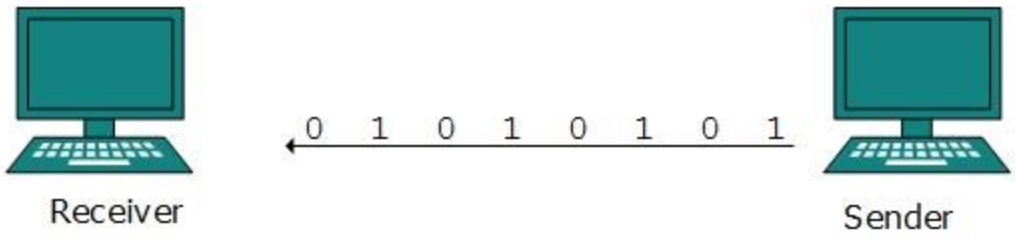
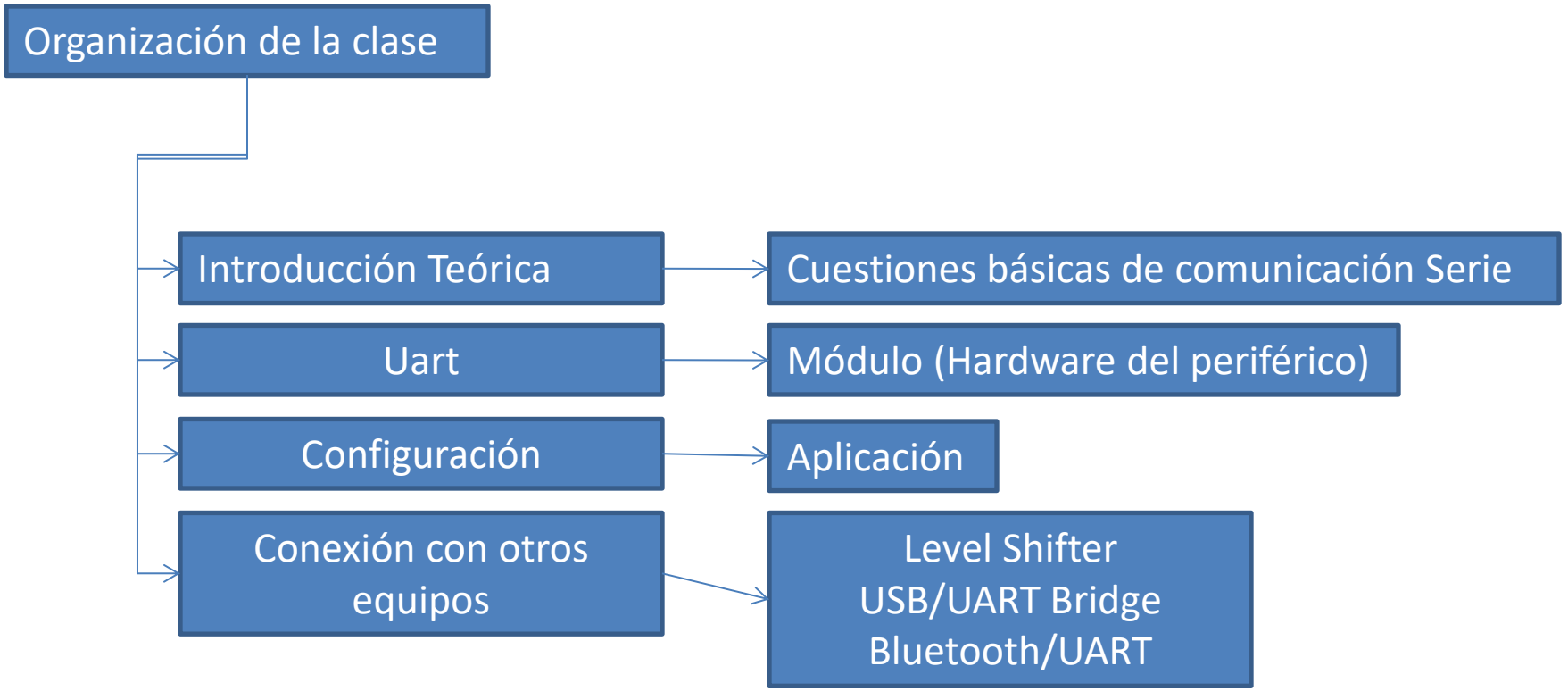
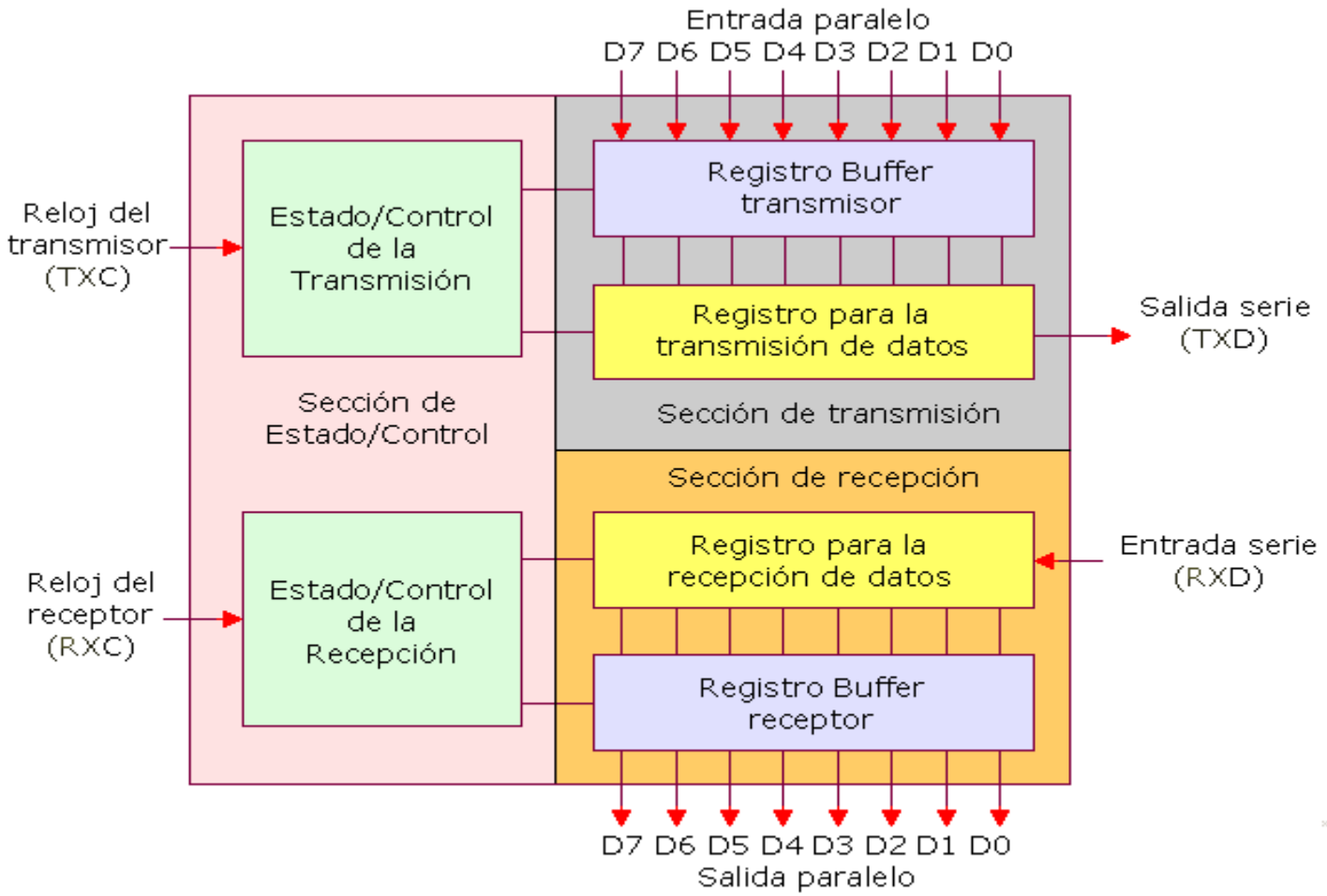
Comunicación Serie




Organización de la clase
Introducción Teórica
Cuestiones básicas de comunicación Serie
Uart
Módulo (Hardware del periférico)
Configuración
Aplicación
Conexión con otros equipos
Level Shifter
USB/UART Bridge
Bluetooth/UART

Introducción Teórica → Cuestiones básicas de comunicación Serie

LA idea básica de la comunicación serie es transmitir bits uno detrás de otro.



