Punto 5

Las funciones utilizadas para el envío de mensajes de depuración por puerto serie son

debugPrintConfigUart(…) y debugPrintString(…).

Ambas pertenecen a la librería sapi\_debugPrint.h, ubicada en /firmware\_v2/modules/lpc4337\_m4/sapi/inc/sapi\_debugPrint.h

Allí se tienen las siguientes declaraciones:

**#define** debugPrintConfigUart(uart,baudRate) printConfigUart(&(debugPrint),(uart),(baudRate))

**#define** debugPrintString(string) printString((debugPrint),(string))

Es decir que en realidad se están llamando implícitamente a las funciones printConfigUart(…) y printString(…) que se encuentran definidas en /firmware\_v2/modules/lpc4337\_m4/sapi/src/sapi\_print.c (Figura 1).

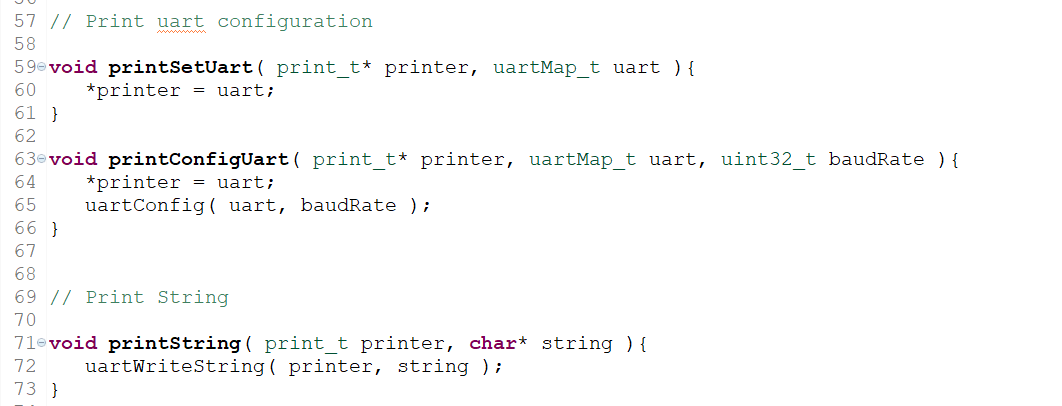


Figura 1.

Dentro de los parámetros que recibe printConfigUart se encuentra el tipo de dato uartMap\_t, que está definido de la siguiente forma:

**typedef** **enum**{

UART\_GPIO = 0, // Hardware UART0 via GPIO1(TX), GPIO2(RX) pins on header P0

UART\_485 = 1, // Hardware UART0 via RS\_485 A, B and GND Borns

//UART\_1 = 2, // Hardware UART1 not routed

UART\_USB = 3, // Hardware UART2 via USB DEBUG port

UART\_ENET = 4, // Hardware UART2 via ENET\_RXD0(TX), ENET\_CRS\_DV(RX) pins on header P0

UART\_232 = 5 // Hardware UART3 via 232\_RX and 232\_tx pins on header P1

} uartMap\_t;

Con eso se establece la vía mediante la cual se establecerá la comunicación. En este caso, se elige la opción UART\_USB = 3, es decir que la comunicación será a través del puerto USB DEBUG.

También recibe como parámetro el baud rate, que marcará la tasa con que se transmita la información y está tipificado como un entero de 32 bits.

Esta función llama a su vez a otra función: uartConfig(uart, baudRate) que está ubicada en /firmware\_v2/modules/lpc4337\_m4/sapi/src/sapi\_uart.c

Esta función inicializa y configura varios parámetros de la UART (Figura 2).

