uc3m

Departamento Tecnología Electrónica
dte UC3M

Tema 10: Comunicación Serie Asíncrona (USART)

Sistemas Digitales Basados en Microprocesadores

Universidad Carlos III de Madrid

Dpto. Tecnología Electrónica

Nota: Las figuras utilizadas para ilustrar las características y funcionalidades del microcontrolador del curso se han obtenido de la documentación técnica disponible en https://www.st.com/en/microcontrollers-microprocessors/stm32l151-152.html

Índice

- Conceptos Generales
  - Comunicaciones Serie
  - Comunicación Serie Asíncrona
- Arquitectura, Configuración y Uso
- ¿Cómo utilizar y probar la USART?
- Ejemplos
  - Tx por Espera Activa
  - Rx y Tx por Espera Activa
  - Rx por IRQ y Tx por Espera Activa
- Uso de la USART a nivel de Registros:
  - Registros de Control
  - Registros de Datos
  - Registros de Estado
  - Ejemplos a nivel de Registros

Conceptos Generales

Clasificación de la Comunicación 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

Nivel de Carácter

- La comunicación 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

ht https://www.meccanismocomplesso.org/en/serial-communication/

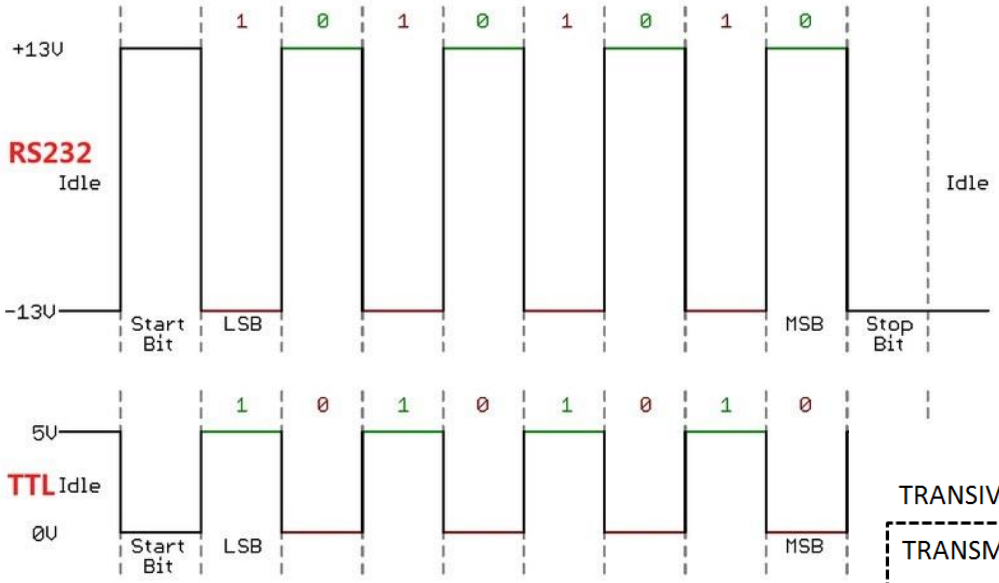
Parámetros de Comunicación

- 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)

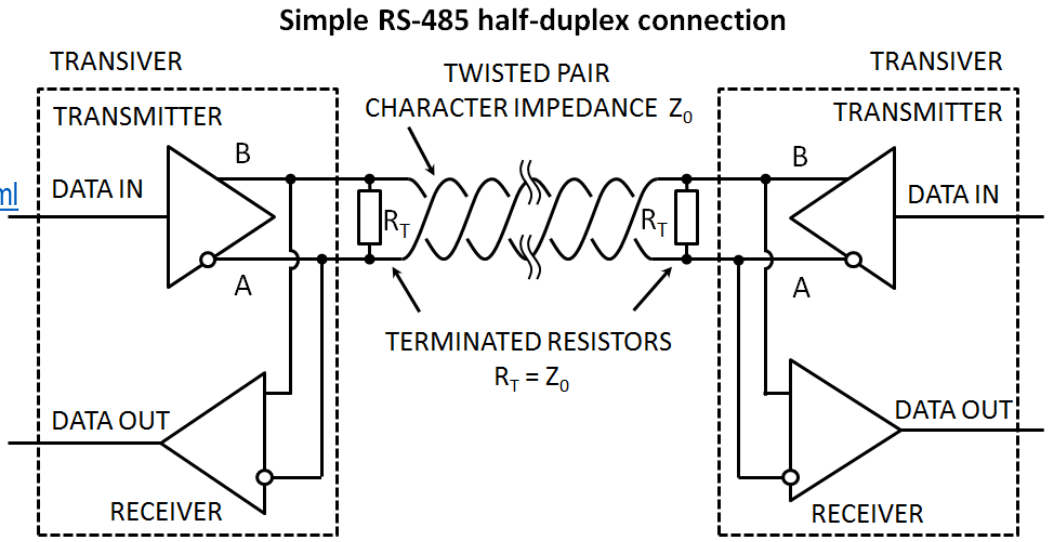
Capa Física

- 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.

Capa Física



http://231.261.boot.msr-recycling.de/rs232-timing-diagram.html



DATA IN	A	B	DATA OUT
0	1	0	0
1	0	1	1

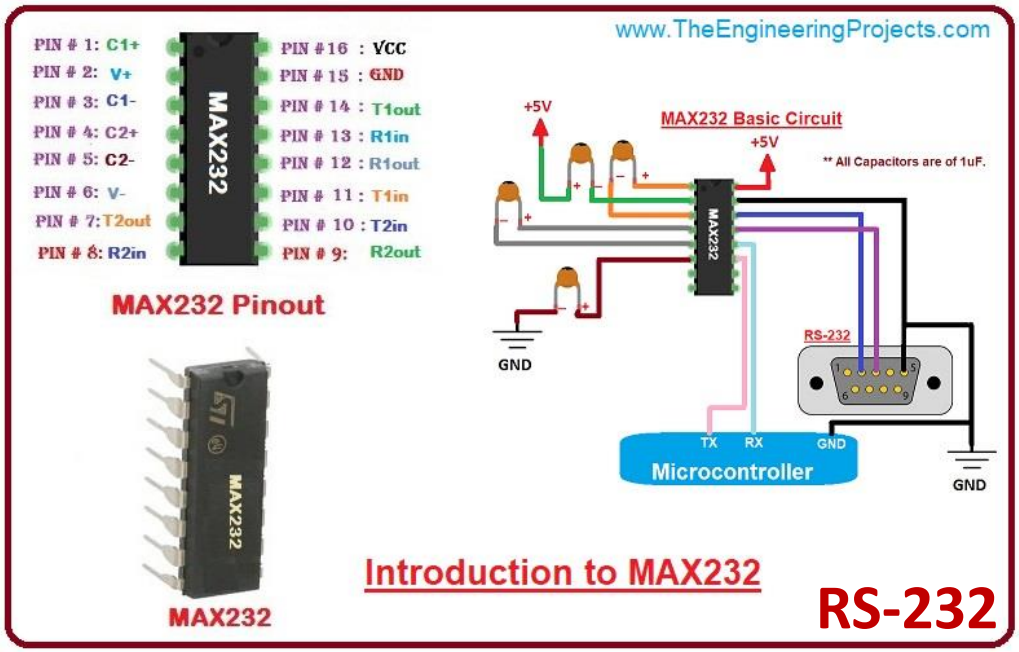
https://stackoverflow.com/tags/rs485/info

