Diseño Basado en Microprocesadores

Tema 2. Microcontroladores

- 2.1. Introducción a los microcontroladores
- 2.2. Entradas/Salidas Digitales
- 2.3. Temporizadores
- 2.4. Excepciones
- 2.5. Conversión Analógica/Digital
- 2.6. Comunicación serie RS232C
- 2.7. Teclado, conversión D/A y sonido
- 2.8. Interfaz I2C

2.5. Comunicación serie RS232C

UM10562 Cap. 18: UART0/2/3

- 2.5.1. Tipos de comunicación
- 2.5.2. Protocolos de comunicación
- 2.5.3. Interfaz RS232C
- 2.5.4. Formato y parámetros de comunicación
- 2.5.5. La UART
- 2.5.6. Las UARTs del LPC4088
- 2.5.7. Arquitectura de las UARTs 0/2/3
- 2.5.8. Registros
- 2.5.9. Pasos para la programación
- 2.5.10. Ejemplo

2.5.1. Tipos de comunicación

Serie:

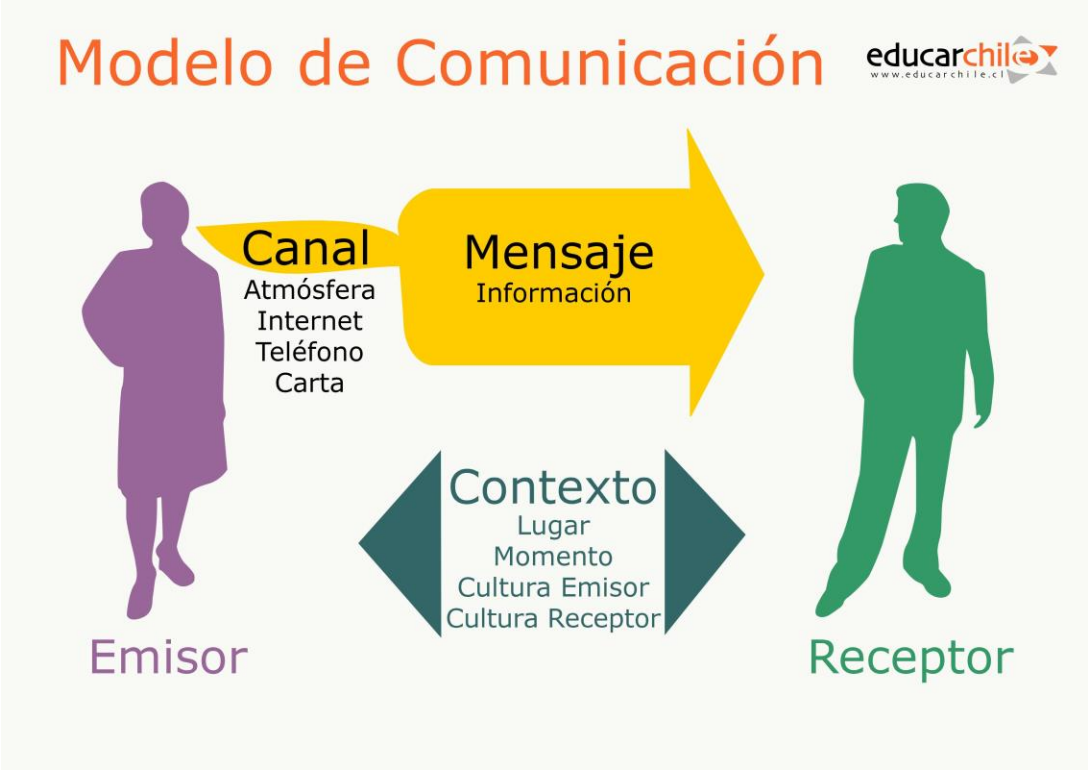
- RS232C, RS485, RS422, USB, SPI, I2C.
- Comunicación, económica y simple
- Grandes distancias (modem: cable telefónico)
- Comunicación lenta

Paralelo:

- GPIB, VME-VXI, ……

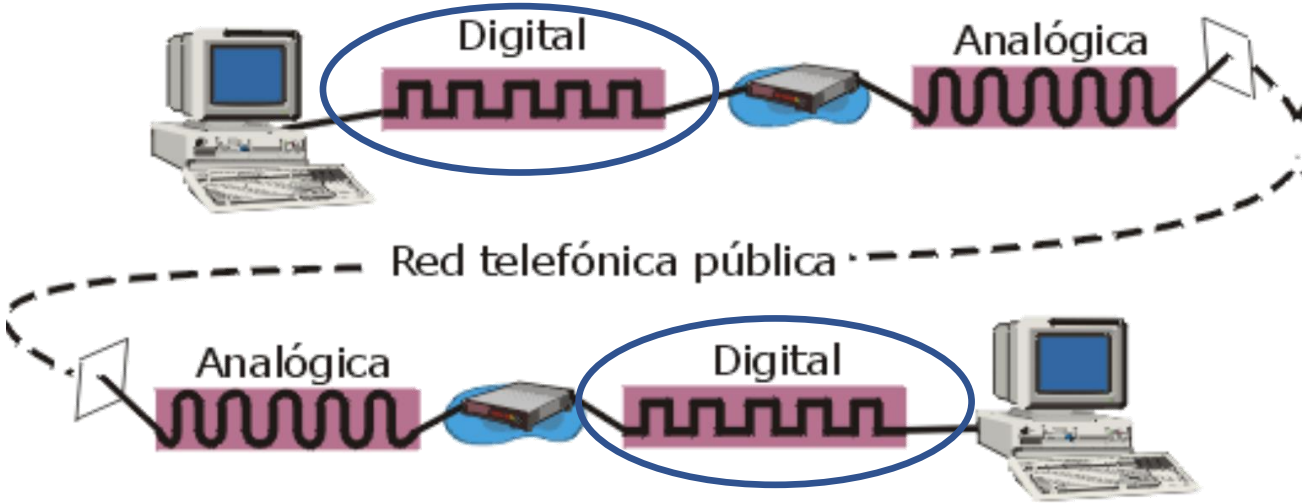
2.5.2. Protocolos de comunicación

- Software de control de:
  - Flujo de datos
  - Exactitud de la transferencia
- Conjunto de reglas entre emisor y receptor para el intercambio de datos
- Control del Hardware, parámetros de comunicación



2.5.3. Interfaz RS232C (1)

- Desarrollado por la EIA (Electronics Industries Association) en los años 60
- Título de la norma: Interfaz Between Data Terminal Equipment (DTE) and Data Circuit-Terminating Equipment (DCE) Employing Serial Binary Data Interchange
- El objetivo inicial fue estandarizar la comunicación entre teletipos y módems y posteriormente entre terminales y módems (RS-232C)



Comunicación a través de módems

2.5.3. Interfaz RS232C (2)

- Muchos fabricantes adaptaron la norma a dispositivos diferentes de terminales y módems haciendo una interpretación propia del estándar lo que generó:
  - Asignación arbitraria de señales en los conectores
  - Problemas para interconectar esos equipos
  - Mercado de cajas adaptadoras, instrumentos de test, cables,..

