

Comunicación serie asíncrona: La UART en el LPC1768

Versión 2.0

Sistemas Electrónicos Digitales

**Universidad de Alcalá
Departamento de Electrónica**

1 Introducción al documento

Este documento presenta un resumen del sistema de comunicación serie asíncrona que incorpora el LPC1768, conocido como UART.

La información está obtenida fundamentalmente de los capítulos 14 y 15 del LPC17xx User Manual proporcionado por NXP.

2 Índice de contenidos

1	INTRODUCCIÓN AL DOCUMENTO	3
2	ÍNDICE DE CONTENIDOS.....	3
3	INTRODUCCIÓN A LA COMUNICACIÓN SERIE ASÍNCRONA.....	5
4	LA COMUNICACIÓN SERIE ASÍNCRONA EN EL LPC1768.....	8
4.1	CONFIGURACIÓN DEL FORMATO DE LA COMUNICACIÓN.....	12
4.2	GENERADOR DE LA VELOCIDAD DE COMUNICACIÓN	13
4.3	FUNCIONAMIENTO DE LA UART POR INTERRUPCIÓN.....	16
4.4	FUNCIONAMIENTO DE LA UART CON FIFO	19
5	LA UART1	19
6	REGISTROS DE CONFIGURACIÓN Y CONTROL DE LA UART	22

3 Introducción a la comunicación serie asíncrona

La comunicación serie asíncrona constituye una de las formas más simples de comunicar un microcontrolador con multitud dispositivos periféricos externos, entre ellos un ordenador u otro microcontrolador, con tan solo dos líneas de conexión (además de la señal de referencia de tensión o masa). Sólo se precisa de una línea de transmisión (TxD) y otra de recepción (RxD) si se desea una comunicación en ambos sentidos, entre el microcontrolador y el periférico (figura 3.1).

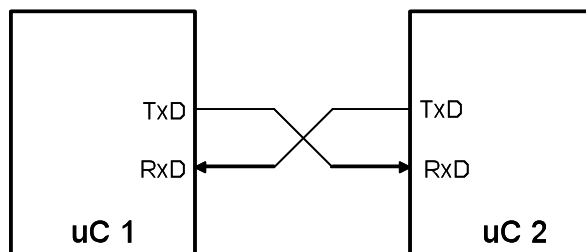


Fig. 3.1. Esquema de interconexión de dos microcontroladores a través de la interfaz serie asíncrona

Este tipo de comunicación se caracteriza porque no se hace necesaria una señal de reloj adicional o sincronismo entre el transmisor y receptor que marque el instante de llegada de los bits.

La transmisión serie de un dato se realiza enviando y recibiendo tramas de bits por los puertos TxD y RxD respectivamente. Asumiendo que el estado de reposo de la línea de transmisión y recepción es un nivel alto, el envío o recepción de cada trama comienza con un bit de START ('0' lógico). A continuación se envía el dato comenzando por el bit LSB, que puede tener una longitud de bits variable (en este caso 5, 6, 7 u 8 bits) seguido, si se desea, de un bit de paridad para detectar errores en la transmisión/recepción de datos, para finalizar con uno o dos bits (a configurar) de STOP a nivel alto.

La figura 3.2 muestra un ejemplo de transmisión para el carácter ASCII correspondiente a la letra 'A', donde se muestra que la transmisión de un dato comienza con el bit START, seguido de los bits que codifican el dato, el bit de paridad (paridad par), para finalizar con un bit de STOP antes de quedar la señal de nuevo en estado de reposo (nivel alto).

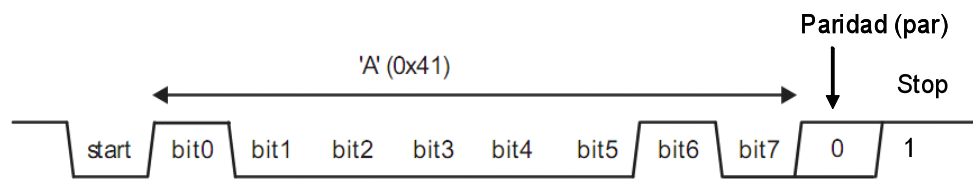


Fig. 3.2. Formato de la comunicación serie asíncrona

La velocidad de la comunicación se mide en baudios o bits por segundo, y constituye el inverso del tiempo de bit. Esta es programable y depende de la frecuencia de CPU.

Para una transmisión continua de caracteres (figura 3.3), estos se envían uno a continuación del otro. Nótese que el bit de STOP además de ser una forma de indicar el final de un carácter o *frame*, garantiza que el estado de reposo de la señal TxD se mantenga nivel alto antes de iniciar una transmisión de un nuevo dato.

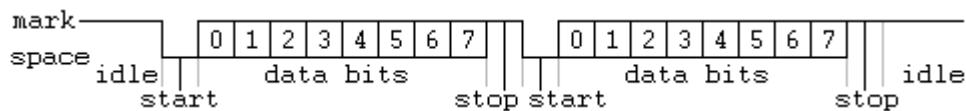


Fig. 3.3. Ejemplo del formato asíncrono para una transmisión de dos caracteres

Estas características junto con otras relacionadas con aspectos físicos, de niveles de tensión, de funcionalidades de otras señales adicionales necesarias para el control de un MODEM, etc. están definidas en la norma RS232C. Se trata de una de las normas más populares empleadas en la comunicación serie (su inserción en el PC incrementó su popularidad, aunque hoy en día la tendencia es a desaparecer sobre todo en los ordenadores portátiles). Fue desarrollada en la década de los 60 para definir un estándar de interconexión entre equipos terminales (DTE) y MODEMs o equipos de comunicación de datos (DCE).

Para más información acceder a: <http://perso.wanadoo.es/pictob/comserie.htm>

El periférico interno del microcontrolador que implementa este tipo de comunicación recibe el nombre de UART (*Universal Asynchronous Receiver-Transmitter*) y está basado en un registro de desplazamiento serie-paralelo para el receptor, y otro paralelo-serie para el transmisor debido a los buses paralelos internos que maneja la CPU.

La transmisión se inicia al escribir en el registro de datos de salida, dando lugar a la generación de la señal de reloj que fija la velocidad de transmisión. El dato se transfiere a un registro de desplazamiento al que se han insertado automáticamente el bit de START y STOP (figura 3.4). Los bits salen por la línea TxD comenzando por el bit

START. Con el envío del bit de STOP se activa un flag indicador que el registro transmisor está vacío y listo para cargar un nuevo dato para ser transmitido. Esta condición puede generar una interrupción, si se habilita.

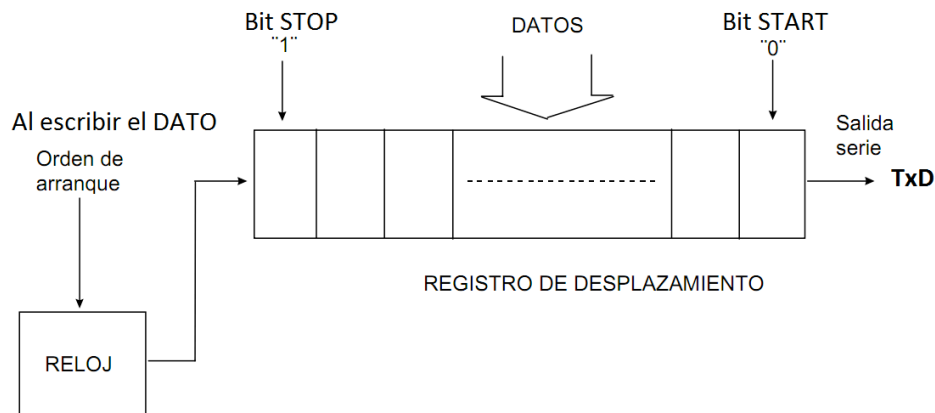


Fig. 3.4. Esquema del circuito transmisor

El receptor (figura 3.5) muestrea la línea de recepción RxD a una frecuencia 16 veces superior a la velocidad de recepción. Al detectar un flanco de bajada se sincroniza el reloj para iniciar el proceso de detección del bit START, que se valida si se detecta un cero a mitad del tiempo de bit (figura 3.6). El resto de bits se muestrean a mitad de su tiempo de bit previsto, de ahí que sea necesario (con un pequeño margen de error), que la velocidad de transmisión y recepción sean idénticas, para que el dato recibido coincida con el transmitido. (<http://pdfserv.maxim-ic.com/en/an/AN2141.pdf>).

