	115.108 KBps	8.6875	0.08	115.108 KBps	17.375	0.08
8	230.4 KBps	231.884 KBps	4.3125	0.64	230.216 KBps	8.6875	0.08
9	460.8 KBps	457.143 KBps	2.1875	0.79	463.768 KBps	4.3125	0.64
10	921.6 KBps	941.176 KBps	1.0625	2.12	914.286 KBps	2.1875	0.79
11	2 MBps	NA	NA	NA	2000 KBps	1	0
12	4 MBps	NA	NA	NA	NA	NA	NA

1. The lower the CPU clock the lower the accuracy for a particular baud rate. The upper limit of the achievable baud rate can be fixed with these data.

ARQUITECTURA: GENERADOR DE BAUDRATES

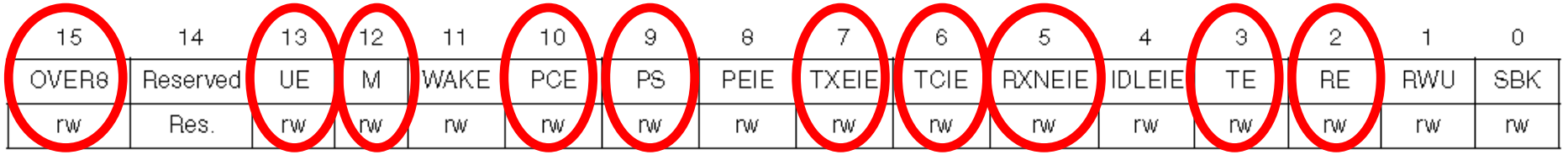
Table 91. Error calculation for programmed baud rates at f_{PCLK} = 16 MHz or f_{PCLK} = 32 MHz), oversampling by 8(1)

Oversampling by 8 (OVER8 = 1)							
Baud rate		f_{PCLK} = 16 MHz			f_{PCLK} = 32 MHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated - Desired)B.Rate / Desired B.Rate	Actual	Value programmed in the baud rate register	% Error
1	1.2 KBps	1.2 KBps	1666.625	0	1.2 KBps	3333.375	0
2	2.4 KBps	2.4 KBps	833.375	0	2.4 KBps	1666.625	0
3	9.6 KBps	9.598 KBps	208.375	0.02	9.601 KBps	416.625	0.01
4	19.2 KBps	19.208 KBps	104.125	0.04	19.196 KBps	208.375	0.02
5	38.4 KBps	38.369 KBps	52.125	0.08	38.415 KBps	104.125	0.04
6	57.6 KBps	57.554 KBps	34.75	0.08	57.554 KBps	69.5	0.08
7	115.2 KBps	115.108 KBps	17.375	0.08	115.108 KBps	34.75	0.08
8	230.4 KBps	231.884 KBps	8.625	0.64	230.216 KBps	17.375	0.08
9	460.8 KBps	457.143 KBps	4.375	0.79	463.768 KBps	8.625	0.64
10	921.6 KBps	941.176 KBps	2.125	2.12	914.286 KBps	4.375	0.79
11	2 MBps	2000 KBps	1	0	2000 KBps	2	0
12	4 MBps	NA	NA	NA	4000 KBps	1	0

1. The lower the CPU clock the lower the accuracy for a particular baud rate. The upper limit of the achievable baud rate can be fixed with these data.

USART: Registros de Control

- USARTx→CR1 – Control Register 1:
  - OVER8 – Modo de sobremuestreo:
    - 0 – Oversampling por 16; 1 – Oversampling por 8
  - UE – Habilitación de la USART
  - M – Tamaño de Palabra:
    - 0 – 8 bits; 1 – 9 bits
  - WAKE: no se utiliza
  - PCE: Control de paridad
    - 0 – Deshabilitado; 1 – Habilitado
  - PS: Selección de paridad
    - 0 – Paridad par; 1 – Paridad impar
  - PEIE: no se utiliza
  - TXEIE: Habilitación de IRQ por buffer de transmisión vacío
  - TCIE: Habilitación de IRQ por transmisión completada
  - RXNEIE: Habilitación de IRQ por carácter recibido
  - IDLIE : no se utiliza
  - TE: Habilitación de transmisión
  - RE: Habilitación de recepción
  - RWU, SBK: no se utiliza



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	Reserved	UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
rw	Res.	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

USART: Registros de Control

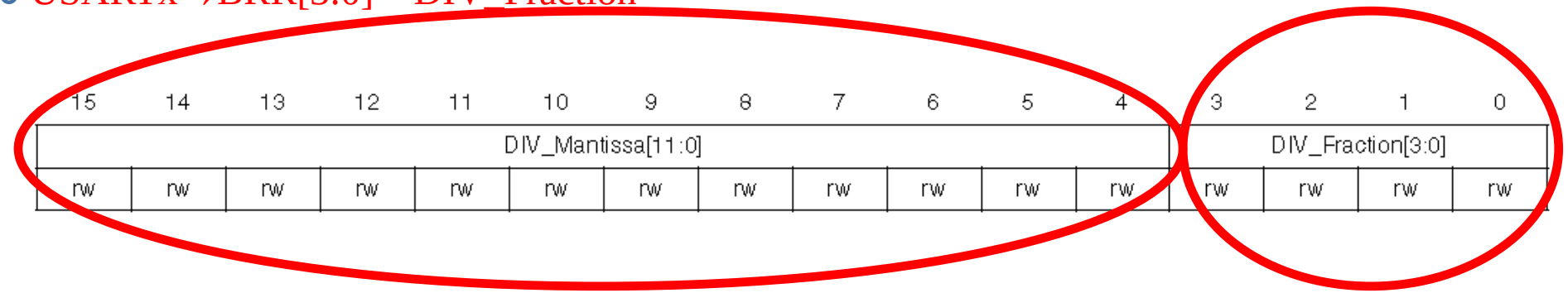
- USARTx→CR2 – Control Register 2:
  - LINEN: No se usa en este curso
  - STOP – Números de bit de parada:
    - 00 – 1 bit; 01 – 0.5 bits; 10 – 2 bits; 11 – 1,5 bits
  - CLKEN, CPOL, CPHA, LBCL, LBDIE, LBDL, ADD: no se usan

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	Res.	ADD[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

- USARTx→CR3 – Control Register 3:
  - Pensado para controlar las funciones adicionales de la USART (DMA, tarjetas inteligentes, IrDA, etc.)
  - No se usa en este curso
- USARTx→GTPR – Guard Time and Prescaler Register:
  - No se usa en este curso