Arquitectura interna de una UART

Introducción Teórica → Cuestiones básicas de comunicación Serie

La comunicación Serie Puede ser

Síncrona

Existe además de la transmisión de datos o información, la transmisión de un tren de pulsos de clock encargados de sincronizar los bits enviados.
Es decir, el momento exacto en que un bit nuevo está disponible está coordinado por una señal eléctrica

Asíncrona

La llegada de cada nuevo bit no está sincronizada por una señal de clock, por lo que el Dispositivo transmisor y receptor deben ponerse de acuerdo en varios aspectos para que la comunicación pueda llevarse a cabo..

Introducción Teórica → Cuestiones básicas de comunicación Serie

La comunicación serie puede ser de las siguientes formas

SIMPLEX

Podemos recibir o enviar, solo una de las 2 opciones

HALF DUPLEX

Podemos enviar o recibir datos
Pero no simultáneamente

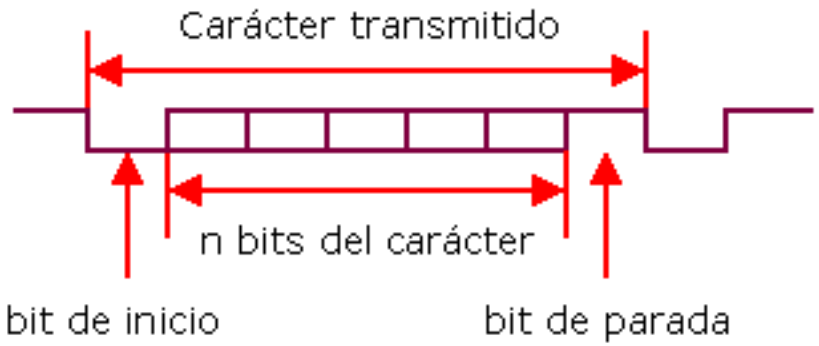
FULL DUPLEX

Podemos enviar o recibir datos
simultáneamente

Introducción Teórica → Cuestiones básicas de comunicación Serie

Trama serie

Al no existir un clock de sincronización, tanto el receptor como el transmisor deben conocer la velocidad de transmisión.
Además de la velocidad deben detectar el momento en que se produce el inicio de la comunicación. El fin de la misma y cuántos Bits se estás comunicando.



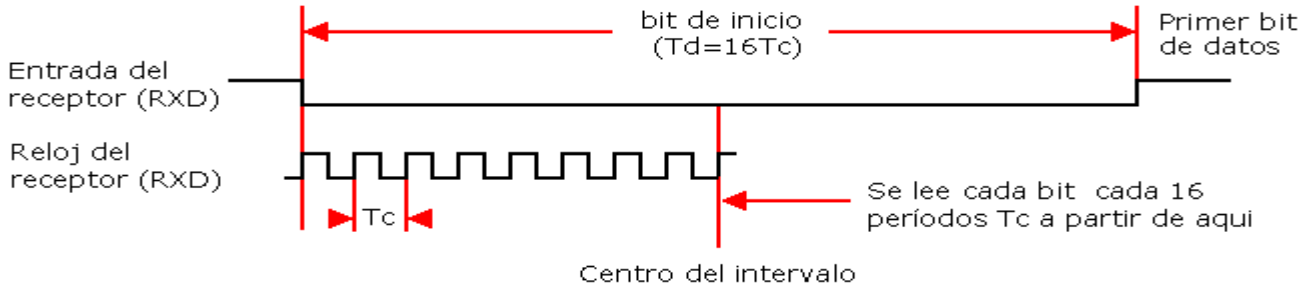
Formato básico de transmisión asincrónica

Como puede verse, en la trama de n bits se incluyen un bit de start y un bit de stop. La señal de stop pueden ser 1 o 2 bits.

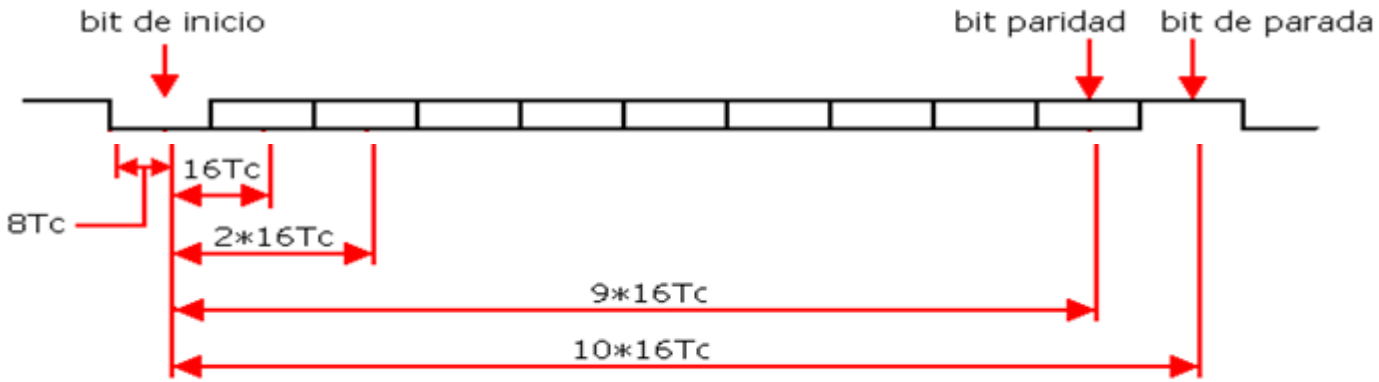
El bit de start y el de stop no intervienen en la interpretación del dato. La UART se encarga de agregarlos al momento de enviar el dato y de quitarlos al momento de recibirlo.

También puede estar presente un bit de paridad si lo configuramos para que exista.