2.5.3. Interfaz RS232C (3)

DTE (Equipo Terminal de Datos):

- Dispositivo que convierte la información introducida por el usuario en señales para su transmisión y reconvierte las señales recibidas en información que presenta al usuario
- Es un dispositivo que actúa como fuente y/o destino de información.
- Ejemplos: terminal, impresora

DCE (Equipo de Comunicación de Datos):

- Dispositivo que convierte y/o codifica las señales del DTE para poder transmitirlas y recibirlas a través de un determinado canal de comunicación
- Ejemplo: modem

2.5.3. Interfaz RS232C (4)

DB-25

- La norma establece el uso de conectores de 25 patillas, pero no indica un tipo concreto
- Los conectores DB-25 de 25 patillas han sido los usados habitualmente

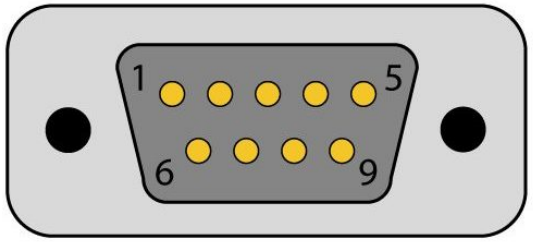
DB-9

- Tienen solo las 9 señales imprescindibles

Conector

- El DTE debe llevar un conector macho
- El DCE debe llevar un conector hembra

DB9M Connector



RS232 Pin Out

Pin #	Signal
1	DCD
2	RX
3	TX
4	DTR
5	GND
6	DSR
7	RTS
8	CTS
9	RI

2.5.3. Interfaz RS232C (5)

TxD (Transmit Data)

- Salida de datos en serie del DTE
- Entrada de datos en serie en el DCE

RxD (Receive Data)

- Entrada de datos en serie del DTE
- Salida de datos en serie en el DCE

DTR (Data Terminal Ready)

- Salida en el DTE (entrada en el DCE) que indica al DCE que el DTE está activo

DSR (Data Set Ready)

- Salida en el DCE (entrada en el DTE) que indica al DTE que el DCE está activo

RTS (Request To Send)

- Salida en el DTE (entrada en el DCE) que se activa para **solicitar al DCE permiso para transmitirle datos**

CTS (Clear To Send)

- Salida en el DCE (entrada en el DTE) para **indicarle al DTE que está preparado para recibir datos**

DCD (Data Carrier Detect)

- Salida en el DCE (entrada en el DTE) con la que el DCE **indica que está recibiendo una señal portadora de datos correcta del DCE remoto**

RI (Ring Indicator)

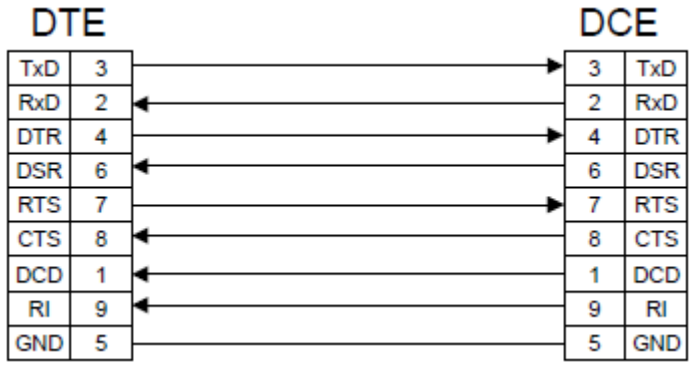
- Salida en el DCE (entrada en el DTE) con la que el DCE de tipo modem telefónico **indica una llamada entrante en la línea telefónica a la que está conectado**

GND (Ground)