Capa de Protocolo

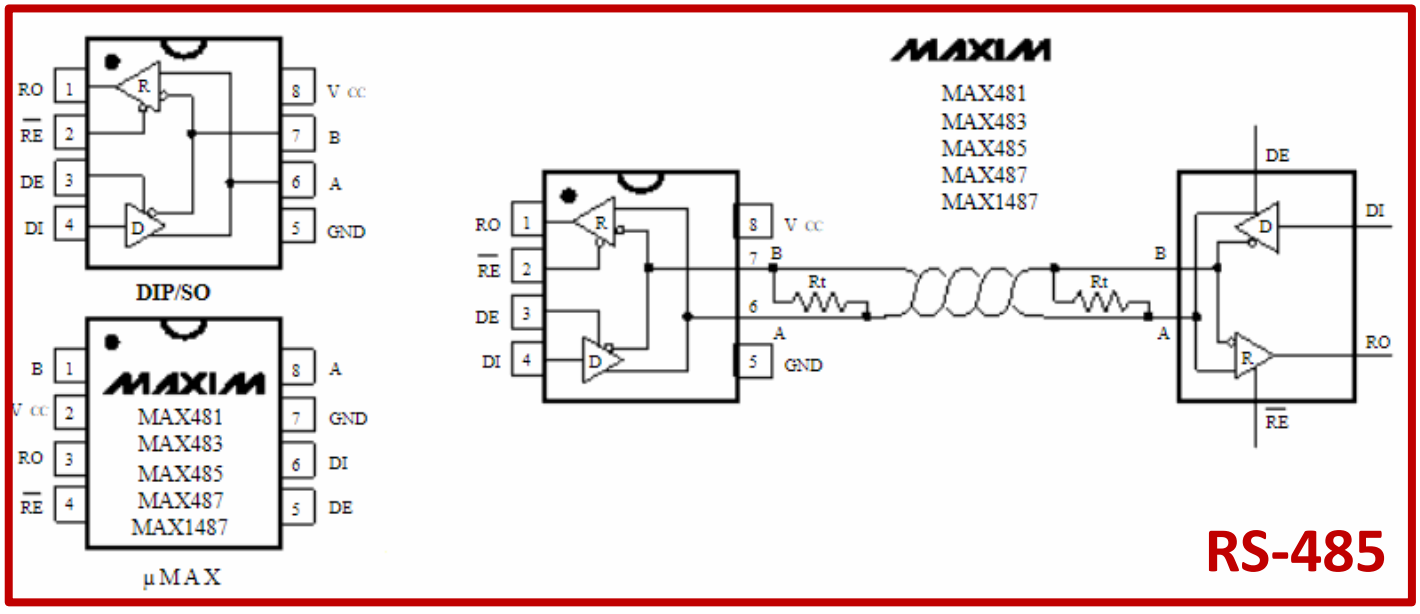
- La comunicación serie asíncrona suele ser ***full-duplex***, ya que se contempla una línea de Tx y otra de Rx
  - En concreto, se utiliza frecuentemente la conexión **null-modem** (o módem nulo):
    - Tx del dispositivo 1 conectado al Rx del dispositivo 2
    - Rx del dispositivo 1 conectado al Tx 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
  - O también control de flujo software (Xon – Xoff)
- En la actualidad no se usa **ningún control de flujo** en las capas bajas de la comunicación, sino que se deja a nivel de capa de aplicación