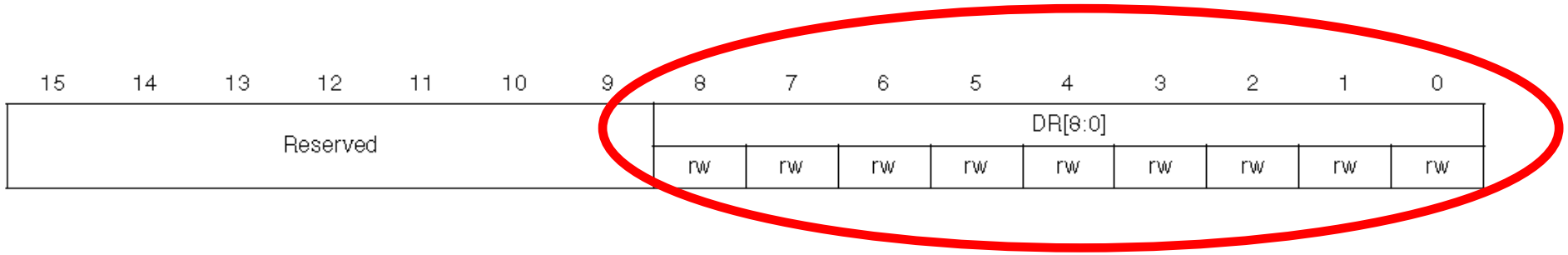
USART: Registros de Control

- USARTx→BRR – Baud Rate Register:
  - Registro de 32 bits, donde sólo se usan los 16 menos significativos:
    - USARTx→BRR[15:4] – DIV_Mantissa
    - USARTx→BRR[3:0] – DIV_Fraction



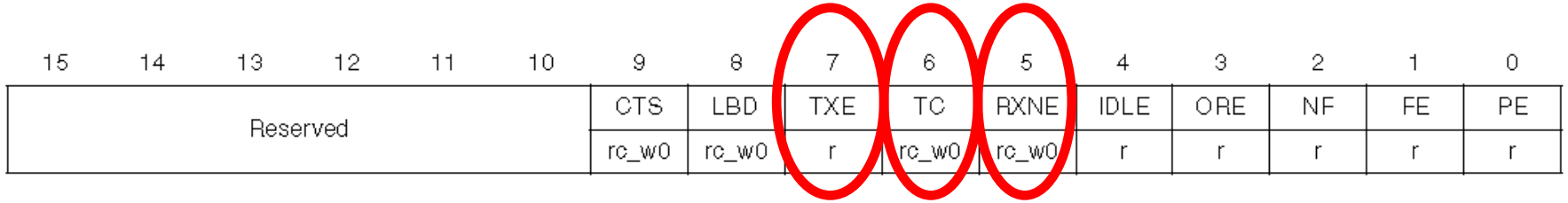
USART: Registros de Datos

- USARTx→DR – Data Register
  - Es un registro de 32 bits, pero del cual sólo se usan los 8 bits menos significativos
  - Internamente son dos registros, uno de recepción y otro de transmisión, pero a nivel de software es uno sólo, en el que si se escribe, se transmite, y si se lee, se lee el último byte recibido.



USART: Registros de Estado

- USARTx→SR – Status Register:
  - Registro de 32 bits, que contiene los 10 flags de eventos:
    - CTS y LBD: no se van a utilizar
    - TXE: Transmit Data Register Empty.
      - Se limpia al escribir en USARTx→ DR
    - TC: Transmission Complete
      - Se limpia con la secuencia: 1.- Lee el USARTx→ SR, 2.- Escribe en USARTx→ DR
    - RXNE: Read data register Not Empty
      - Se limpia leyendo el USARTx→ DR
    - IDLE: no se va a utilizar
    - ORE: Overrun Error.
      - Se limpia por secuencia.
    - NF, FE, PE: no se van a utilizar.

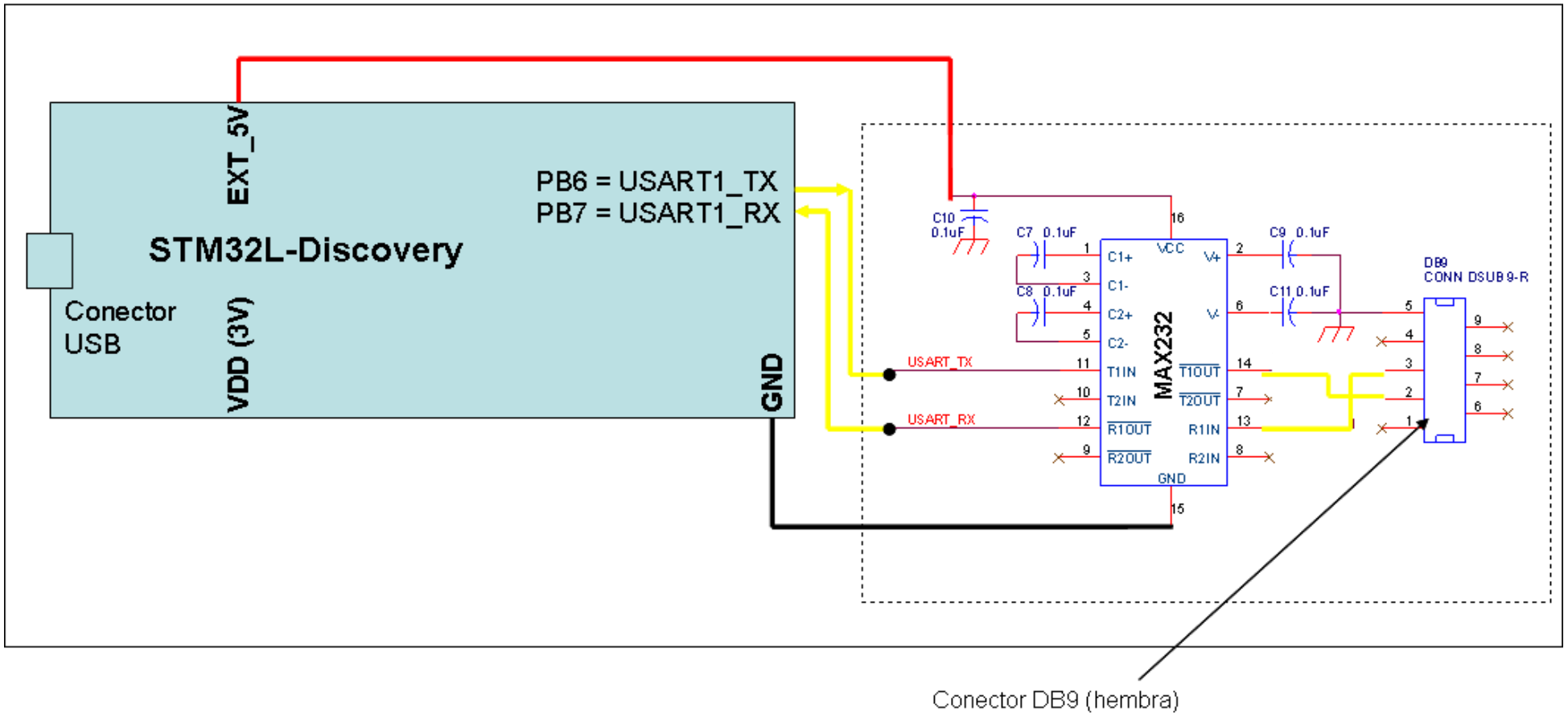


TEMA 9: COMUNICACIÓN SERIE ASÍNCRONA (USART)

¿CÓMO UTILIZAR Y PROBAR LA USART?