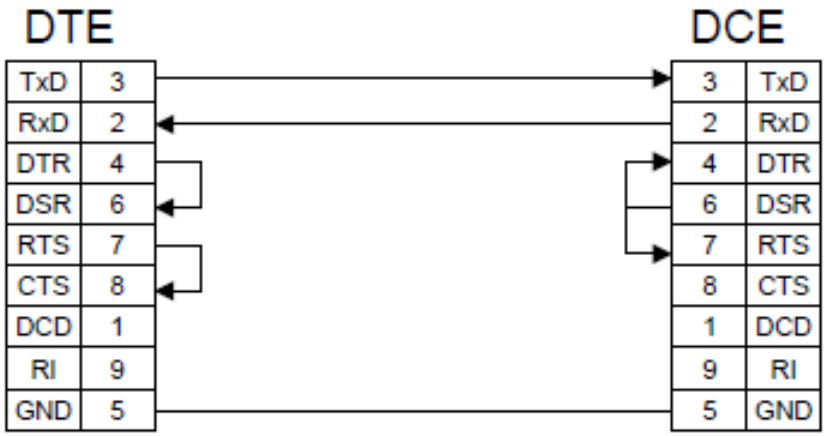
- Conexión de tierra o masa común

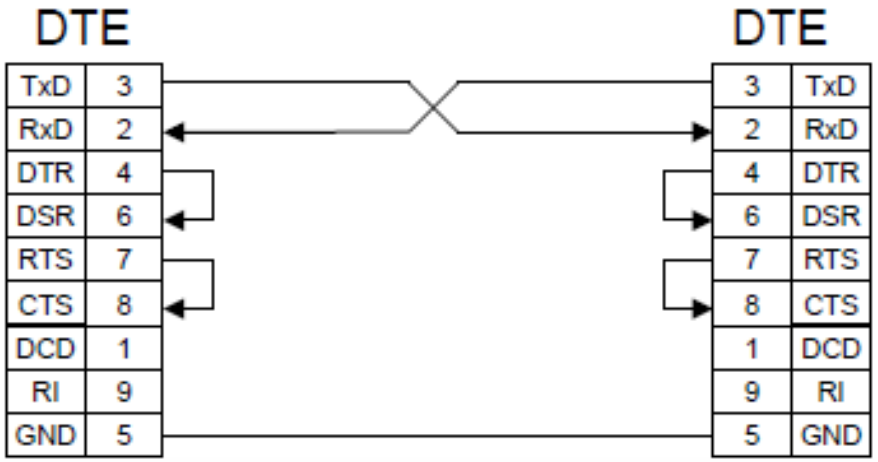
2.5.3. Interfaz RS232C (6)

Cable DTE-DCE completo (DB-9)



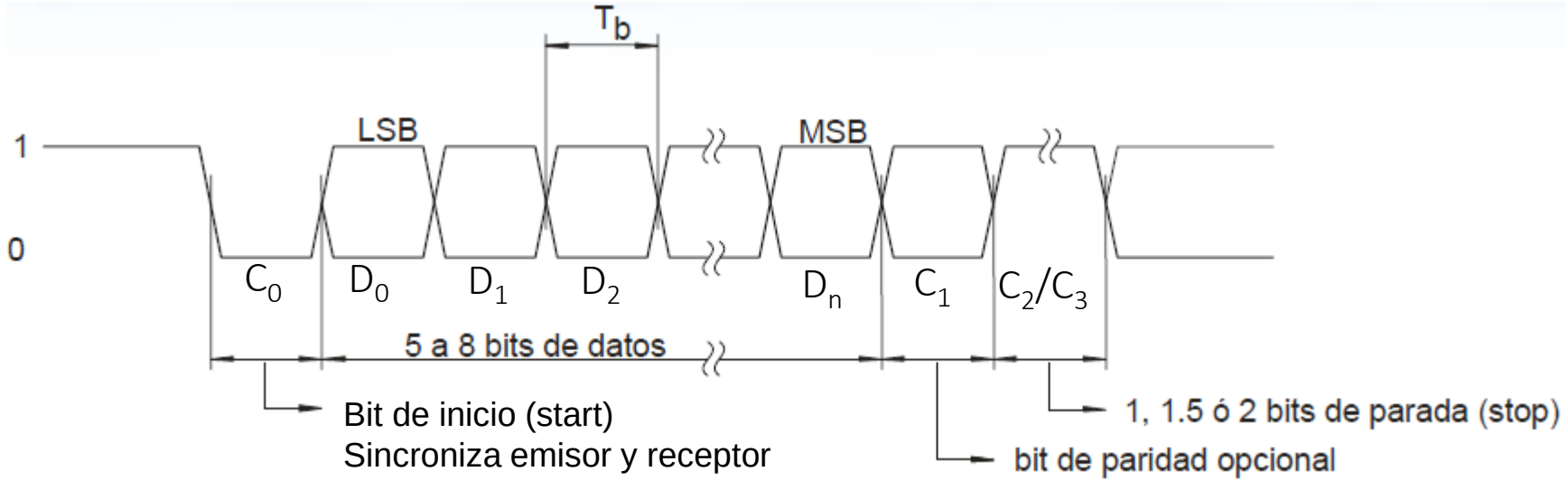
Cable DTE-DCE de solo 3 hilos "engañando" a las señales de protocolo (DB-9)





Cable "modem nulo" (null modem) de 3 hilos para conectar entre sí dos DTEs, por ejemplo, dos PCs (DB-9)

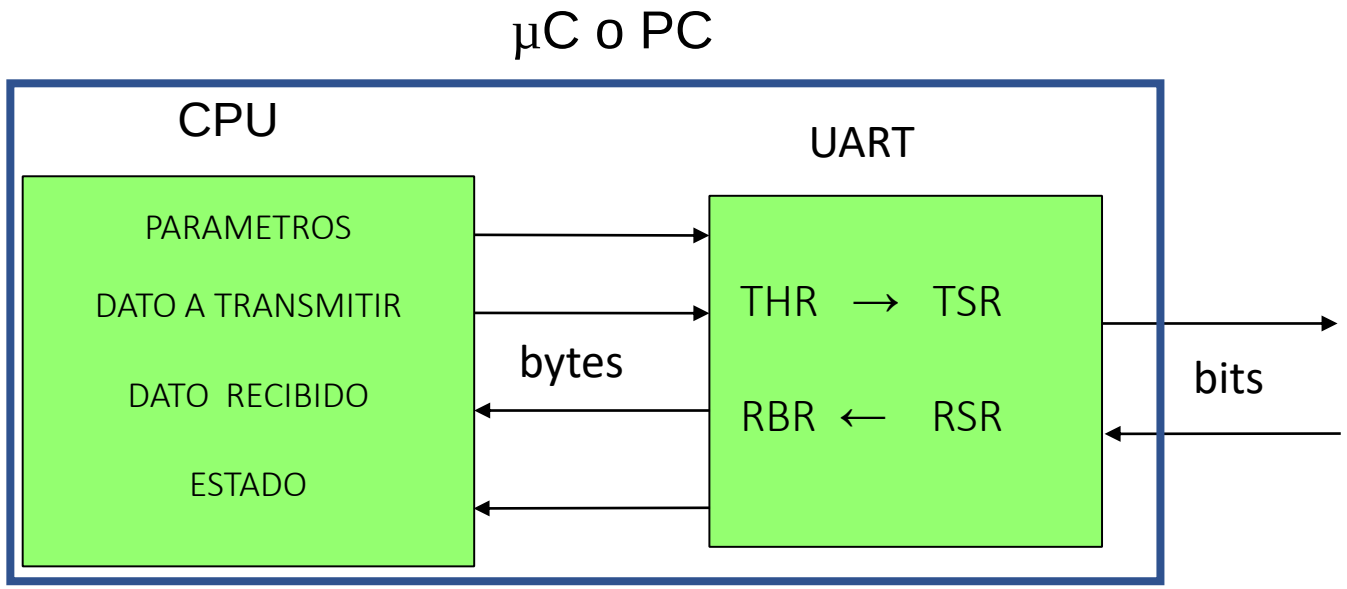
2.5.4. Formato y parámetros de la comunicación RS232C