Convertidores / Transductores

https://www.theengineeringprojects.com/2017/07/introduction-to-max232.html

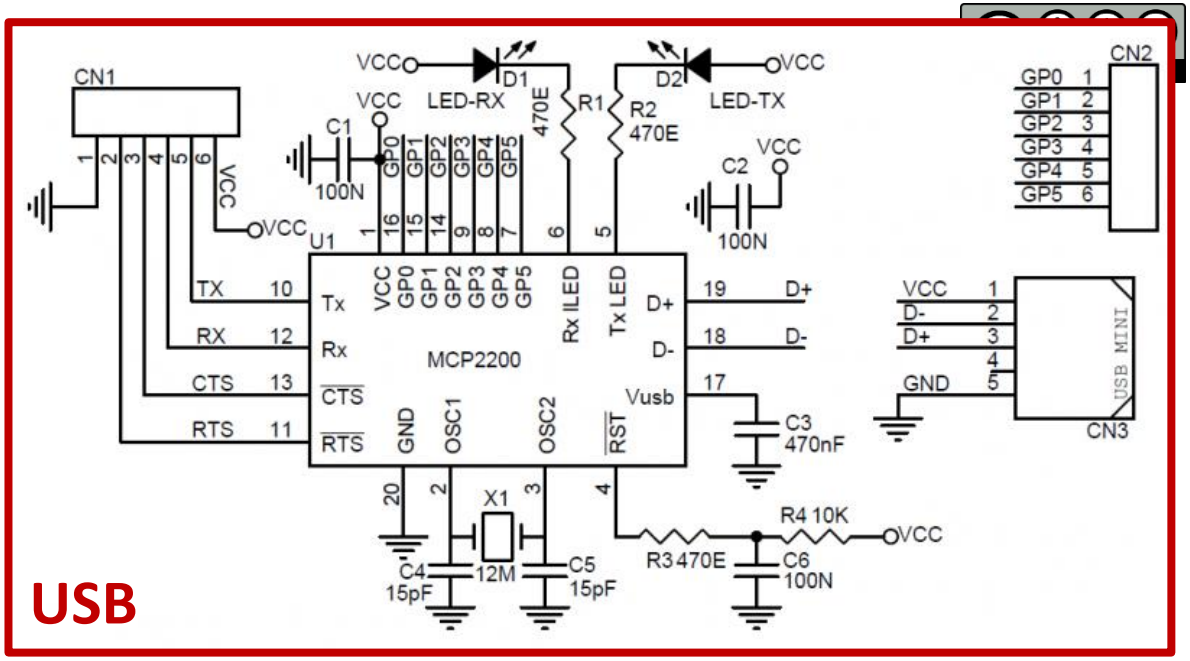


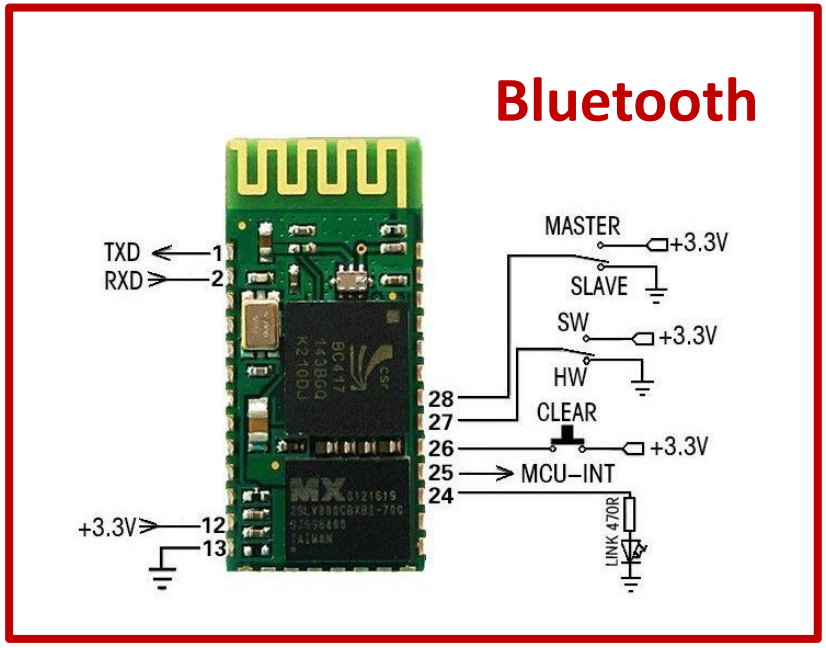
https://www.industrialshields.com/es_ES/support/help/ardbox-analog-hf-8/ardbox-analog-rs-485-25



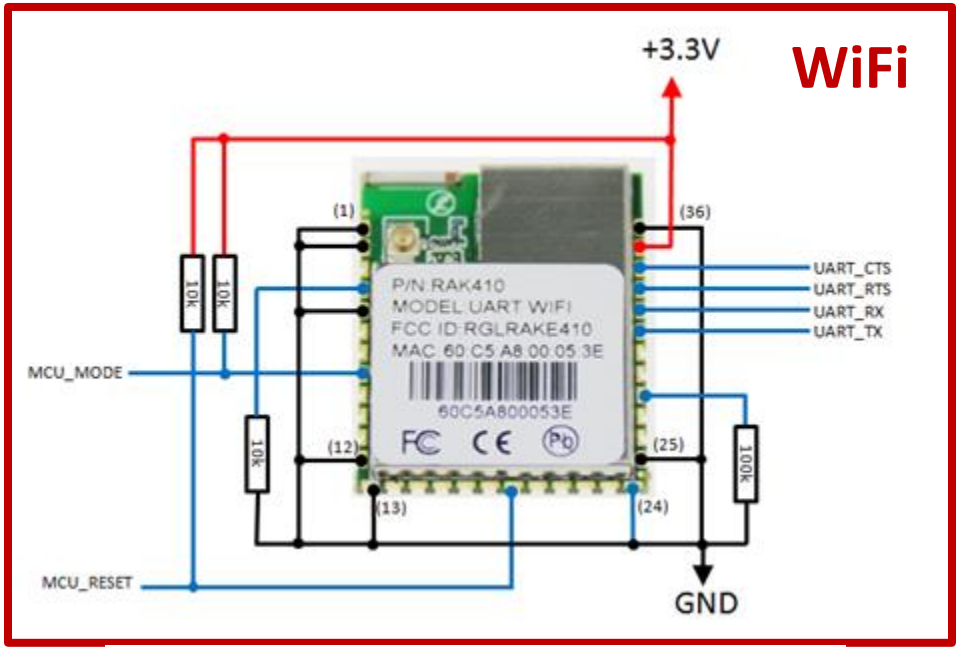
Convertidores / Transductores

https://www.electronics-lab.com/project/usb-uart-converter-gpio-mcp220/





https://www.ebay.com/itm/BC04-B-Bluetooth-to-UART-Module-Industrial-Master-Slave-Wireless-Bluetooth-AK-/281782868122



