# *OMAP5910 Dual-Core Processor*
# *Universal Asynchronous Receiver/Transmitter (UART) Devices*
# *Reference Guide*

Literature Number: SPRU676
October 2003

OMAP™
TEXAS INSTRUMENTS TECHNOLOGY

TEXAS INSTRUMENTS

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

| **Products** | | **Applications** | |
|---|---|---|---|
| Amplifiers | amplifier.ti.com | Audio | www.ti.com/audio |
| Data Converters | dataconverter.ti.com | Automotive | www.ti.com/automotive |
| DSP | dsp.ti.com | Broadband | www.ti.com/broadband |
| Interface | interface.ti.com | Digital Control | www.ti.com/digitalcontrol |
| Logic | logic.ti.com | Military | www.ti.com/military |
| Power Mgmt | power.ti.com | Optical Networking | www.ti.com/opticalnetwork |
| Microcontrollers | microcontroller.ti.com | Security | www.ti.com/security |
| | | Telephony | www.ti.com/telephony |
| | | Video & Imaging | www.ti.com/video |
| | | Wireless | www.ti.com/wireless |

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265

Copyright © 2003, Texas Instruments Incorporated

# Preface

# Read This First

## About This Manual

This document describes the three universal asynchronous receiver/transmitter (UART) devices in the OMAP5910 multimedia processor.

## Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.

## Related Documentation From Texas Instruments

The following documents describe the OMAP5910 device and related peripherals. Copies of these documents are available on the Internet at www.ti.com. *Tip:* Enter the literature number in the search box provided at www.ti.com.

***OMAP5910 Dual-Core Processor MPU Subsystem Reference Guide*** (literature number SPRU671)

***OMAP5910 Dual-Core Processor DSP Subsystem Reference Guide*** (literature number SPRU672)

***OMAP5910 Dual-Core Processor Memory Interface Traffic Controller Reference Guide*** (literature number SPRU673)

***OMAP5910 Dual-Core Processor System DMA Controller Reference Guide*** (literature number SPRU674)

***OMAP5910 Dual-Core Processor LCD Controller Reference Guide*** (literature number SPRU675)

***OMAP5910 Dual-Core Processor Universal Asynchronous Receiver/Transmitter (UART) Devices Reference Guide*** (literature number SPRU676)

***OMAP5910 Dual-Core Processor Universal Serial Bus (USB) and Frame Adjustment Counter (FAC) Reference Guide*** (literature number SPRU677)

***OMAP5910 Dual-Core Processor Clock Generation and System Reset Management Reference Guide*** (literature number SPRU678)

***OMAP5910 Dual-Core Processor General-Purpose Input/Output (GPIO) Reference Guide*** (literature number SPRU679)

***OMAP5910 Dual-Core Processor MMC/SD Reference Guide*** (literature number SPRU680)

***OMAP5910 Dual-Core Processor Inter-Integrated Circuit (I2C) Controller Reference Guide*** (literature number SPRU681)

***OMAP5910 Dual-Core Processor Timer Reference Guide*** (literature number SPRU682)

***OMAP5910 Dual-Core Processor Inter-Processor Communication Reference Guide*** (literature number SPRU683)

***OMAP5910 Dual-Core Processor Camera Interface Reference Guide*** (literature number SPRU684)

***OMAP5905 Dual-Core Processor Multichannel Serial Interface (MCSI) Reference Guide*** (literature number SPRU685)

***OMAP5910 Dual-Core Processor Micro-Wire Interface Reference Guide*** (literature number SPRU686)

***OMAP5910 Dual-Core Processor Real-Time Clock (RTC) Reference Guide*** (literature number SPRU687)

***OMAP5910 Dual-Core Processor HDQ/1-Wire Interface Reference Guide*** (literature number SPRU688)

***OMAP5910 Dual-Core Processor PWL, PWT, and LED Peripheral Reference Guide*** (literature number SPRU689)

***OMAP5910 Dual-Core Processor Multichannel Buffered Serial Port (McBSP) Reference Guide*** (literature number SPRU708)

## *Trademarks*

OMAP and the OMAP symbol are trademarks of Texas Instruments.

# Contents

# Figures

# Tables

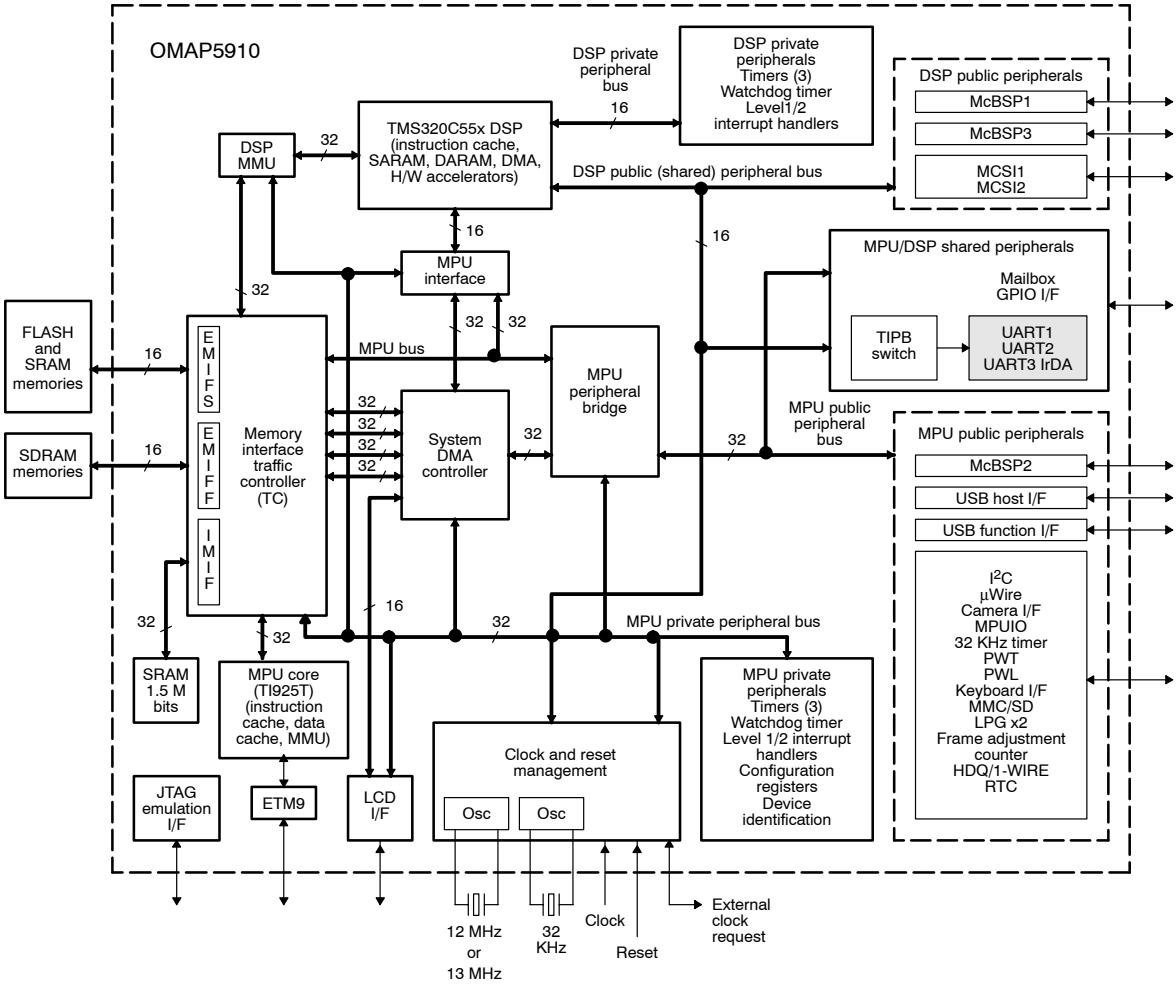
# UART Peripherals

This document describes the three universal asynchronous receiver/transmitter (UART) peripherals in the OMAP5910 multimedia processor.

## 1 UART Introduction

The OMAP5910 multimedia processor contains three universal asynchronous receiver/transmitter (UART) peripherals. UART1 and UART2 are UART modems with autobaud capability. UART3 is a modem with IrDA. Either the MPU (default) or the DSP controls the three UARTs via three TIPB switches (one for each UART). Figure 1 shows the OMAP5910 device with the UART peripherals highlighted.

*Figure 1. UART Modem Module*



## 1.1 Main UART Features (UART1/2/3)

The main features are as follows:

- ❏ Selectable UART/autobaud modes (UART1 and 2 only)
- ❏ Dual 64-entry FIFOs for received and transmitted data payloads
- ❏ Programmable and selectable transmit and receive FIFO trigger levels for DMA and interrupt generation
- ❏ Programmable sleep mode
- ❏ Complete status reporting capabilities in both normal and sleep mode
- ❏ Frequency prescaler values from 0 to 65535 to generate the appropriate baud rates
- ❏ An interrupt request to the system if there are multiple DMA requests

### 1.1.1 UART/Modem Functions (UART1/2/3)

- ❏ Baud rate from 300 bits/s up to 1.5M bits/s
- ❏ Autobaud between 1200 bits/s and 115.2K bits/s
- ❏ Software/hardware flow control
  - ■ Programmable XON/XOFF characters
  - ■ Programmable AUTO_RTS and AUTO_CTS
- ❏ Programmable serial interface characteristics
  - ■ 5-, 6-, 7-, or 8-bit characters
  - ■ Even-, odd-, or no-parity bit generation and detection
  - ■ 1, 1.5, or 2 stop bit generation
- ❏ False start bit detection
- ❏ Line break generation and detection
- ❏ Fully prioritized interrupt system controls
- ❏ Internal test and loopback capabilities

Modem control functions ($\overline{\text{CTS}}$, $\overline{\text{RTS}}$, $\overline{\text{DSR}}$, and $\overline{\text{DTR}}$)

### 1.1.2 IrDA Functions (UART3 Only)

- ❏ Slow infrared (SIR) operations
- ❏ Framing error, cyclic redundancy check (CRC) error, and abort pattern (SIR) detection

- An 8-entry status FIFO (with selectable trigger levels) available to monitor the frame length and frame errors

Table 1 describes the I/O module at the module level.

### 1.1.3 UART Signals

The signals available on the UART modules are illustrated in Figure 2. These signals are described in Table 1.

*Figure 2. UART Signals*

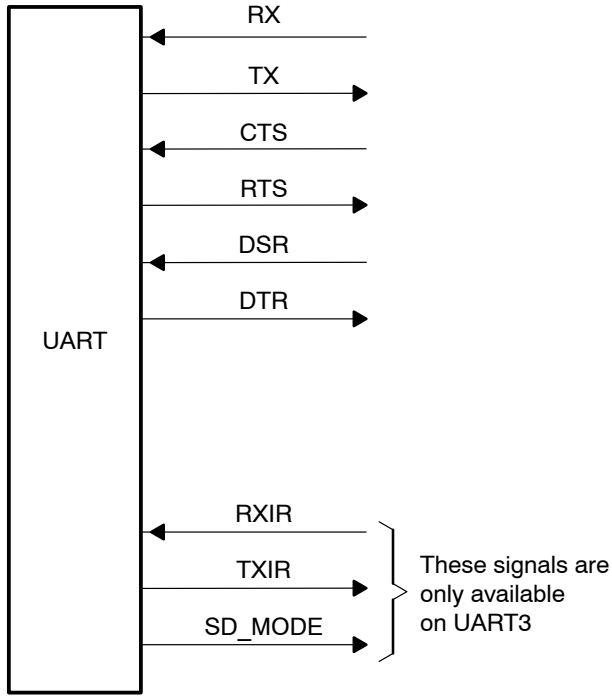


*Table 1. I/O Description*

| Signal | I/O | Description | Reset Value |
|---|---|---|---|
| **UART/MODEM Signals** | | | |
| RX | I | Serial data input | – |
| TX | O | Serial data output | 1 |
| $\overline{\text{CTS}}$ | I | Clear to send.<br>Active-low modem status signal. Reading bit 4 of the modem status register checks the condition of $\overline{\text{CTS}}$. Reading bit 0 of that register checks for a change of state of $\overline{\text{CTS}}$ since the last read of the modem status register. $\overline{\text{CTS}}$ is used in automatic CTS mode to control the transmitter. | – |
| $\overline{\text{RTS}}$ | O | Request to send.<br>When active (low), the module is ready to receive data. Setting the modem control register bit 1 activates $\overline{\text{RTS}}$. It becomes inactive as a result of a module reset, loop back mode or by clearing the MCR1. In automatic RTS mode, it becomes inactive as a result of the receiver threshold logic. | 1 |
| $\overline{\text{DSR}}$ | I | Data set ready<br>Active-low modem status signal. Reading bit 5 of the modem status register checks the condition of $\overline{\text{DSR}}$. Reading bit 1 of that register checks for a change of state of $\overline{\text{DSR}}$ since the last read of the modem status register. | – |
| $\overline{\text{DTR}}$ | O | Data transmit ready<br>Active-low modem control signal. Reading bit 0 of the modem control register checks the condition of $\overline{\text{DTR}}$. | 1 |
| **IrDA Signals (UART3 Only)** | | | |
| RXIR | I | Serial data input | – |
| TXIR | O | Serial data output | 0 |
| SD_MODE | O | Signal used to configure transceivers | 1 |

# 2 UART Environments

Each UART is controllable through a TIPB switch, either by the MPU (default) or the DSP.