La frecuencia d clock de la UART es 16 veces mayor que la velocidad de transmisión.
Por lo tanto, la UART se posicionará en medio de cada bit y contará 16 ciclos entre bit y bit para determinar si es "1" o "0"

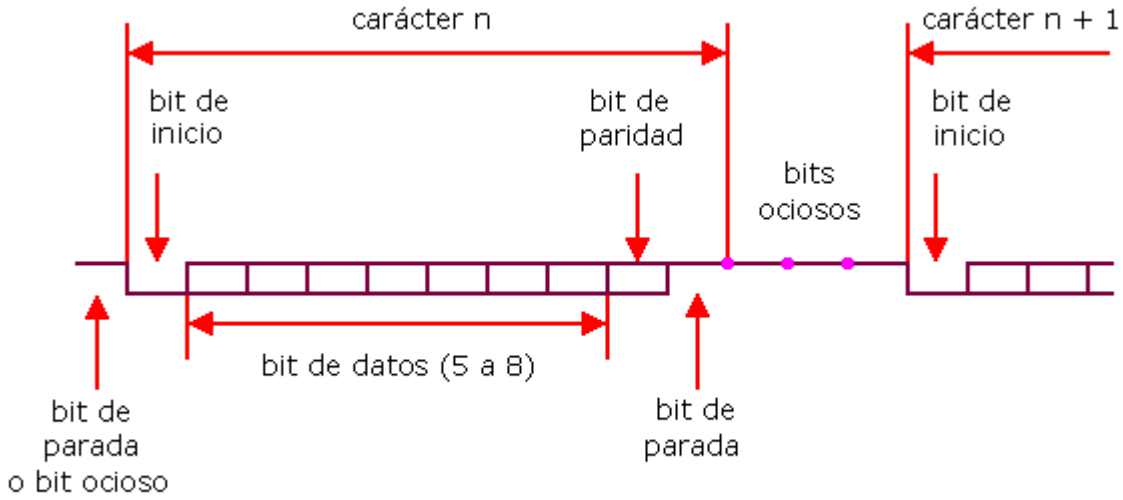


Deteccion del bit de inicio.



Comprobación de la informacion en la mitad del intervalo de cada bit

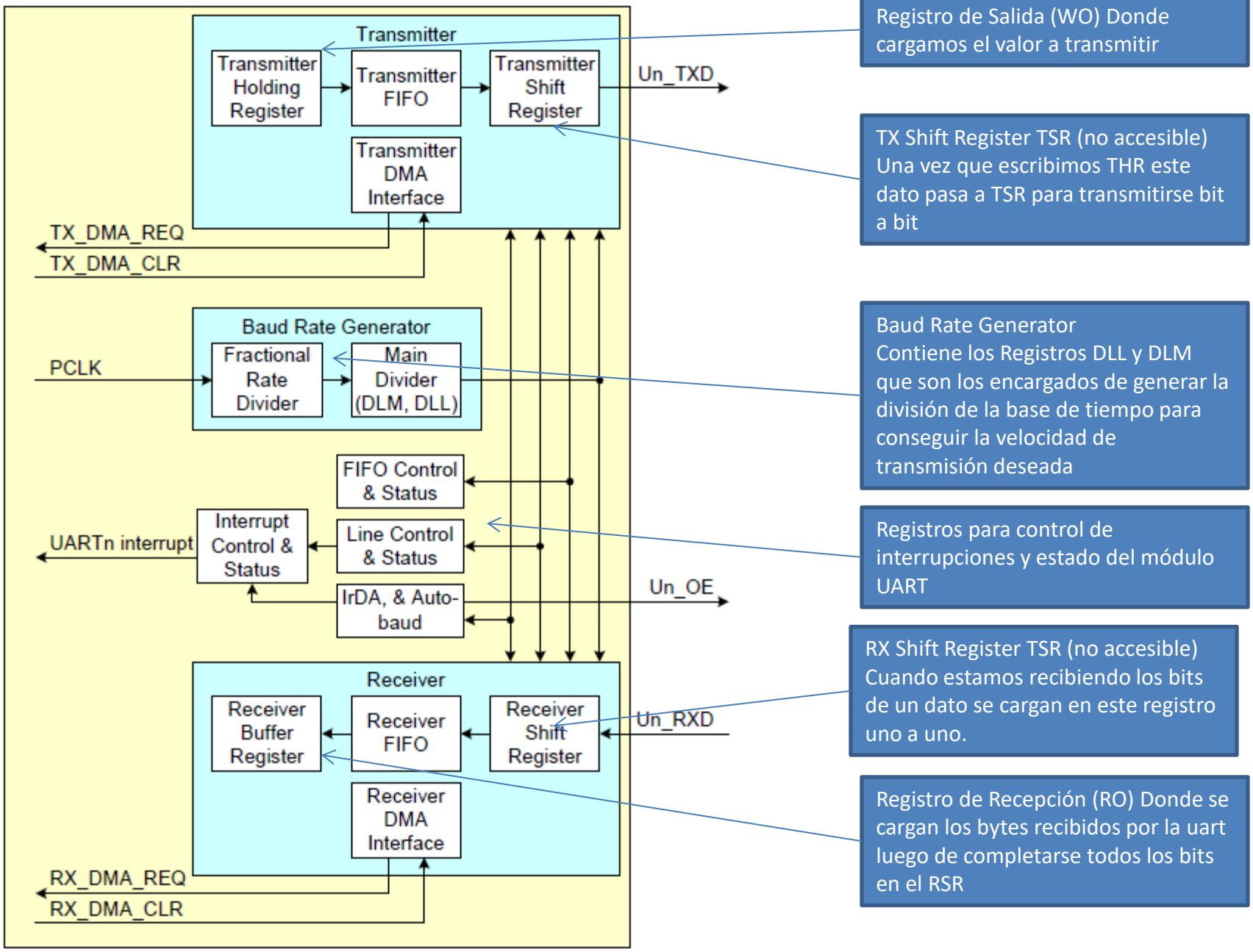
Además de los bits de datos, stop y start, contamos con un bit de paridad que puede ser útil en la detección de errores.



Transmisión asincrónica con velocidad menor que la máxima posible

El Bit de paridad nos indica si la cantidad de “1” contenidos en el carácter enviado es par o impar de acuerdo a la configuración que le demos.


Uart
Módulo (Hardware del periférico)



Registro de Salida (WO) Donde cargamos el valor a transmitir

TX Shift Register TSR (no accesible) Una vez que escribimos THR este dato pasa a TSR para transmitirse bit a bit

Baud Rate Generator
Contiene los Registros DLL y DLM que son los encargados de generar la división de la base de tiempo para conseguir la velocidad de transmisión deseada

Registros para control de interrupciones y estado del módulo UART

RX Shift Register TSR (no accesible) Cuando estamos recibiendo los bits de un dato se cargan en este registro uno a uno.

Registro de Recepción (RO) Donde se cargan los bytes recibidos por la uart luego de completarse todos los bits en el RSR

. UART0, 2 and 3 block diagram

Uart → Módulo (Hardware del periférico)