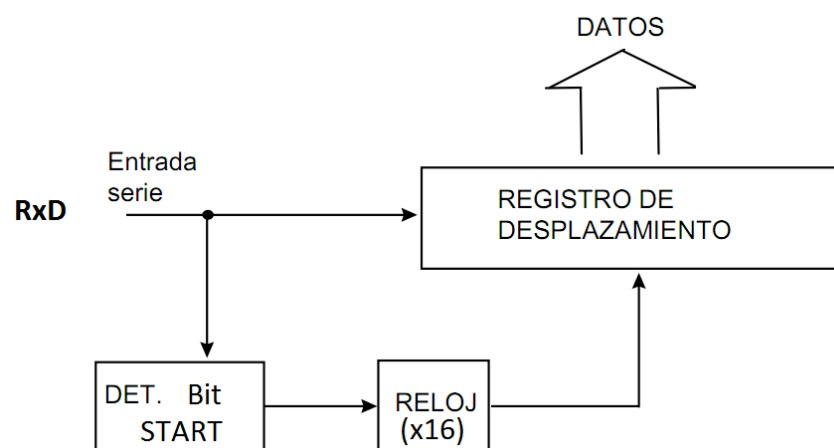


Fig. 3.5. Esquema del circuito receptor

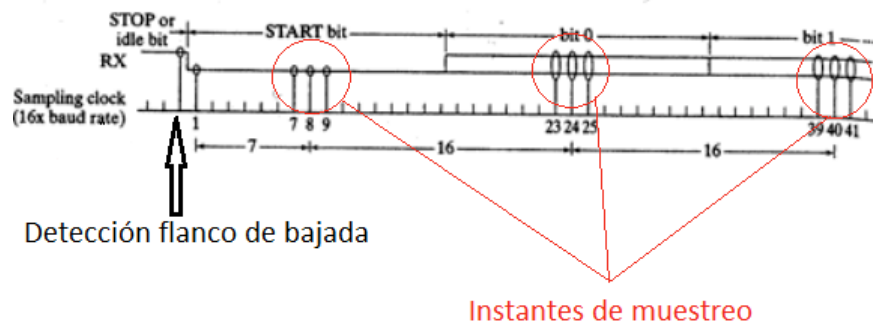


Fig. 3.6. Proceso de muestreo en recepción

Con la llegada del bit de STOP se activa un flag indicador que el registro receptor está lleno y listo para ser leído por la CPU. Esta condición puede generar una interrupción, si se habilita.

Las UARTs más modernas que incorporan hoy en día los microcontroladores más avanzados incorporan una pequeña memoria FIFO en el transmisor y en el receptor. Con ello, reducen la sobrecarga de CPU pues no se precisa, por ejemplo en recepción, leer cada dato individual recibido en la función de interrupción, sino sólo cuando la FIFO se llena. De esta forma la frecuencia de interrupción de la interfaz serie se reduce en un factor igual al tamaño de la memoria FIFO interna, y esto hoy en día en los sistemas empujados que hacen uso de sistemas operativos de tiempo real, es prácticamente una necesidad para aliviar a la CPU en la tarea asociada a la interfaz serie, sobre todo cuando se trabaja a velocidades elevadas.

4 La comunicación serie asíncrona en el LPC1768

El LPC1768 incorpora cuatro interfaces de comunicación serie asíncronas o UARTs (figura 4.1). Las UARTs 0, 2 y 3 tienen un funcionamiento idéntico con sus respectivos registros de configuración y control. La UART 1 añade funcionalidades para servir de interfaz con un MODEM utilizando unas señales adicionales de control de flujo o *handshake*.

Para realizar la transmisión serie bastan sólo dos líneas, una de transmisión (TxD_n) y otra de recepción (RxD_n), además de la señal de masa o referencia de tensión, donde **n** es el número de UART (0, 1, 2, 3).

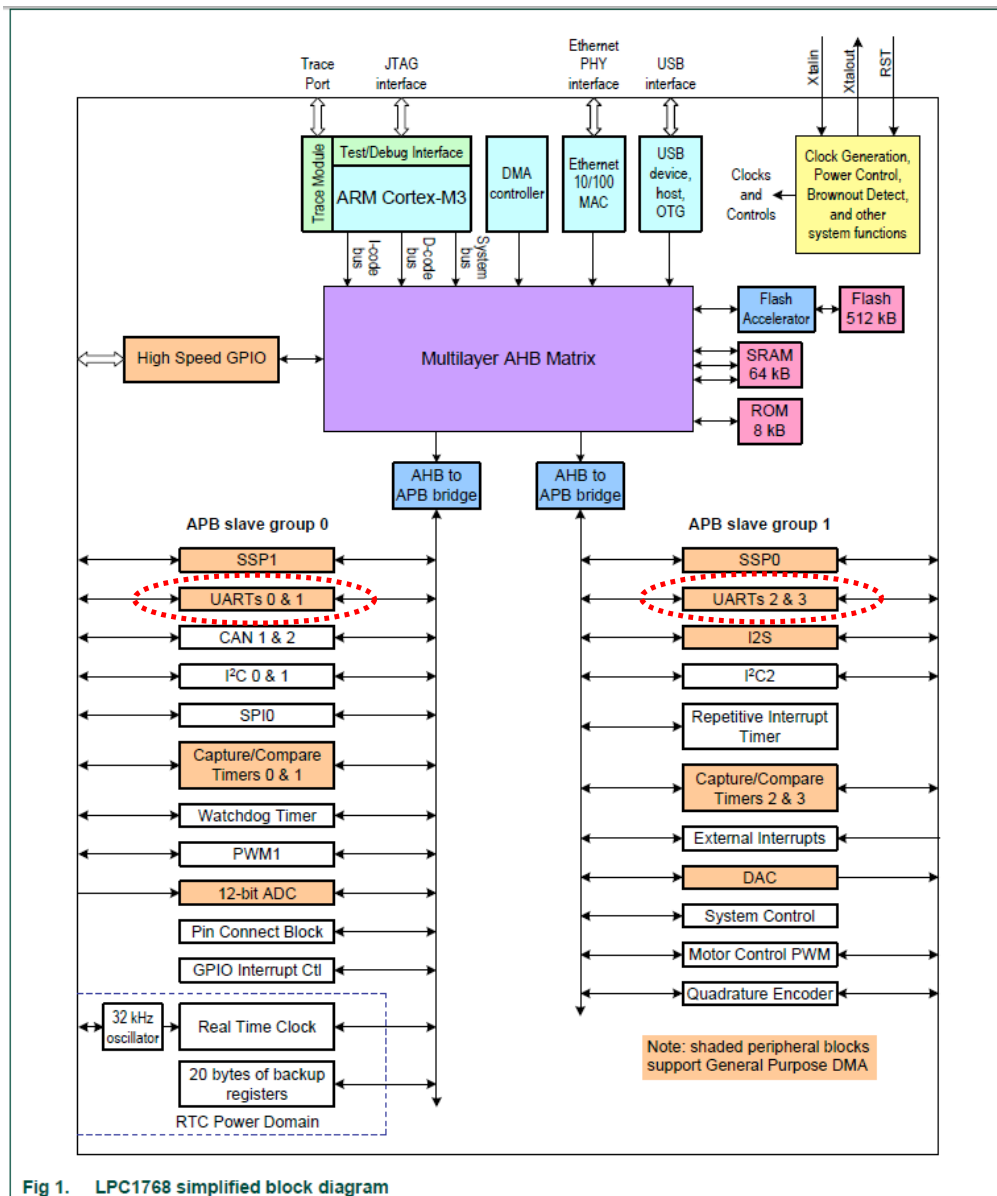


Fig. 4.1. Interfaces UART en el LPC1768

Para poder usar cada una de las UARTs, éstas deben estar alimentadas (registro PCONP), los pines asociados configurados como RxD y TxD (registro PINSELn), y si la comunicación se gestiona por interrupción, su habilitación, además de su prioridad a través de los registros del NVIC. También debe de haberse establecido la velocidad de transmisión en baudios deseada (en los registros de la UART que corresponda).

Algunas características a destacar de las UARTs 0, 2 y 3 del LPC1768 son las memorias FIFO de 16 bytes que incorporan tanto en los módulos de transmisión, como de recepción y la posibilidad de usar el DMA en la transmisión y recepción de datos. También soporta formato IrDA (para transmisión por infrarrojos) y control de flujo mediante software.