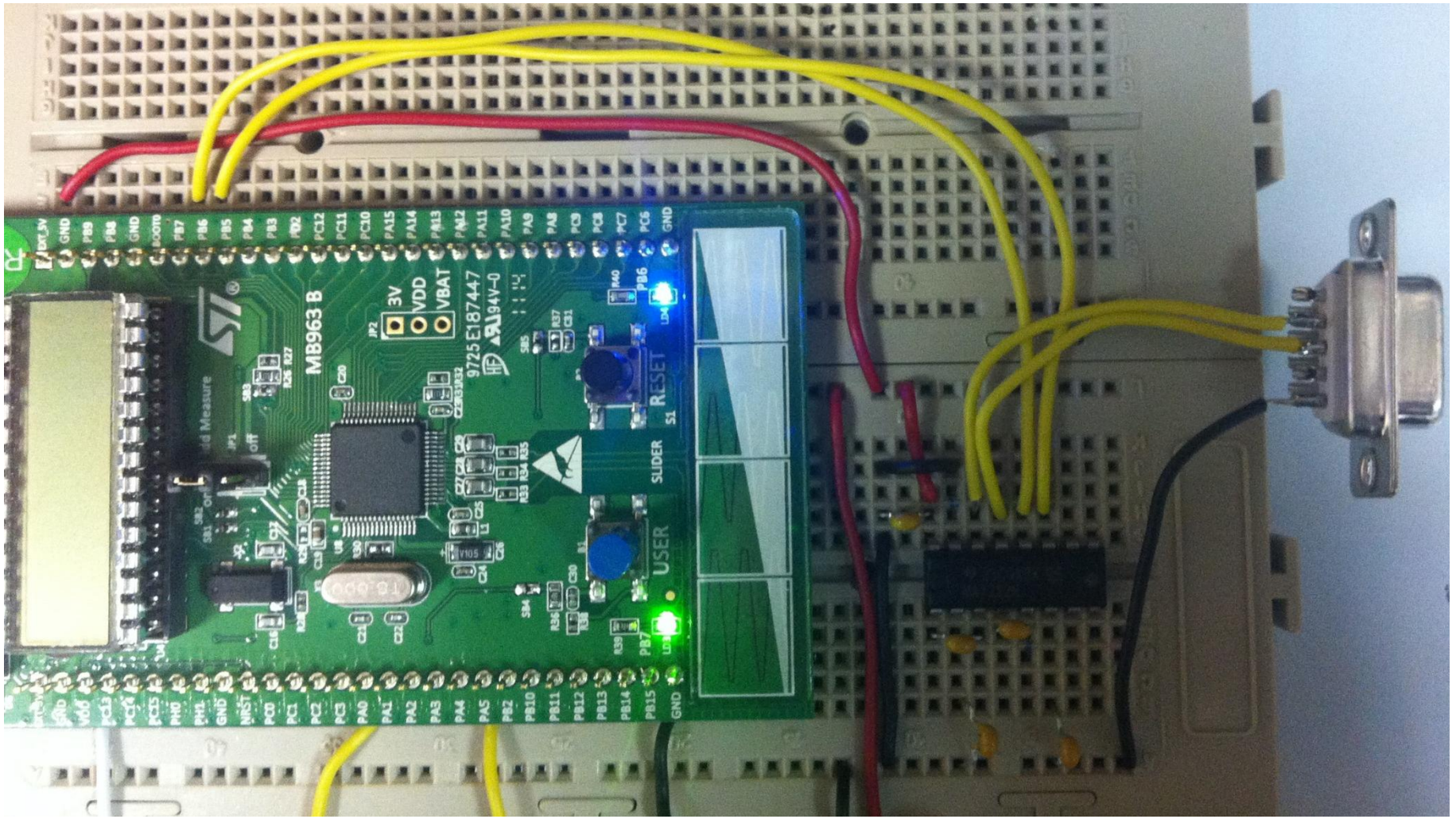
EJEMPLOS: ¿CÓMO UTILIZAR Y PROBAR LA USART? (1)

- Una primera solución para poder probar la comunicación serie es conectar los pines de la USART a un transductor RS-232 (por ejemplo, el MAX232).
  - Como la placa STM32L-Discovery tiene disponibles los pines PB6 y PB7, se hace la conexión como se indica en la figura



EJEMPLOS: ¿CÓMO UTILIZAR Y PROBAR LA USART? (2)

- Es importante tener en cuenta que habrá que utilizar un conector DB9 hembra para dicha conexión. La siguiente fotografía muestra el trabajo de montaje necesario:



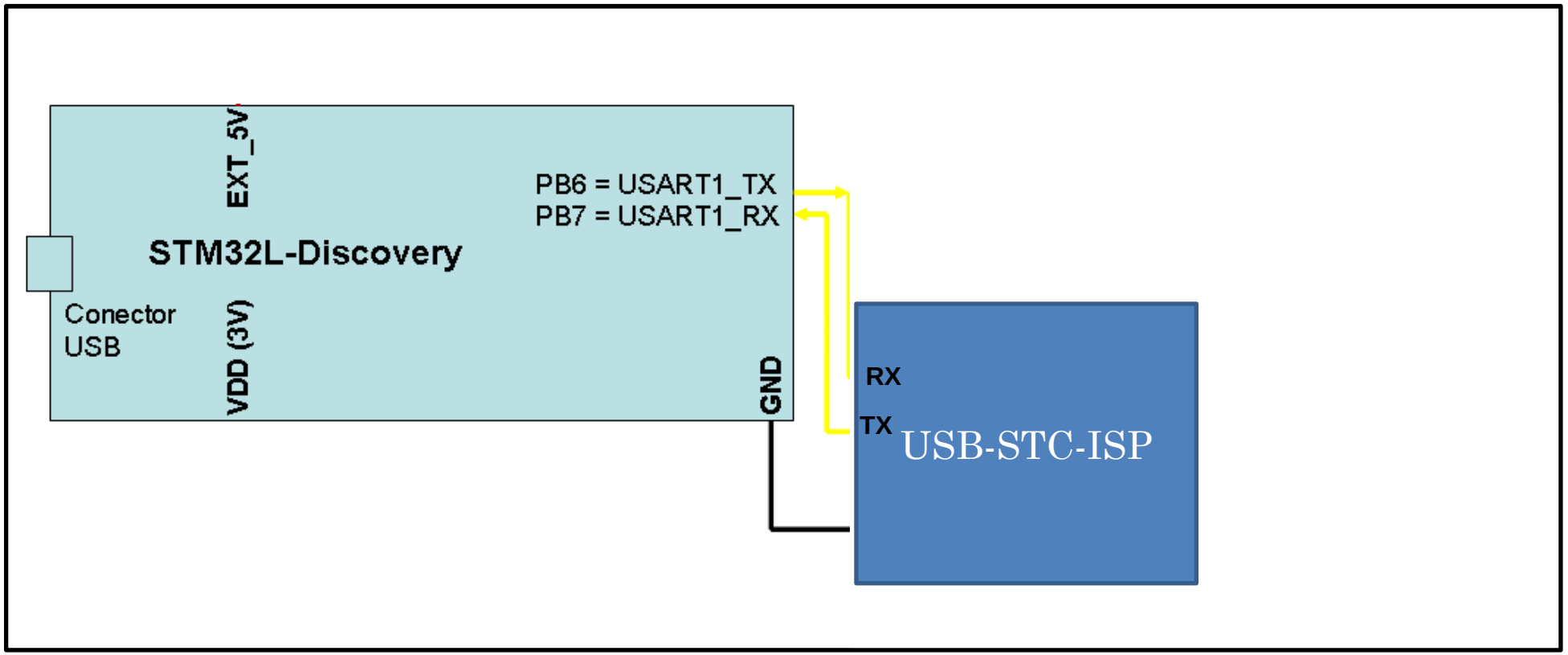
EJEMPLOS: ¿CÓMO UTILIZAR Y PROBAR LA USART? (3)