Registros asociados al módulo UART

Table 271. UART0/2/3 Register Map

Generic Name	Description	Access	Reset value[1]	UARTn Register Name & Address
RBR (DLAB =0)	Receiver Buffer Register. Contains the next received character to be read.	RO	NA	U0RBR - 0x4000 C000 U2RBR - 0x4009 8000 U3RBR - 0x4009 C000
THR (DLAB =0)	Transmit Holding Register. The next character to be transmitted is written here.	WO	NA	U0THR - 0x4000 C000 U2THR - 0x4009 8000 U3THR - 0x4009 C000
DLL (DLAB =1)	Divisor Latch LSB. Least significant byte of the baud rate divisor value. The full divisor is used to generate a baud rate from the fractional rate divider.	R/W	0x01	U0DLL - 0x4000 C000 U2DLL - 0x4009 8000 U3DLL - 0x4009 C000
DLM (DLAB =1)	Divisor Latch MSB. Most significant byte of the baud rate divisor value. The full divisor is used to generate a baud rate from the fractional rate divider.	R/W	0x00	U0DLM - 0x4000 C004 U2DLM - 0x4009 8004 U3DLM - 0x4009 C004
IER (DLAB =0)	Interrupt Enable Register. Contains individual interrupt enable bits for the 7 potential UART interrupts.	R/W	0x00	U0IER - 0x4000 C004 U2IER - 0x4009 8004 U3IER - 0x4009 C004
IIR	Interrupt ID Register. Identifies which interrupt(s) are pending.	RO	0x01	U0IIR - 0x4000 C008 U2IIR - 0x4009 8008 U3IIR - 0x4009 C008
FCR	FIFO Control Register. Controls UART FIFO usage and modes.	WO	0x00	U0FCR - 0x4000 C008 U2FCR - 0x4009 8008 U3FCR - 0x4009 C008
LCR	Line Control Register. Contains controls for frame formatting and break generation.	R/W	0x00	U0LCR - 0x4000 C00C U2LCR - 0x4009 800C U3LCR - 0x4009 C00C
LSR	Line Status Register. Contains flags for transmit and receive status, including line errors.	RO	0x60	U0LSR - 0x4000 C014 U2LSR - 0x4009 8014 U3LSR - 0x4009 C014
SCR	Scratch Pad Register. 8-bit temporary storage for software.	R/W	0x00	U0SCR - 0x4000 C01C U2SCR - 0x4009 801C U3SCR - 0x4009 C01C
ACR	Auto-baud Control Register. Contains controls for the auto-baud feature.	R/W	0x00	U0ACR - 0x4000 C020 U2ACR - 0x4009 8020 U3ACR - 0x4009 C020
ICR	IrDA Control Register. Enables and configures the IrDA mode.	R/W	0x00	U0ICR - 0x4000 C024 U2ICR - 0x4009 8024 U3ICR - 0x4009 C024
FDR	Fractional Divider Register. Generates a clock input for the baud rate divider.	R/W	0x10	U0FDR - 0x4000 C028 U2FDR - 0x4009 8028 U3FDR - 0x4009 C028
TER	Transmit Enable Register. Turns off UART transmitter for use with software flow control.	R/W	0x80	U0TER - 0x4000 C030 U2TER - 0x4009 8030 U3TER - 0x4009 C030

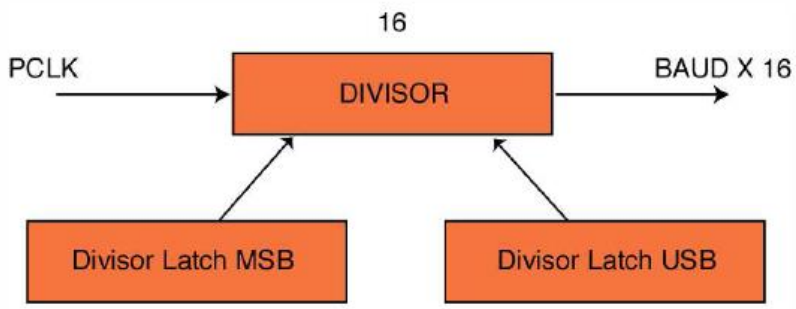
- Contiene el dato recibido
- Reg. Donde escribimos el dato que queremos transmitir
- Divisor Latch LSB
- Divisor Latch MSB
- Habilita las Fuentes de interrupción del módulo UART
- Interrupt ID Register
- Habilita la FIFO Por defesto desabilitada
- Controla La trama
- Estado de transmisión y recepción y estados de error
- Lo dejamos por defecto
- No usamos autobaud lo dejamos por defecto
- Por defecto
- No usamos un Baud Rate Fraccional
- Para desabilitar la transmisión para gestionar flujo de datos por software

Uart → Módulo (Hardware del periférico)

- Divisor Latch LSB
- Divisor Latch MSB
- Baud rate Fraccional

Para entender cómo funcionan estos registros tenemos la siguiente expresión

$$UART0_{baudrate} = \frac{PCLK}{16 \times (256 \times U0DLM + U0DLL) \times \left(1 + \frac{DivAddVal}{MulVal}\right)}$$



Cálculo de BAUDRATE
El Pclk es dividido por DLM y el DLL para conseguir la velocidad de transmisión Baudrate. A los registros DLL y DLM tenemos acceso poniendo a “1” el bit DLAB del registro LCR.
El 16 que puede verse en el denominador aparece por el hecho de que la UART debe tener un clock 16 veces más rápido que la velocidad de transmisión

Uart → Módulo (Hardware del periférico)

LCR Controla La trama

Table 280: UARTn Line Control Register (U0LCR - address 0x4000 C00C, U2LCR - 0x4009 800C, U3LCR - 0x4009 C00C) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	Word Length Select	00	5-bit character length	0
		01	6-bit character length	
		10	7-bit character length	
		11	8-bit character length	
2	Stop Bit Select	0	1 stop bit.	0
		1	2 stop bits (1.5 if UnLCR[1:0]=00).	
3	Parity Enable	0	Disable parity generation and checking.	0
		1	Enable parity generation and checking.	
5:4	Parity Select	00	Odd parity. Number of 1s in the transmitted character and the attached parity bit will be odd.	0
		01	Even Parity. Number of 1s in the transmitted character and the attached parity bit will be even.	
		10	Forced "1" stick parity.	
		11	Forced "0" stick parity.	
6	Break Control	0	Disable break transmission.	0
		1	Enable break transmission. Output pin UARTn TXD is forced to logic 0 when UnLCR[6] is active high.	
7	Divisor Latch Access Bit (DLAB)	0	Disable access to Divisor Latches.	0
		1	Enable access to Divisor Latches.	
31:8	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

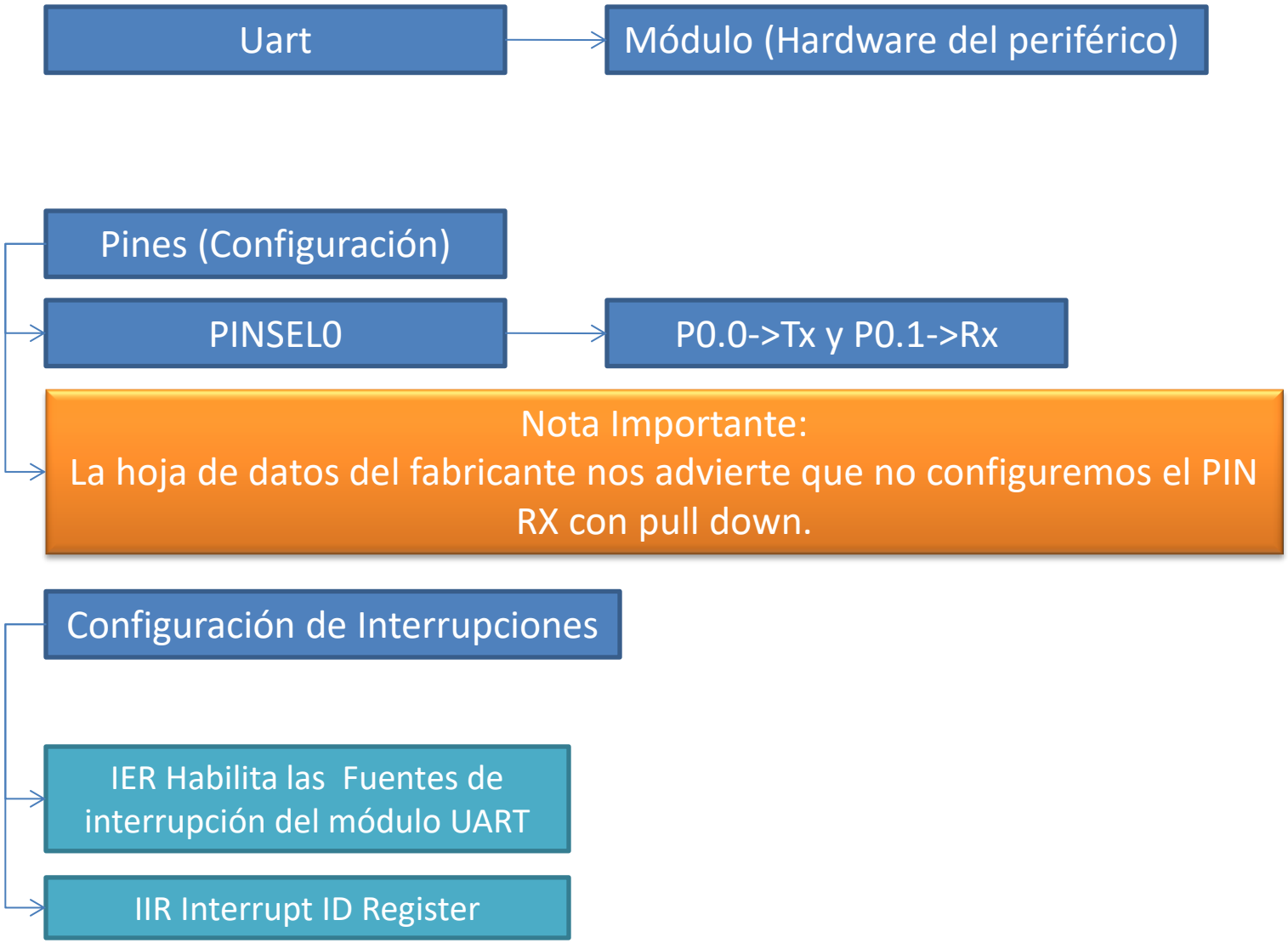
Uart → Módulo (Hardware del periférico)