En la figura 4.3 se resaltan las direcciones, dentro del mapa de memoria de periféricos, correspondientes a las UARTs.

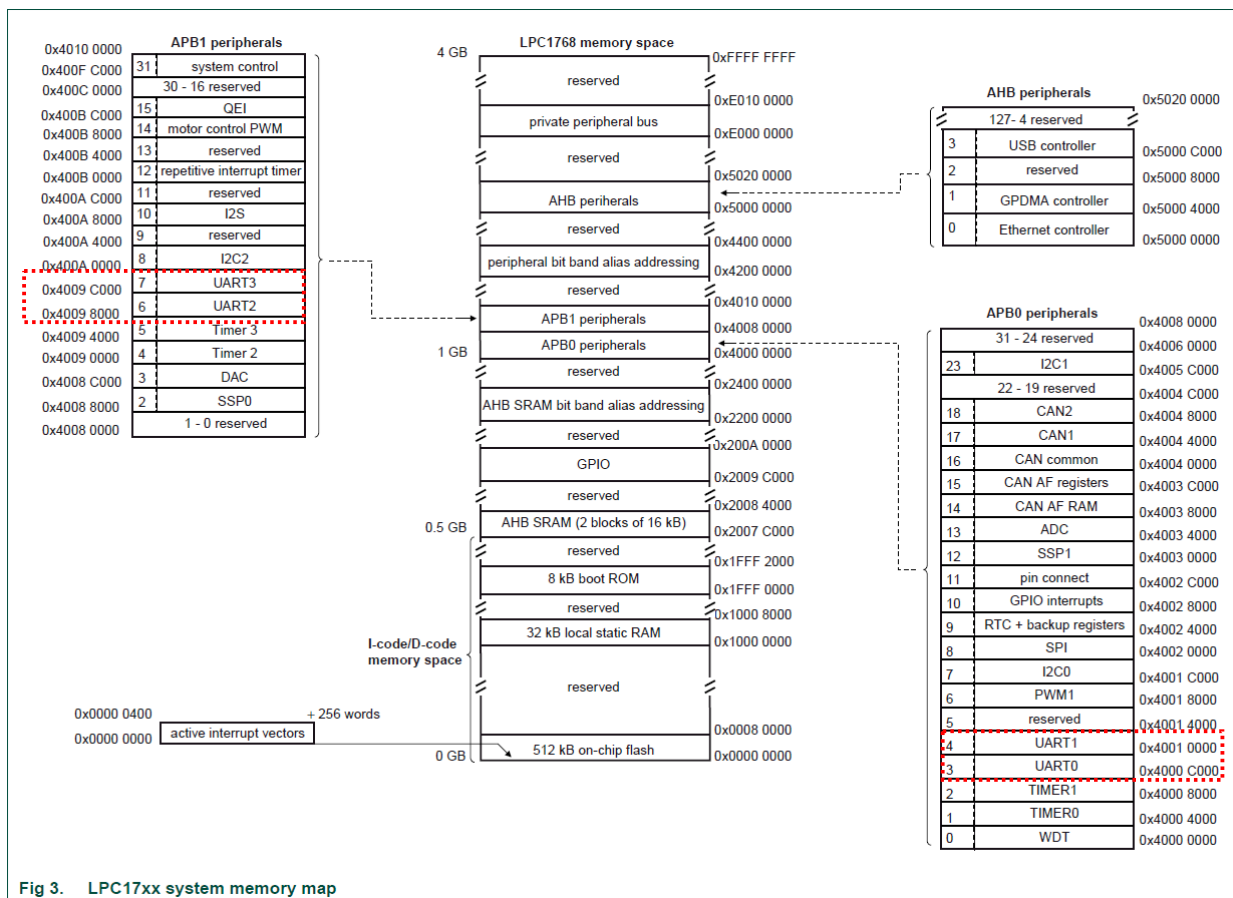


Fig. 4.3. Rango de direccionamiento de las UARTs en el mapa de memoria de periféricos

En la figura 4.4 se observa el diagrama de bloques correspondiente a las UARTs 0, 2 y 3 donde se ven claramente los distintos bloques internos que intervienen en su funcionamiento:

- Bloque de transmisión de datos
- Bloque de recepción de datos.
- Bloque generador de baudios o de velocidad de transmisión,
- Bloque de control y estado de funcionamiento.

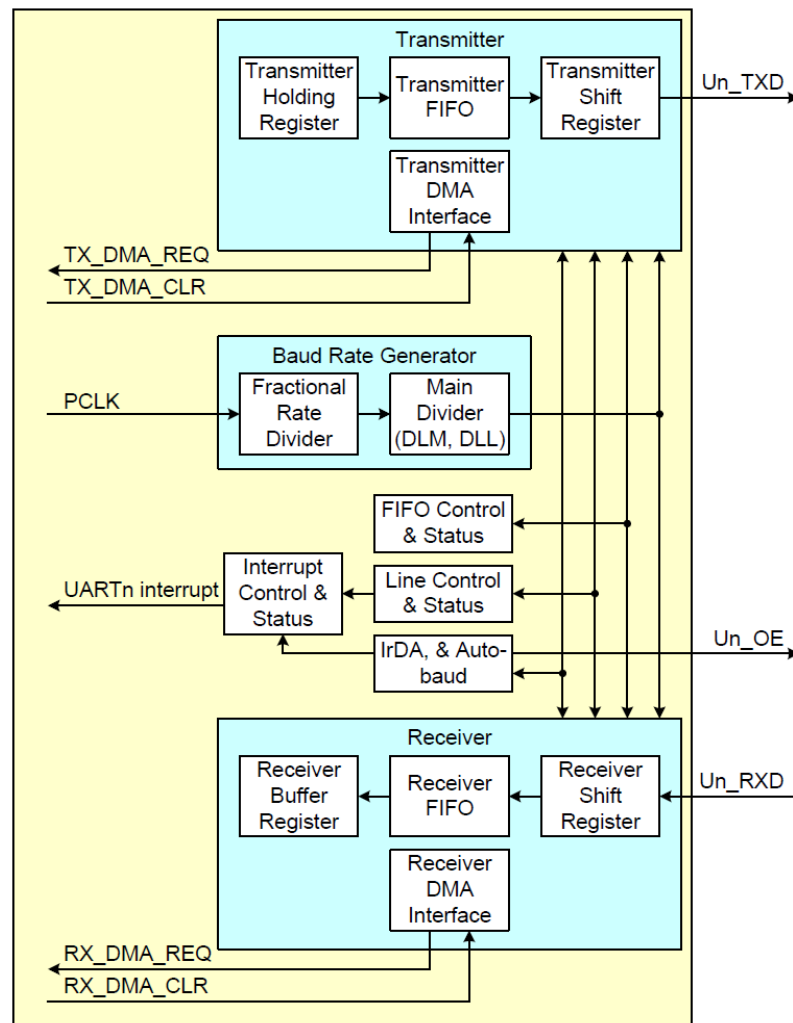


Fig. 4.4. Diagrama de bloques funcional de las UART0, 2 y 3

El bloque de recepción de datos (*Receiver*) monitoriza la entrada serie de datos y se asegura de que sea una entrada válida. El dato recibido pasa por un registro de desplazamiento (*Receiver Shift Register* o RSR) y de ser válido será guardado en una posición de la FIFO, donde se almacenará hasta que sea leída por la CPU. El byte más alto de la FIFO del receptor, y que puede ser leído por la CPU se corresponde con los 8 primeros bits (RBR) del registro UnRBR o *Receiver Buffer Register* (siendo n el número de UART 0, 2 ó 3). Para poder acceder a este registro el bit 7 del registro UnLCR debe estar a nivel bajo (DLAB = '0').

El bloque de transmisión acepta el dato que se desea transmitir en el *UARTn TX holding Register*, el cual se corresponde con el registro UnTHR y su escritura da lugar a que se inicie la transmisión. De ahí pasa directamente a la FIFO de transmisión. Cuando el dato alcanza el byte más bajo de la FIFO y el transmisor esté disponible tras haber enviado el dato anterior, el dato escrito anteriormente será enviado. Para poder acceder a este registro el bit 7 del registro UnLCR debe estar a nivel bajo (DLAB = '0').