- Como la mayoría de los ordenadores actuales no poseen puertos serie (COM), sino solamente puertos USB, hay que añadir a este montaje un adaptador USB-RS232, que instalan un driver que crea un puerto COM virtual

- Finalmente, para probar la comunicación hay que utilizar un programa de comunicaciones, como el Hyperterminal de Windows
  - En Windows 7 no se incluye el Hyperterminal, por lo que hay que copiarlo de una versión de XP, o utilizar otro tipo de terminal como:
    - http://en.sourceforge.jp/projects/ttssh2/downloads/54081/teraterm-4.72.exe/
  - Hay que asegurarse que se selecciona correctamente el puerto, así como los parámetros de comunicación
  - Hay que quitar todo tipo de control de flujo (dejarlo en Control de Flujo Nulo).

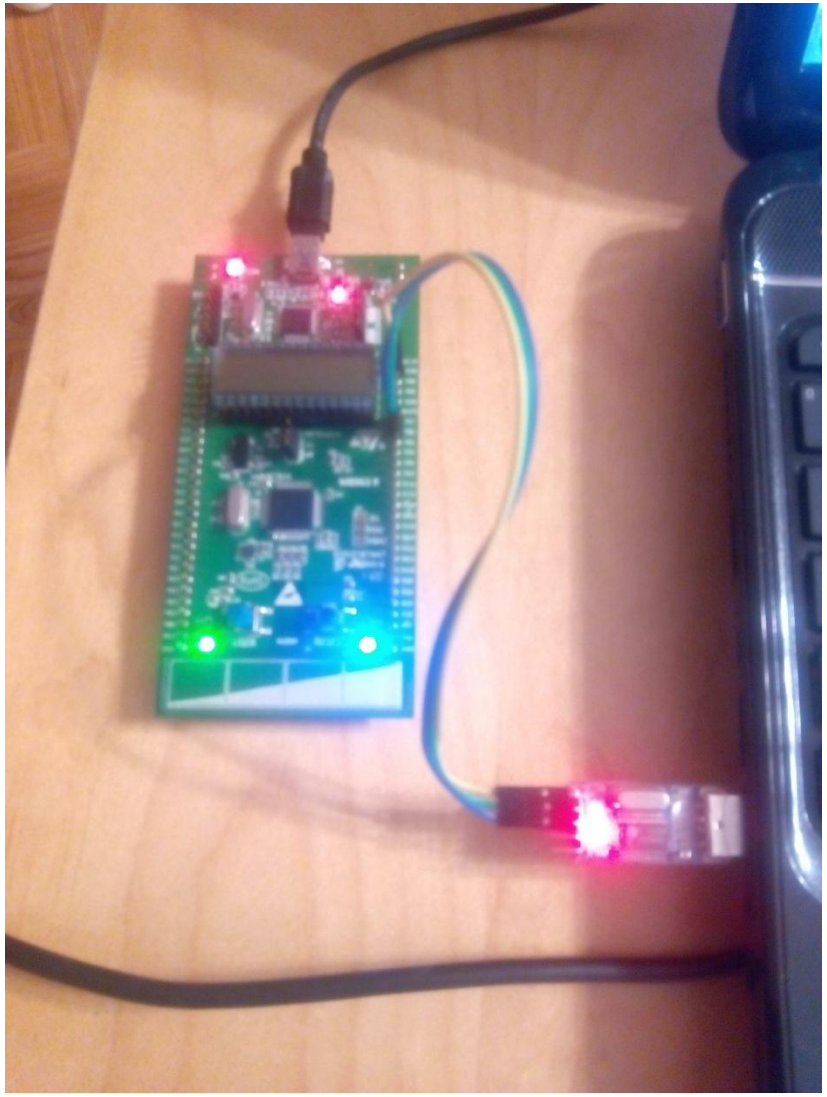
EJEMPLOS: ¿CÓMO UTILIZAR Y PROBAR LA USART? (4)

- Otra solución es conectar los pines de la USART a un transductor TTL-USB (por ejemplo, el USB-STC-ISP).
  - Como la placa STM32L-Discovery tiene disponibles los pines PB6 y PB7, se hace la conexión como se indica en la figura



EJEMPLOS: ¿CÓMO UTILIZAR Y PROBAR LA USART? (2)

- La conexión con el PC sería la mostrada en la siguiente figura:



TEMA 9: COMUNICACIÓN SERIE ASÍNCRONA (USART)

EJEMPLO: TRANSMISIÓN POR ESPERA ACTIVA

EJEMPLO: TX POR ESPERA ACTIVA (1)

- Se transmite un carácter al PC tras la pulsación del botón USER, empezando por la ‘A’ e incrementando en una unidad. La comunicación es a 9600,8,N,1