LSR Estado de transmisión y recepción y estados de error

Bits
0- si hay un dato para leer
1- error de sobre-escritura
2- error de paridad
3- error de encuadre
5-indica si el THR está libre
6- indica si el THR y el TSR están libres

Table 281: UARTn Line Status Register (U0LSR - address 0x4000 C014, U2LSR - 0x4009 8014, U3LSR - 0x4009 C014) bit description

Bit	Symbol	Value	Description	Reset Value
0	Receiver Data Ready (RDR)		UnLSR0 is set when the UnRBR holds an unread character and is cleared when the UARTn RBR FIFO is empty.	0
		0	The UARTn receiver FIFO is empty.	
		1	The UARTn receiver FIFO is not empty.	
1	Overrun Error (OE)		The overrun error condition is set as soon as it occurs. An UnLSR read clears UnLSR1. UnLSR1 is set when UARTn RSR has a new character assembled and the UARTn RBR FIFO is full. In this case, the UARTn RBR FIFO will not be overwritten and the character in the UARTn RSR will be lost.	0
		0	Overrun error status is inactive.	
		1	Overrun error status is active.	
2	Parity Error (PE)		When the parity bit of a received character is in the wrong state, a parity error occurs. An UnLSR read clears UnLSR[2]. Time of parity error detection is dependent on UnFCR[0]. **Note:** A parity error is associated with the character at the top of the UARTn RBR FIFO.	0
		0	Parity error status is inactive.	
		1	Parity error status is active.	
3	Framing Error (FE)		When the stop bit of a received character is a logic 0, a framing error occurs. An UnLSR read clears UnLSR[3]. The time of the framing error detection is dependent on UnFCR0. Upon detection of a framing error, the Rx will attempt to resynchronize to the data and assume that the bad stop bit is actually an early start bit. However, it cannot be assumed that the next received byte will be correct even if there is no Framing Error. **Note:** A framing error is associated with the character at the top of the UARTn RBR FIFO.	0
		0	Framing error status is inactive.	
		1	Framing error status is active.	
4	Break Interrupt (BI)		When RXDn is held in the spacing state (all zeroes) for one full character transmission (start, data, parity, stop), a break interrupt occurs. Once the break condition has been detected, the receiver goes idle until RXDn goes to marking state (all ones). An UnLSR read clears this status bit. The time of break detection is dependent on UnFCR[0]. **Note:** The break interrupt is associated with the character at the top of the UARTn RBR FIFO.	0
		0	Break interrupt status is inactive.	
		1	Break interrupt status is active.	
5	Transmitter Holding Register Empty (THRE))		THRE is set immediately upon detection of an empty UARTn THR and is cleared on a UnTHR write.	1
		0	UnTHR contains valid data.	
		1	UnTHR is empty.	
6	Transmitter Empty (TEMT)		TEMT is set when both UnTHR and UnTSR are empty; TEMT is cleared when either the UnTSR or the UnTHR contain valid data.	1
		0	UnTHR and/or the UnTSR contains valid data.	
		1	UnTHR and the UnTSR are empty.	


Uart
Módulo (Hardware del periférico)
Pines (Configuración)
PINSEL0
P0.0->Tx y P0.1->Rx
Nota Importante:
La hoja de datos del fabricante nos advierte que no configuremos el PIN RX con pull down.
Configuración de Interrupciones
IER Habilita las Fuentes de interrupción del módulo UART
IIR Interrupt ID Register

Configuración → Aplicación

```
52 //----------------------------------Configuracion del módulo UART-------------------------------------//
53
54
55 LPC_SC->PCONP|=1<<25;   //Encendemos el periférico
56
57 /*en este punto deberíamos asignar la frecuencia de clock del modulo UART3
58 pero debido a que por defecto está en PCLK_UART3=Cclock/4 lo dejaremos así de lo contrario deberíamos
59 ingresar al PCLKSEL1 en el que tenemos los dos bits necesarios para seleccionar
60 la frecuencia de Pclk para el módulo UART3 */
61
62 //-----a continuación configuraremos el formato de la trama de datos de la transmisión UART3LCR-----//
63
64 LPC_UART3->LCR|=11;
65 /*usaremos 1 bit de stop, pero como por defecto se tiene 1 bit de stop no es necesario configurar
66 parámetro*/
67
68 /*Configuramos el módulo uart para que no tenga paridad pero como por defecto está desabilitado no tocamos
69 este bit*/
70
71 /*También por defecto dejamos desabilitado el Break Control, por defecto está desabilitado*/
72
73
74 /*ahora colocamos en un 1 en al bit 7 del LCR, el bit Divisor Latch Access Bit para acceder a DLL y DLM*/
75 LPC_UART3->LCR|=(1<<7);
76
77                                          //-----Trama de Datos-----//
78
79 /*A continuación configuramos el Baud rate, es decir, la tasa de transmisión en baudios por segundo*/
80 /*es decir DLL y DLM*/
81
82
83 /*Configuraremos DLL y DLM para obtener un Baud Rate de 9600 baudios
84  * para esto tenemos */
```

Configuración → Aplicación

```
/*Configuraremos DLL y DLM para obtener un Baud Rate de 9600 baudios
 * para esto tenemos */

//
//                              Pclk
//Baudrate   =    ---------------------------------
//                      16 X (256*DLM+DLL)
//
//no haremos caso al divisor de frecuencia fraccional dado que el error que cometemos no es significativo
//
//
LPC_UART3->DLL=0b10100001; //DLL = 161 -> 9585 baudios -> error de 14 -> 0.1458%
LPC_UART3->DLM=0;

LPC_UART3->LCR &= 0b01111111; /*Colocamos el Bit DLAB nuevamente a cero para que
esto nos impide el acceso a DLL y DLM pero nos permite acceder a RBR, THR e IER*/

/*en IER seleccionamos la fuente de interrupción del módulo UART3 que queremos que dispare la interrupción */
LPC_UART3->IER=1; // Corresponde a la interrupción por Receive Data Available

```

Configuración → Aplicación

Configuración NVIC

```
107
108 //Configuramos NVIC
109
110 NVIC_EnableIRQ(UART3_IRQn);
111
112 //Configuramos los pines de RX y TX
113
114 LPC_PINCON->PINSEL0|=0b1010; //configuramos los pines P0.0 como TX y P0.1 como RX
115
116     while(1)
117     {
118
119
120     }
121     return 0 ;
122 }
123
```