4.1 Configuración del formato de la comunicación

El formato del dato ó carácter que será transmitido/recibido se configura en el registro **UnLCR** (*Line Control Register*):

- UnLCR[0,1], los dos primeros bits (Word Length Select) configuran el tamaño del dato transmitir (“00”: 5 bits, “01”: 6 bits, “10”: 7 bits, “11”: 8 bits).
- En UnLCR[2] (Stop Bit Select) se selecciona el número de bits de STOP (‘0’: 1 bit, ‘1’: 2 bits).
- UnLCR[3] (Parity Enable) a ‘1’, indica que se transmitirán bits de paridad, y en UnLCR[4,5] (Parity Select) se seleccionará el valor lógico del bit de paridad (“00”: paridad par, “01”: paridad impar, “10”: forzado a ‘1’, “11”: forzado a ‘0’).
 - NOTA: Paridad par quiere decir que el bit de paridad será ‘0’ si el número total de ‘1’s a transmitir es par. El bit de paridad, por tanto, trata de mantener un número par o impar de unos en el carácter en función de la paridad elegida.
- UnLCR[6] (Break Control) a ‘1’ fuerza el pin de transmisión a ‘0’, es decir, corta la transmisión.
- UnLCR[7] (**Divisor Latch Access Bit, DLAB**). **Es necesario activarlo para acceder a los registros de configuración de la velocidad de la comunicación o al byte alto de cada FIFO.**

Las UARTs tienen el registro UnICR (*IrDA Control Register*) para habilitar y configurar el modo IrDA, de manera que el dispositivo está preparado para la transmisión de datos por infrarojo. En IrDA[0] (IrDAEn) se habilita este modo, en IrDA[1] (IrDAInv) se habilita la entrada serial invertida (la salida no se invierte), en IrDA[2] (FixPulseEn) se habilita el modo de ancho de pulso fijo y en los bits IrDA[3..5] (PulseDiv) se configura el pulso según la tabla 4.1.

Tabla 4.1 Ancho de pulso de modo IrDA

FixPulseEn	PulseDiv	IrDA Transmitter Pulse width (us)
0	X	3 / (16 x baud rate)
1	0	2 x T _{PCLK}
1	1	4 x T _{PCLK}
1	2	8 x T _{PCLK}
1	3	16 x T _{PCLK}
1	4	32 x T _{PCLK}
1	5	64 x T _{PCLK}
1	6	128 x T _{PCLK}
1	7	256 x T _{PCLK}

El registro UnTER es el registro de habilitación de la transmisión. Sólo es útil UnTER[7] en donde se habilita o deshabilita la implementación de control de flujo de transmisión por software. Esto implica (de estar activado a '1') que el transmisor estará enviando datos mientras éstos estén disponibles. Tan pronto como UnTER[7] (campo TXEN) sea 0, la transmisión finalizará. Tras un reset TXEN = '1'.

4.2 Generador de la velocidad de comunicación

El bloque generador de baudios establece la velocidad a la que va a trabajar la UART, es decir, la velocidad a la que se van a recibir/transmitir los datos. La generación del *baud rate* o velocidad en baudios puede establecerse de forma manual o automática.

De forma manual, hay que configurar los registros **UnDLL** y **UnDLM** (n se corresponde con el número de UART 0, 2 ó 3). Cada uno de tamaño byte, el primero se corresponde con la parte baja y el segundo con la parte alta del divisor de la señal de reloj de entrada (PCLK_UARTn, del registro PCLK). En conjunto el divisor del reloj o prescaler es de 16 bits. **Para poder acceder a estos registros, el bit 7 del registro UnLCR debe estar a nivel alto (DLAB = '1').**

Además, si se desea, se puede añadir un valor fraccional o factor divisor mediante el registro UnFDR con objeto de aproximar con mayor exactitud la velocidad de transmisión deseada. Este registro cuenta con los campos DIVADDVAL (UnFDR[0..3]) y MULVAL (UnFDR[4..7]) que son configurables. Para usarlos hay que cumplir las siguientes reglas:

1. $1 \leq \text{MULVAL} \leq 15$
2. $0 \leq \text{DIVADDVAL} \leq 14$
3. $\text{DIVADDVAL} \leq \text{MULVAL}$

Además, si $\text{DIVADDVAL} > 0$ y $\text{UnDLM}=0$, entonces **UnDLL** debe ser mayor que 2.

Finalmente el *baud rate* o *velocidad de la comunicación* puede calcularse mediante la fórmula:

$$\text{UARTn}_{\text{baudrate}} = \frac{\text{PCLK}}{16 \times (256 \times \text{UnDLM} + \text{UnDLL}) \times \left(1 + \frac{\text{DivAddVal}}{\text{MulVal}}\right)}$$

Donde, PCLK es la frecuencia de reloj de periféricos (por defecto, la cuarta parte de la de CPU).

Las UARTs pueden funcionar sin utilizar el registro UnFDR escribiendo un “0” en el campo DIVADDVAL (valor por defecto tras el reset) y un “1” en MULVAL (valor por defecto tras el reset).

Como ejemplo, para configurar la UART0 a 19200 baudios, sin utilizar FR calcularíamos:

$$U0DL_{16} = \frac{PCLK[Hz]}{16 \cdot V_i[baudios]} = \frac{100^6 / 4}{16 \cdot 19200} = 81.38$$

Como **no** se obtiene un **valor entero**, ajustamos al valor entero más próximo (83) para cometer el menor error en la velocidad real y escribimos en U0DLL y U0DLM (es necesario que DLAB=1):

```
LPC_UART0->LCR |= DLAB_ENABLE;    // importante poner a 1
LPC_UART0->DLM = 0;
LPC_UART0->DLL = 81;
LPC_UART0->LCR &= ~DLAB_ENABLE;    // importante poner a 0
```

Si el error cometido en la velocidad real es grande, podemos hacer uso del factor FR despejando de la fórmula general y obteniendo los valores DivAddVal y MulVal de la tabla 4.2.

El fabricante proporciona el diagrama de flujo mostrado en la figura 4.5 como método para elegir los valores más adecuados de los registros a partir de la tabla 4.2, que hacen que para un valor de FR estimado entre 1.1 y 1.9, se obtenga la velocidad real lo más aproximada a la teórica.