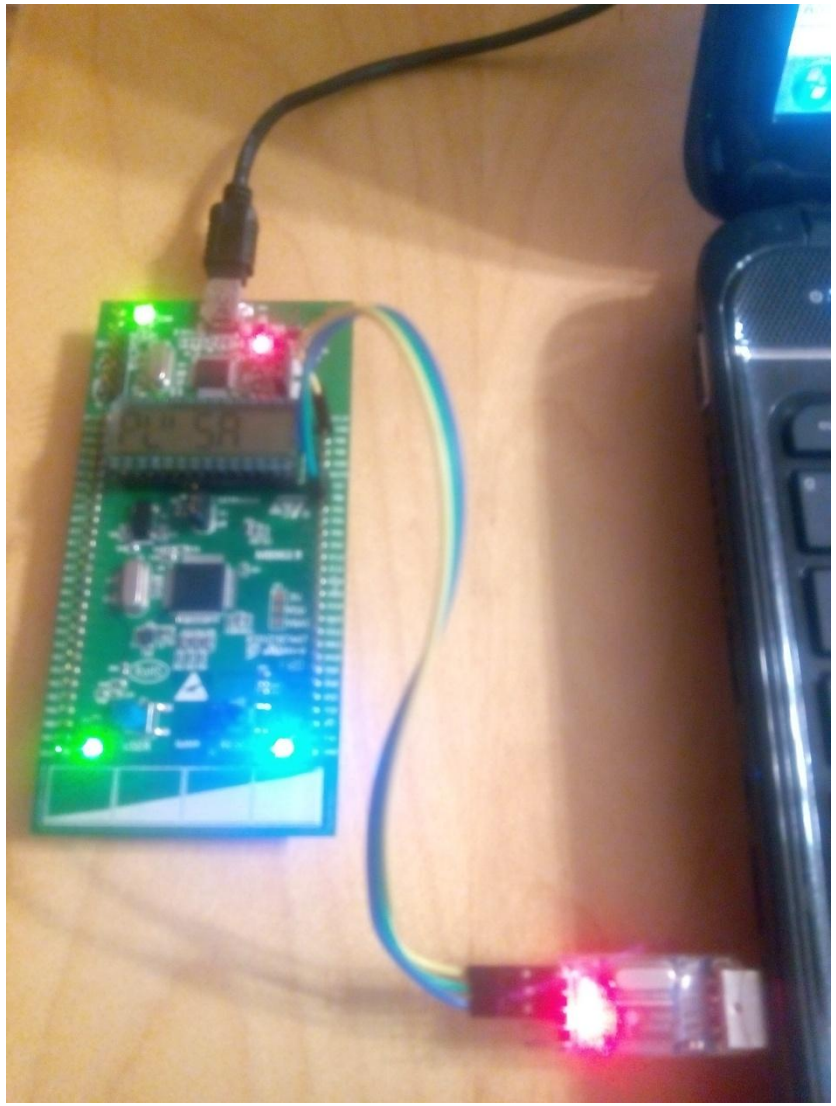
```
#include "stm32l1xx.h"
#include "Biblioteca_SDM.h"
#include "Utiles_SDM.h"

int main(void){

   unsigned char valor = 'A';
   Init_SDM();
   Init_LCD();

   // PA0 (Boton User) como entrada (00)
   GPIOA->MODER &= ~(1 << (0*2 +1));
   GPIOA->MODER &= ~(1 << (0*2));

   // Entrada PA0 sin pull-up, pull-down (00)
   GPIOA->PUPDR &= ~(11 << (0*2));

   // Configuración del USART

   // PB7 y PB6 como AF - USART
   GPIOB->MODER |= (0x01 << (2*7+1)); // (10 la posición de PB6 en MODER)
   GPIOB->MODER &= ~(0x01 << (2*7));
   GPIOB->MODER |= (0x01 << (2*6+1));
   GPIOB->MODER &= ~(0x01 << (2*6));  // (10 la posición de PB7 en MODER)
   GPIOB->AFR[0] = 0x77000000;        // AF7 = 0111 -> La función AF es la función USART
                                      // 0111 en la posición del PB6 y del PB7 en AFR[0])
```

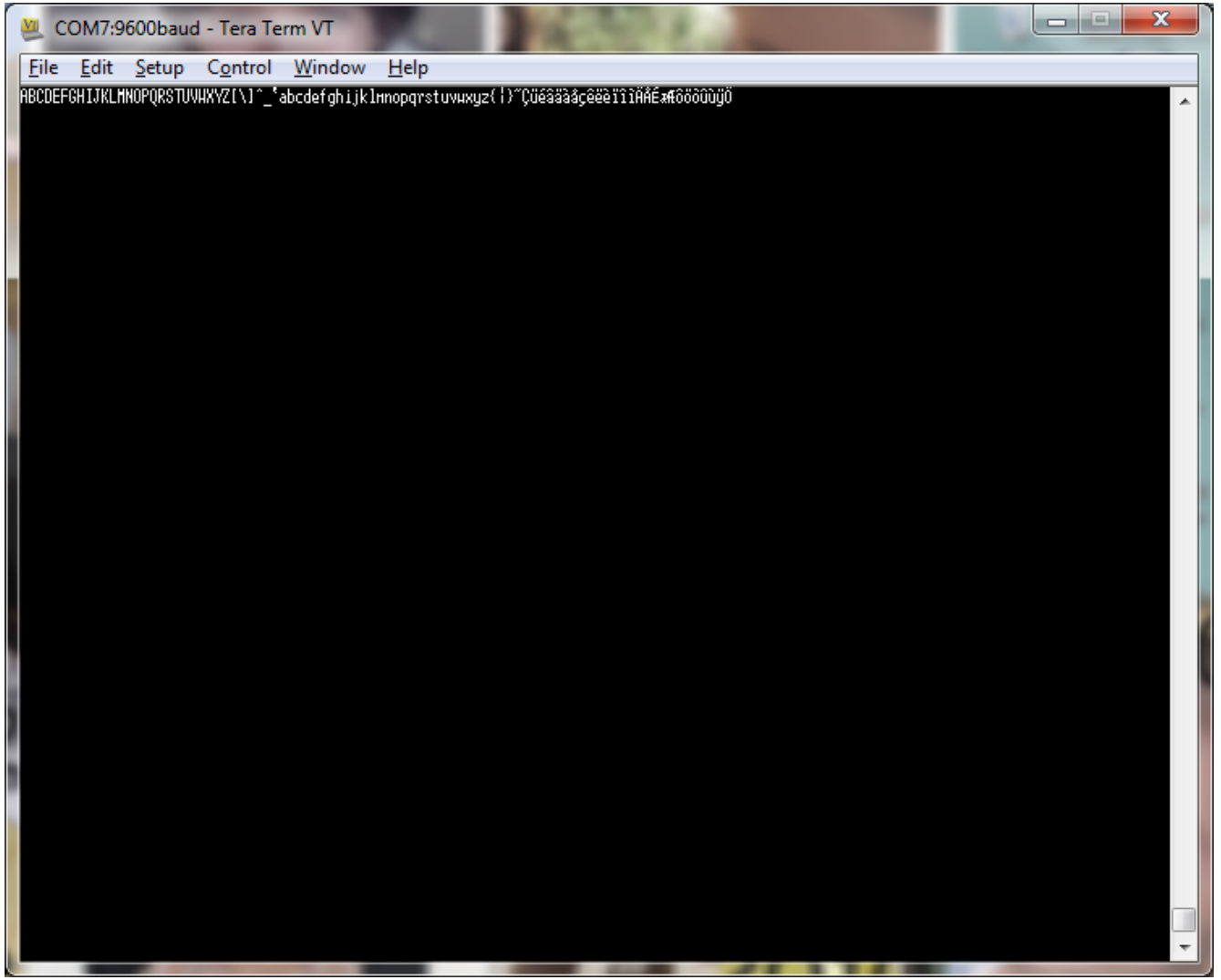
EJEMPLO: TX POR ESPERA ACTIVA (2)