Configuración Systick_Timer

```
48 int main(void)
49 {
50
51 SysTick_Config(1000000); //Configuramos el SysTick Timer para que interrumpa periódicamente
52 //-----------------------------------Configuracion del módulo UART-------------------------------------//
53
```

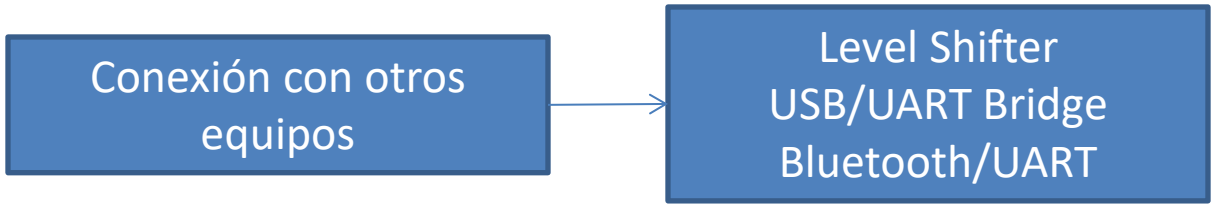
Configuración → Aplicación

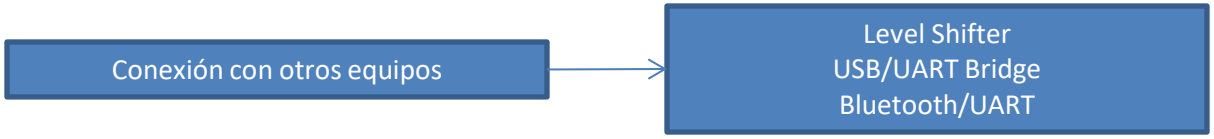
Definición de los handler

```
27 void SysTick_Handler(void)
28 {
29     i--;
30     if(i==0)
31     {
32         i=100;
33         j++;
34         enviar(j);
35     }
36 }
37 /*interrupción que se genera por el módulo UART3
38  * la única fuente de interrupción habilitada en el módulo UART3 es la de Receive data available*/
39
40 void UART3_IRQHandler(void)
41 {
42     k=LPC_UART3->RBR;
43     enviar(k);
44 }
45
46
```

Definición de variables y funciones

```
15 #include <cr_section_macros.h>
16 volatile static int i = 100;
17 uint8_t k = 0;
18 uint8_t j = 0;
19 uint32_t IIR_temp;
20
21 void enviar (uint8_t c)
22 {
23     while((LPC_UART3->LSR&(1<<5))==0){}
24     LPC_UART3->THR=c;
25 }
26
```


Conexión con otros equipos
Level Shifter
USB/UART Bridge
Bluetooth/UART



Para entrar en esta parte del tema tenemos que tener en cuenta que existen varios Protocolos, cada uno con sus especificaciones, especificaciones:

- Trama
- Velocidades máximas
- Alcance en metros
- Niveles de tensión
- etc

Consideraremos a continuación los diferentes protocolos, características principales y hardware para estar dentro de las especificaciones de esa norma

Conexión con otros equipos → Level Shifter
USB/UART Bridge
Bluetooth/UART

La norma RS232 determina las tensiones correspondientes a los niveles lógicos “0” y “1” además de las distancias máximas a las que pueden transmitirse datos.

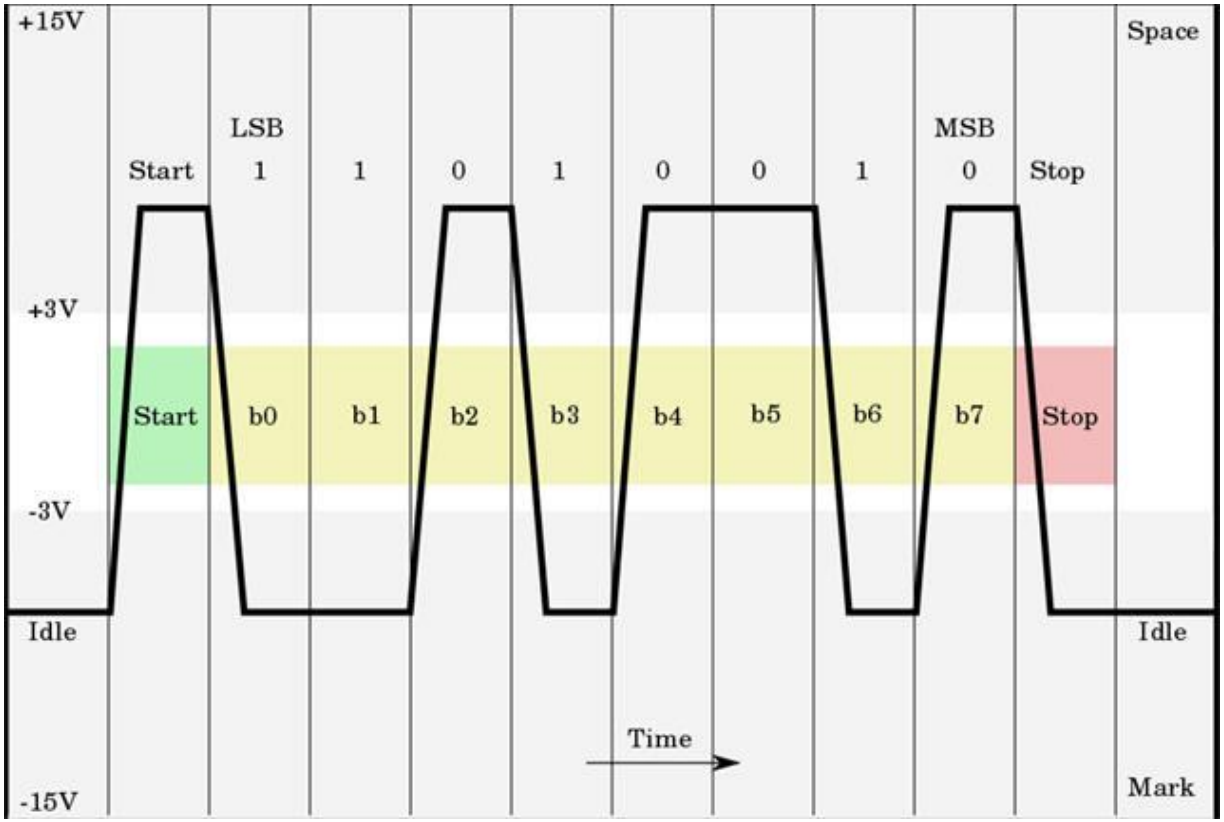
Se establece que la longitud máxima del cable no debe ser superior a los 15 metros y la velocidad máxima de transmisión es, en principio, 128.000 bps. Comunicación Full Duplex. Los niveles lógicos no son compatibles TTL, considerando:

1 lógico entre -3V y -15V
0 lógico entre +3V y +15V

La idea de usar estos niveles es:
•Aumentar el nivel de tensión para conseguir mayores distancias
•Usar como “1” y “0” valores de tensión diferentes de 0V para detectar posibles cortes en la línea.