## 2.1 UART1 Environment

UART1 is a UART modem with autobaud capability. Table 2 lists the UART1 modem signals accessible at the OMAP5910 level.

*Table 2. Available UART1 Signals*

| Generic UART Signal Name | Description | UART1 Signal Name |
|---|---|---|
| RX | Serial data input | UART1.RX |
| TX | Serial data output | UART1.TX |
| CTS | Clear to send input | UART1.CTS |
| RTS | Request to send input | UART1.RTS |
| DTR | Data transmit ready output | UART1.DTR |
| DSR | Data set ready input | UART1.DSR |

The functional clock is either a 12-MHz or a 48-MHz clock. The desired clock can be selected with the CONF_MOD_UART1_CLK_MODE_R bit (29) of the MOD_CONF_CTRL_0 register:

- ❏ CONF_MOD_UART1_CLK_MODE_R = 0: 12 MHz (default)
- ❏ CONF_MOD_UART1_CLK_MODE_R = 1: 48 MHz

NDMA_REQ [1:0] are connected to the DMA request [13:12] of both the MPU system DMA controller and the DSP DMA controller.

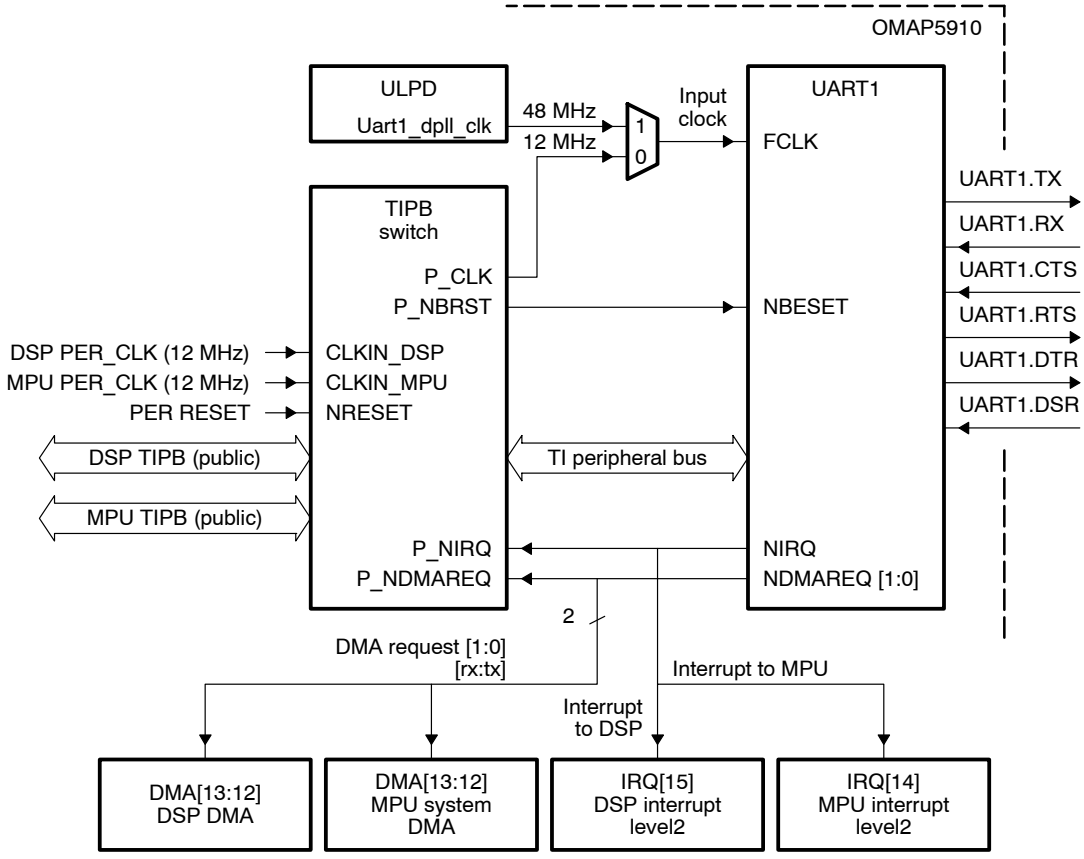
NDMA_REQ[1] is a RX request, and NDMA_REQ[0] is a TX request.

NIRQ from UART1 is connected to:

- ❏ The interrupt line IRQ[14] of the MPU level 2 interrupt handler
- ❏ The interrupt line IRQ[5] of the DSP level 2 interrupt handler

Figure 3 shows the UART1 environment.

*Figure 3. UART1 Environment*



## 2.2 UART2 Environment

UART2 is an UART modem with autobaud capability.

Table 3 lists the UART2 modem signals accessible at the OMAP5910 level.

*Table 3. Available UART2 Signals*

| Generic UART Signal Name | Description | UART2 Signal Name |
|---|---|---|
| RX | Serial data input | RX2 |
| TX | Serial data output | TX2 |
| CTS | Clear to send input | CTS2 |
| RTS | Request to send input | RTS2 |
| FSR | Receive frame (input only) | Not available (internal feedback from FSX) |
| BDCLK | 16x baud clock input | BDCLK2 |

The functional clock is either a 32-kHz/12-MHz or a 48-MHz clock. The desired clock can be selected with the CONF_MOD_UART2_CLK_MODE_R bit (30) of the MOD_CONF_CTRL_0 register as follows:

- CONF_MOD_UART2_CLK_MODE_R = 0: 32 kHz/12 MHz (default)
- CONF_MOD_UART2_CLK_MODE_R = 1: 48 MHz

The frequency of the 32-kHz/12-MHz clock depends on the OMAP5910 system state:

- 32 kHz in deep sleep modes
- 12 MHz in big sleep and awake mode

Note that the UPLD clock control register (CLOCK_CTRL_REG) bit 0 MODEM_32K_EN must be controlled as follows:

- MODEM_32K_EN = 0: Disables 32-kHz on the UART clock when in deep sleep
- MODEM_32K_EN = 1: Enables 32-kHz on the UART clock when in deep sleep

The reset condition is 0.

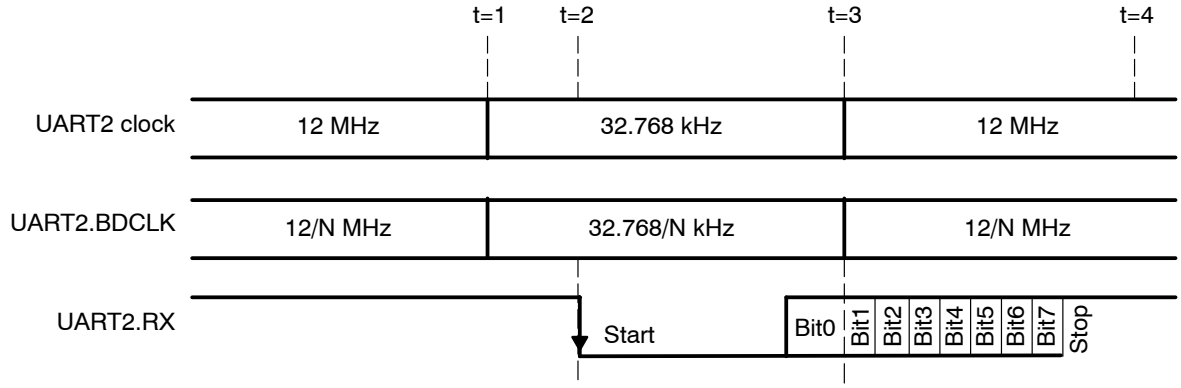
When the CONF_MOD_UART2_CLK_MODE_R = 0 and MODEM_32K_ EN = 1, the UART2 operates at 32kHz input clock while the device is in deep sleep.

UART2.BDCLK (UART2 baud clock) automatically switches to a lower frequency in deep sleep based on the 12MHz to 32kHz clock switch in the ULPD.

Putting the baud clock outside of the device enables an external UART to use this baud clock to remain synchronized even while the OMAP5910 device is in deep sleep. If the external UART sends a byte when the OMAP5910 device is in deep sleep, it is received by UART2 correctly without loss of data, although at a slow rate. The activity detection circuit monitors UART2.RX activity using the UART2.BDCLK clock and requests the ULPD to wakeup by the periph_clk_nreq signal. The activity detection logic uses a two-out-of-three voting logic. The following sampled combinations of UART2.RX will produce a periph_clk_nreq: 001, 010, and 100.

Figure 4 shows the sequence of the wakeup by UART2.RX.

*Figure 4. UART2.RX Wakeup Sequence*




- T = #1: The MPU goes to standby and enters “Deep Sleep” state. The 12MHz osc is turned off. The Baud clock automatically switches to 32,768 / N. ( N = Uart Div ratio)
- T = #2: A falling edge on UART2.RX is sensed and causes wake up of the 12MHz OSC, the wakeup time depends primarily on the analog wait timer (12MHz osc start up delay).
- T = #3: The ULPD transitions to “Awake” mode after 12MHz is stable. The UART2 Baud clock will switch back to 12/N MHz. Note the MPU is still sleeping now waiting for an interrupt to wakeup.
- T = #4: There are two ways the MPU wakes up:
  - RHR interrupt
  - RX Timeout interrupt (shown in Figure 4): When UART2.RX has been high for a time equivalent to (4 * Programmed word length + 12bits) / baud rate, the RX Timeout interrupt occurs and wakes up the MPU.