- Se transmite un carácter al PC tras la pulsación del botón USER, empezando por la 'A' e incrementando en una unidad. La comunicación es a 9600,8,N,1

```
// Comunicación 9600,8,N,1 con PCLK = 32MHz
USART1->CR1 = 0x00000008;  // OVER8 = 0 -> Oversampling = 16 -> DIV_fraction[3:0] en el USARTx->BRR usa 4 bits
                           // UE = 0 -> Deshabilitación de la USART
                           // M = 0 -> Tamaño de palabra: 8 bits
                           // PCE = 0 -> Control de paridad deshabilitado
                           // TXEIE, TCIE, RXNEIE = 0 -> Sin IRQs en la comunicación
                           // TE = 1 -> Habilitación de la transmisión
                           // RE = 0 -> Deshabilitación de la recepción
USART1->CR2 = 0x00000000;  // STOP = 00 -> Números de bit de parada = 1
USART1->BRR = 0x00000D05;  // 9600 baudios -> USARTDIV = 208,3125 (mirar tabla de teoría para 32Mhz)
                           // Mantisa = 208 = 0xD0;
                           // Fracción = 0,31*16 = 5 = 0X05;

// Habilitación del USART
USART1->CR1 |= 0x01 << 13; // UE = 1 en el bit 13 del CR1

LCD_Texto("PULSA");        // Nada más arrancar sale PULSA en el display
while (1) {
   if ((GPIOA->IDR&0x00000001)!=0)       // Si pulso PA0 comunico, si no, no hago nada
     {
      while ((GPIOA->IDR&0x00000001)!=0) // Espero un poco para evitar rebotes
        {
         espera(70000);
        }
      while ((USART1->SR & 0x0080)== 0); // Si no puedo enviar (TXE = 0), espero

      USART1->DR = valor;                // En cuanto pueda, coloco el dato en USART1->DR en envío
      valor++;                           // Aumento el carácter a envía en la siguiente pulsación
      }
   }
}
```

PRUEBA DEL EJEMPLO EXPLICADO





Tema 9: Comunicación Serie Asíncrona (USART)

Ejemplo: Tx y Rx por Espera Activa

Ejemplo: Tx y Rx por Espera Activa (1)

- Se espera la recepción de un carácter desde el PC, se incrementa en uno y se transmite ese nuevo valor

```
#include "stm32l1xx.h"
#include "Biblioteca_SDM.h"
#include "Utiles_SDM.h"

int main(void){

   unsigned char valor = 'A';
   Init_SDM();

   // Configuración del USART

   // PB7 y PB6 como AF - USART
   GPIOB->MODER |= (0x01 << (2*7+1)); // (10 la posición de PB6 en MODER)
   GPIOB->MODER &= ~(0x01 << (2*7));
   GPIOB->MODER |= (0x01 << (2*6+1));
   GPIOB->MODER &= ~(0x01 << (2*6));  // (10 la posición de PB7 en MODER)
   GPIOB->AFR[0] = 0x77000000;        // AF7 = 0111 -> La función AF es la función USART
                                      // 0111 en la posición del PB6 y del PB7 en AFR[0])

   // Comunicación 9600,8,N,1 con PCLK = 32MHz
   USART1->CR1 = 0x0000000C;  // OVER8 = 0 -> Oversampling = 16 -> DIV_fraction[3:0] en el USARTx->BRR usa 4 bits
                              // UE = 0 -> Deshabilitación de la USART
                              // M = 0 -> Tamaño de palabra: 8 bits
                              // PCE = 0 -> Control de paridad deshabilitado
                              // TXEIE, TCIE, RXNEIE = 0 -> Sin IRQs en la comunicación
                              // TE = 1 -> Habilitación de la transmisión
                              // RE = 1 -> Habilitación de la recepción
```

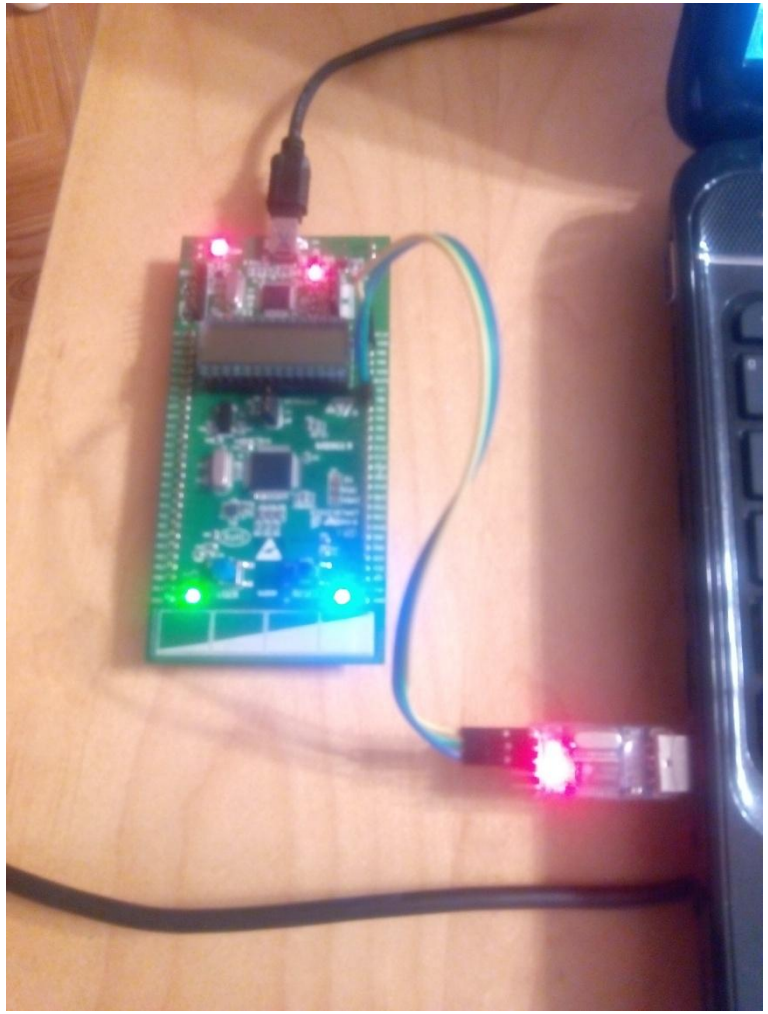
EJEMPLO: TX Y RX POR ESPERA ACTIVA (2)