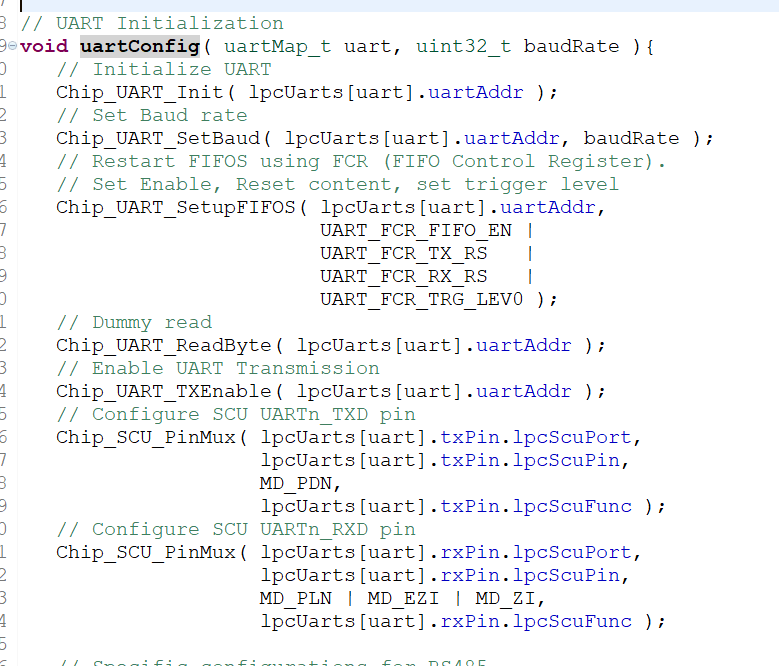


Figura 2.

Donde lpcUarts[] se define como:

**static** **const** uartLpcConfig\_t lpcUarts[] = {

// { uartAddr, { txPort, txpin, txfunc }, { rxPort, rxpin, rxfunc }, uartIrqAddr },

// UART\_GPIO (GPIO1 = U0\_TXD, GPIO2 = U0\_RXD)

{ LPC\_USART0, { 6, 4, FUNC2 }, { 6, 5, FUNC2 }, USART0\_IRQn }, // 0

// UART\_485 (RS485/Profibus)

{ LPC\_USART0, { 9, 5, FUNC7 }, { 9, 6, FUNC7 }, USART0\_IRQn }, // 1

// UART not routed

{ LPC\_UART1, { 0, 0, 0 }, { 0, 0, 0 }, UART1\_IRQn }, // 2

// UART\_USB

{ LPC\_USART2, { 7, 1, FUNC6 }, { 7, 2, FUNC6 }, USART2\_IRQn }, // 3

// UART\_ENET

{ LPC\_USART2, { 1,15, FUNC1 }, { 1,16, FUNC1 }, USART2\_IRQn }, // 4

// UART\_232

{ LPC\_USART3, { 2, 3, FUNC2 }, { 2, 4, FUNC2 }, USART3\_IRQn } // 5

};

Es decir que es aquí donde se especifican los números de puerto y de pin para transmisión y recepción para cada UART.

La función printString recibe entre sus parámetros un puntero al inicio de una cadena de caracteres y llama a la función uartWriteString( printer, string ) que se encuentra también en /firmware\_v2/modules/lpc4337\_m4/sapi/src/sapi\_uart.c. Una captura de la función se ve en la figura 3.

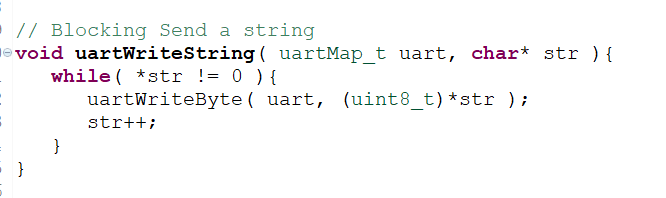


Figura 3.

Se ve que la función recibe la cadena de caracteres y la vía de comunicación e invoca a la función uartWriteByte hasta que se termine de escribir la cadena entera. La función uartWriteByte está definida en el mismo lugar y lo que hace es recibir de a un byte por vez y escribirlo.

Los mensajes enviados en este caso son:

debugPrintString( "DEBUG c/sAPI\r\n" );

a modo de inicialización luego de configurar la UART y

debugPrintString( "LED Toggle\n" );

que se enviará cada vez que se invoque a la función

gpioToggle(LED3);