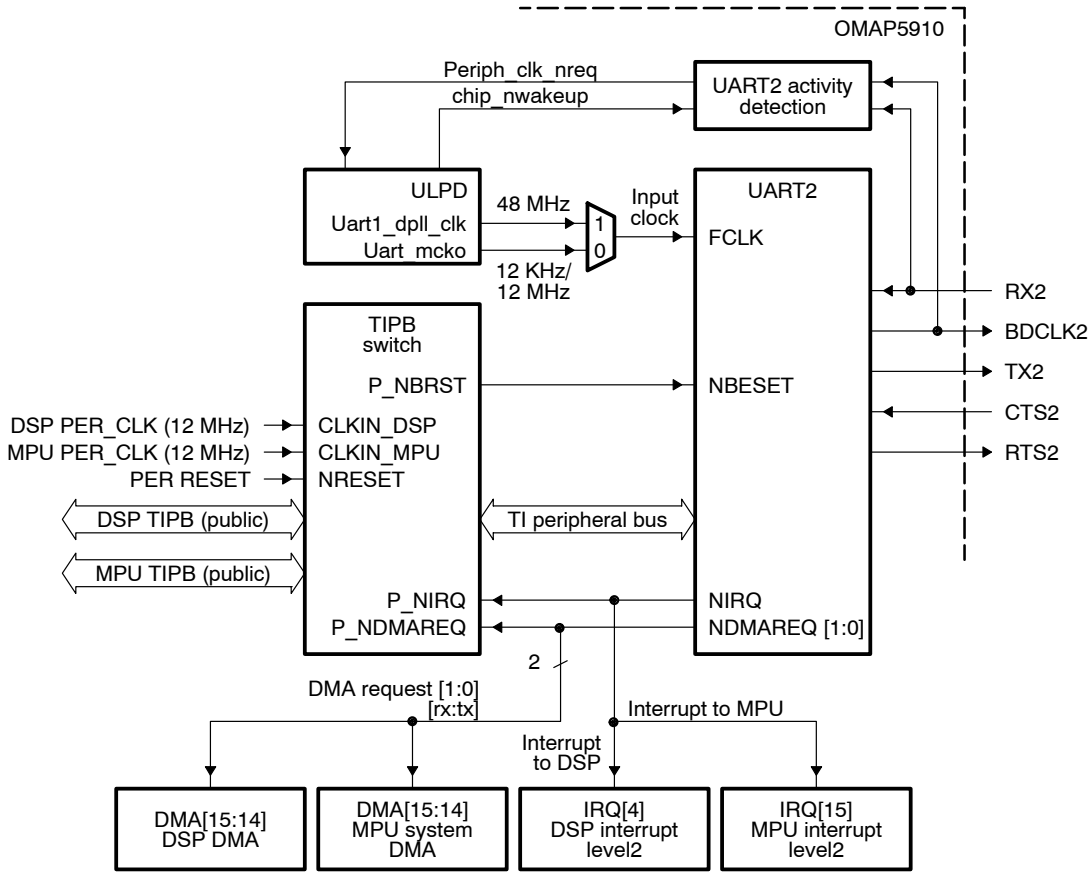
There is an alternate wakeup from deep sleep: By programming bit 4 of the UART2 SCR register, it is possible to create an interrupt by an edge going low on RX or CTS. The interrupt would be generated by RX at time #2 in Figure 4.

NDMA_REQ [1:0] are connected to the DMA request [15:14] of both the MPU system DMA controller and the DSP DMA controller. NDMA_REQ[1] is a RX request, and the NDMA_REQ[0] is a TX request. The NIRQ from UART2 is connected to the following:

- Interrupt line IRQ[15] of the MPU level 2 interrupt handler
- Interrupt line IRQ[4] of the DSP level 2 interrupt handler

Figure 5 shows the UART2 environment.

*Figure 5. UART2 Environment*



## 2.3 UART3 Environment

The UART3 is a UART modem with IrDA capability.

The IrDA mode (SIR) is selectable by setting the MODE_SELECT bits (2:0) of the UART3 MDRI register to 001. Set the IRDA_SELECT signal to 1.

The IRDA_SELECT signal can be used to control the multiplexing on the UART3 I/Os between the UART3 modem signals and the UART3 IrDA signals.

Table 4 lists the UART3 IrDA signals accessible at the OMAP5910 level when IRDA_SELECT = 1.

*Table 4. Available UART3 Signals in IrDA = 1 Mode*

| Generic UART Signal Name | Description | UART3 Signal Name |
|---|---|---|
| TXIR | IrDA serial data input | TX3 |
| RXIR | IrDA serial data output | UART3.RX |
| RX | Serial data input | High |
| SD_MODE | Signal used to configure transceivers | RTS3 |

Table 5 lists the UART3 IrDA signals accessible at the OMAP5910 level when IRDA_SELECT = 0.

*Table 5. Available UART3 Signals in IrDA = 0 Mode*

| Generic UART Signal Name | Description | UART3 Signal Name |
|---|---|---|
| TX | Serial data output | TX3 |
| RX | Serial data input | UART3.RX |
| RXIR | IrDA serial data input | High |
| RTS | Request to send output | RTS3 |
| CTS | Clear to send input | CTS3 |
| DTR | Data transmit ready output | DTR3 |
| DSR | Data set ready input | DSR3 |

The functional clock is either a 12-MHz or a 48-MHz clock. The desired clock can be selected with the CONF_MOD_UART3_CLK_MODE_R bit (30) of the MOD_CONF_CTRL_0 register as follows:

- ❏ CONF_MOD_UART3_CLK_MODE_R = 0: 12 MHz (default)
- ❏ CONF_MOD_UART3_CLK_MODE_R = 1: 48 MHz

The NDMA_REQ [1:0] are connected to DMA request [19:18] of both the MPU system DMA controller and the DSP DMA controller.

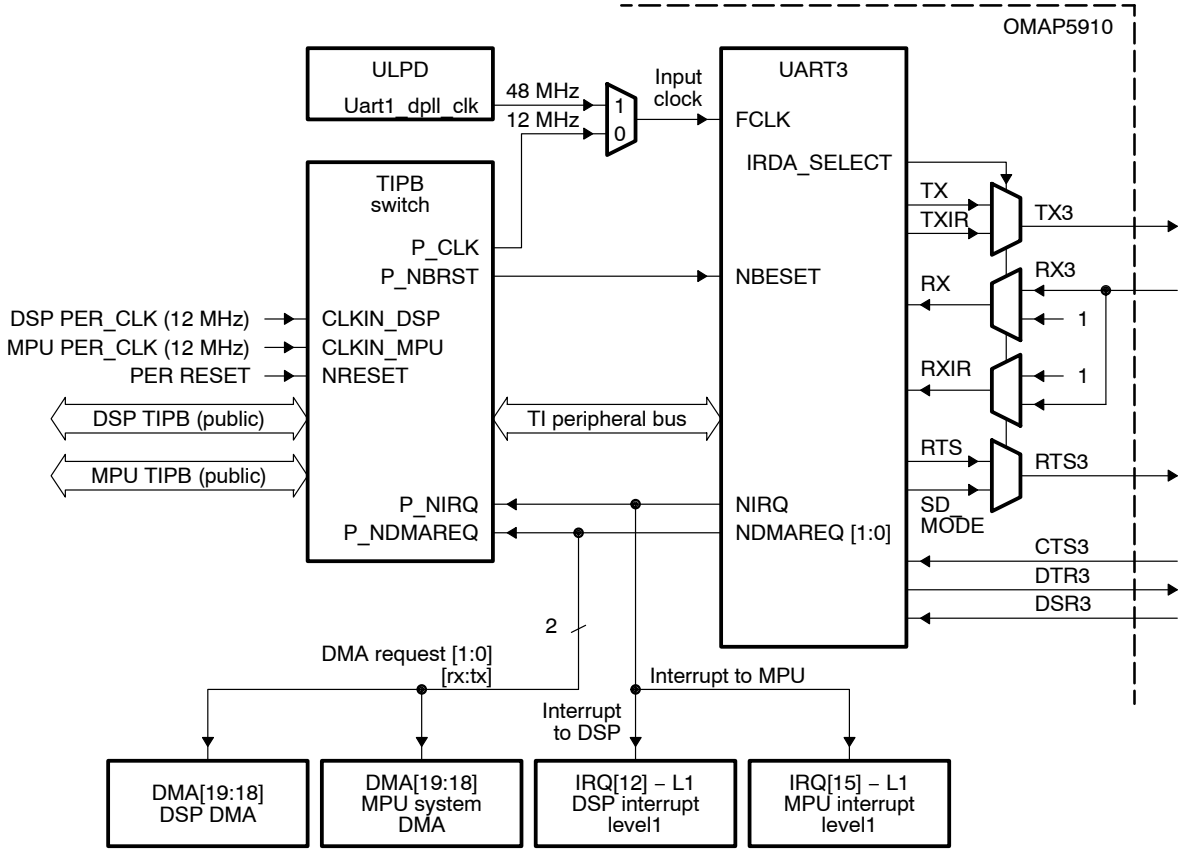
The NDMA_REQ[1] is an RX request and the NDMA_REQ[0] is a TX request.

NIRQ from UART3 is connected to:

- The interrupt line IRQ[15] of the MPU level 1 interrupt handler
- The interrupt line IRQ[12] of the DSP level 1 interrupt handler

Figure 6 shows the UART3 environment.

*Figure 6. UART3 Environment*



## 2.4 TIPB Switch

By default, the three UARTs are controllable from the MPU public TIPB.

The three TIPB switch modules allow the default configurations to be changed individually and thus to control the UARTs from the DSP public TIPB.

This change can only be done during the boot time. Dynamic switches are not supported.

This switch is software programmable, so each TIPB switch has two sets of registers:

- The MPU-accessible registers are listed in Table 6. Table 7 and Table 8 describe the register bits.
- The DSP-accessible registers are listed in Table 9. Table 10 and Table 11 describe the register bits.

*Table 6. MPU Registers*

| UART | Register | Description | R/W | Bits | Address |
|---|---|---|---|---|---|
| UART1 | RHSW_ARM_CNF | TIPB switch configuration | R/W | 16 | FFFB:C800 |
| UART1 | RHSW_ARM_STA | TIPB switch status | R | 16 | FFFB:C804 |
| UART2 | RHSW_ARM_CNF | TIPB switch configuration | R/W | 16 | FFFB:C840 |
| UART2 | RHSW_ARM_STA | TIPB switch status | R | 16 | FFFB:C844 |
| UART3 | RHSW_ARM_CNF | TIPB switch configuration | R/W | 16 | FFFB:C880 |
| UART3 | RHSW_ARM_STA | TIPB switch status | R | 16 | FFFB:C884 |

*Table 7. TIPB Switch Configuration MPU Register (RHSW_ARM_CNF) Field Descriptions*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 15–2 | Reserved | | – | – | – |
| 1 | DSP_PERIPH_LOCK | 0 | No lock | R | 0 |
| | | 1 | DSP bus is allocated. | | |
| 0 | ARM_PERIPH_LOCK | 0 | No lock | R/W | 1 |
| | | 1 | MPU bus is allocated. | | |

*Table 8. TIPB Switch Status MPU Register (RHSW_ARM_STA) Field Descriptions*

| Bits | Field | Value | Descriptions | R/W | Reset Value |
|---|---|---|---|---|---|
| 15–4 | Reserved | | – | – | – |
| 3 | RHSW_BOTH_LCK_ERR | 0 | Normal operation | R | 0 |
| | | 1 | Lock error | | |
| 2 | RHSW_ITPEND_ERR | 0 | Normal operation | R | 0 |
| | | 1 | DMA request error | | |
| 1 | RHSW_DMAREQ_ERR | 0 | Normal operation | R | 0 |
| | | 1 | IT pending error | | |
| 0 | RHSW_ERR_NIRQ | 0 | Clears IRQ line and all others status bits of register | R/W | 1 |
| | | 1 | Normal operation | | |

*Table 9. DSP Registers*

| UART | Register | Description | R/W | Bits | Address |
|---|---|---|---|---|---|
| UART1 | RHSW_DSP_CNF | TIPB switch control | R/W | 16 | 001:C800 |
| UART1 | RHSW_DSP_STA | TIPB switch status | R | 16 | 001:C802 |
| UART2 | RHSW_DSP_CNF | TIPB switch control | R/W | 16 | 001:C820 |
| UART2 | RHSW_DSP_STA | TIPB switch status | R | 16 | 001:C822 |
| UART3 | RHSW_DSP_CNF | TIPB switch control | R/W | 16 | 001:C840 |
| UART3 | RHSW_DSP_STA | TIPB switch status | R | 16 | 001:C842 |

*Table 10. TIPB Switch Configuration DSP Register (RHSW_DSP_CNF) Field Descriptions*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 15–2 | Reserved | | – | – | – |
| 1 | DSP_PERIPH_LOCK | 0 | No lock | R/W | 0 |
| | | 1 | DSP bus is allocated. | | |
| 0 | ARM_PERIPH_LOCK | 0 | No lock | R | 1 |
| | | 1 | MPU bus is allocated. | | |

*Table 11. TIPB Switch Status DSP Register (RHSW_DSP_STA) Field Descriptions*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 15–4 | Reserved | | – | – | – |
| 3 | RHSW_BOTH_LCK_ERR | 0 | Normal operation | R | 0 |
| | | 1 | Lock error | | |
| 2 | RHSW_ITPEND_ERR | 0 | Normal operation | R | 0 |
| | | 1 | DMA request error | | |
| 1 | RHSW_DMAREQ_ERR | 0 | Normal operation | R | 0 |
| | | 1 | IT pending error | | |
| 0 | RHSW_ERR_NIRQ | 0 | Clears the IRQ line and all others status bits of register | R/W | 1 |
| | | 1 | Normal operation | | |

## 2.5 Switching Procedures

The following procedures enable switching from the MPU to the DSP.