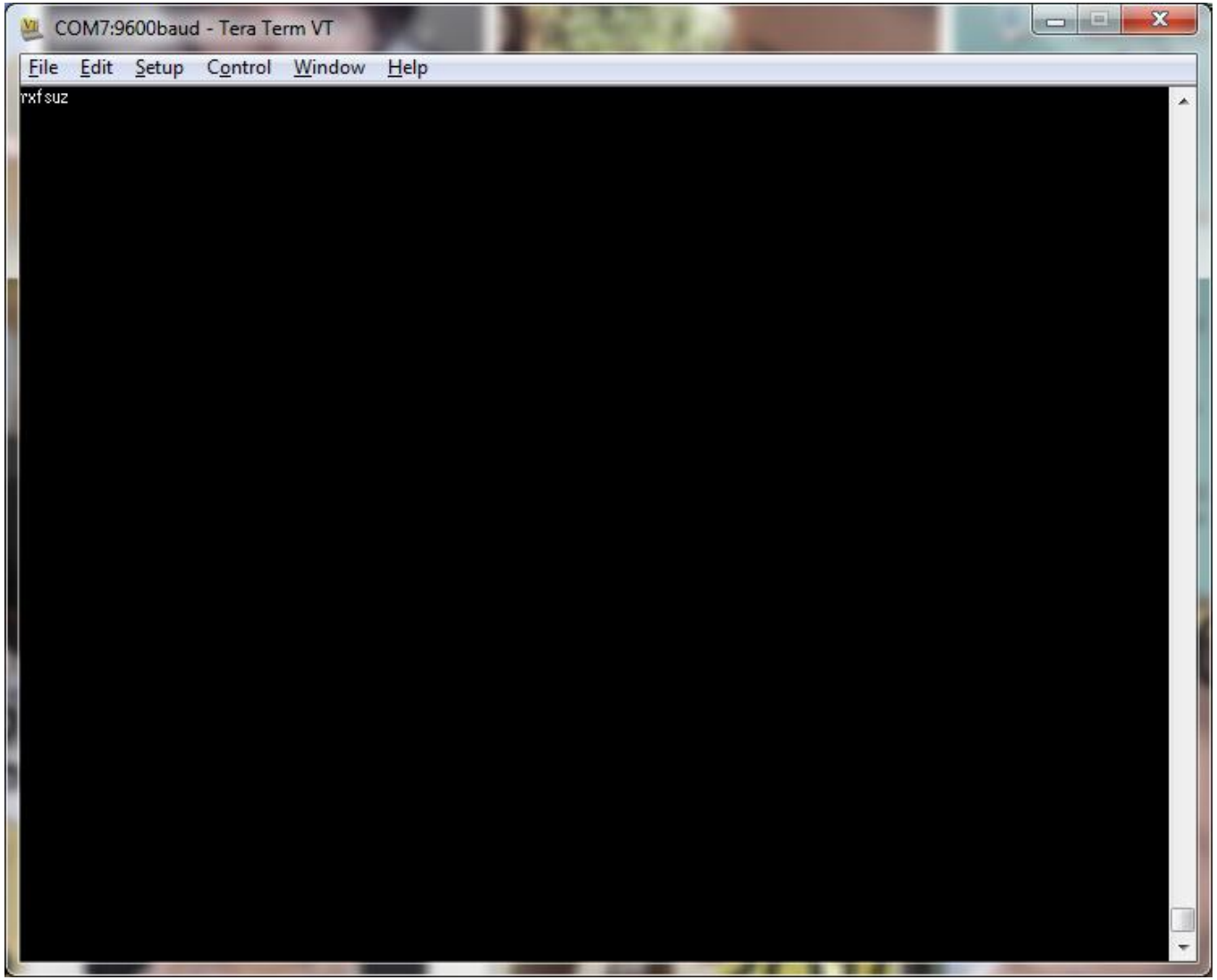
- Se espera la recepción de un carácter desde el PC, se incrementa en uno y se transmite ese nuevo valor

```
    USART1->CR2 = 0x00000000;  // STOP = 00 -> Números de bit de parada = 1
    USART1->BRR = 0x00000D05;  // 9600 baudios -> USARTDIV = 208,3125 (mirar tabla de teoría para 32Mhz)
                               // Mantisa = 208 = 0xD0;
                               // Fracción = 0,31*16 = 5 = 0X05;

    // Habilitación del USART
    USART1->CR1 |= 0x01 << 13; // UE = 1 en el bit 13 del CR1

    while (1) {

            while ((USART1->SR & 0x0020)==0);    // Si no he recibido nada (RXNE = 0), espero

            valor = USART1->DR;                  // Cuando haya recibido, cojo el dato
            valor++;                             // E incremento en un caracter

            while ((USART1->SR & 0x0080)== 0);   // Si no puedo enviar (TXE = 0), espero

            USART1->DR = valor;                  // Cuando pueda enviar, envío el carácter
        }
}
```

PRUEBA DEL EJEMPLO EXPLICADO

- Se pulsa querty en el teclado y el micro devuelve lo mostrado





Tema 9: Comunicación Serie Asíncrona (USART)

Ejemplo: Recepción por Interrupciones

Ejemplo de Uso con Rx por IRQ (1)

- El mismo ejemplo pero con IRQs en la Recepción

```
#include "stm32l1xx.h"
#include "Biblioteca_SDM.h"
#include "Utiles_SDM.h"

unsigned char valor=0;

void USART1_IRQHandler(void)          // Interrupción de la USART
   {
   if ((USART1->SR & 0x0020)!=0)      // Si he recibido algo (RXNE = 1), ejecuto lo siguiente
                        {
      valor = USART1->DR;             // Cojo el valor recibido
                       valor++;       // Incremento en un carácter
      }
   }

int main(void){

    Init_SDM();

    // Configuración del USART

    // PB7 y PB6 como AF - USART
    GPIOB->MODER |= (0x01 << (2*7+1)); // (10 la posición de PB6 en MODER)
    GPIOB->MODER &= ~(0x01 << (2*7));
    GPIOB->MODER |= (0x01 << (2*6+1));
    GPIOB->MODER &= ~(0x01 << (2*6));  // (10 la posición de PB7 en MODER)
    GPIOB->AFR[0] = 0x77000000;        // AF7 = 0111 -> La función AF es la función USART
                                       // 0111 en la posición del PB6 y del PB7 en AFR[0])
```

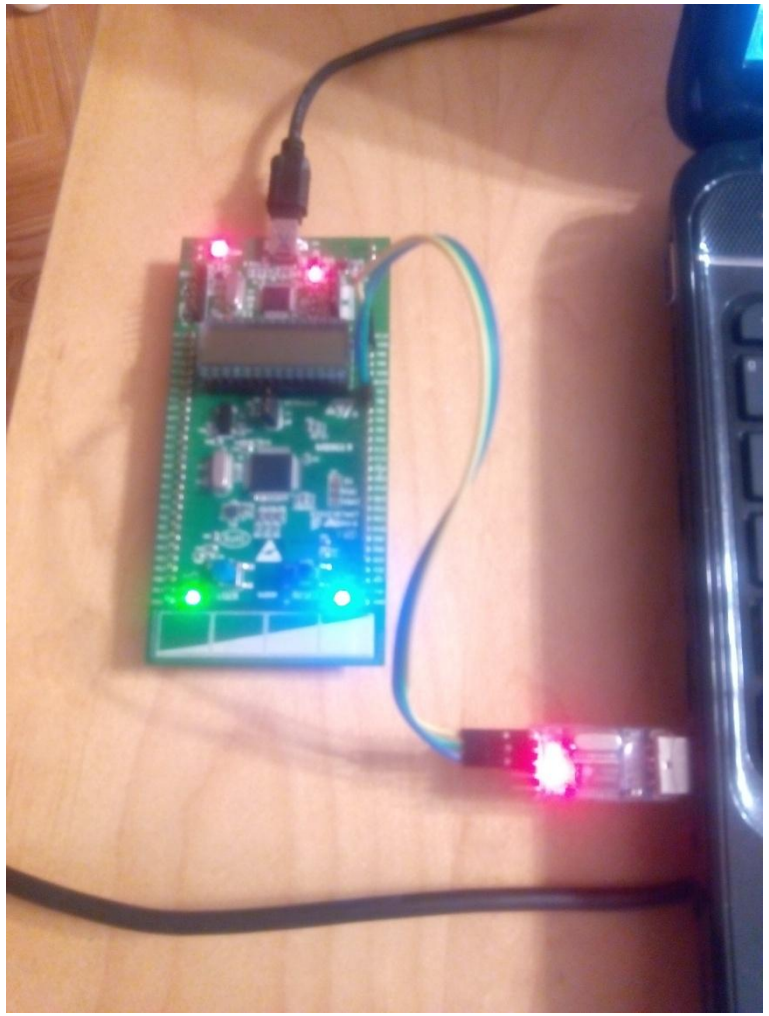
Ejemplo de Uso con Rx por IRQ (2)