- Se muestran niveles lógicos, no niveles eléctricos
- Si se usa paridad puede ser par o impar
- El inverso del periodo de bit, Tb, se denomina velocidad de comunicación y se mide en baudios (bits/s)
- Velocidades estándar: 75, 110, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600 y 115200 baudios
- Los equipos conectados deben estar configurados con los mismos parámetros: el mismo número de bits de datos, bits de stop, paridad (si se usa) y velocidad en baudios

2.5.5. UART (Universal Asynchronous Receiver-Transmitter) (1)

- Dispositivo que transforma datos en el formato paralelo manejado por la CPU hacia/desde un formato serie, típicamente RS232
- Se fabrican como circuitos integrados independientes o se integran en los chipsets de las placas base de ordenador o en el interior de los µc
- Añade/retira automáticamente los bits de start, stop y paridad
- Comprueba errores de recepción tales como bits de start, stop o paridad incorrectos



2.5.5. UART (2)

- Incluye un generador de reloj programable para obtener la velocidad de comunicación en baudios deseada
- Permite generar y comprobar las señales de protocolo DTR, DSR, RTS, CTS, DCD y RI
- La CPU interactúa con la UART mediante un conjunto de registros internos que ésta posee.
- Modelos de UART estándar de la industria:
  - 8250: diseño original de National Semiconductor en los ’80 y fabricado por muchos otros
  - 16C550: versión mejorada del 8250 con buffers FIFO de transmisión y recepción

2.5.5. UART (3)

Comunicación a corta distancia usando niveles CMOS

- Hay dispositivos que usan el formato trama de datos RS-232 pero emplean niveles lógicos CMOS (5 V ó 3.3 V tipicamente).
- Pensados para conectarse distancias cortas (misma PCB que el microcontrolador o cables cortos).
- La conexión puede ser directa, sin intercalar transceivers.
- Ejemplos: pequeños módulos inalámbricos, GPS y RFID.

Convertidores USB-RS232

- Permite usar periféricos RS-232 en ordenadores que no tienen Interfaz RS-232 pero si USB.

2.5.6. Las UARTs del LPC4088

- 5 UARTs: UART0 a UART4
- En el reset las UARTs 0 y 1 están habilitadas, las UARTs 2, 3 y 4 no
- Se usan buffers FIFO de 16 bytes para la transmisión y recepción de los bytes que se envían o reciben por la UART
- Dispone de un generador de baudios
- DMA
- Modo IrDA para comunicación por infrarrojos
- UART1: permite funcionar como modem
- UART4: modo Smart Card para comunicación con tarjetas inteligentes
- Estudiaremos las UART0/2/3 por ser las más sencillas