http://waijung.aimagin.com/index.htm?hardware_setup.htm

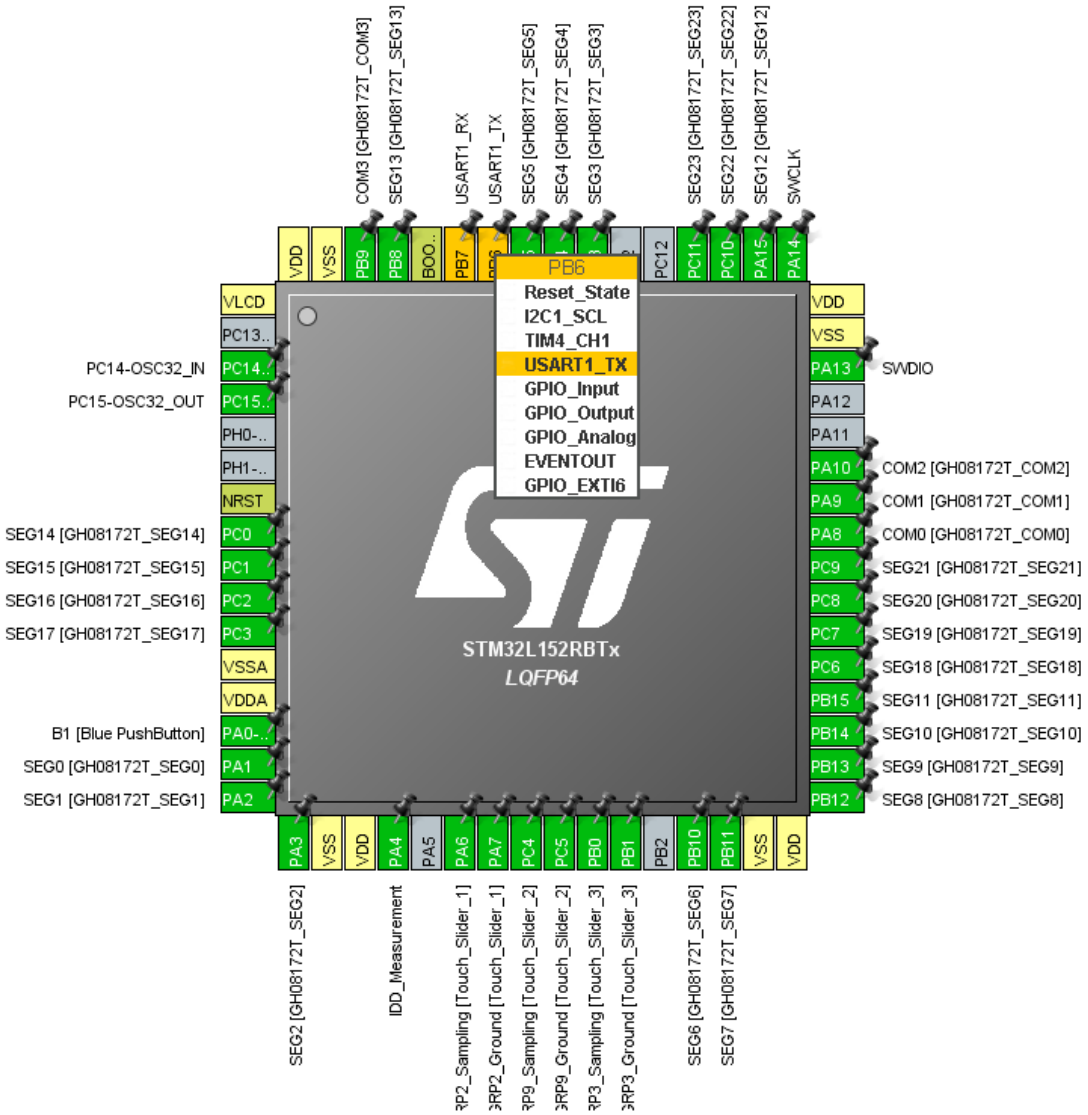
Arquitectura, Configuración y Uso

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

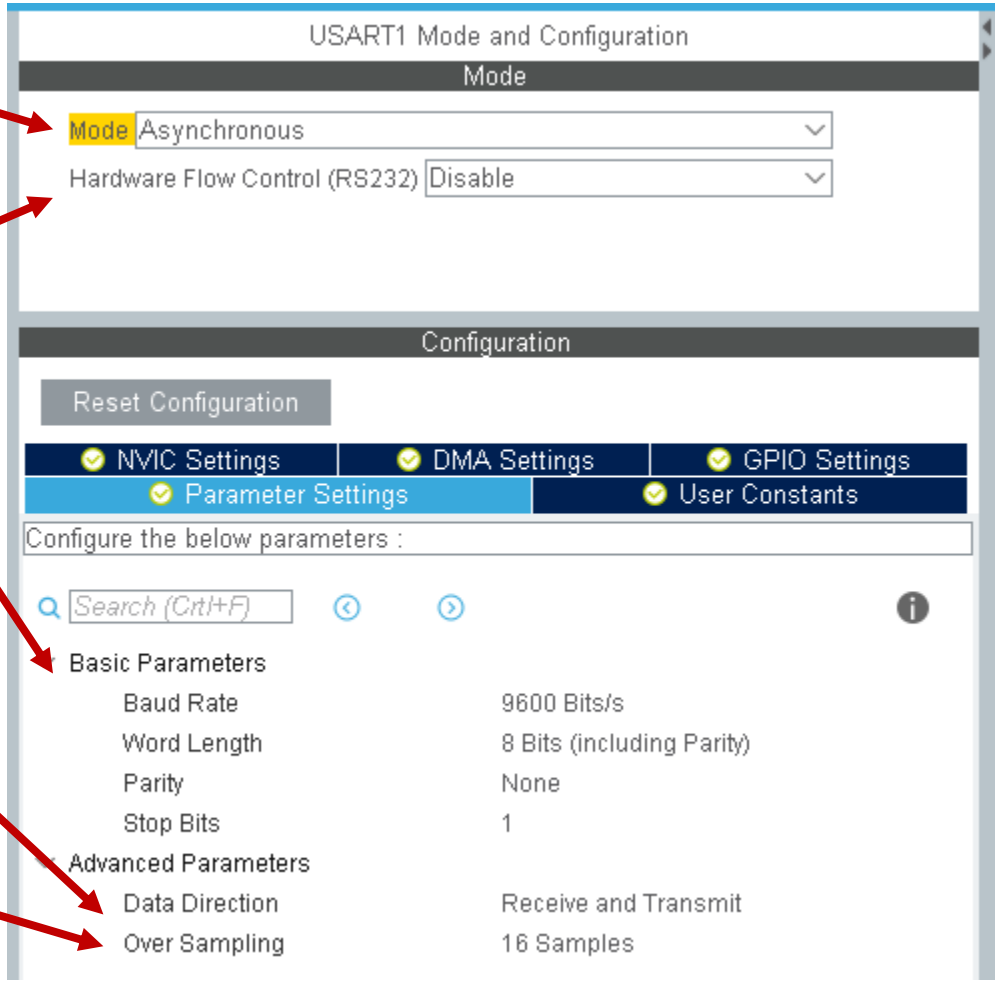
Configuración

- Se configuran los pines (PB6 y PB7) como funcionalidad USART (USART1)



Configuración

- Configure la USART en modo Asíncrono
- Deshabilite el control de flujo hardware
- Seleccione:
  - Baud Rate
  - Número de bits de datos
  - Paridad
  - Bits de parada
- Decida si va a comunicar en un sentido o en ambos
- Decida el sobre-muestreo (8 o 16)
  - Si no está seguro, déjelo como recomiende el programa



Uso de la USART

Por Espera Activa

- Inicialización:
  - `HAL_UART_Init(h)`
- Transmisión:
  - `HAL_UART_Transmit(h, b, s, t)`
- Recepción:
  - `HAL_UART_Receive(h, b, s, t)`

Por Interrupciones

- Inicialización:
  - `HAL_UART_Init(h)`
- Transmisión:
  - `HAL_UART_Transmit_IT(h, b, s)`
  - `HAL_UART_TxCpltCallback(h)`
- Recepción:
  - `HAL_UART_Receive_IT(h, b, s)`
  - `HAL_UART_RxCpltCallback(h)`
- Gestión de Errores:
  - `HAL_UART_ErrorCallback(h)`

Parámetros:

- **h** – handler: `UART_HandleTypeDef *`
- **b** – buffer: `uint8_t *`
- **s** – tamaño buffer: `uint16_t`
- **t** – timeout en ms: `uint32_t`

¿Cómo utilizar y probar la USART?