For switching UART1 to DSP:

1) MPU: Write 0 into the UART1 TIPB switch configuration MPU register (RHSW_ARM_CNF) to unlock UART1.

2) DSP: Write 2 into the UART1 TIPB switch status DSP register (RHSW_DSP_CNF) to lock UART1.

For switching UART2 to DSP:

1) MPU: Write 0 into the UART2 TIPB switch configuration MPU register (RHSW_ARM_CNF) to unlock UART2.

2) DSP: Write 2 into UART2 TIPB switch configuration DSP register (RHSW_DSP_CNF) to lock UART2.

For switching UART3 to DSP:

1) MPU: Write 0 into the UART3 TIPB switch configuration MPU register (RHSW_ARM_CNF) to unlock UART3.

2) DSP: Write 2 into UART3 TIPB switch configuration DSP register (RHSW_DSP_CNF) to lock UART3.

**Note:PERIF_LOCK Bits**

If either the DSP_PERIF_LOCK (in the RHSW_DSP_CNF register) or the ARM_PERIF_LOCK bit (in the RHSW_ARM_CNF register) is already set to 1, then a write to the other PERIF_LOCK bit has no effect on the TIPB switches, even though such a write may be performed. Before attempting to write 1 to a PERIF_LOCK bit, the DSP and MPU software must always read the corresponding read-only PERIF_LOCK bits to confirm that the other processor PERIF_LOCK bit is not already set.

# 3 UART/Autobaud Control and Status Registers

The programming combinations for register selection are shown in Table 12.

## 3.1 UART/Autobaud Modem Register Mapping

UART1 and UART2 are accessible as follows:

- MPU (32-bit-byte aligned address) from the following base addresses:
  - UART1: 0xFFFB 0000
  - UART2: 0xFFFB 0800
- DSP (16-bit-aligned word address) from the following base addresses:
  - UART1: 0x008000
  - UART2: 0x008400

*Table 12. UART Modem Register Program*

| MPU Byte Offset | DSP Byte Offset | Registers | | | | | |
|---|---|---|---|---|---|---|---|
| | | LCR[7] = 0 | | LCR[7] = 1 LCR[7:0] ≠ 0xBF | | LCR[7:0] = 0xBF | |
| | | READ | WRITE | READ | WRITE | READ | WRITE |
| 0x00 | 0x00 | RHR | THR | DLL | DLL | DLL | DLL |
| 0x04 | 0x02 | IER† | IER† | DLH | DLH | DLH | DLH |
| 0x08 | 0x04 | IIR | FCR† | IIR | FCR† | EFR | EFR |
| 0x0C | 0x06 | LCR | LCR | LCR | LCR | LCR | LCR |
| 0x10 | 0x08 | MCR† | MCR† | MCR† | MCR† | XON1 | XON1 |
| 0x14 | 0x0A | LSR | - | LSR | - | XON2 | XON2 |
| 0x18 | 0x0C | MSR/TCR‡ | TCR‡ | MSR/TCR‡ | TCR‡ | XOFF1/TCR‡ | XOFF1/TCR‡ |
| 0x1C | 0x0E | SPR/TLR‡ | SPR/TLR‡ | SPR/TLR‡ | SPR/TLR‡ | XOFF2/TLR‡ | XOFF2/TLR‡ |
| 0x20 | 0x10 | MDR1 | MDR1 | MDR1 | MDR1 | MDR1 | MDR1 |
| 0x24 | 0x12 | - | - | - | - | - | - |
| 0x28 | 0x14 | - | - | - | - | - | - |
| 0x2C | 0x16 | - | - | - | - | - | - |
| 0x30 | 0x18 | - | - | - | - | - | - |

† MCR[7:5], FCR[5:4], and IER[7:4] can only be written when EFR[4] = 1.
‡ Transmission control register (TCR) and trigger level register (TLR) are accessible only when EFR[4] = 1 and MCR[6] = 1.

*Table 12. UART Modem Register Program (Continued)*

| MPU Byte Off-set | DSP Byte Off-set | Registers | | | | | |
|---|---|---|---|---|---|---|---|
| | | LCR[7] = 0 | | LCR[7] = 1 LCR[7:0] ≠ 0xBF | | LCR[7:0] = 0xBF | |
| | | READ | WRITE | READ | WRITE | READ | WRITE |
| 0x34 | 0x1A | - | - | - | - | - | - |
| 0x38 | 0x1C | - | - | UASR | - | UASR | - |
| 0x3C | 0x1E | - | - | - | - | - | - |
| 0x40 | 0x20 | SCR | SCR | SCR | SCR | SCR | SCR |
| 0x44 | 0x22 | SSR | - | SSR | - | SSR | - |
| 0x48 | 0x24 | - | - | - | - | - | - |
| 0x4C | 0x26 | - | OSC_12M_ SEL | - | - | - | - |
| 0x50 | 0x28 | MVR | - | MVR | - | MVR | - |

† MCR[7:5], FCR[5:4], and IER[7:4] can only be written when EFR[4] = 1.
‡ Transmission control register (TCR) and trigger level register (TLR) are accessible only when EFR[4] = 1 and MCR[6] = 1.

Table 13 lists the UART/autobaud registers. Table 14 through Table 41 describe specific register bits.

*Table 13. UART/Autobaud Registers*

| Register | Description | Size | Access |
|---|---|---|---|
| RHR | Receive holding | 8-bit | R |
| THR | Transmit holding | 8-bit | W |
| FCR | FIFO control | 8-bit | W |
| SCR | Supplementary control | 8-bit | R/W |
| LCR | Line control | 8-bit | R/W |
| LSR | UART mode (LSR) | 8-bit | R |
| SSR | Supplementary status | 8-bit | R |
| MCR | Modem control | 8-bit | R/W |
| MSR | Modem status | 8-bit | R |
| IER | Interrupt enable (IER) | 8-bit | R/W |

*Table 13. UART/Autobaud Registers (Continued)*

| Register | Description | Size | Access |
|---|---|---|---|
| IIR | Interrupt identification (IIR) | 8-bit | R |
| EFR | Enhanced feature | 8-bit | R/W |
| XON1 | XON1 | 8-bit | R/W |
| XON2 | XON2 | 8-bit | R/W |
| XOFF1 | XOFF1 | 8-bit | R/W |
| XOFF2 | XOFF2 | 8-bit | R/W |
| SPR | Scratchpad | 8-bit | R/W |
| DLL | Divisor latch low | 8-bit | R/W |
| DLH | Divisor latch high | 8-bit | R/W |
| TCR | Transmission control | 8-bit | R/W |
| TLR | Trigger level | 8-bit | R/W |
| MDR1 | Mode definition 1 | 8-bit | R/W |
| UASR | UART autobauding status | 8-bit | R |
| OSC_12M_SEL | 12-MHz oscillator select | 8-bit | R |
| MVR | Module version | 8-bit | R |

The receiver section consists of the receiver holding register (RHR) and the receiver shift register. The RHR is actually a 64-byte FIFO. The receiver shift register receives serial data from RX input. The data is converted to parallel data and moved to the RHR. If the FIFO is disabled, location zero of the FIFO is used to store the single data character.

**Note:**

If an overflow occurs, data in the RHR is not overwritten.

*Table 14. Receive Holding Register (RHR) Field Description*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–0 | RHR | Receive holding register | R | Undefined |

The transmitter section consists of the transmit holding register (THR) and the transmit shift register. The THR is actually a 64-byte FIFO. The host (MPU or DSP) writes data to the THR. The data is placed in the transmit shift register where it is shifted out serially on the TX output. If the FIFO is disabled, location 0 of the FIFO is used to store the data.

*Table 15. Transmit Holding Register (THR) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–0 | THR | Transmit holding register | W | Undefined |

*Table 16. FIFO Control Register (FCR) Field Descriptions*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 7–6 | RX_FIFO_TRIG | | Sets the trigger level for the RX FIFO: | W | 00 |
| | | | If SCR[7] = 0 and TLR[7:4] = 0000: | | |
| | | 00 | 8 characters | | |
| | | 01 | 16 characters | | |
| | | 10 | 56 characters | | |
| | | 11 | 60 characters | | |
| | | | If SCR[7] = 0 and TLR[7:4] ≠ 0000, RX_FIFO_TRIG is not considered. | | |
| | | | If SCR[7] = 1, RX_FIFO_TRIG is two LSBs of the trigger level (1-63 on 6 bits) with granularity of 1. | | |

**Notes:** 1) Bits 4 and 5 can only be written when EFR[4] = 1.

2) Bits 0 to 3 can be changed only when baud clock is not running (DLL and DLH set to 0).

3) See Table 36 for FCR[5:4] setting restriction when SCR[6] = 1.

4) See Table 37 for FCR[7:6] setting restriction when SCR[7] = 1.

*Table 16. FIFO Control Register (FCR) Field Descriptions (Continued)*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 5–4 | TX_FIFO_TRIG | | Sets the trigger level for the TX FIFO:<br>If SCR[6] = 0 and TLR[3:0] = 0000: | W | 00 |
| | | 00 | 8 characters | | |
| | | 01 | 16 characters | | |
| | | 10 | 56 characters | | |
| | | 11 | 60 characters | | |
| | | | If SCR[6] = 0 and TLR[3:0] ≠ 0000, TX_FIFO_TRIG is not considered.<br>If SCR[6] = 1, TX_FIFO_TRIG is two LSBs of the trigger level (1 - 63 on 6 bits) with granularity of 1. | | |
| 3 | DMA_MODE | 0 | DMA_MODE 0 (No DMA) | W | 0 |
| | | 1 | DMA_MODE 1 (UART_nDMA_REQ0 in TX, UART_nDMA_REQ1 in RX)<br>This register only has effect if SCR[0] = 0. | | |
| 2 | TX_FIFO_CLEAR | 0 | No change | W | 0 |
| | | 1 | Clears the transmit FIFO and resets its counter logic to zero. Returns to zero after clearing the FIFO. | | |
| 1 | RX_FIFO_CLEAR | 0 | No change | W | 0 |
| | | 1 | Clears the receive FIFO and resets its counter logic to zero. Returns to zero after clearing the FIFO. | | |
| 0 | FIFO_EN | 0 | Disables the transmit and receive FIFOs | W | 0 |
| | | 1 | Enables the transmit and receive FIFOs | | |

**Notes:** 1) Bits 4 and 5 can only be written when EFR[4] = 1.
2) Bits 0 to 3 can be changed only when baud clock is not running (DLL and DLH set to 0).
3) See Table 36 for FCR[5:4] setting restriction when SCR[6] = 1.
4) See Table 37 for FCR[7:6] setting restriction when SCR[7] = 1.