2.5.7. Arquitectura de las UARTs0/2/3 (1)

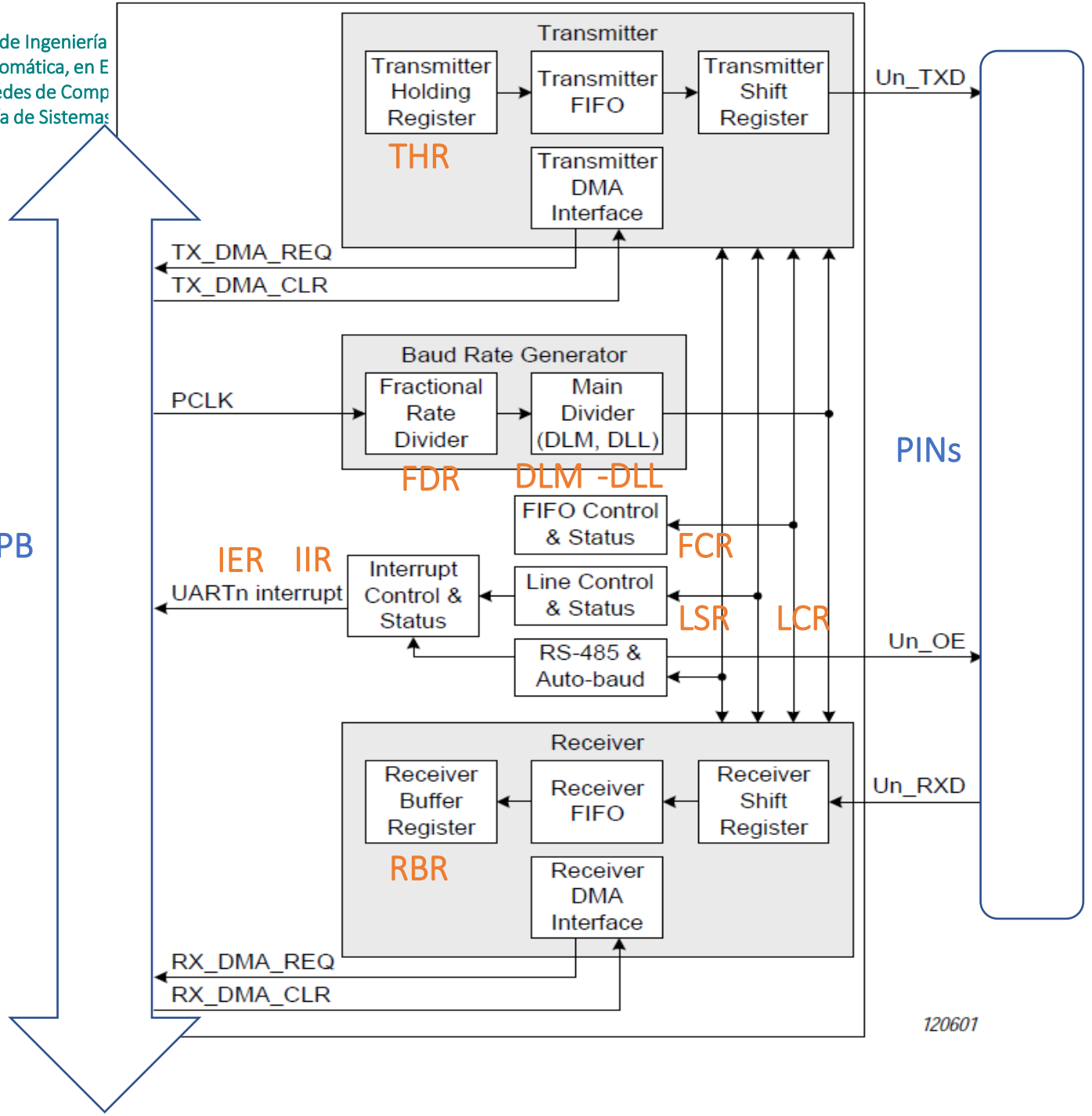
Table 391: UARTn Pin description

Pin	Type	Description
U0_RXD, U2_RXD, U3_RXD	Input	**Serial Input**. Serial receive data.
U0_TXD, U2_TXD, U3_TXD	Output	**Serial Output**. Serial transmit data.
U0_OE, U2_OE, U3_OE	Output	**Output Enable**. RS-485/EIA-485 output enable.

Table 84. Type D I/O Control registers: FUNC values and pin functions

	Value of FUNC field in IOCON register							
Register	**000**	**001**	**'010**	**011**	**100**	**101**	**110**	**111**
IOCON_P0_0	P0[0]	CAN_RD1	U3_TXD	I2C1_SDA	U0_TXD			
IOCON_P0_1	P0[1]	CAN_TD1	U3_RXD	I2C1_SCL	U0_RXD			
IOCON_P0_2	P0[2]	U0_TXD	U3_TXD					
IOCON_P0_3	P0[3]	U0_RXD	U3_RXD					
IOCON_P0_4	P0[4]	I2S_RX_SCK	CAN_RD2	T2_CAP0		CMP_ROSC		LCD_VD[0]
IOCON_P0_5	P0[5]	I2S_RX_WS	CAN_TD2	T2_CAP1		CMP_RESET		LCD_VD[1]
IOCON_P0_6	P0[6]	I2S_RX_SDA	SSP1_SSEL	T2_MAT0	U1_RTS	CMP_ROSC		LCD_VD[8]
IOCON_P0_10	P0[10]	U2_TXD	I2C2_SDA	T3_MAT0				LCD_VD[5]
IOCON_P0_11	P0[11]	U2_RXD	I2C2_SCL	T3_MAT1				LCD_VD[10]

2.5.7. Arquitectura de las UARTs0/2/3 (2)



2.5.8. Registros (1)

Nombre	Descripción	Bit (7.. 0)	Tipo	Reset
Un**RBR**	Registro Buffer Receptor	Dato recibido más antiguo	RO	NA
Un**THR**	Registro Holding Transmisor	Dato a enviar	WO	NA
Un**DLL**	Latch Divisor LSB del PCLK	Parte baja vel.	R/W	0x01
Un**DLM**	Latch Divisor MSB del PCLK	Parte alta vel.	R/W	0x00
Un**IER**	Registro Enable Interrupciones	Habilitar interrupciones	R/W	0x00
Un**IIR**	Registro ID Interrupciones	Identificador de Interrupciones	RO	0x01
Un**FCR**	Registro Control FIFO	Buffers FIFO	WO	0x00

2.5.8. Registros (2)

Nombre	Descripción	Bit (7..0)	Tipo	Reset
Un**LCR**	Registro Control Línea	Parámetros de comunicación	R/W	0x00
Un**LSR**	Registro Status Línea	Estado registros receptor/transmisor	RO	0x60
Un**SCR**	Registro Scratch Pad	-	R/W	0x00
Un**ACR**	Registro Control Auto Baudios	-	R/W	0x00
Un**ICR**	Registro Control IrDA	-	R/W	0x00
Un**FDR**	Registro Divisor Fraccional	Parte Fraccionaria	R/W	0x10
Un**TER**	Registro Enable Transmisor	-	R/W	0x80

2.5.8. Registro Fractional Divider Register FDR

Añaden una parte fraccionaria al factor de división conseguido

Table 400: UARTn Fractional Divider Register (FDR - address 0x4000 C028 (UART0), 0x4009 8028 (UART2), 0x4009 C028 (UART3)) bit description

Bit	Function	Value	Description	Reset value
3:0	DIVADDVAL	0	Baud Rate generation pre-scaler divisor value. If this field is 0, fractional baud rate generator will not impact the UARTn baud rate.	0
7:4	MULVAL	1	Baud Rate pre-scaler multiplier value. This field must be greater or equal 1 for UARTn to operate properly, regardless of whether the fractional baud rate generator is used or not.	1
31:8	-		Reserved. Read value is undefined, only zero should be written.	0

La tasa en baudios resultante es:

$$UART_{baudrate} = \frac{PCLK}{16 \cdot (256 \cdot DLM + DLL) \cdot \left(1 + \frac{DIVADDVAL}{MULVAL}\right)}$$

$$1 \leq \text{MULVAL} \leq 15;\ 0 \leq \text{DIVADDVAL} \leq 14;\ \text{DIVADDVAL} < \text{MULVAL}$$

2.5.8. Registro Line Control Register LCR

Table 396: UARTn Line Control Register (LCR - address 0x4000 C00C (UART0), 0x4009 800C (UART2), 0x4009 C00C (UART3)) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	WLS		Word Length Select.	0
		0x0	5-bit character length	
		0x1	6-bit character length	
		0x2	7-bit character length	
		0x3	8-bit character length	
2	SBS		Stop Bit Select	0
		0	1 stop bit.	
		1	2 stop bits (1.5 if UnLCR[1:0]=00).	
3	PE		Parity Enable.	0
		0	Disable parity generation and checking.	
		1	Enable parity generation and checking.	
5:4	PS		Parity Select	0
		0x0	Odd parity. Number of 1s in the transmitted character and the attached parity bit will be odd.	
		0x1	Even Parity. Number of 1s in the transmitted character and the attached parity bit will be even.	
		0x2	Forced 1 stick parity.	
		0x3	Forced 0 stick parity.	
6	BC		Break Control	0
		0	Disable break transmission.	
		1	Enable break transmission. Output pin UARTn TXD is forced to logic 0 when UnLCR[6] is active high.	
7	DLAB		Divisor Latch Access Bit	0
		0	Disable access to Divisor Latches.	
		1	Enable access to Divisor Latches.	
31:8	-		Reserved. Read value is undefined, only zero should be written.	NA

2.5.8. Registro Line Status Register LSR (1)

Table 397: UARTn Line Status Register (LSR - address 0x4000 C014 (UART0), 0x4009 8014 (UART2), 0x4009 C014 (UART3)) bit description

Bit	Symbol	Value	Description	Reset Value
0	RDR		Receiver Data Ready. UnLSR[0] 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	OE		Overrun Error. The overrun error condition is set as soon as it occurs. An UnLSR read clears UnLSR[1]. UnLSR[1] 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	PE		Parity Error. 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	FE		Framing Error. 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 UnFCR[0]. 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.	