Velocidades Estandarizadas según RS232

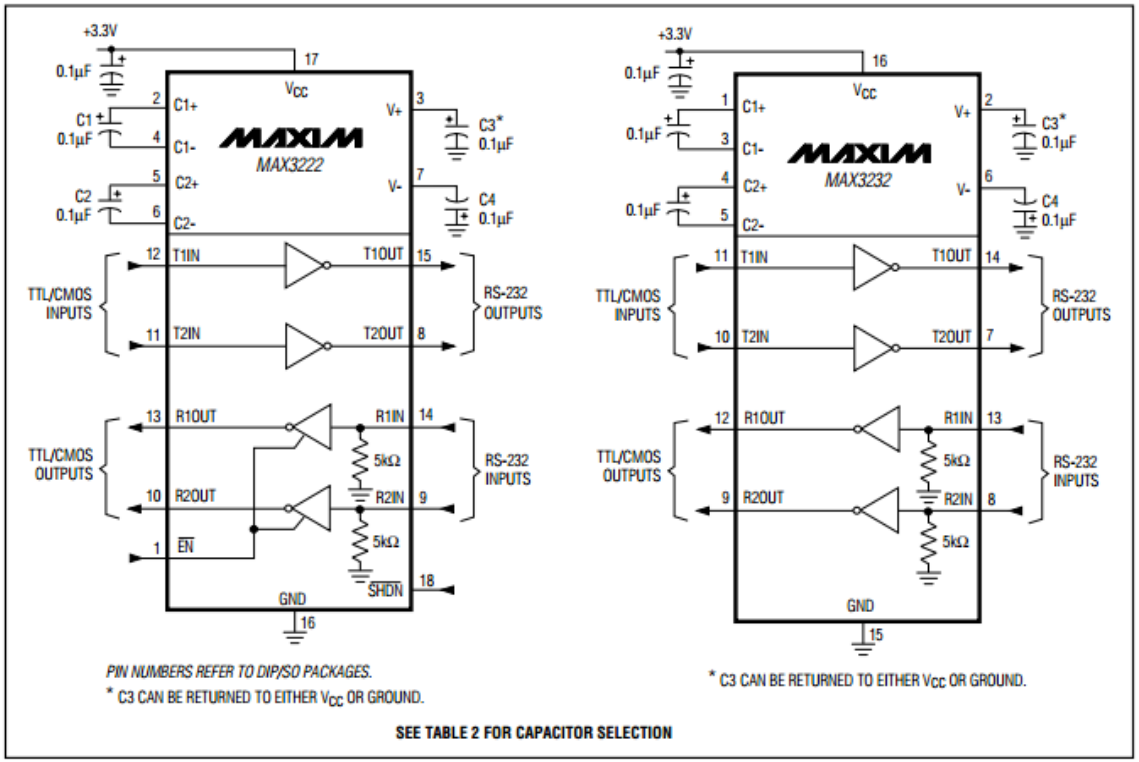
•75
•110
•150
•300
•600
•1200
•2400
•4800
•9600
•19200
•Fuera de la norma:
•38400
•57600
•76800
•115200



Conexión con otros equipos → Level Shifter, USB/UART Bridge, Bluetooth/UART

Como pudimos ver RS232 no es Compatible con TTL por lo tanto necesitamos utilizar un Level Shifter en nuestro caso consideraremos el MAX3232 que es una variante del MAX232 que puede alimentarse con 3.3V a continuación el esquema de conexionado. También podría ser un SP3232, etc.

Typical Operating Circuits



PIN NUMBERS REFER TO DIP/SO PACKAGES.

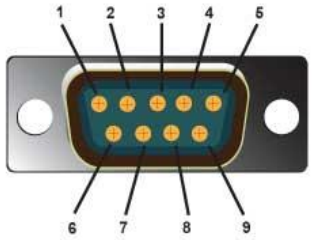
* C3 CAN BE RETURNED TO EITHER V_{CC} OR GROUND.

SEE TABLE 2 FOR CAPACITOR SELECTION

* C3 CAN BE RETURNED TO EITHER V_{CC} OR GROUND.

Una vez que contamos con los niveles de tensión apropiados solo basta agregar un conector DB9 y con eso podemos conectarnos a cualquier dispositivo que cuente con una conexión RS232C

DB-9 Connector Pin Outs



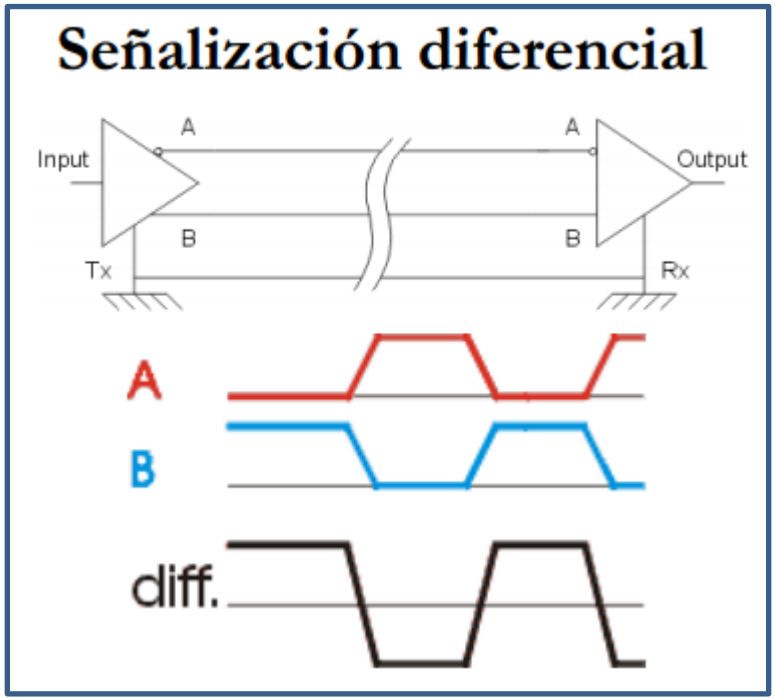
Pin #	Signal	Direction	Description
1	CD	in → computer	Carrier Detect
2	RXD	in → computer	Receive Data
3	TXD	out ← computer	Transmit Data
4	DTR	out ← computer	Data Terminal Ready
5	GND	-	Ground
6	DSR	in → computer	Data Set Ready
7	RTS	out ← computer	Request To Send
8	CTS	in → computer	Clear To Send
9	RI	in → computer	Ring Indicator (Cordless Serial Adapter: optional power input 3.3V-5.0V)

De todos estos pines solo utilizamos Receive Data, Transmit Data y GND (muy importante)
Tener en cuenta cuando se trate de conector DB9 Macho o hembra, dado que los pines Tx y Rx están cruzados

Conexión con otros equipos → Level Shifter, USB/UART Bridge, Bluetooth/UART

RS485

Niveles de tensión:
+-7V - +12V
Velocidad/Distancia máxima:
10Mbps hasta 10m, 100kbps hasta 1200m
Tipo de señal: Diferencial (balanceada) sobre un par trenzado.
Tipo de comunicación:
Half Duplex. Aunque existen versiones de 2 pares trenzados para conseguir comunicaciones Full Duplex
Puede usarse un MAX485 para adaptar las señales.



La señal se transmite en forma directa (A) e invertida (B).
El ruido aditivo ataca tanto a la señal A como a la B.
Al llegar ambas al final de la línea se restan y el ruido aditivo se elimina a sí mismo.

Supongamos S(t): señal, N(t): Ruido
Lo que hacemos es, enviamos A=S(t) y B=-S(t).
Al final de la línea A y B se verán ambas afectadas por el ruido. O sea que al final de la línea tenemos
A=S(t)+N(t) y B=-S(t)+N(t)
El circuito receptor se encarga de hacer la diferencia entre ambas señales
Con lo que la señal recibida será:
A-B=S(t)+N(t)-(-S(t)+N(t))=2S(t)
Y el ruido se elimina a sí mismo.