*Table 17. Supplementary Control Register (SCR)*

| Bit | Name | Value | Function | R/W | Reset Value |
|---|---|---|---|---|---|
| 7 | RX_TRIG_GRANU1 | 0 | Disables the granularity of 1 for trigger RX level | R/W | 0 |
| | | 1 | Enables the granularity of 1 for trigger RX level | | |
| 6 | TX_TRIG_GRANU1 | 0 | Disables the granularity of 1 for trigger TX level | R/W | 0 |
| | | 1 | Enables the granularity of 1 for trigger TX level | | |
| 5 | DSR_IT | 0 | Disables $\overline{\text{DSR}}$ interrupt | R/W | 0 |
| | | 1 | Enables $\overline{\text{DSR}}$ interrupt | | |
| 4 | RX_CTS_DSR_ WAKE_UP_ENABLE | 0 | Disables the wake up interrupt and clears SSR1 | R/W | 0 |
| | | 1 | Waits for a falling edge of pins RX, $\overline{\text{CTS}}$, or $\overline{\text{DSR}}$ to generate an interrupt | | |
| 3 | TX_EMPTY_CTL_IT | 0 | Normal mode for THR interrupt (see Table 23) | R/W | 0 |
| | | 1 | The THR interrupt is generated when the TX FIFO and TX shift register are empty. | | |
| 2–1 | DMA_MODE_2 | | Used to specify the DMA mode. Valid if SCR[0] = 1 | R/W | 00 |
| | | 00 | DMA mode 0 (no DMA) | | |
| | | 01 | DMA mode 1 (UART_nDMA_REQ0 in TX, UART_nDMA_REQ1 in RX) | | |
| | | 10 | DMA mode 2 (UART_nDMA_REQ0 in RX) | | |
| | | 11 | DMA mode 3 (UART_nDMA_REQ0 in TX) | | |
| 0 | DMA_MODE_CTL | 0 | The DMA_MODE is set with FCR3. | R/W | 0 |
| | | 1 | The DMA_MODE is set with SCR2:1. | | |

**Note:** Bit 4 enables the wake-up interrupt, but this interrupt is not mapped on the IIR register. Therefore, when an interrupt occurs and if there is no interrupt pending in IIR, SSR[1] must be checked. To clear the wake-up interrupt, SCR[4] must be reset to 0.

*Table 18. Line Control Register (LCR) Field Descriptions*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 7 | DIV_EN | 0 | Normal operating condition | R/W | 0 |
| | | 1 | Divisor latch enable. Allows access to DLL, DLH, and other registers (see the register mapping). | | |
| 6 | BREAK_EN | | Break control bit. | R/W | 0 |
| | | 0 | Normal operating condition | | |
| | | 1 | Forces the transmitter output to go low to alert the communication terminal | | |
| 5 | PARITY_TYPE2 | | Selects the forced parity format (if LCR[3] = 1) | R/W | 0 |
| | | | If LCR[5] = 1 and LCR[4] = 0, the parity bit is forced to 1 in the transmitted and received data. | | |
| | | | If LCR[5] = 1 and LCR[4] = 1, the parity bit is forced to 0 in the transmitted and received data. | | |
| 4 | PARITY_TYPE1 | 0 | Odd parity is generated (if bit 3 = 1). | R/W | 0 |
| | | 1 | Even parity is generated (if bit 3 = 1). | | |
| 3 | PARITY_EN | 0 | No parity | R/W | 0 |
| | | 1 | A parity bit is generated during transmission and the receiver checks for the received parity. | | |
| 2 | NB_STOP | | Specifies the number of stop bits: | R/W | 0 |
| | | 0 | 1 stop bits (word length = 5, 6, 7, 8) | | |
| | | 1 | 1.5 stop bits (word length = 5) | | |
| | | 1 | 2 stop bits (word length = 6, 7, 8) | | |

**Note:** As soon as LCR[6] is set to 1, the RX line is forced to 0 and remains in this state as long as LCR[6] = 1.

*Table 18. Line Control Register (LCR) Field Descriptions (Continued)*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 1–0 | CHAR_LENGTH | | Specifies the word length to be transmitted or received. | R/W | 00 |
| | | 00 | 5 bits | | |
| | | 01 | 6 bits | | |
| | | 10 | 7 bits | | |
| | | 11 | 8 bits | | |

**Note:** As soon as LCR[6] is set to 1, the RX line is forced to 0 and remains in this state as long as LCR[6] = 1.

*Table 19. UART Mode Line Status Register (LSR) Field Descriptions*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 7 | RX_FIFO_STS | 0 | Normal operation | R | 0 |
| | | 1 | At least one parity error, framing error or break indication in the receiver FIFO. Bit 7 is cleared when no more errors are present in the FIFO. | | |
| 6 | TX_SR_E | 0 | Transmitter hold and shift registers are not empty. | R | 1 |
| | | 1 | Transmitter hold and shift registers are empty. | | |
| 5 | TX_FIFO_E | 0 | Transmit hold register is not empty. | R | 1 |
| | | 1 | Transmit hold register is empty. The processor can now load up to 64 bytes of data into the THR if the TX FIFO is enabled. | | |
| 4 | RX_BI | 0 | No break condition | R | 0 |
| | | 1 | A break was detected while the data being read from the RX FIFO was being received (i.e., the RX input was low for one character time frame). | | |

*Table 19. UART Mode Line Status Register (LSR) Field Descriptions (Continued)*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 3 | RX_FE | 0 | No framing error in data being read from RX FIFO | R | 0 |
| | | 1 | A framing error occurred in the data being read from the RX FIFO (i.e., the received data did not have a valid stop bit). | | |
| 2 | RX_PE | 0 | No parity error in the data being read from the RX FIFO | R | 0 |
| | | 1 | Parity error in the data being read from the RX FIFO | | |
| 1 | RX_OE | 0 | No overrun error | R | 0 |
| | | 1 | An overrun error has occurred. Set when the character held in the receive shift register is not transferred to the RX FIFO. This case can occur only when the receive FIFO is full. | | |
| 0 | RX_FIFO_E | 0 | No data in the receive FIFO | R | 0 |
| | | 1 | At least one data character is in the RX_FIFO | | |

When the LSR is read, LSR[4:2] reflects the error bits [BI, FE, PE] of the character at the top of the RX FIFO (next character to be read). Therefore, reading the LSR and then reading the RHR identifies the errors in a character.

Reading RHR updates BI, FE, and PE (see Table 14).

LSR[7] is set when there is an error anywhere in the RX FIFO and is cleared only when there are no more errors remaining in the FIFO.

**Note:**

Reading the LSR does not cause an increment of the RX FIFO read pointer. The RX FIFO read pointer is incremented by reading the RHR.

Reading LSR clears OE, if OE is set (see Table 19).

*Table 20. Supplementary Status Register (SSR) Field Descriptions*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 7–2 | – | | Reserved | R | 000000 |
| 1 | RX_CTS_DSR_WAKE_UP_STS | 0 | No falling edge event on RX, $\overline{\text{CTS}}$ and $\overline{\text{DSR}}$ | R | 0 |
| | | 1 | A falling edge occurred on RX, $\overline{\text{CTS}}$ or $\overline{\text{DSR}}$. | | |
| 0 | TX_FIFO_FULL | 0 | TX FIFO not full | R | 0 |
| | | 1 | TX FIFO full | | |

**Note:** Bit 1 is reset only when SCR[4] is reset to 0.

The modem control register (MCR)[3:0] controls the interface with the modem, data set, or peripheral device that is emulating the modem.

*Table 21. Modem Control Register (MCR) Field Descriptions*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 7 | CLKSEL | 0 | No action | R/W | 0 |
| | | 1 | Divide clock input by 4 | | |
| 6 | TCR_TLR | 0 | No action | R/W | 0 |
| | | 1 | Enables access to the TCR and TLR registers | | |
| 5 | XON_EN | 0 | Disable XON for any function | R/W | 0 |
| | | 1 | Enable XON for any function | | |
| 4 | LOOPBACK_EN | 0 | Normal operating mode | R/W | 0 |
| | | 1 | Enable the local loop back mode (internal). In this mode the MCR[3:0] signals are looped back into MSR | | |
| 3 | RESERVED | | Reserved. This bit must always be written as 0. | R/W | 0 |
| 2 | RESERVED | | Reserved. This bit must always be written as 0. | R/W | 0 |

**Note:** Bits 5, 6, and 7 can be written only when EFR[4] = 1.

*Table 21. Modem Control Register (MCR) Field Descriptions (Continued)*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 1 | RTS | 0 | 0: Forces $\overline{\text{RTS}}$ output to inactive (high) | R/W | 0 |
| | | 1 | Forces $\overline{\text{RTS}}$ output to active (low)<br>In loopback mode, controls MSR[4].<br>If automatic RTS is enabled, the $\overline{\text{RTS}}$ output is controlled by the hardware flow control. | | |
| 0 | DTR | 0 | Forces $\overline{\text{DTR}}$ output to inactive (high) | R/W | 0 |
| | | 1 | Forces $\overline{\text{DTR}}$ output to active (low) | | |

**Note:** Bits 5, 6, and 7 can be written only when EFR[4] = 1.

The modem status register (MSR) provides information about the current state of the control lines from the modem, data set, or peripheral device to the host (MPU or DSP). It also indicates when a control input from the modem changes state.

*Table 22. Modem Status Register (MSR) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–6 | RESERVED | Reserved | R | Input signal |
| 5 | NDSR_STS | This bit is the complement of the $\overline{\text{DSR}}$ input. In loopback mode it is equivalent to MCR[0]. | R | Input signal |
| 4 | NCTS_STS | This bit is the complement of the $\overline{\text{CTS}}$ input. In loopback mode it is equivalent to MCR[1]. | R | Input signal |
| 3–2 | RESERVED | Reserved | R | 0 |
| 1 | DSR_STS | 1: Indicates that $\overline{\text{DSR}}$ input (or MCR[0] in loopback) has changed state. It is cleared on a read. | R | 0 |
| 0 | CTS_STS | 1: Indicates that $\overline{\text{CTS}}$ input (or MCR[1] in loopback) has changed state. It is cleared on a read. | R | 0 |

The interrupt enable register (IER) can be programmed to enable/disable any of the following interrupts:

- Receiver error
- RHR
- THR
- XOFF received
- $\overline{\text{CTS}}$/$\overline{\text{RTS}}$ change of state from low to high

These interrupts can be enabled/disabled individually. There is also a sleep mode enable bit.

*Table 23. UART Mode Interrupt Enable Register (IER) Field Descriptions*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 7 | CTS_IT | 0 | Disables the $\overline{\text{CTS}}$ interrupt | R/W | 0 |
| | | 1 | Enables the $\overline{\text{CTS}}$ interrupt | | |
| 6 | RTS_IT | 0 | Disables the $\overline{\text{RTS}}$ interrupt | R/W | 0 |
| | | 1 | Enables the $\overline{\text{RTS}}$ interrupt | | |
| 5 | XOFF_IT | 0 | Disables the XOFF interrupt | R/W | 0 |
| | | 1 | Enables the XOFF interrupt | | |
| 4 | SLEEP_MODE | 0 | Disables sleep mode | R/W | 0 |
| | | 1 | Enables the sleep mode (stops baud rate clock when the module is inactive) | | |
| 3 | MODEM_STS_IT | 0 | Disables the modem status register interrupt | R/W | 0 |
| | | 1 | Enables the modem status register interrupt | | |
| 2 | LINE_STS_IT | 0 | Disables the receiver line status interrupt | R/W | 0 |
| | | 1 | Enables the receiver line status interrupt | | |
| 1 | THR_IT | 0 | Disables the THR interrupt | R/W | 0 |
| | | 1 | Enables the THR interrupt | | |
| 0 | RHR_IT | 0 | Disables the RHR interrupt and time-out interrupt. | R/W | 0 |
| | | 1 | Enables the RHR interrupt and time-out interrupt. | | |

**Note:** Bits 4, 5, 6, and 7 can only be written when EFR[4] = 1.

The IIR is a read-only register that provides the source of the interrupt in a prioritized manner.