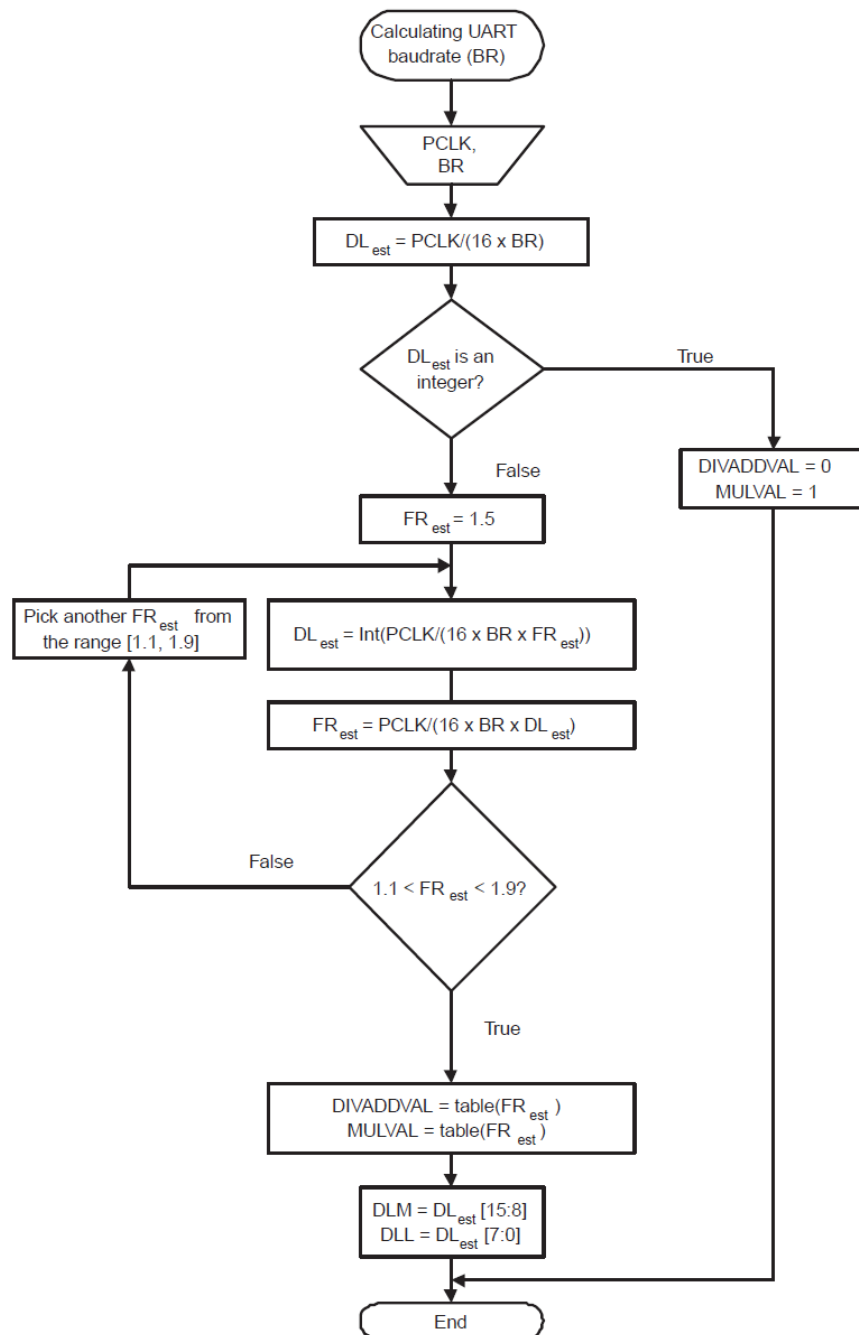


Fig. 4.5. Método para programar el *baud-rate* de las UART 0, 2 y 3 a partir del factor FR estimado

La tabla 4.2 muestra los valores de DIVADDVAL y MULVAL para obtener el factor divisor FR, que dará una mejor aproximación de la velocidad en caso de ser necesaria.

Tabla 4.2 Valores predeterminados DivAddVal/MulV que reducen el error de *baud rate*.

FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal
1.000	0/1	1.250	1/4	1.500	1/2	1.750	3/4
1.067	1/15	1.267	4/15	1.533	8/15	1.769	10/13
1.071	1/14	1.273	3/11	1.538	7/13	1.778	7/9
1.077	1/13	1.286	2/7	1.545	6/11	1.786	11/14
1.083	1/12	1.300	3/10	1.556	5/9	1.800	4/5
1.091	1/11	1.308	4/13	1.571	4/7	1.818	9/11
1.100	1/10	1.333	1/3	1.583	7/12	1.833	5/6
1.111	1/9	1.357	5/14	1.600	3/5	1.846	11/13
1.125	1/8	1.364	4/11	1.615	8/13	1.857	6/7
1.133	2/15	1.375	3/8	1.625	5/8	1.867	13/15
1.143	1/7	1.385	5/13	1.636	7/11	1.875	7/8
1.154	2/13	1.400	2/5	1.643	9/14	1.889	8/9
1.167	1/6	1.417	5/12	1.667	2/3	1.900	9/10
1.182	2/11	1.429	3/7	1.692	9/13	1.909	10/11
1.200	1/5	1.444	4/9	1.700	7/10	1.917	11/12
1.214	3/14	1.455	5/11	1.714	5/7	1.923	12/13
1.222	2/9	1.462	6/13	1.727	8/11	1.929	13/14
1.231	3/13	1.467	7/15	1.733	11/15	1.933	14/15

4.3 Funcionamiento de la UART por interrupción

Cada una de las UARTs tiene asociado un vector de interrupción y su función de interrupción asociada correspondiente, siendo varias las causas que pueden ocasionar una interrupción (por recepción, transmisión, errores, etc.), motivo por el cual es necesario consultar la causa de interrupción al entrar en la función de interrupción.

Para que las interrupciones de la UART puedan tener lugar, es preciso habilitarlas primero en el NVIC (ISERT_UARTn). Después, ya en los registros de la UART podemos encontrar el registro de habilitación de interrupciones UNIER y el registro de identificación de la fuente de interrupción UNIIR. En el primero se habilitan las posibles interrupciones que se puedan dar, y en el segundo se encuentran los flags de las interrupciones, con lo que podemos identificar en la rutina de interrupción cuál ha sido la fuente.

Las posibles fuentes de interrupción de las UARTs que pueden ser habilitadas escribiendo un “1” en el bit correspondiente del registro UNIER son:

- RDA (*Receive Data Available*) en UnIER[0].
- THRE (*Transmitter Holding Register Empty*) en UnIER[1].
- RLS (*Receive Line Status*) en UnIER[2]
- ABEO en UnIER[8] y ABTO en UnIER[9].

UnIER[0] habilita la interrupción por RDA (*Receive Data Available*) que se activa cuando hay un dato disponible en UnRBR o cuando se ha llenado la FIFO (si está habilitada).

UnIER[1] habilita la interrupción THRE (*Transmitter Holding Register Empty*), que se activa no exista un dato a transmitir en UnTHR, es decir, cuando el registro transmisor esté vacío y listo para transmitir un nuevo dato. En el caso de estar habilitada la FIFO, este flag indica que ya se ha transmitido todo su contenido (máximo, 16 caracteres) y que está dispuesta para recibir más datos. El flag generado por esta interrupción se borra automáticamente cuando tiene lugar una escritura en UnTHR ó cuando se lee el registro UnIIR, siempre que la interrupción THRE sea la de mayor prioridad y esté activa.

Habilitar las interrupciones RLS (*Receive Line Status*) implica que cada vez que ocurra un error en la UART, se producirá una interrupción. Las posibles fuentes de error son:

- **Overrun Error** (OE): se da cuando llega un carácter a la UART, pero el registro UnRBR no ha sido leído con el dato anterior (es decir, contiene otro dato).
- **Parity Error** (PE): Error de paridad. Ocurre cuando el bit de paridad que es recibido (en el byte escrito en UnRBR) no es correcto.
- **Framing Error** (FE): El bit de stop es '0' en lugar de '1' (en el byte escrito en UnRBR). Tomará como válido el carácter, pero no el siguiente, esté bien o no.
- **Break Interrupt** (BI): Se da cuando todos los bits del carácter recibido son ceros (en el byte escrito en UnRBR).

Cuando se produzca una interrupción procedente de la UART, debemos leer el registro **UnIIR** (*Interrupt Identification Register*) para conocer a qué se ha debido esa interrupción:

- El bit UnIIR[0] se corresponde con el campo IntStatus, que de ser '1' indica que hay **alguna** interrupción de la UART activa o pendiente.
- Los bits UnIIR[3..1] (IntId) nos indican la fuente de **interrupción activa**:
 - **011**: RLS (*Receive Line Status*)
 - **010**: RDA (*Receive Data Available*)
 - **110**: CTI (*Character Time-out Indication*)
 - **001**: THRE (*Transmitter Holding Register Empty*)

- La interrupción CTI (*Character Time-out Indication*), se activa cuando no habiéndose llenado la FIFO, transcurre un tiempo superior a 1.5 veces la duración de un dato. Esto garantiza la lectura de los datos de la FIFO en caso de no alcanzar su nivel de llenado.
- Los bits UnIIR[6,7] (FIFO Enable) son copias de UnFCR[0], que indica si las FIFOs están habilitadas ('1') o no ('0'). Los campos UnIIR[8] (ABEOInt) y UnIIR[9] (ABTOInt) contienen los flags de las interrupciones ABEO y ABTO de configuración automática del *Baud Rate* respectivamente.

Ejemplo de lectura del registro U0IIR que se ha de hacer dentro de la función de interrupción para detectar cual ha sido la causa de la interrupción (sólo se consultan dos fuentes)

```
switch(LPC_UART0->IIR&0x0E) {  
  
    case 0x04:                                /* RBR, Receiver Buffer Ready */  
  
        break;  
  
    case 0x02:                                /* THRE, Transmit Holding Register empty */  
  
        break;  
  
    ...  
}
```

En el caso de que se produzca una interrupción RLS, la lectura de UnIIR no es suficiente para conocer el origen de la interrupción, por lo que se debe leer el registro de estado de línea ó **UnLSR**:

- El bit UnLSR[0] (Receptor de datos preparado, RDR), es '1' cuando a la FIFO aún le quedan caracteres por leer, y '0' cuando está vacía.
- Los bits **UnLSR[1..4]** hacen referencia a las posibles fuentes de error, que son **OE**, **PE**, **FE** y **BI** respectivamente.
- El bit UnLSR[5] se corresponde con el flag THRE y se pone a '1' cuando el registro UnTHR está vacío. Se borra al escribir en dicho registro. El bit UnLSR[6] (TEMT) se activa cuando UnTHR está vacío y además UnTSR también está vacío (UnTSR es el *Transmitter Shift Register*, ó registro de desplazamiento que saca el dato de la FIFO y lo manda al pin TxD).
- El bit UnLSR[7], que se encarga de mostrar que se ha producido uno de los posibles errores vistos (OE, PE, FE y BI). Será puesto a cero cuando se lee UnLSR y no hay ningún error activo.