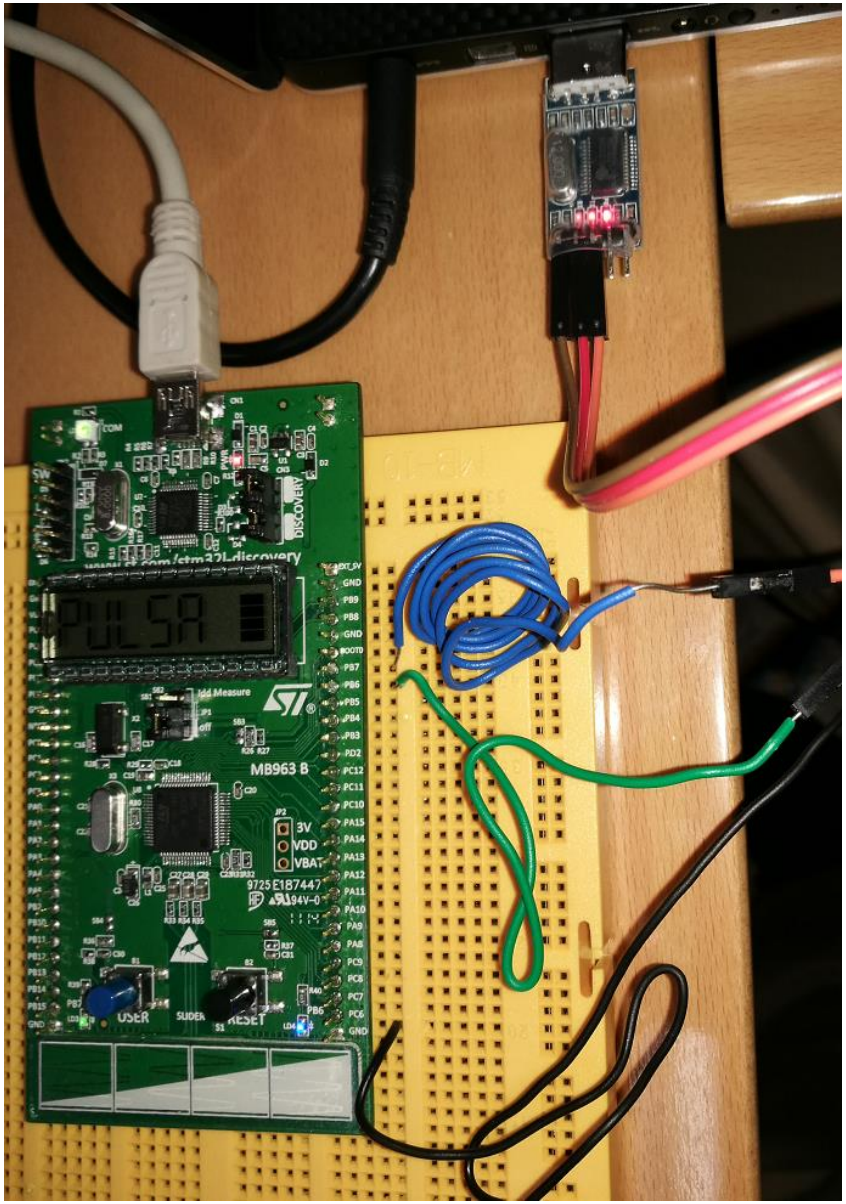
¿Cómo utilizar y probar la USART?

- La mayoría de los ordenadores actuales no poseen puertos serie (COM), sino solamente puertos USB
- La opción más sencilla es utilizar un módulo de conversión TTL-USB
  - Por ejemplo, módulos basados en el PL-2303 como https://www.amazon.es/s/ref=nb_sb_noss?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&url=search-alias%3Daps&field-keywords=ttl-usb+converter+pl2303&rh=i%3Aaps%2Ck%3Attl-usb+converter+pl2303)

https://cherryjulong.en.ec21.com/USB_2.0_To_Ttl_Uart--5610297_5636844.html

¿Cómo utilizar y probar la USART?

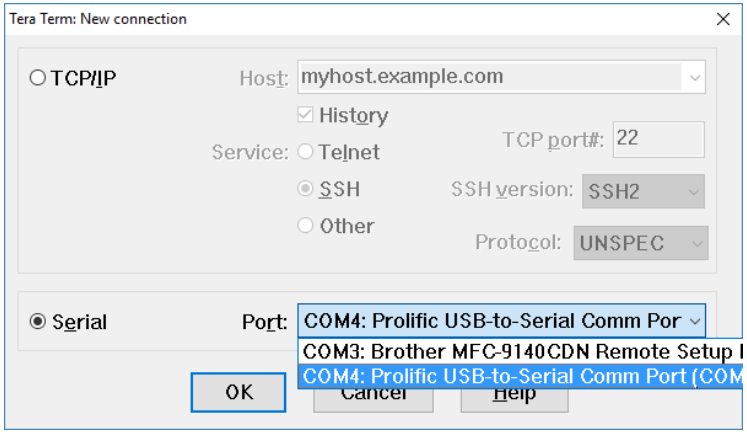
- El conexionado es sencillo:
  - Conecta el terminal GND del conversor USB al GND de la placa
  - Conecta el terminal RxD del conversor USB al TxD de la placa
  - Conecta el terminal TxD del conversor USB al RxD de la placa
  - NOTA: los terminales de alimentación NO hay que conectarlos

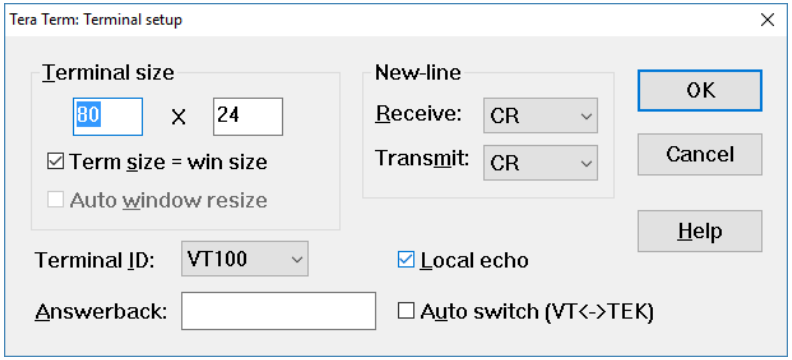


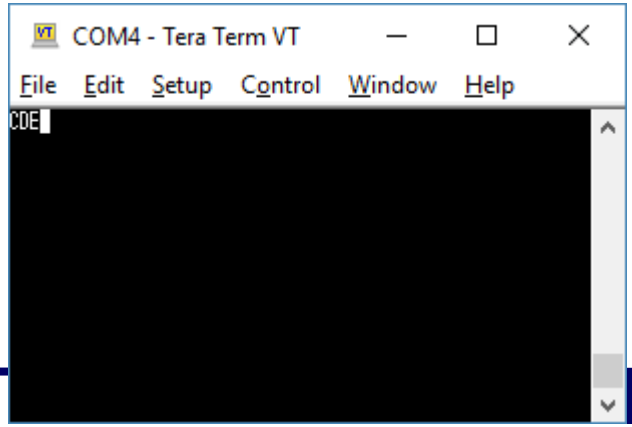
¿Cómo utilizar y probar la USART?