2.5.8. Registro Line Status Register LSR (2)

Table 397: UARTn Line Status Register (LSR - address 0x4000 C014 (UART0), 0x4009 8014 (UART2), 0x4009 C014 (UART3)) bit description

Bit	Symbol	Value	Description	Reset Value
4	BI		Break Interrupt. 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	THRE		Transmitter Holding Register Empty. 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	TEMT		Transmitter Empty. 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.	
7	RXFE		Error in RX FIFO . UnLSR[7] is set when a character with a Rx error such as framing error, parity error or break interrupt, is loaded into the UnRBR. This bit is cleared when the UnLSR register is read and there are no subsequent errors in the UARTn FIFO.	0
		0	UnRBR contains no UARTn RX errors or UnFCR[0]=0.	
		1	UARTn RBR contains at least one UARTn RX error.	
31:8	-		Reserved. The value read from a reserved bit is not defined.	NA

2.5.8. Registro FIFO Control Register FCR

Table 395: UARTn FIFO Control Register, write only (FCR - address 0x4000 C008 (UART0), 0x4009 8008 (UART2), 0x4007 C008 (UART3)) bit description

Bit	Symbol	Value	Description	Reset Value
0	FIFOEN		FIFO Enable.	0
		0	UARTn FIFOs are disabled. Must not be used in the application.	
		1	Active high enable for both UARTn Rx and TX FIFOs and UnFCR[7:1] access. This bit must be set for proper UART operation. Any transition on this bit will automatically clear the related UART FIFOs.	
1	RXFIFORES		RX FIFO Reset.	0
		0	No impact on either of UARTn FIFOs.	
		1	Writing a logic 1 to UnFCR[1] will clear all bytes in UARTn Rx FIFO, reset the pointer logic. This bit is self-clearing.	
2	TXFIFORES		TX FIFO Reset.	0
		0	No impact on either of UARTn FIFOs.	
		1	Writing a logic 1 to UnFCR[2] will clear all bytes in UARTn TX FIFO, reset the pointer logic. This bit is self-clearing.	
3	DMAMODE		DMA Mode Select. When the FIFO enable (bit 0 of this register) is set, this bit selects the DMA mode. See Section 18.6.6.1.	0
5:4	-		Reserved. Read value is undefined, only zero should be written.	NA
7:6	RXTRIGLVL		RX Trigger Level. These two bits determine how many receiver UARTn FIFO characters must be written before an interrupt or DMA request is activated.	0
		0x0	Trigger level 0 (1 character or 0x01).	
		0x1	Trigger level 1 (4 characters or 0x04).	
		0x2	Trigger level 2 (8 characters or 0x08).	
		0x3	Trigger level 3 (14 characters or 0x0E).	
31:8	-		Reserved. Read value is undefined, only zero should be written.	NA

2.5.8. Registro Interrupt Enable Register IER

Table 392: UARTn Interrupt Enable Register when DLAB = 0 (IER - address 0x4000 C004 (UART0), 0x4009 8004 (UART2), 0x4009 C004 (UART3)) bit description

Bit	Symbol	Value	Description	Reset Value
0	RBRIE		RBR Interrupt Enable. Enables the Receive Data Available interrupt for UARTn. It also controls the Character Receive Time-out interrupt.	0
		0	Disable the RDA interrupts.	
		1	Enable the RDA interrupts.	
1	THREIE		THRE Interrupt Enable. Enables the THRE interrupt for UARTn. The status of this can be read from UnLSR[5].	0
		0	Disable the THRE interrupts.	
		1	Enable the THRE interrupts.	
2	RXIE		RX Line Status Interrupt Enable. Enables the UARTn RX line status interrupts. The status of this interrupt can be read from UnLSR[4:1].	0
		0	Disable the RX line status interrupts.	
		1	Enable the RX line status interrupts.	
7:3	-		Reserved. Read value is undefined, only zero should be written.	NA
8	ABEOINTEN		Enables the end of auto-baud interrupt.	0
		0	Disable end of auto-baud Interrupt.	
		1	Enable end of auto-baud Interrupt.	
9	ABTOINTEN		Enables the auto-baud time-out interrupt.	0
		0	Disable auto-baud time-out Interrupt.	
		1	Enable auto-baud time-out Interrupt.	
31:10	-		Reserved. Read value is undefined, only zero should be written.	NA

2.5.9. Pasos para la configuración de las UARTs0/2/3 (1)

Declarar las variables de selección para configurar los parámetros de comunicación (longitud de dato, bit de parada, habilitar paridad y tipo paridad)

↓

Comprobar los parámetros de comunicación dados a la función (velocidad, bits de datos, bits de parada y paridad)

↓

Asignar a las variables de selección los valores apropiados para la configuración

↓

Activar la alimentación de la UART0/2/3: poner a 1 el bit PCONP[3] (por seguridad)

↓

Configurar los pines para los TXDn y RXDn (IOCON)

↓

Activar registros UnDLL y UnDLM: poner a 1 el bit UnLCR[7]

2.5.9. Pasos para la configuración de las UARTs0/2/3 (2)

Ajustar la velocidad (PCLK=PeripheralClock)

valor Hz = PCLK Hz/(16*velocidad estándar) (parte entera)

UnDLL = valor & 0xFF y UnDLM = valor >> 8

Ajustar los demás parámetros de comunicación en el UnLCR en los campos correspondientes