4.4 Funcionamiento de la UART con FIFO

El control de las FIFOs se realiza en el registro UnFCR (*FIFO Control Register*). El bit UnFCR[0] (FIFO Enable) habilita ('1') o deshabilita ('0') las FIFOs. El bit UnFCR[1] (RX FIFO Reset) resetea la FIFO de recepción y el bit UnFCR[2] (TX FIFO Reset) la de transmisión de datos (todas las posiciones de las FIFOs son borradas).

Cuando UnFCR[3] (DMA Mode Select) está activado, el modo DMA está activo, de manera que se usa la DMA para recibir y transmitir datos con las UARTs.

En los bits UnFCR[6,7] (RX Trigger Level) se configura el número de caracteres que serán recibidos en la FIFO ("00": 1 carácter, "01": 4 caracteres, "10": 8 caracteres, "11": 14 caracteres), antes de que se produzca una interrupción o *DMA request* si se ha activado el modo DMA.

5 La UART1

La UART1 difiere en algunas características de la UART 0, 2 y 3. A continuación se exponen las principales diferencias:

- La UART1 dispone de un modo de comunicación con módem. Para ello dispone de un mecanismo de control denominado "apretón de manos" o *handshaking* que consiste en una comunicación previa entre UART1 y módem para establecer las condiciones de la comunicación que tendrá lugar para el intercambio de información.
- Soporta el protocolo de comunicaciones RS-485, también conocido como EIA-485.
- No soporta el protocolo de comunicación infrarroja IrDA.
- El control de flujo por software no está disponible.

Para usar la UART1, en primer lugar hay que alimentarla mediante el campo PCUART1 del registro PCON, aunque se habilita automáticamente tras un reset. Requiere de una entrada de reloj configurable en el campo PCLK_UART1 del registro PCLKSEL0. También es necesario configurar correctamente los pines que hagan falta mediante los registros PINSEL y PINMODE, así como las interrupciones (en NVIC y registros propios) y de ser necesario, la DMA. El propio dispositivo requiere de una correcta configuración de la velocidad de comunicación (*baud rate*) y de las FIFOs (de estar habilitadas).

El hecho de que sea posible establecer una comunicación mediante un módem, hace necesario que la UART uno disponga de los pines necesarios para la comunicación con este dispositivo:

- **Serial Input (RXD1) y Serial Output (TXD1):** Realizan la misma función que en el resto de las UARTs.
- **Clear To Send (CTS1):** Señal asíncrona de entrada activa a nivel bajo. Indica si el modem externo está preparado para aceptar el dato a transmitir desde la UART1.
- **Data Carrier Detect (DCD1):** Señal asíncrona de entrada activa a nivel bajo. Indica si el modem externo ha establecido una comunicación con la UART1.
- **Data Set Ready (DSR1):** Señal asíncrona de entrada activa a nivel bajo. Indica si el módem externo está preparado para establecer una comunicación con la UART1.
- **Data Terminal Ready (RTS1):** Señal asíncrona de salida activa a nivel bajo. Indica si el módem externo está preparado para establecer una comunicación con la UART1.
- **Ring Indicator (DTR1):** Señal asíncrona de entrada activa a nivel bajo. Indica una señal de llamada de teléfono ha sido detectada por el módem.
- **Request To Send (RTS1):** Señal asíncrona de salida activa a nivel bajo. Indica que a la UART1 le gustaría transmitir datos al módem externo. En modo auto-rts, este pin se usa para controlarla lógica de transmisión de la FIFO. Este pin puede ser también usado como señal de habilitación de salida del bus RS-485/EIA-485.

Todos los registros referentes a las UARTs 0, 2 y 3 están disponibles para la UART1 (salvo los que hacen referencia a las características que esta UART no posee). Por ello, el funcionamiento básico de la UART 1 es exactamente igual, y se controla de la misma manera que el resto de UARTs. Los nombres de los registros y su significado son idénticos, solo que referidos a la UART1. Sin embargo, al haber otros modos de funcionamiento habrá nuevos registros que controlen estas nuevas funcionalidades: comunicación con módem y comunicación con bus RS-485.

Para la comunicación con módem existen dos registros: el *Modem Control Register* (U1MCR) y el *Modem Status Register* (U1MSR).

El U1MCR sirve para habilitar el modo *loopback modem* que consiste en enviar un dato al módem para que le sea devuelto de nuevo. El objetivo es comprobar que el dato devuelto sea igual que el enviado. De esta forma se pueden localizar errores de

comunicación. Además, se puede configurar el modo con el que la UART 1 se comunicará con el módem, habiendo 2 posibles modos: auto-RTS y auto-CTS.

El U1MSR es un registro de sólo lectura que sirve para proporcionar información acerca del estado de las señales de entrada del módem. Como estas señales del módem no tienen efectos inmediatos en la UART1, se facilita la implementación software de las operaciones con el módem.

Para la comunicación a través del protocolo RS-485 se usan tres registros: el *UART1 RS485 Control Register* (U1RS485CTRL), el *UART1 RS485 Address Match* (U1RS485ADRMATCH) y el *UART1 RS485 Delay Value Register* (U1RS485DLY). En el U1RS485CTRL se configura el modo de intercambio de datos mediante el protocolo RS-485/EIA-485. Existen tres posibles modos de comunicación: *Normal Multidrop mode* (NMM), *Auto Address Detection mode* (AAD) y *Auto Direction Control*.

El registro U1RS485ADRMATCH se guarda la dirección de dispositivo que tomará el LPC1768 durante la comunicación con el protocolo RS-485. En el registro U1RS485DLY puede configurarse un retardo entre la transmisión de distintas tramas de datos.

La configuración de ambas interfaces (para comunicación con módem y protocolo RS-485) es algo compleja. Para ver información más detallada de cómo llevar a cabo estas comunicaciones, revisar el capítulo 15 del *datasheet*.

En la figura 5.1 puede verse un diagrama de bloques funcional de la UART1, en el que pueden apreciarse bastantes similitudes con el de la UART 0, 2 y 3.

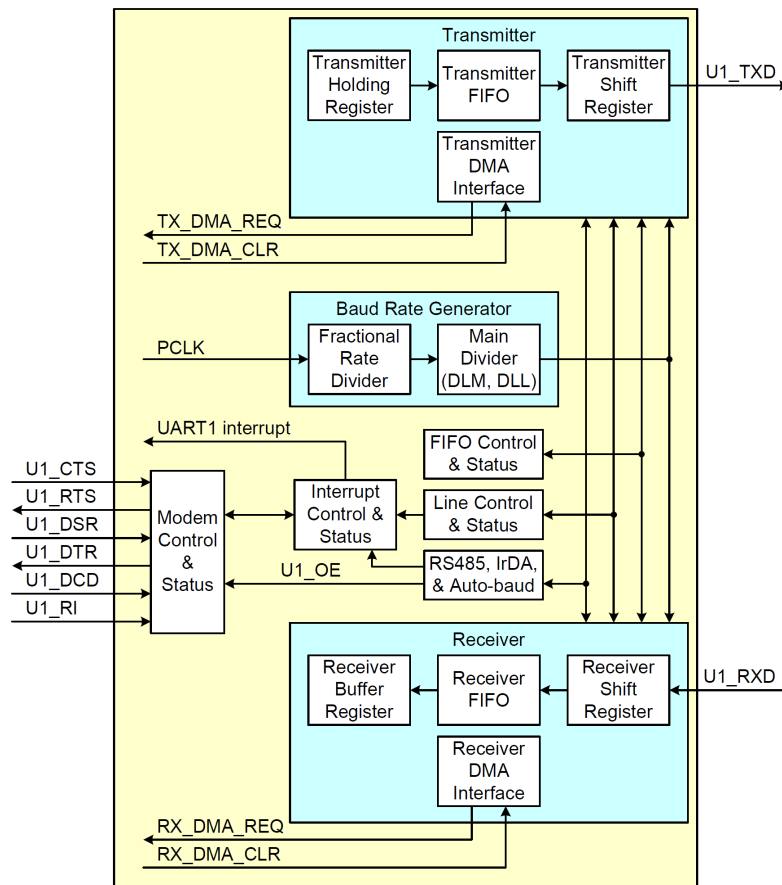


Fig. 5.1. Diagrama de bloques funcional de la UART1

Para programar la UART es necesario crear al menos dos funciones: una función de configuración y la función de interrupción que gestionará el proceso de recepción y transmisión de los datos. En la función de configuración se han de programar los parámetros relacionados con el formato deseado de la comunicación, si se desea o no habilitar la FIFO y sus parámetros de configuración, la habilitación de las interrupciones, y su prioridad en caso de ser necesario.