- En cualquiera de los dos casos será necesario que se instalen unos drivers del conversor en el sistema operativo.
  - En algunos casos dichos drivers se instalan automáticamente, en otros casos hay que buscar el driver en la red e instalarlo a mano.
  - En Aula Global están los drivers para Windows, tanto en su versión de 32-bits como de 64-bits.
- El driver convierte la conexión USB en un puerto serie, y entonces puede ser utilizado por cualquier programa de conexión serie, como por ejemplo el Hyperterminal de Windows (que desde Windows 7 NO se proporciona directamente con el sistema operativo) o el Tera Term (https://osdn.net/projects/ttssh2/releases/)
- Para el caso del Tera Term, estos son los pasos para usarlo una vez instalado y arrancado:
  - Selecciona comunicación Serial y ahí localice el puerto serie asociado
  - Activa “Local echo” en el Setup->Terminal, para así poder ver los caracteres que tecleas en el Tera Term
  - Lo que envía la placa lo verás en la pantalla del Tera Term
  - Lo que escribas en la pantalla del Tera Term se enviará a la placa
  - Recuerda que lo que se ve y se escribe en la pantalla son caracteres ASCII

¿Cómo utilizar y probar la USART?







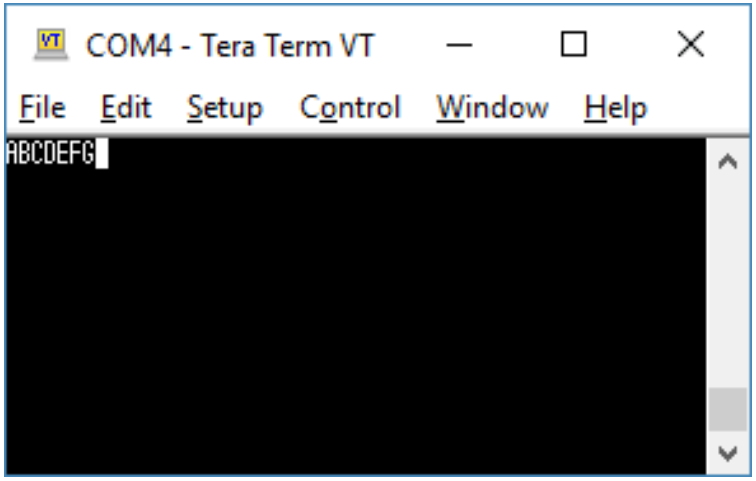
Ejemplos

Ejemplo: Tx por Espera Activa

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

```
/* USER CODE BEGIN 1 */
uint8_t valor = 'A';
/* USER CODE END 1 */
```

```
/* USER CODE BEGIN 2 */
BSP_PB_Init(BUTTON_USER, BUTTON_MODE_GPIO);
/* USER CODE END 2 */
```



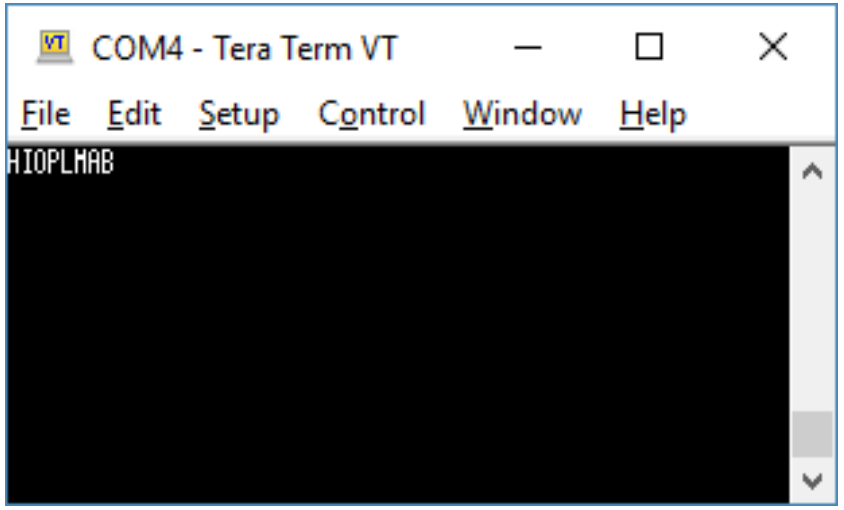

```
/* USER CODE BEGIN WHILE */
while (1)
{
  if (BSP_PB_GetState(BUTTON_USER) == 1) {
    HAL_UART_Transmit(&huart1, &valor, 1, 10000);
    valor++;
    espera(10000);
    while (BSP_PB_GetState(BUTTON_USER)==1);
  }
  /* USER CODE END WHILE */
```

Ejemplo: Tx y Rx por Espera Activa

- Se espera la recepción de un carácter, se muestra en el LCD, se incrementa en una unidad y se transmite ese nuevo valor
  - El ejemplo envía sucesivamente los caracteres HOLA (con local echo en Tera Term)

```
/* USER CODE BEGIN 1 */
uint8_t texto[7];
/* USER CODE END 1 */
```

```
/* USER CODE BEGIN 2 */
BSP_LCD_GLASS_Init();
BSP_LCD_GLASS_BarLevelConfig(0);
BSP_LCD_GLASS_Clear();
/* USER CODE END 2 */
```




```
/* USER CODE BEGIN WHILE */
while (1)
{
    if (HAL_UART_Receive(&huart1, texto, 1, 10000)==HAL_OK){
       BSP_LCD_GLASS_DisplayString((uint8_t *)texto);
       texto[0]++;
       HAL_UART_Transmit(&huart1, texto, 1, 10000);
       }
    /* USER CODE END WHILE */
```