*Table 24. UART Mode Interrupt Identification Register (IIR) Field Descriptions*

| Bits | Field | Value | Description | | | | | | | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 7–6 | FCR_MIRROR | | Mirror the contents of FCR[0] on both bits | | | | | | | R | 00 |
| 5–1 | IT_TYPE | | **Priority** | **5** | **4** | **3** | **2** | **1** | **Source** | R | 00000 |
| | | | **1** | 0 | 0 | 0 | 1 | 1 | Receiver line status error | | |
| | | | **2** | 0 | 0 | 1 | 1 | 0 | RX time-out | | |
| | | | **2** | 0 | 0 | 0 | 1 | 0 | RHR interrupt | | |
| | | | **3** | 0 | 0 | 0 | 0 | 1 | THR interrupt | | |
| | | | **4** | 0 | 0 | 0 | 0 | 0 | Modem interrupt | | |
| | | | **5** | 0 | 1 | 0 | 0 | 0 | Xoff/special character | | |
| | | | **6** | 1 | 0 | 0 | 0 | 0 | CTS, RTS, DSR change state from active (low) to inactive (high) | | |
| 0 | IT_PENDING | 0 | An interrupt is pending (nIRQ active). | | | | | | | R | 1 |
| | | 1 | No interrupt is pending (nIRQ inactive). | | | | | | | | |

The enhanced feature register (EFR) enables or disables enhanced features.

*Table 25. Enhanced Feature Register (EFR) Field Descriptions*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 7 | AUTO_CTS_EN | | Automatic CTS enable bit | R/W | 0 |
| | | 0 | Normal operation | | |
| | | 1 | Automatic CTS flow control is enabled; that is, transmission is halted when the $\overline{\text{CTS}}$ pin is high (inactive). | | |
| 6 | AUTO_RTS_EN | | Automatic RTS enable bit | R/W | 0 |
| | | 0 | Normal operation | | |
| | | 1 | Automatic RTS flow control is enabled; that is, $\overline{\text{RTS}}$ pin goes high (inactive) when the receiver FIFO HALT trigger level, TCR[3:0], is reached, and goes low (active) when the receiver FIFO RESTORE transmission trigger level is reached. | | |

*Table 25. Enhanced Feature Register (EFR) Field Descriptions (Continued)*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 5 | SPECIAL_CHAR_ DETECT | 0 | Normal operation | R/W | 0 |
| | | 1 | Special character detect enable<br>Received data is compared with the XOFF2 data. If a match occurs, the received data is transferred to the FIFO and IIR bit 4 is set to 1 to indicate that a special character has been detected. | | |
| 4 | ENHANCED_EN | | Enhanced functions write enable bit | R/W | 0 |
| | | 0 | Disables writing to IER bits 4-7, FCR bits 4-5, and MCR bits 5-7. | | |
| | | 1 | Enables writing to IER bits 4-7, FCR bits 4-5, and MCR bits 5-7. | | |
| 3–0 | SW_FLOW_CON TROL | | Combinations of software flow control can be selected by programming bit 3-bit 0. See Section 5.10, *Software Flow Control.* | R/W | 0 |

*Table 26. EFR[0-3]: Software Flow Control Options*

| Bit 3 | Bit 2 | Bit 1 | Bit 0 | TX,RX Software Flow Controls |
|---|---|---|---|---|
| 0 | 0 | X | X | No transmit flow control |
| 1 | 0 | X | X | Transmit XON1, XOFF1 |
| 0 | 1 | X | X | Transmit XON2, XOFF2 |
| 1 | 1 | X | X | Transmit XON1, XON2: XOFF1, XOFF2 |
| X | X | 0 | 0 | No receive flow control |
| X | X | 1 | 0 | Receiver compares XON1, XOFF1 |
| X | X | 0 | 1 | Receiver compares XON2, XOFF2 |
| X | X | 1 | 1 | Receiver compares XON1, XON2: XOFF1, XOFF2 |

**Note:** XON1 and XON2 must be set to different values if the software flow control is enabled.

*Table 27. XON1 Register (XON1) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–0 | XON_WORD1 | Used to store the 8-bit XON1 character | R/W | 0x00 |

*Table 28. XON2 Register (XON2) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–0 | XON_WORD2 | Used to store the 8-bit XON2 character | R/W | 0x00 |

*Table 29. XOFF1 Register (XOFF1) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–0 | XOFF_WORD1 | Used to store the 8-bit XOFF1 character | R/W | 0x00 |

*Table 30. XOFF2 Register (XOFF2) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–0 | XOFF_WORD2 | Used to store the 8-bit XOFF2 character | R/W | 0x00 |

The scratchpad register (SPR) does not control the module in any way; rather, it is used by the programmer to hold temporary data.

*Table 31. Scratchpad Register (SPR) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–0 | SPR_WORD | Scratchpad register | R/W | 0x00 |

The divisor latch low register (DLL) and divisor latch high register (DLH) store the 16-bit divisor for generation of the baud clock in the baud rate generator. DLH stores the most significant part of the divisor; DLL stores the least significant part of the divisor.

**Note:**

The DLL and DLH latch registers can only be written to before sleep mode is enabled (that is, before IER[4] is set).

*Table 32. Divisor Latch Low Register (DLL) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–0 | CLOCK_LSB | Used to store the 8-bit LSB divisor value | R/W | 0x00 |

*Table 33. Divisor Latch High Register (DLH) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–0 | CLOCK_MSB | Used to store the 8-bit MSB divisor value | R/W | 0x00 |

To achieve the required baud rate, DLL/DLH must be programmed with the integer part of the divisor value.

Choosing the appropriate divisor value:

UART: Divisor value = Operating Frequency / (16 x Baud Rate)

Just as in autobaud mode, the input frequency of the UART modem must be fixed to the operating frequency (here 12 MHz; no CLKSEL bit setting) and the the OSC_12M_SEL bit must be set to be able to reach the desired baud rate. Setting OSC_12M_SEL to 1 enables turning on the 6.5 division factor. For instance, 12 MHz/16/6.5 = 115200 bps; in case OSC_12M_SEL is not set, the reached baud rate is either 12 MHz/16/6 or 12 MHz/16/7, which are out of permitted tolerance.

The transmission control register (TCR) stores the receive FIFO threshold levels to start/stop transmission during hardware/software flow control.

*Table 34. Transmission Control Register (TCR) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–4 | RX_FIFO_TRIG_START | RCV FIFO trigger level to RESTORE transmission (0 – 60) | R/W | 0000 |
| 3–0 | RX_FIFO_TRIG_HALT | RCV FIFO trigger level to HALT transmission (0 – 60) | R/W | 1111 |

**Notes:** 1) Trigger levels from 0 - 60 bytes are available with a granularity of four (Trigger level = 4 x [4-bit register value]).

2) The programmer must ensure that TCR[3:0] > TCR[7:4] whenever automatic RTS or software flow control is enabled to avoid a faulty operation of the device.

3) In FIFO interrupt mode with flow control, the programmer must also ensure that the trigger level to HALT transmission is greater than or equal to the receive the FIFO trigger level (either TLR[7:4] or FCR[7:6]): otherwise, FIFO operation stalls. In FIFO DMA mode with flow control, this issue does not exist because a DMA request is sent each time a byte is received.

The trigger level register (TLR) stores the programmable transmit and receive FIFO trigger levels used for DMA and IRQ generation.

*Table 35. Trigger Level Register (TLR) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–4 | RX_FIFO_TRIG_DMA | Receive FIFO trigger level | R/W | 0000 |
| 3–0 | TX_FIFO_TRIG_DMA | Transmit FIFO trigger level | R/W | 0000 |

Table 36 and Table 37 summarize the different ways to set the trigger levels for the transmit FIFO and the receive FIFO.

*Table 36. TX FIFO Trigger Level Setting Summary*

| SCR[6] | TLR[3:0] | TX FIFO Trigger Level |
|---|---|---|
| 0 | 0000 | Defined by FCR5:4 (either 8, 16, 32, 56 characters) |
| 0 | ≠ 0000 | Defined by TLR[3:0] (from 4 to 60 spaces with a granularity of 4 characters) |
| 1 | Any value | Defined by the concatenated value of TLR[3:0] and FCR[5:4] (from 1 to 63 characters with a granularity of 1 character).<br>The combination of TLR[3:0] = 0000 and FCR[5:4] = 00 (all zeros) is not supported (minimum 1 character required). All zeros result in unpredictable behavior. |

**Note:** The protocol to set the concatenation of TLR and FCR is:
- Set SCR[6] = 0
- Set the value of threshold into FCR and TLR
- Set SCR[6] = 1

*Table 37. RX FIFO Trigger Level Setting Summary*

| SCR[7] | TLR[7:4] | RX FIFO Trigger Level |
|---|---|---|
| 0 | 0000 | Defined by FCR[7:6] (either 8, 16, 56, 60 characters) |
| 0 | ≠ 0000 | Defined by TLR[7:4] (from 4 to 60 characters with a granularity of 4 characters) |
| 1 | Any value | Defined by the concatenated value of TLR[7:4] and FCR[7:6] (from 1 to 63 characters with a granularity of 1 character).<br>The combination of TLR[7:4] = 0000 and FCR[7:6] = 00 (all zeros) is not supported (minimum 1 character required). All zeros result in unpredictable behavior. |

**Note:** The protocol to set the concatenation of TLR and FCR is:
- Set SCR[7] = 0
- Set the value of threshold into FCR and TLR
- Set SCR[7] = 1

The mode of operation can be programmed by writing to MDR1[2:0]; therefore the MDR1 must be programmed on start-up after configuration of the configuration registers (DLL, DLH, LCR). The value of MDR1[2:0] must not be changed again during normal operation.

*Table 38. Mode Definition Register 1 (MDR1) Field Descriptions*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 7–3 | – | | Reserved | R/W | 00000 |
| 2–0 | MODE_SELECT† | | 000: UART | R/W | 111 |
| | | 010 | UART with autobauding | | |
| | | 111 | Disables UART/default state | | |
| | | | All the other values are reserved. | | |

† The MODE_SELECT = 0x7 setting disables the UART module by disabling the FIFO and the state machine. It does not gate the functional clock to the module. The lowest power state is not achieved by setting MODE_SELECT = 0x7, but by putting the UART into sleep mode. The lowest power state is achieved when in sleep mode witht DLL = 0xFFFF and DLH = 0xFFFF.

The UART autobauding status register (UASR) returns the speed, the number of bits by characters, and the type of the parity in UART autobaud mode.

*Table 39. Autobauding Status Register (UASR) Field Description*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 7–6 | PARITY_TYPE | 00 | 00: No parity identified | R | 00 |
| | | 01 | Parity space | | |
| | | 10 | Even parity | | |
| | | 11 | Odd parity | | |
| 5 | BIT_BY_CHAR | 0 | 7-bit character identified | R | 0 |
| | | 1 | 8-bit character identified | | |
| 4–0 | SPEED | | Used to report the speed identified | R | 0000 |
| | | 00000 | No speed identified | | |
| | | 00001 | 115 200 baud | | |
| | | 00010 | 57 600 baud | | |
| | | 00011 | 38 400 baud | | |
| | | 00100 | 28 800 baud | | |
| | | 00101 | 19 200 baud | | |
| | | 00110 | 14 400 baud | | |
| | | 00111 | 9 600 baud | | |
| | | 01000 | 4 800 baud | | |
| | | 01001 | 2 400 baud | | |
| | | 01010 | 1 200 baud | | |

**Note:** This register is used to set up the transmission according to the characteristics of the previous reception instead of the LCR, DLL, and DLH registers when the UART is in autobaud mode.To reset the autobauding hardware (to start a new AT detection) or to set the UART in standard mode (no autobaud), MDR1[2:0] must be set to reset state 111 then to the UART in autobaud mode 010 or UART in standard mode 000.

*Table 40. OSC_12_MHz Register Select (OSC_12M_SEL) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–1 | – | Reserved | R | 0000000 |
| 0 | OSC_12M_SEL† | When 1, selects 6.5 division factor with a 12-MHz system clock. | W | 0 |

† This register is write-only and cannot be read.

*Table 41. Module Version Register (MVR) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7-4 | MAJOR_REV | Major revision number of the module | R | ---† |
| 3-0 | MINOR_REV | Minor revision number of the module | R | --- |

† For example: MVR = 0x11 => Version 1.1

# 4 UART/Autobaud Modes of Operation

The UART/autobaud module can operate in two different modes: UART mode and UART with autobaud mode.