6 Registros de configuración y control de la UART

A continuación se enumeran los registros más importantes necesarios para programar las UARTs 0, 2 y 3 con objeto de configurar y gestionar el proceso de transferencia de datos en transmisión y recepción. Se remarcan en rojo los más importantes en la tabla 270, y se ven en detalle cada uno de ellos de manera individual mostrando sus bits de configuración y control a partir de las tablas obtenidas del manual de referencia. Ya se comentó anteriormente que la UART1 tiene idénticos registros, y otra serie de ellos asociados con el la función de MODEM.

Table 270. 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

Registro de activación de la alimentación de los periféricos.**Table 46. Power Control for Peripherals register (PCONP - address 0x400F C0C4) bit description**

Bit	Symbol	Description	Reset value
0	-	Reserved.	NA
1	PCTIM0	Timer/Counter 0 power/clock control bit.	1
2	PCTIM1	Timer/Counter 1 power/clock control bit.	1
3	PCUART0	UART0 power/clock control bit.	1
4	PCUART1	UART1 power/clock control bit.	1
5	-	Reserved.	NA
6	PCPWM1	PWM1 power/clock control bit.	1
7	PCI2C0	The I ² C0 interface power/clock control bit.	1
8	PCSPI	The SPI interface power/clock control bit.	1
9	PCRTC	The RTC power/clock control bit.	1
10	PCSSP1	The SSP 1 interface power/clock control bit.	1
11	-	Reserved.	NA
12	PCADC	A/D converter (ADC) power/clock control bit. Note: Clear the PDN bit in the AD0CR before clearing this bit, and set this bit before setting PDN.	0
13	PCCAN1	CAN Controller 1 power/clock control bit.	0
14	PCCAN2	CAN Controller 2 power/clock control bit.	0
15	PCGPIO	Power/clock control bit for IOCON, GPIO, and GPIO interrupts.	1
16	PCRIT	Repetitive Interrupt Timer power/clock control bit.	0
17	PCMCPWM	Motor Control PWM	0
18	PCQEI	Quadrature Encoder Interface power/clock control bit.	0
19	PCI2C1	The I ² C1 interface power/clock control bit.	1
20	-	Reserved.	NA
21	PCSSP0	The SSP0 interface power/clock control bit.	1
22	PCTIM2	Timer 2 power/clock control bit.	0
23	PCTIM3	Timer 3 power/clock control bit.	0
24	PCUART2	UART 2 power/clock control bit.	0
25	PCUART3	UART 3 power/clock control bit.	0
26	PCI2C2	I ² C interface 2 power/clock control bit.	1
27	PCI2S	I ² S interface power/clock control bit.	0
28	-	Reserved.	NA
29	PCGPDMA	GPDMA function power/clock control bit.	0
30	PCENET	Ethernet block power/clock control bit.	0
31	PCUSB	USB interface power/clock control bit.	0

Registro de configuración de la funcionalidad de los pines

Table 79. Pin function select register 0 (PINSEL0 - address 0x4002 C000) bit description

PINSEL0	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P0.0	GPIO Port 0.0	RD1	TXD3	SDA1	00
3:2	P0.1	GPIO Port 0.1	TD1	RXD3	SCL1	00
5:4	P0.2	GPIO Port 0.2	TXD0	AD0.7	Reserved	00
7:6	P0.3	GPIO Port 0.3	RXD0	AD0.6	Reserved	00
9:8	P0.4 ^[1]	GPIO Port 0.4	I2SRX_CLK	RD2	CAP2.0	00
11:10	P0.5 ^[1]	GPIO Port 0.5	I2SRX_WS	TD2	CAP2.1	00
13:12	P0.6	GPIO Port 0.6	I2SRX_SDA	SSEL1	MAT2.0	00
15:14	P0.7	GPIO Port 0.7	I2STX_CLK	SCK1	MAT2.1	00
17:16	P0.8	GPIO Port 0.8	I2STX_WS	MISO1	MAT2.2	00
19:18	P0.9	GPIO Port 0.9	I2STX_SDA	MOSI1	MAT2.3	00
21:20	P0.10	GPIO Port 0.10	TXD2	SDA2	MAT3.0	00
23:22	P0.11	GPIO Port 0.11	RXD2	SCL2	MAT3.1	00
29:24	-	Reserved	Reserved	Reserved	Reserved	0
31:30	P0.15	GPIO Port 0.15	TXD1	SCK0	SCK	00

Registro de configuración del formato de la transmisión

Table 279: 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 U0LCR[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 U0LCR[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

Registros de transmisión y recepción de un dato

Table 271: UARTn Receiver Buffer Register (U0RBR - address 0x4000 C000, U2RBR - 0x4009 8000, U3RBR - 0x4009 C000 when DLAB = 0) bit description

Bit	Symbol	Description	Reset Value
7:0	RBR	The UARTn Receiver Buffer Register contains the oldest received byte in the UARTn Rx FIFO.	Undefined
31:8	-	Reserved, the value read from a reserved bit is not defined.	NA

Table 272: UARTn Transmit Holding Register (U0THR - address 0x4000 C000, U2THR - 0x4009 8000, U3THR - 0x4009 C000 when DLAB = 0) bit description

Bit	Symbol	Description	Reset Value
7:0	THR	Writing to the UARTn Transmit Holding Register causes the data to be stored in the UARTn transmit FIFO. The byte will be sent when it reaches the bottom of the FIFO and the transmitter is available.	NA
31:8	-	Reserved, user software should not write ones to reserved bits.	NA

Registros de configuración del divisor de velocidad de transmisión

Table 273: UARTn Divisor Latch LSB register (U0DLL - address 0x4000 C000, U2DLL - 0x4009 8000, U3DLL - 0x4009 C000 when DLAB = 1) bit description

Bit	Symbol	Description	Reset Value
7:0	DLLSB	The UARTn Divisor Latch LSB Register, along with the UnDLM register, determines the baud rate of the UARTn.	0x01
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 274: UARTn Divisor Latch MSB register (U0DLM - address 0x4000 C004, U2DLM - 0x4009 8004, U3DLM - 0x4009 C004 when DLAB = 1) bit description

Bit	Symbol	Description	Reset Value
7:0	DLMSB	The UARTn Divisor Latch MSB Register, along with the U0DLL register, determines the baud rate of the UARTn.	0x00
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