(longitud de dato UnLCR[1:0], bit de parada UnLCR[2], habilitar paridad UnLCR[3] y tipo paridad UnLCR[5:4])

Activar y limpiar los FIFOS en UnFCR: Poner a 1

FIFOEN en UnFCR[0] RXFIFORES en UnFCR[1] TXFIFORES en UnFCR[2]

2.5.9. Pasos para la recepción y transmisión de datos (3)

Transmisión:

- Comprobar si el transmisor está libre: Verificar si el bit 5 (THRE) de UnLSR está a 1
- Enviar dato (byte) por THR

Recepción:

- Comprobar si hay dato disponible: Verificar si el bit 0 (RDR) de UnLSR está a 1
- Obtener dato (byte) en RBR

Biblioteca específica

Cabecera de la biblioteca uart_lpc40xx.h

- Contiene constantes y los prototipos de las funciones

Biblioteca uart_lpc40xx.c

- Contiene las definiciones de las funciones

Fichero uart_lpc40xx.h (1)

```
/**********************************************************************
 * \file    uart_lpc40xx.h
 * \brief   Funciones de manejo de las UARTs del LPC40xx.*/

#ifndef UART_LPC40XX_H
#define UART_LPC40XX_H

#include <LPC407x_8x_177x_8x.h>
#include "gpio_lpc40xx.h"
#include "tipos.h"

#define LPC_UART1_  ((LPC_UART_TypeDef*)LPC_UART1_BASE)
#define LPC_UART4_  ((LPC_UART_TypeDef*)LPC_UART4_BASE)

/*===== Constantes ==========================================*/
/* Símbolos para referirse a los bloques de registros de las UARTs.*/

#define UART0   LPC_UART0
#define UART1   LPC_UART1_
#define UART2   LPC_UART2
#define UART3   LPC_UART3
#define UART4   LPC_UART4_
```

Fichero uart_lpc40xx.h (2)

```
/* Símbolos para especificar baudrates estándar.*/

#define UART_BAUDRATE_110       110u
#define UART_BAUDRATE_300       300u
#define UART_BAUDRATE_600       600u
#define UART_BAUDRATE_1200      1200u
#define UART_BAUDRATE_2400      2400u
#define UART_BAUDRATE_4800      4800u
#define UART_BAUDRATE_9600      9600u
#define UART_BAUDRATE_14400     14400u
#define UART_BAUDRATE_19200     19200u
#define UART_BAUDRATE_28800     28800u
#define UART_BAUDRATE_38400     38400u
#define UART_BAUDRATE_57600     57600u
#define UART_BAUDRATE_115200    115200u
```

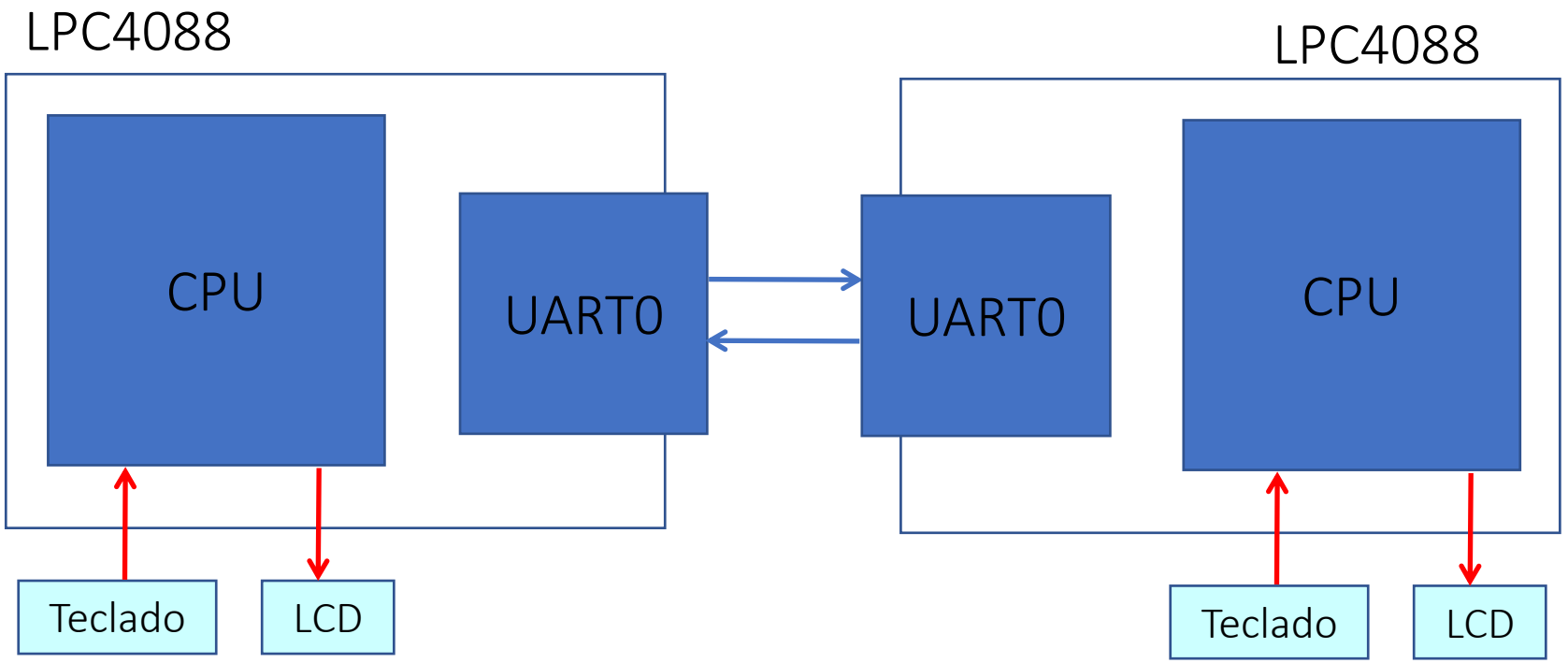
Fichero uart_lpc40xx.h (3)

```
/* Símbolos para especificar el número de bits de datos.*/

#define UART_BITS_DATOS_5       5u
#define UART_BITS_DATOS_6       6u
#define UART_BITS_DATOS_7       7u
#define UART_BITS_DATOS_8       8u

/* Símbolos para especificar el número de bits de stop.*/

#define UART_BITS_STOP_1        1u
#define UART_BITS_STOP_2        2u

/* Tipos de paridad.*/

typedef enum
{ UART_PARIDAD_NINGUNA,
  UART_PARIDAD_IMPAR,
  UART_PARIDAD_PAR
} uart_tipo_paridad_t;
```