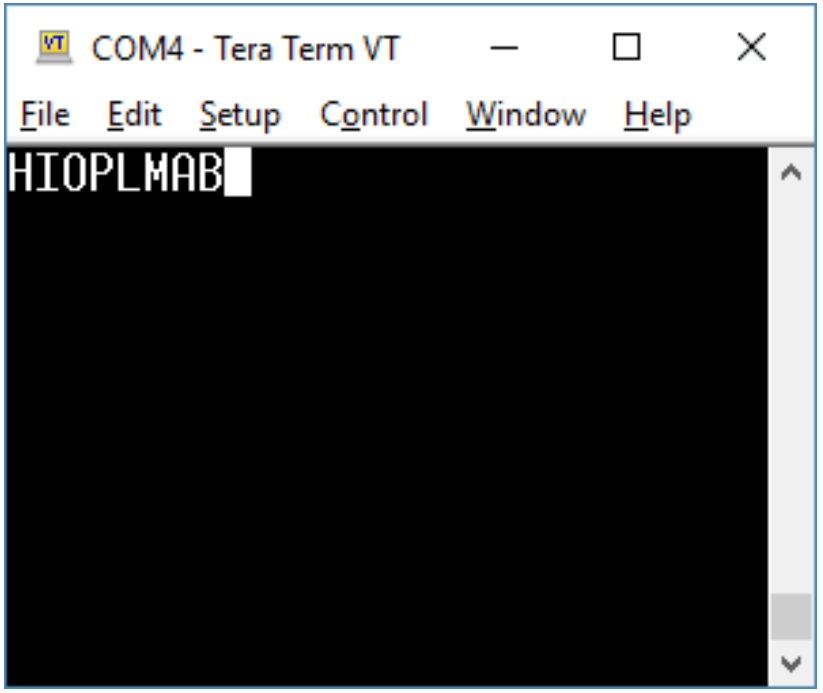
Ejemplo: Rx por IRQ y Tx por Espera Activa

- El mismo ejemplo pero con IRQs en la Recepción

```
/* USER CODE BEGIN PV */
uint8_t texto[7] = "      ";
/* USER CODE END PV */
```

```
/* USER CODE BEGIN 2 */
BSP_LCD_GLASS_Init();
BSP_LCD_GLASS_BarLevelConfig(0);
BSP_LCD_GLASS_Clear();
HAL_UART_Receive_IT(&huart1, texto, 1);
/* USER CODE END 2 */
```

```
/* USER CODE BEGIN WHILE */
while (1)
{
  while (texto[0]==0);
  BSP_LCD_GLASS_DisplayString(texto);
  texto[0]++;
  HAL_UART_Transmit(&huart1, texto, 1, 10000);
  texto[0]=0;
  /* USER CODE END WHILE */
```




```
/* USER CODE BEGIN 4 */
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
  HAL_UART_Receive_IT(huart, texto, 1); // Vuelve a activar Rx por haber acabado el buffer
}
/* USER CODE END 4 */
```

Notas sobre el uso de Tx por IRQ

- Las IRQs saltan cuando el flag correspondiente se activa
- El flag de Tx de la USART siempre está activo, por lo que saltaría constantemente la IRQ
- Ese flag sólo se desactiva cuando está transmitiendo, y se vuelve a poner cuando está listo para transmitir más
- Por tanto, las IRQs en la Tx sólo se utilizan para enviar grandes tramas de datos (bastantes bytes), y en ese caso el procedimiento sería:
  - Definir de forma global una variable que sea capaz de albergar la mayor trama de datos a definir
  - En el programa principal, rellenar dicha variable con los datos a transmitir
    - Puede también tenerse que compartir una variable que indique la longitud de dichos datos
  - Activar las interrupciones por Tx
  - Por su parte la RAI debe:
    - Coger el byte que toca de dicho mensaje y transmitirlo
    - Incrementar el contador de bytes
    - Y si el contador de bytes llega al valor de la longitud de los datos a transmitir, inhibir la IRQ.
- Al gestionar las interrupciones por Tx con las HAL, estas precauciones pueden no tener aplicación, ya que la HAL puede resolver todos estos conflictos

Uso de la USART a nivel de Registros

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. Pon a 1 el bit TE en USARTx→CR1
  3. Decide el número de bits de parada en USARTx→CR2
  4. Selecciona el baudrate en USARTx→BRR
  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.

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. Habilita la recepción poniendo RE=1 en USARTx→CR1
  3. Programa el número de bits de parada deseado en USARTx→CR2
  4. Selecciona el baudrate deseado en USARTx→BRR
  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

Generación 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

Generación 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.

Generación de Baudrates

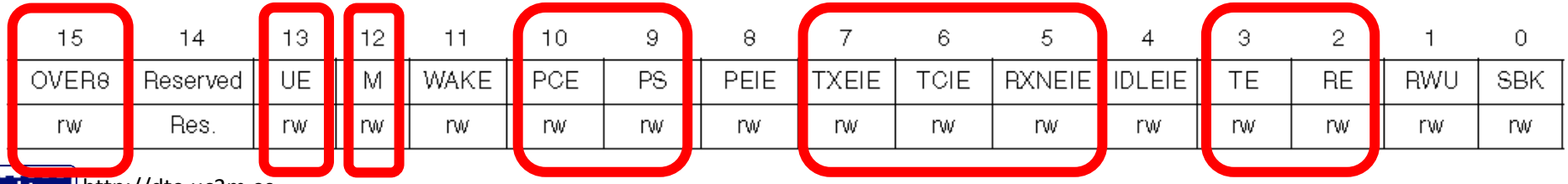
Table 91. Error calculation for programmed baud rates at f_{PCLK} = 16 MHz or f_{PCLK} = 32 MHz), oversampling by 8(1)

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

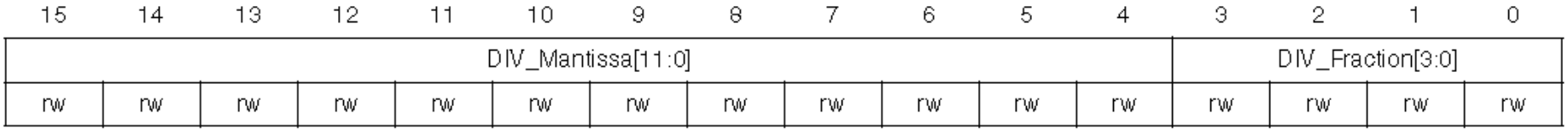
- **USARTx→CR2** – Control Register 2:
  - *LINEN: no se usa*
  - STOP – números de bit de parada:
    - 00 – 1 bit; 01 – 0.5 bits; 10 – 2 bits; 11 – 1,5 bits
  - *CLKEN, CPOL, CPHA, LBCL, LBDIE, LBDL, ADD: no se usan*

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*
- **USARTx→GTPR** – Guard time and prescaler Register:
  - *No se usa*

USART: Registros de Control

- **USARTx→BRR** – Baud Rate Register:
  - Registro de 16 bits:
    - USARTx→BRR[15:4] – DIV_Mantissa
    - USARTx→BRR[3:0] – DIV_Fraction