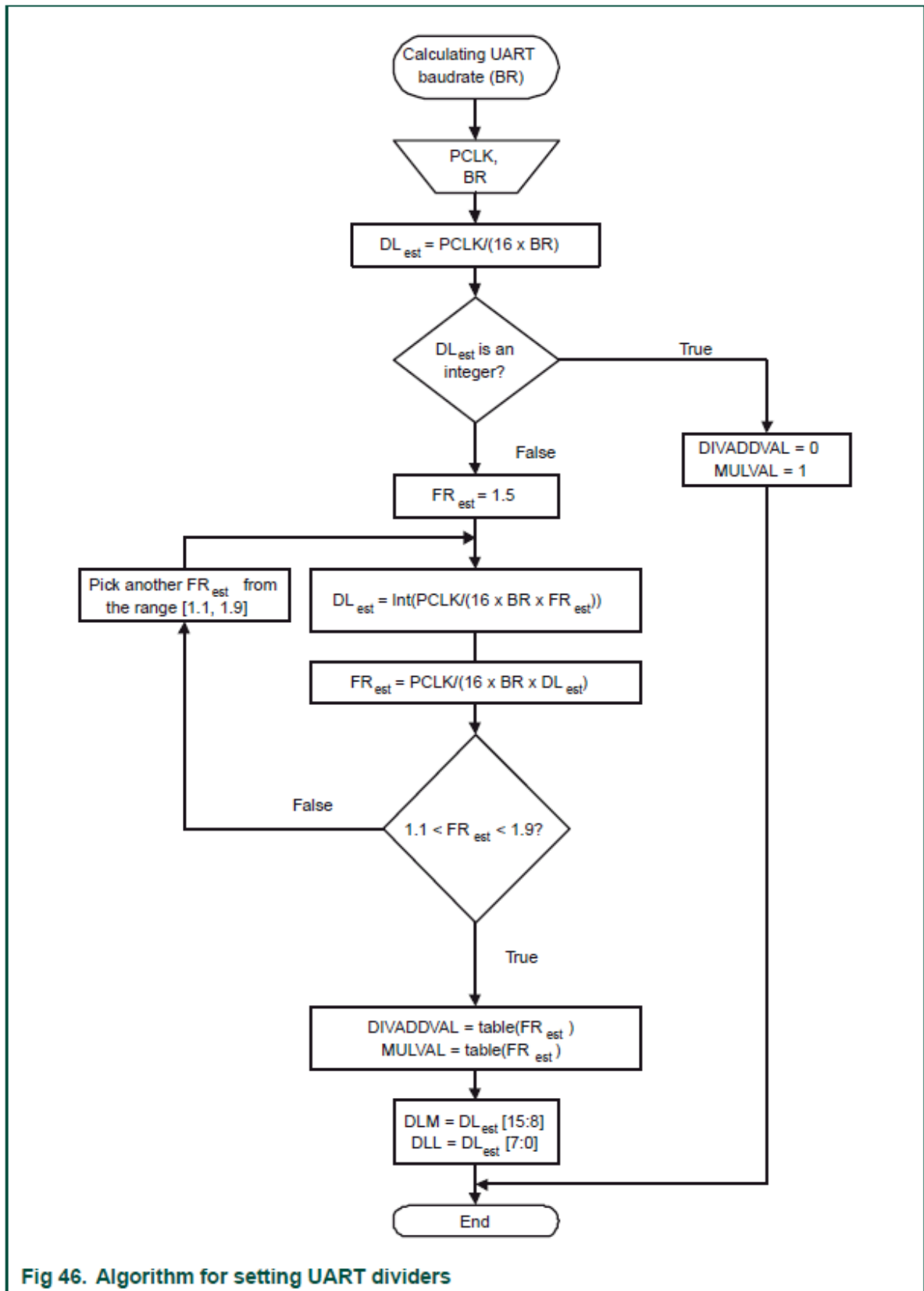


Table 286. Fractional Divider setting look-up table

FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal
1.000	0/1	1.250	1/4	1.500	1/2	1.750	3/4
1.067	1/15	1.267	4/15	1.533	8/15	1.769	10/13
1.071	1/14	1.273	3/11	1.538	7/13	1.778	7/9
1.077	1/13	1.286	2/7	1.545	6/11	1.786	11/14
1.083	1/12	1.300	3/10	1.556	5/9	1.800	4/5
1.091	1/11	1.308	4/13	1.571	4/7	1.818	9/11
1.100	1/10	1.333	1/3	1.583	7/12	1.833	5/6
1.111	1/9	1.357	5/14	1.600	3/5	1.846	11/13
1.125	1/8	1.364	4/11	1.615	8/13	1.857	6/7
1.133	2/15	1.375	3/8	1.625	5/8	1.867	13/15
1.143	1/7	1.385	5/13	1.636	7/11	1.875	7/8
1.154	2/13	1.400	2/5	1.643	9/14	1.889	8/9
1.167	1/6	1.417	5/12	1.667	2/3	1.900	9/10
1.182	2/11	1.429	3/7	1.692	9/13	1.909	10/11
1.200	1/5	1.444	4/9	1.700	7/10	1.917	11/12
1.214	3/14	1.455	5/11	1.714	5/7	1.923	12/13
1.222	2/9	1.462	6/13	1.727	8/11	1.929	13/14
1.231	3/13	1.467	7/15	1.733	11/15	1.933	14/15

Registro de configuración del divisor fraccional de la velocidad**Table 285: UARTn Fractional Divider Register (U0FDR - address 0x4000 C028, U2FDR - 0x4009 8028, U3FDR - 0x4009 C028) 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 baudrate.	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	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

Registro de control y configuración de la FIFO

Table 278: UARTn FIFO Control Register (U0FCR - address 0x4000 C008, U2FCR - 0x4009 8008, U3FCR - 0x4007 C008) bit description

Bit	Symbol	Value	Description	Reset Value
0	FIFO Enable	0	UARTn FIFOs are disabled. Must not be used in the application.	0
		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	RX FIFO Reset	0	No impact on either of UARTn FIFOs.	0
		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	TX FIFO Reset	0	No impact on either of UARTn FIFOs.	0
		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	DMA Mode Select		When the FIFO enable bit (bit 0 of this register) is set, this bit selects the DMA mode. See Section 14.4.6.1 .	0
5:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	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
		00	Trigger level 0 (1 character or 0x01)	
		01	Trigger level 1 (4 characters or 0x04)	
		10	Trigger level 2 (8 characters or 0x08)	
		11	Trigger level 3 (14 characters or 0x0E)	
31:8	-	-	Reserved, user software should not write ones to reserved bits.	NA

Registro de habilitación de interrupciones

Table 275: UARTn Interrupt Enable Register (U0IER - address 0x4000 C004, U2IER - 0x4009 8004, U3IER - 0x4009 C004 when DLAB = 0) bit description

Bit	Symbol	Value	Description	Reset Value
0	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	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	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, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	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, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Registro de identificación de la fuente de interrupción

Table 276: UARTn Interrupt Identification Register (U0IIR - address 0x4000 C008, U2IIR - 0x4009 8008, U3IIR - 0x4009 C008) bit description

Bit	Symbol	Value	Description	Reset Value
0	IntStatus		Interrupt status. Note that UnIIR[0] is active low. The pending interrupt can be determined by evaluating UnIIR[3:1].	1
		0	At least one interrupt is pending.	
		1	No interrupt is pending.	
3:1	IntId		Interrupt identification. UnIER[3:1] identifies an interrupt corresponding to the UARTn Rx or TX FIFO. All other combinations of UnIER[3:1] not listed below are reserved (000,100,101,111).	0
		011	1 - Receive Line Status (RLS).	
		010	2a - Receive Data Available (RDA).	
		110	2b - Character Time-out Indicator (CTI).	
		001	3 - THRE Interrupt	
5:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	FIFO Enable		Copies of UnFCR[0].	0
8	ABEOInt		End of auto-baud interrupt. True if auto-baud has finished successfully and interrupt is enabled.	0
9	ABTOInt		Auto-baud time-out interrupt. True if auto-baud has timed out and interrupt is enabled.	0
31:10	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Registro de identificación de fuentes de error

Table 280: 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.	
7	Error in RX FIFO (RXFE)		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

Registro de configuración del reloj de los periféricos

Table 40. Peripheral Clock Selection register 0 (PCLKSEL0 - address 0x400F C1A8) bit description

Bit	Symbol	Description	Reset value
1:0	PCLK_WDT	Peripheral clock selection for WDT.	00
3:2	PCLK_TIMER0	Peripheral clock selection for TIMER0.	00
5:4	PCLK_TIMER1	Peripheral clock selection for TIMER1.	00
7:6	PCLK_UART0	Peripheral clock selection for UART0.	00
9:8	PCLK_UART1	Peripheral clock selection for UART1.	00
11:10	-	Reserved.	NA
13:12	PCLK_PWM1	Peripheral clock selection for PWM1.	00
15:14	PCLK_I2C0	Peripheral clock selection for I ² C0.	00
17:16	PCLK_SPI	Peripheral clock selection for SPI.	00
19:18	-	Reserved.	NA
21:20	PCLK_SSP1	Peripheral clock selection for SSP1.	00
23:22	PCLK_DAC	Peripheral clock selection for DAC.	00
25:24	PCLK_ADC	Peripheral clock selection for ADC.	00
27:26	PCLK_CAN1	Peripheral clock selection for CAN1. [1]	00
29:28	PCLK_CAN2	Peripheral clock selection for CAN2. [1]	00
31:30	PCLK_ACF	Peripheral clock selection for CAN acceptance filtering. [1]	00

PCLKSEL0 and PCLKSEL1 individual peripheral's clock select options	Function	Reset value
00	PCLK_peripheral = CCLK/4	00
01	PCLK_peripheral = CCLK	
10	PCLK_peripheral = CCLK/2	
11	PCLK_peripheral = CCLK/8, except for CAN1, CAN2, and CAN filtering when "11" selects = CCLK/6.	