The modules perform serial-to-parallel conversion on data characters received and parallel-to-serial conversion on data characters transmitted by the processor. The complete status of each channel of the modules and each received character/frame can be read at any time during functional operation via the line status register (LSR).

The modules can be placed in an alternate mode (FIFO mode) to relieve the processor of excessive software overhead by buffering received/transmitted characters. Both the receiver and transmitter FIFOs can store up to 64 bytes of data (plus three additional bits of error status per byte for the receiver FIFO) and have selectable trigger levels.

Both interrupts and DMA are available to control the data-flow between the host (MPU or DSP) and the module.

## 4.1 UART Mode

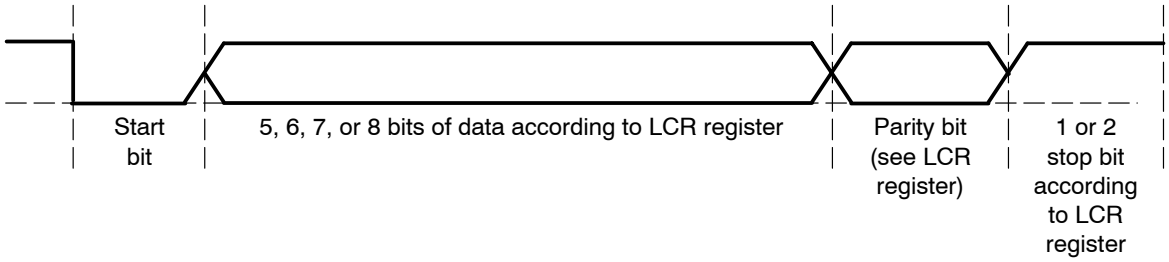
The UART modem uses a wired interface for serial communication with a remote device.

The UART modem module is functionally compatible with the TL16C750 UART and is also functionally compatible with earlier designs such as the TL16C550. The UART modem module can use hardware or software flow controls to manage transmission/reception. Hardware flow control significantly reduces software overhead and increases system efficiency by automatically controlling serial data flow using the $\overline{\text{RTS}}$ output and $\overline{\text{CTS}}$ input signals. Software flow control automatically controls data flow by using programmable XON/XOFF characters.

## 4.2 UART Mode With Autobauding

The UART modem module is enhanced with an autobauding functionality which in control mode allows automatically setting the speed, the number of bits per character, and the parity selected (see Figure 7).

*Figure 7. UART Data Format*

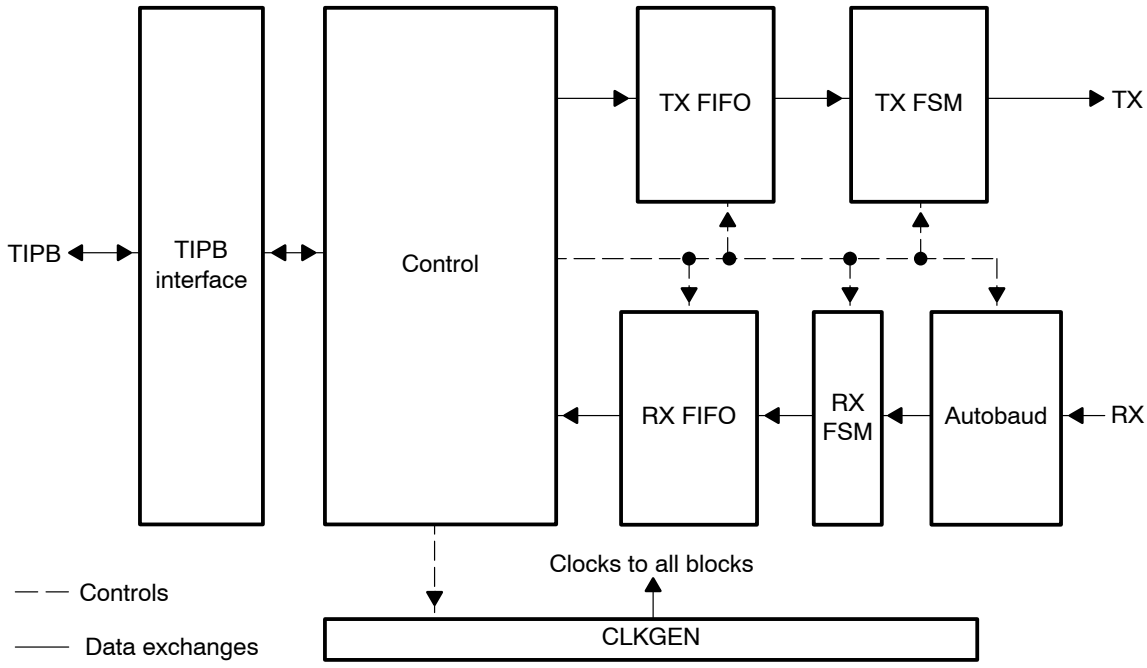


# 5 UART/Autobaud Functional Description

## 5.1 UART/Autobaud Functional Block Diagram

Figure 8 shows the UART/autobaud (FSM stands for finite state machine).

*Figure 8. Functional Block Diagram*




## 5.2 Trigger Levels

The UART provides programmable trigger levels for both receiver and transmitter DMA and interrupt generation. After reset, both the transmitter and receiver FIFOs are disabled (in effect, the trigger level is the default value of one byte). A programmable trigger level is an enhanced feature available via the trigger level register (TLR).

## 5.3 Interrupts

The UART generates interrupts on the UART_nIRQ output pin. All interrupts can be enabled/disabled by writing to the appropriate bit in the interrupt enable register (IER). The interrupt status of the device can be checked at any time by reading the interrupt identification register (IIR).

### 5.3.1 Generic Interrupts Description

There are seven possible interrupts, prioritized to six different levels.

When an interrupt is generated, the interrupt identification register (IIR) indicates a pending interrupt by bringing IIR[0] to logic 0, and it specifies the type of interrupt through IIR[5-1]. Table 42 summarizes the interrupt control functions.

*Table 42. Generic Interrupt Descriptions in Modem Mode*

| IIR[5-0] | Priority Level | Interrupt Type | Interrupt Source | Interrupt Reset Method |
|---|---|---|---|---|
| 0 0 0 0 0 1 | None | None | None | None |
| 0 0 0 1 1 0 | 1 | Receiver line status | OE, FE, PE, or BI errors occur in characters in the RX FIFO | FE, PE, BI: All erroneous characters are read from the RX FIFO. OE: Read LSR |
| 0 0 1 1 0 0 | 2 | RX time-out | Stale data is in the RX FIFO | Read the RHR |
| 0 0 0 1 0 0 | 2 | RHR interrupt | DRDY (data ready) (FIFO disable)<br>RX FIFO is above the trigger level (FIFO enable) | Read the RHR until the interrupt condition disappears. |
| 0 0 0 0 1 0 | 3 | THR interrupt | TFE (THR empty) (FIFO disable)<br>TX FIFO is below trigger level (FIFO enable) | Write to the THR until the interrupt condition disappears. |
| 0 0 0 0 0 0 | 4 | Modem status | MSR[1:0] = 0 | Read the MSR |
| 0 1 0 0 0 0 | 5 | XOFF interrupt/special character interrupt | Receive XOFF characters(s)/ special character | Receive XON character(s), if XOFF interrupt/read of the IIR, if special character interrupt |
| 1 0 0 0 0 0 | 6 | $\overline{CTS}$, $\overline{RTS}$, $\overline{DSR}$ | $\overline{RTS}$ pin, $\overline{CTS}$ pin or $\overline{DSR}$ pin change state from active (low) to inactive (high). | Read the IIR |

**Note:** Once LSR[7] (RX_FIFO_STS) is set to FIFO disable (FCR[0]=0), this bit bit cannot be cleared by reading LSR. First, FCR[1] (RX_FIFO_CLERA) must be set to 1, then LSR[7] can be cleared.

LSR[7] generates the receiver line status interrupt.

For the XOFF interrupt, if an XOFF flow character detection caused the interrupt, the interrupt is cleared by an XON flow character detection. If special

character detection caused the interrupt, the interrupt is cleared by a read of the interrupt identification register (IIR).

### 5.3.2 Wake-Up Interrupt

The wake-up interrupt is uniquely designed and is enabled when SCR[4] is set to 1. The interrupt identification register (IIR) is not modified when this interrupt occurs; SSR[1] must be checked to detect a wake-up event. When a wake-up interrupt occurs, the only way to clear it is to reset SCR[4] to 0.

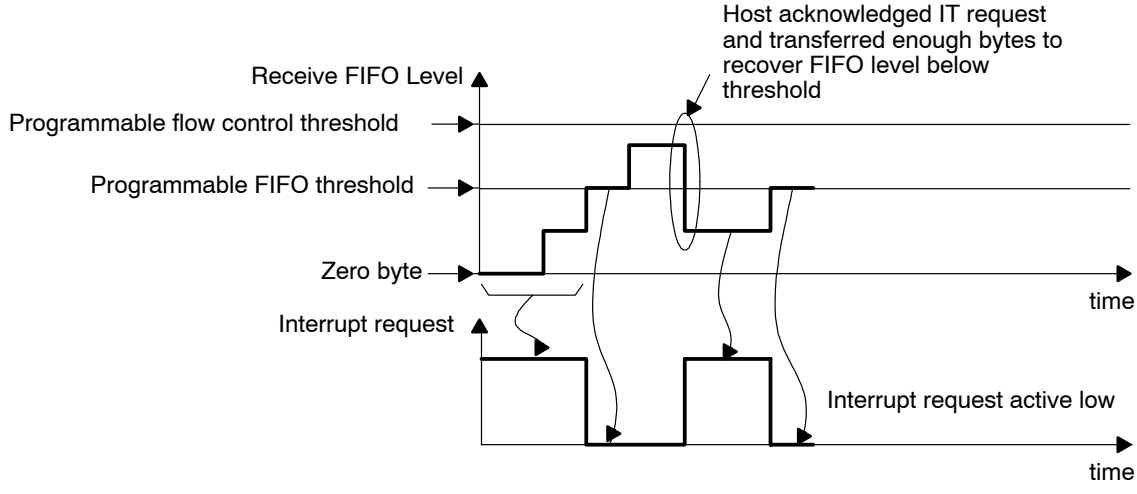
### 5.3.3 FIFO Interrupt Mode

In FIFO interrupt mode, FCR[0] = 1 and relevant interrupts are enabled via the interrupt enable register (IER). The processor is informed of the status of the receiver and transmitter by an interrupt signal, nIRQ. These interrupts are raised when the receive/transmit FIFO thresholds (respectively TLR[7:4] and TLR[3:0] or FCR[7:6] and FCR[5:4]) are reached; they instruct the host (MPU or DSP) to transfer data to the destination (from the UART module in receive mode and from any source to UART FIFO in transmit mode).

When UART flow control is enabled along with interrupt capabilities, ensure that the UART flow control FIFO threshold (TCR[3:0]) is greater than or equal to the receive FIFO threshold.

Figure 9 shows receive IT operations; Figure 10 shows transmit IT operations.