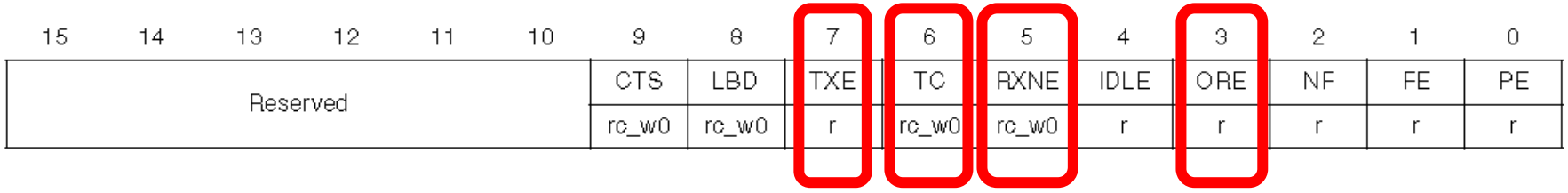
USART: Registros de Datos

- **USARTx→DR** – Data Register
  - Es un registro de 16 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.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							DR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

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: 1.- Lee el USARTx→ SR, 2.- Lee el USARTx→ DR
    - *NF, FE, PE: no se van a utilizar.*



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
						rc_w0	rc_w0	r	rc_w0	rc_w0	r	r	r	r	r

Ejemplos a nivel de Registros

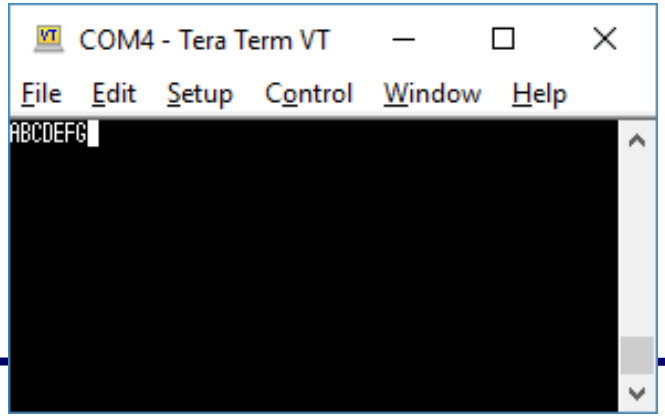
Ejemplo: Tx por Espera Activa

- Se transmite un carácter cada vez, 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();
   GPIOA->MODER &= ~(1 << (0*2 +1));
   GPIOA->MODER &= ~(1 << (0*2));
   GPIOA->PUPDR &= ~(11 << (0*2));
   GPIOB->MODER |= (0x01 << (2*6+1));
   GPIOB->MODER &= ~(0x01 << (2*6));
   GPIOB->AFR[0] &= 0xF0FFFFFF;
   GPIOB->AFR[0] |= 0x07000000;
   USART1->CR1 = 0x00000008;
   USART1->CR2 = 0x00000000;
   USART1->BRR = 0x00000D05;
   USART1->CR1 |= 0x01 << 13;
   LCD_Texto("PULSA");
```

```
   while (1) {
      if ((GPIOA->IDR&0x00000001)!=0) {
         while ((GPIOA->IDR&0x00000001)!=0){
            espera(70000);
            }
         while ((USART1->SR & 0x0080)== 0);
         USART1->DR = valor;
         valor++;
         }
      }
   }
```

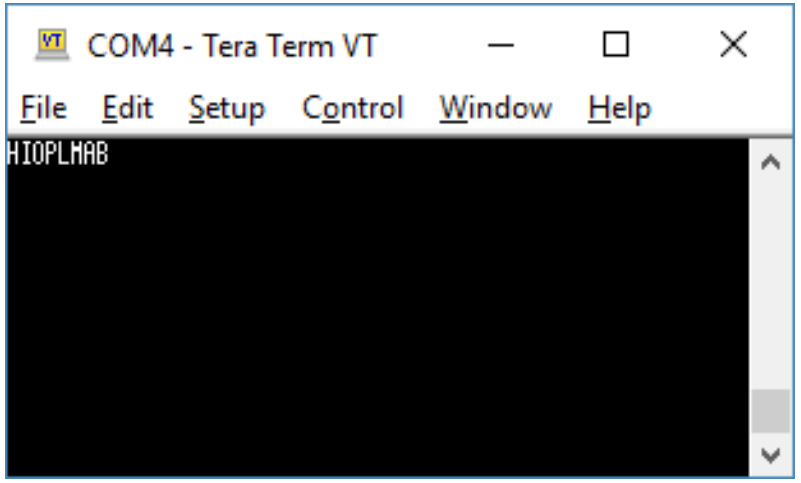


Ejemplo: Tx y Rx por Espera Activa

- Se espera la recepción de un carácter, se muestra en el LCD, se incrementa en una unidad y se transmite ese nuevo valor
  - El ejemplo envía sucesivamente los caracteres HOLA (con local echo en Tera Term)

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

int main(void){
   unsigned char texto[6] = "";
   Init_SDM();
   Init_LCD();
   GPIOB->MODER |= (0x01 << (2*7+1));
   GPIOB->MODER &= ~(0x01 << (2*7));
   GPIOB->MODER |= (0x01 << (2*6+1));
   GPIOB->MODER &= ~(0x01 << (2*6));
   GPIOB->AFR[0] &= 0x00FFFFFF;
   GPIOB->AFR[0] |= 0x77000000;
   USART1->CR1 = 0x0000000C;
   USART1->CR2 = 0x00000000;
   USART1->BRR = 0x00000D05;
   USART1->CR1 |= 0x01 << 13;
   while (1) {
      while ((USART1->SR & 0x0020)==0);
      texto[0] = USART1->DR;
      LCD_Texto(texto);
      texto[0]++;
      while ((USART1->SR & 0x0080)== 0);
      USART1->DR = texto[0];
      }
   }
```



Ejemplo: Rx por IRQ y Tx por Espera Activa

- El mismo ejemplo pero con IRQs en la Recepción

```
unsigned char valor=0;

void USART1_IRQHandler(void) {
   if ((USART1->SR & 0x0020)!=0) {
      valor = USART1->DR;
      }
   }

int main(void){
   unsigned char texto[6] = "";
   Init_SDM();
   Init_LCD();
   GPIOB->MODER |= (0x01 << (2*7+1));
   GPIOB->MODER &= ~(0x01 << (2*7));
   GPIOB->MODER |= (0x01 << (2*6+1));
   GPIOB->MODER &= ~(0x01 << (2*6));
   GPIOB->AFR[0] &= 0x00FFFFFF;
   GPIOB->AFR[0] |= 0x77000000;
   USART1->CR1 = 0x0000000C;
   USART1->CR2 = 0x00000000;
   USART1->BRR = 0x00000D05;
   USART1->CR1 |= 0x01 << 13;
   USART1->CR1 |= 0x01 << 5;
   NVIC->ISER[1] |= (1 << (37-32));
```

```
while (1) {
     while (valor==0);
     texto[0] = valor;
     LCD_Texto(texto);
     while ((USART1->SR & 0x0080)== 0);
     USART1->DR = (++valor);
     valor=0;
     }
  }
```