Fichero uart_lpc40xx.h (4)

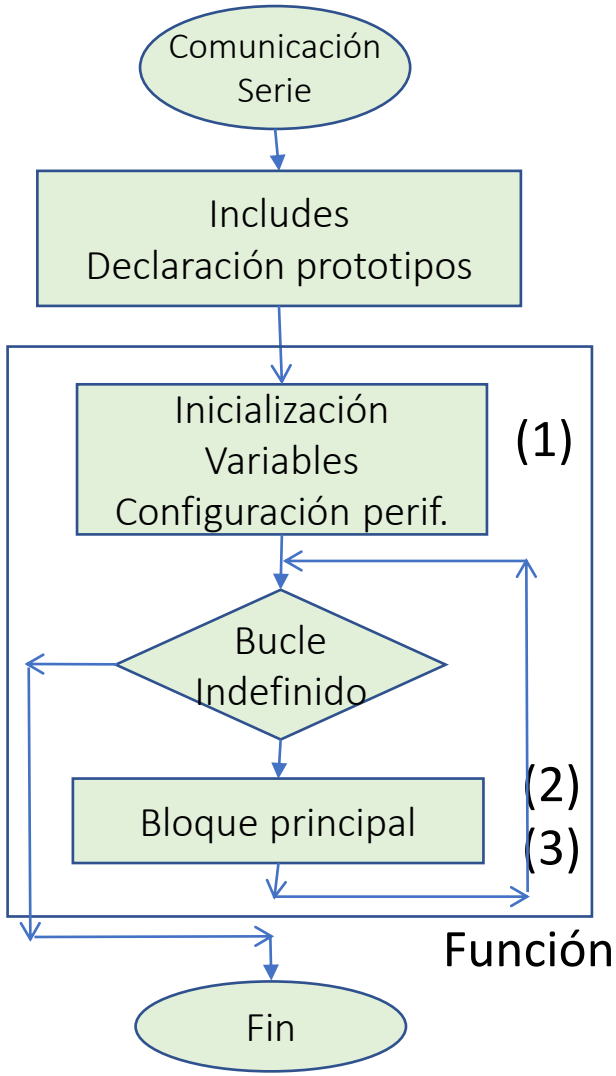
```
/*===== Prototipos de funciones================================*/
void uart_inicializar(LPC_UART_TypeDef *uart_regs,
                      uint32_t baudrate,
                      uint32_t numero_bits_datos,
                      uart_tipo_paridad_t tipo_paridad,
                      uint32_t numero_bits_stop,
                      LPC_GPIO_TypeDef *puerto_txd,
                      uint32_t mascara_pin_txd,
                      LPC_GPIO_TypeDef *puerto_rxd,
                      uint32_t mascara_pin_rxd,
                      float32_t *baudrate_real_obtenido);

void uart_transmitir_dato(LPC_UART_TypeDef *uart_regs,uint8_t dato);
bool_t uart_hay_dato_disponible(LPC_UART_TypeDef *uart_regs);
uint8_t uart_leer_dato(LPC_UART_TypeDef *uart_regs);
uint8_t uart_esperar_recibir_dato(LPC_UART_TypeDef *uart_regs);
void uart_transmitir_cadena(LPC_UART_TypeDef *uart_regs, const char *cadena);
void uart_recibir_cadena(LPC_UART_TypeDef *uart_regs, char *ptr_buffer,
                         uint32_t tamano_buffer);
#endif  /* UART_LPC40XX_H */
```

2.5.10. Ejemplo: Envío de cadenas de caracteres entre 2 µC



Las dos UARTs deben tener los mismos parámetros de comunicación serie

UART0 sin interrupción



Función Main

Funciones	Periféricos
(1) Función Inicializar	UART0
(2) Función Transmitir cadena	LCD
(3) Función Recibir cadena	TECLADO

Bibliotecas específicas:

uart_lpc40xx.h

uart_lpc40xx.c

Función main() (1)

```
/*****************************************************************
 * \file    main.c
 */
#include <LPC407x_8x_177x_8x.h>
#include "glcd.h"
#include "teclado_4x4.h"
#include "error.h"
#include "uart_lpc40xx.h"
#include "stdlib.h"
#include "stdio.h"

int main(void)
{ // Declaracion de las variables necesarias
  char cadena_a_enviar[20];
  char cadena_a_recibir[20];
  // Configuracion perifericos
  glcd_inicializar();
  tec4x4_inicializar();
  uart_inicializar(UART0,UART_BAUDRATE_9600, UART_BITS_DATOS_8, UART_PARIDAD_NINGUNA,
                   UART_BITS_STOP_1, PUERTO0, PIN2, PUERTO0, PIN3, NULL);
```

Función main() (2)

```
    //Imprime textos fijos
    glcd_xprintf(8,20, BLANCO, NEGRO, FONT16X32, "Cadena a enviar:");
    glcd_xprintf(8,120,BLANCO, NEGRO, FONT16X32, "Cadena recibida:");

    while (1)
    { //Muestreo teclado
      if (tec4x4_esperar_estable() != TEC4X4_NINGUNA_PULSADA)
      {glcd_xprintf(164, 20, BLANCO, NEGRO, FONT16X32, "                    ");
       tec4x4_leer_cadena (cadena_a_enviar, sizeof (cadena_a_enviar));
       glcd_xprintf(164, 20, BLANCO, NEGRO, cadena a enviar);
       uart_transmitir_cadena(UART0, cadena_a_enviar);
      }
      //Muestreo UART
      if (uart_hay_dato_disponible())
      { uart_recibir_cadena(UART0,cadena_a_recibir,20);
        glcd_xprintf(164, 120, BLANCO, NEGRO, FONT16X32, "                    ");
        glcd_xprintf(164, 120, BLANCO, NEGRO, FONT16X32, cadena_a_recibir);
      }
    } //Cierra while
} //Cierra main
```