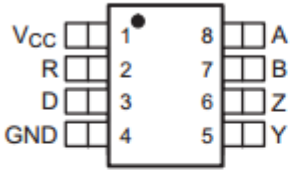
Conexión con otros equipos → Level Shifter, USB/UART Bridge, Bluetooth/UART

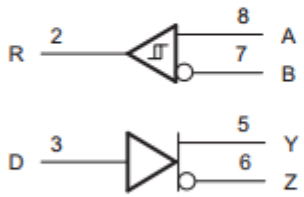
RS485

Diagrama explicativo de un Driver para comunicación RS485 de texas instruments

SN65HVD30, SN65HVD31, SN65HVD32

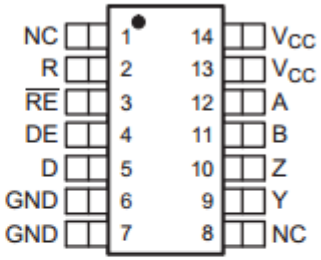
D PACKAGE
(TOP VIEW)



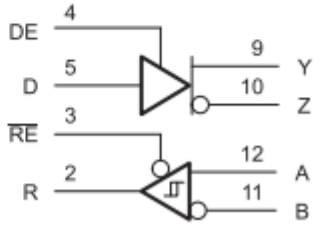


SN65HVD33, SN65HVD34, SN65HVD35

D PACKAGE
(TOP VIEW)



NC - No internal connection



Mediante este tipo de comunicación tenemos la posibilidad de conectar más de 2 dispositivos, con la capacidad de enviar una dirección y luego de detectado el dispositivo enviar los datos

Conexión con otros equipos → Level Shifter
USB/UART Bridge
Bluetooth/UART

USB-Uart Bridge

Existe la posibilidad de conectar el módulo Uart a un puente USB/UART de modo que tenemos un chip dedicado que se encarga de convertir los datos de la trama UART(0V-5V) a protocolo USB, sin embargo, el fabricante de estos chips nos proporciona drivers, aplicaciones que reciben los datos en la PC y los muestran como si se tratara de un puerto serie(COM), no un usb.
De modo que tenemos la versatilidad o comodidad del USB con la simplicidad de configurar una UART

El más popular es el FT232 de la empresa FTDI.
Es popular debido a que es el chip que se encuentra en las placas arduino y en particular en las placas LPCXPRESSOBaseBoard. Por lo que para utilizar el USB UART bridge de la placa LPCXPRESSO baseboard debemos descargarnos el driver de este chip

MCP2200 es otro adaptador USB/UART pero del fabricante MICROCHIP. Por supuesto contará con su driver para la PC

USB-RS232 Bridge

Si Combinamos un Max3232 y un USB-UART Bridge tenemos un adaptador USB-RS232
Eso se compra en forma de cable (Manhattan USB-RS232) o en forma de placa adaptadora (MCP2200 USB to RS232 Board)
De esta manera podemos conectar dispositivos que cuentan con conexión RS232 con dispositivos que cuentan con USB host

Conexión con otros equipos → Level Shifter
USB/UART Bridge
Bluetooth/UART

La placa LPCXPRESSO Baseboard cuenta con un puente USB/UART con el cual nos podemos conectar entre el uart y el USB de una PC.

- Para esto debemos descargar el driver para el Puente FT232 para que la PC reconozca el puente.
-Debemos configurar los Jumpers como indica la Documentación de la placa LPCXPRESSO baseboard
-Luego tener la placa LPCXPRESSO con un programa que use el puerto UART.
-Configurar La velocidad en Baudios de la PC Host.

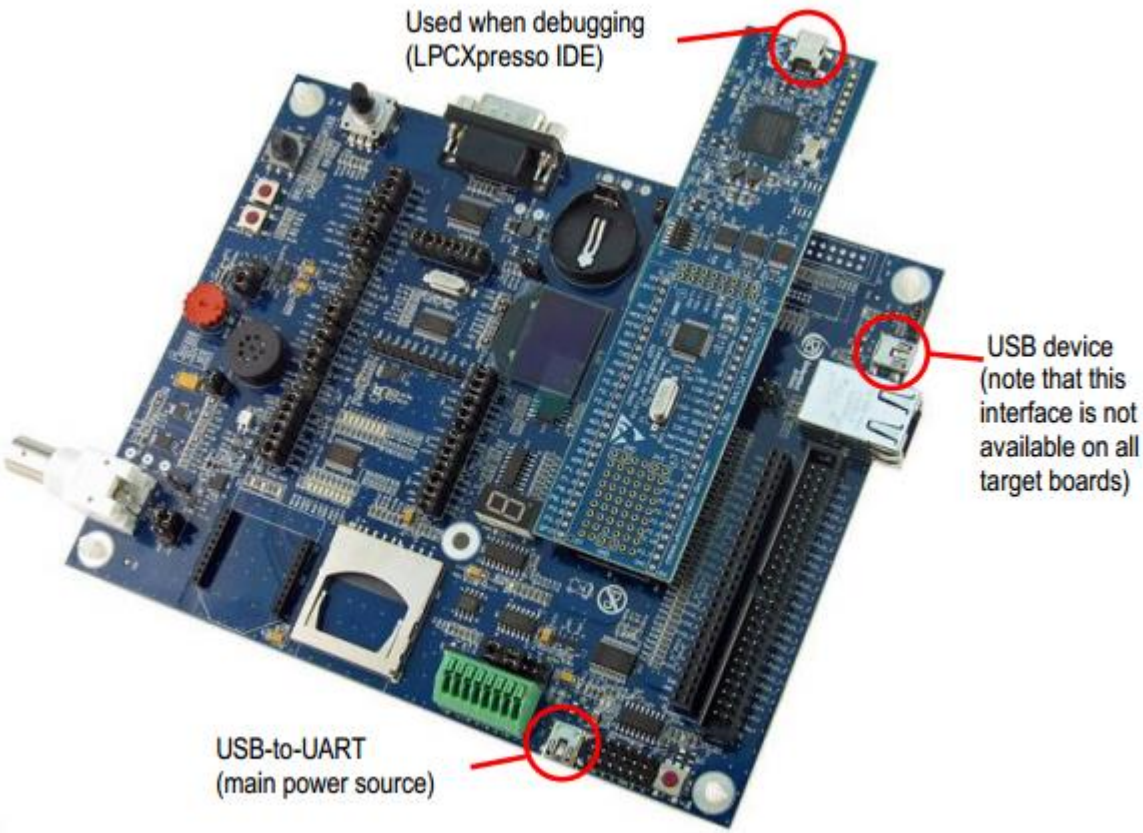
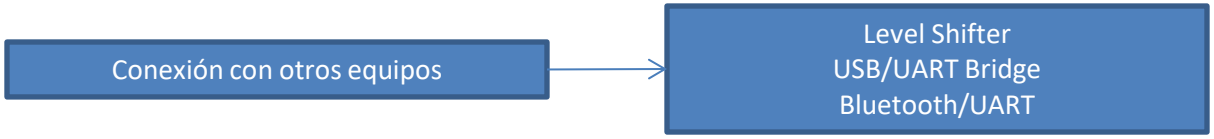


Figure 3 – Base Board with a mounted LPCXpresso LPC1343 board

Conexión con otros equipos → Level Shifter
USB/UART Bridge
Bluetooth/UART

Otra posibilidad de conexión es la de un módulo bluetooth el cual se accede mediante un puerto UART a nivel TTL y nos permite, conectarnos con dispositivos que tengan este tipo de conectividad.
Un ejemplo es el HC06



Lo importante de estos últimos items es tener en cuenta que la configuración del Periférico en el microcontrolador puede ser la misma pero el hardware externo al microcontrolador variará dependiendo del protocolo que estemos teniendo en cuenta