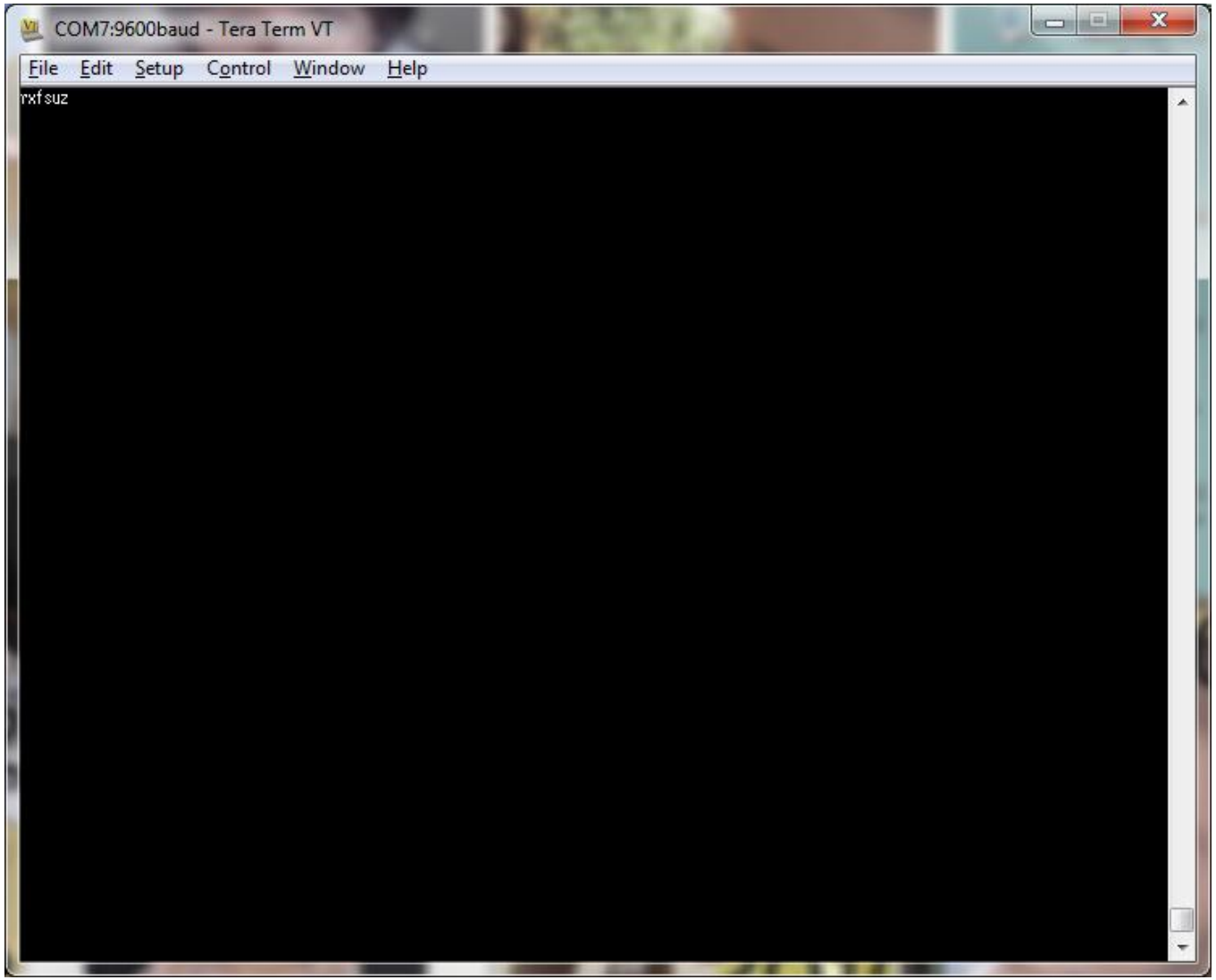
- El mismo ejemplo pero con IRQs en la Recepción

```
// Comunicación 9600,8,N,1 con PCLK = 32MHz
   USART1->CR1 = 0x0000000C;  // OVER8 = 0 -> Oversampling = 16 -> DIV_fraction[3:0] en el USARTx->BRR usa 4 bits
                              // UE = 0 -> Deshabilitación de la USART
                              // M = 0 -> Tamaño de palabra: 8 bits
                              // PCE = 0 -> Control de paridad deshabilitado
                              // TXEIE, TCIE, RXNEIE = 0 -> Sin IRQs en la comunicación
                              // TE = 1 -> Habilitación de la transmisión
                              // RE = 1 -> Habilitación de la recepción
   USART1->CR2 = 0x00000000;  // STOP = 00 -> Números de bit de parada = 1
   USART1->BRR = 0x00000D05;  // 9600 baudios -> USARTDIV = 208,3125 (mirar tabla de teoría para 32Mhz)
                              // Mantisa = 208 = 0xD0;
                              // Fracción = 0,31*16 = 5 = 0X05;

   // Habilitación del USART
   USART1->CR1 |= 0x01 << 13; // UE = 1 en el bit 13 del CR1

   // Habilitacion de la IRQ de la USART
   USART1->CR1 |= 0x01 << 5;          // RXNEIE = 1 -> Habilitación de IRQ por carácter recibido
   NVIC->ISER[1] |= (1 << (37-32));  // Habilita la IRQ de la USART en el NVIC

   while (1) {
      while (valor==0);                    // Si no he recibido nada en la INT de la USART, espero
      while ((USART1->SR & 0x0080)== 0);  // Si no puedo enviar (TXE = 0), espero
      USART1->DR = valor;                  // Cuando pueda enviar, envío el caràcter
      valor=0;                             // Y reseteo "valor" para la siguiente vez
      }
   }
```

PRUEBA DEL EJEMPLO EXPLICADO

- Se pulsa querty en el teclado y el micro devuelve lo mostrado





Ejemplos y Ejercicios

1. ***Análisis de los Ejemplos:*** Realice el diagrama de flujo de los ejemplos, cree los proyectos y al escribir el código comente lo que hace cada línea (a nivel funcional). Ejecútelos y depúrelos.
2. Cree un programa completo que genere señales cuadradas de duty cycle del 50%, en el que la frecuencia venga dada por puerto serie, y el periodo medido se envíe por puerto serie cada dos segundos.
    1. Considere que la frecuencia la da en Hz con 3 caracteres (siempre tres), seguido de un retorno de carro.
    2. Envíe el periodo en milisegundos.
3. Cree un programa que, a través del puerto serie, se pida comprobar cada una de las funcionalidades de la placa de la que dispone.
    1. Para eso, considere juntar y adaptar todos los ejercicios realizados anteriormente durante el curso.