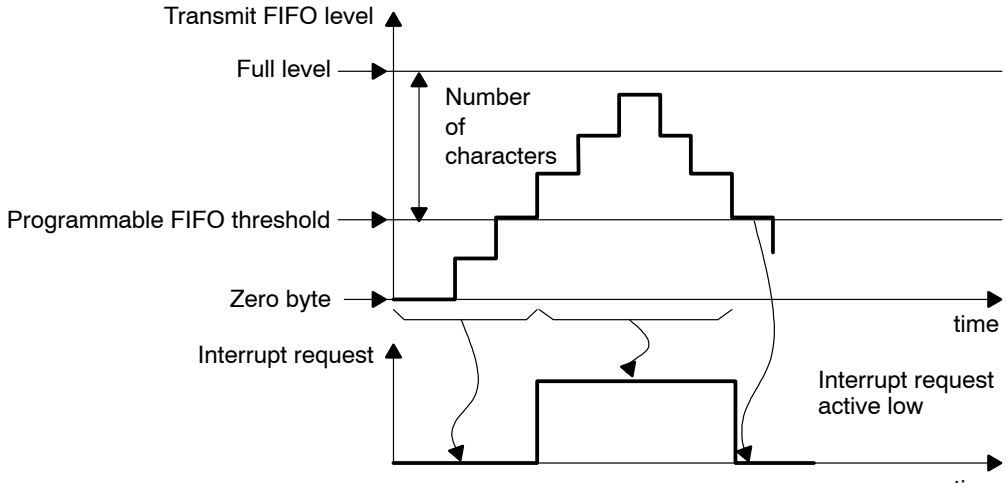
*Figure 9. Receive FIFO IT Request Generation*




In receive mode, no interrupt is generated until the receive FIFO reaches its threshold. A low interrupt can only be deasserted when the host (MPU or DSP)

has handled enough bytes to make the FIFO level below threshold. Notice that the flow control threshold is set at a higher value than the FIFO threshold.

*Figure 10. Transmit FIFO IT Request Generation*




In transmit mode, an interrupt request is automatically asserted when the FIFO is empty. This request is deasserted when the FIFO crosses the threshold level. The interrupt line is deasserted until a sufficient number of elements have been transmitted to go below FIFO threshold.

## 5.4 FIFO Polled Mode

In FIFO polled mode (FCR[0] = 0, relevant interrupts disabled via IER) the status of the receiver and transmitter can be checked by polling the line status register (LSR). This mode is an alternative to the FIFO interrupt mode of operation where the status of the receiver and transmitter is automatically known by means of interrupts sent to the host (MPU or DSP).

## 5.5 FIFO DMA Mode

### 5.5.1 DMA Signalling

There are four modes of DMA operation: DMA mode 0, DMA mode 1, DMA mode 2, and DMA mode 3. They can be selected as follows.

❑ When SCR[0] = 0:

- Setting FCR[3] to 0 enables DMA mode 0.
- Setting FCR[3] to 1 enables DMA mode 1.

- When SCR[0] = 1, SCR[2:1] determine DMA mode 0 to 3 according to SCR register description.

So, for instance:

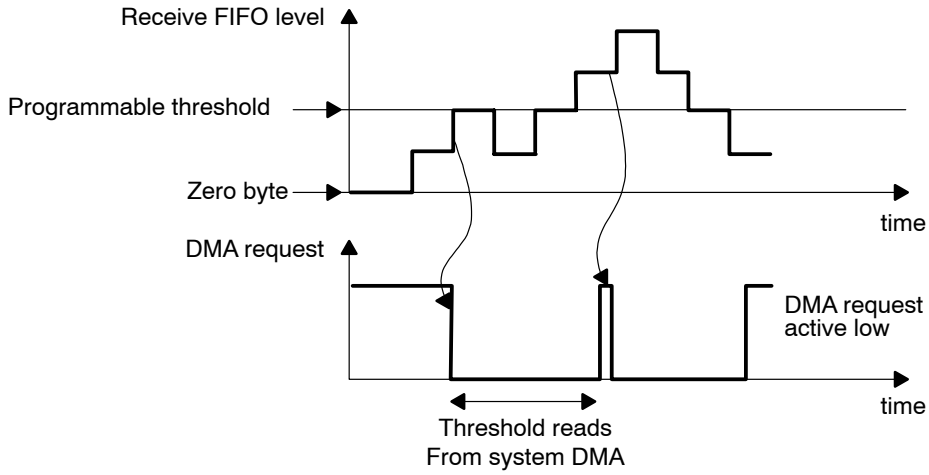
- If no DMA operation is desired, set SCR[0] to 1 and SCR[2:1] to 00 (FCR[3] is disregarded).
- If DMA mode 1 is desired, either set SCR[0] to 0 and FCR[3] to 1 or set SCR[0] to 1 and SCR[2:1] to 01 (FCR[3] is disregarded).

If the FIFOs are disabled (FCR[0] = 0), DMA occurs in single character transfers. When DMA mode 0 has been programmed, the signals associated with DMA operation are not active.

### 5.5.2 DMA Transfers

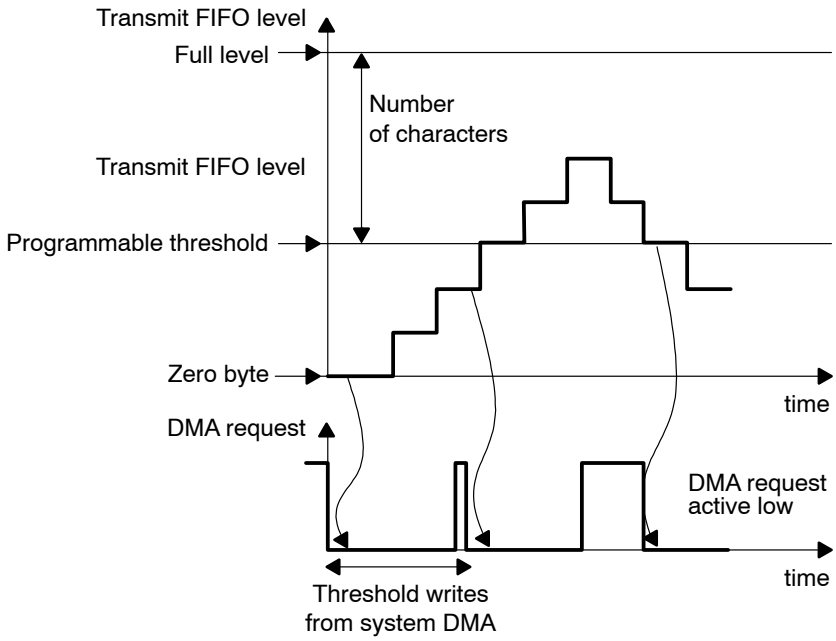
Figure 11 shows DMA operation at receive; Figure 12 shows DMA operation at transmit.

*Figure 11. Receive FIFO DMA Request Generation*




In receive mode, a DMA request is generated as soon as the receive FIFO reaches its threshold. This request is deasserted when the number of bytes defined by the threshold level has been read by the system DMA.

*Figure 12. Transmit FIFO DMA Request Generation*




In transmit mode, a DMA request is automatically asserted when the FIFO is empty. This request is deasserted when the number of bytes defined by the threshold level has been written by the system DMA. The DMA request is again asserted if the FIFO is able to receive the number of bytes defined by the threshold.

## 5.6 Sleep Mode

Sleep mode is a low-power, enhanced feature of the UART that can be enabled by writing a 1 to IER[4] (when EFR[4] = 1).

Sleep mode is entered when:

- ❏ Serial RX data input line is idle.
- ❏ TX FIFO and TX shift registers are empty.
- ❏ No interrupts are pending except transmit holding register (THR) interrupts.

Sleep mode is a good way to lower UART power consumption, but this state can be achieved only when the UART is set in the modem mode. Therefore, even if the UART does not have a functional key role, it must be initialized in a functional mode to take advantage of sleep mode.

In sleep mode, the module clock and baud rate clock are stopped internally. Since most registers are clocked using these clocks, the power consumption

is greatly reduced. The module wakes up when any change is detected on the RX line, when data is written to the TX FIFO, or when there is any change in the state of the modem input pins. An interrupt can be generated on a wake-up event by setting SCR[4] to 1.

**Note:**

Writing to the divisor latches, DLL and DLH, to set the baud clock, BCLK, must not be done during sleep mode. Disable sleep mode using IER[4] before writing to DLL or DLH.

## 5.7 Break and Time-out Conditions

- Time-out counter

  The RX idle condition is detected when the RX line has been high for a time equivalent to (4X programmed word length) plus12 bits. The receiver line is sampled midway through each bit.

  For sleep mode, the counter is reset when there is activity on the RX line.

  For the time-out interrupt, the counter only counts when there is data in the RX FIFO and the count is reset when there is activity on the RX line or when the RHR is read.

- Break condition

  When a break condition occurs, the TX line is pulled low. A break condition is activated by setting LCR[6]. The break condition is not aligned on the word stream; that is, a break condition can occur in the middle of a character. The only way to send a break condition on a full character is as follows:

  - Reset transmit FIFO (if enabled).
  - Wait for transmit shift register to become empty (LSR[6] = 1).
  - Take a guard time according to stop bit definition.
  - Set LCR[6] to 1.

  The break condition is asserted as long as LCR[6] is set to 1.

## 5.8 Programmable Baud Rate Generator

The programmable baud generator takes any clock input and divides it by a divisor between 1 and ($2^{16-1}$). The CLKSEL register bit MCR[7] can be used to select the 1X or 1X/4 clock for the internal baud rate generator. The output frequency of the baud rate generator is 16x the baud rate.

To program the baud rate, a write must be issued to the DLL register (least significant bytes) and DLH register (most significant bytes) of the baud rate divisor.

Writing to these registers can result in wait states being inserted during the write access while the baud rate generator is loaded with the new value. If both registers are 0, the module is effectively disabled and no baud clock is generated.

**Note:**

The programmable baud rate generator selects both the transmit and receive clock rates.

## 5.9 Hardware Flow Control

Hardware flow control is composed of automatic RTS and automatic CTS. Both can be enabled/disabled independently by programming EFR[7:6]. With automatic CTS, $\overline{\text{CTS}}$ must be active before the module can transmit data.

Automatic RTS only activates the $\overline{\text{RTS}}$ output when there is enough room in the FIFO to receive data, and it deactivates the $\overline{\text{RTS}}$ output when the RX FIFO is sufficiently full. The HALT and RESTORE trigger levels in the TCR determine the levels at which $\overline{\text{RTS}}$ is activated/deactivated.

If both automatic CTS and automatic RTS are enabled, data transmission does not occur unless the receiver FIFO has empty space. Thus, overrun errors are eliminated during hardware flow control. If not enabled, overrun errors occur if the transmit data rate exceeds the receive FIFO latency.

- Automatic RTS

  Automatic RTS data flow control originates in the receiver block (see Figure 8). The receiver FIFO trigger levels used in automatic RTS are stored in the TCR. $\overline{\text{RTS}}$ is active if the RX FIFO level is below the HALT trigger level in TCR[3:0]. When the receiver FIFO HALT trigger level is reached, $\overline{\text{RTS}}$ is deasserted. The sending device (for example, another UART) may send an additional byte after the trigger level is reached because it may not recognize the deassertion of $\overline{\text{RTS}}$ until it has begun sending the additional byte. $\overline{\text{RTS}}$ is automatically reasserted once the receiver FIFO reaches the RESUME trigger level programmed via TCR(7:4). This reassertion requests the sending device to resume transmission.

- Automatic CTS

  The transmitter circuitry checks $\overline{CTS}$ before sending the next data byte. When $\overline{CTS}$ is active, the transmitter sends the next byte. To stop the transmitter from sending the following byte, $\overline{CTS}$ must be deasserted before the middle of the last stop bit that is currently being sent. The automatic CTS function reduces interrupts to the host system. When automatic CTS flow control is enabled, the $\overline{CTS}$ state changes need not trigger host interrupts because the device automatically controls its own transmitter. Without automatic CTS, the transmitter sends any data present in the transmit FIFO and a receiver overrun error can result.

## 5.10 Software Flow Control

Software flow control is enabled through the enhanced feature register (EFR) and the modem control register (MCR). Different combinations of software flow control can be enabled by setting different combinations of EFR[3-0].

There are two other enhanced features related to software flow control:

- XON any function [MCR(5)]: Operation resumes after receiving any character after recognizing the XOFF character. The XON-any character is written into the RX FIFO even if it is a software flow character.
- Special character [EFR(5)]: Incoming data is compared to XOFF2. Detection of the special character sets the XOFF interrupt [IIR(4)] but does not halt transmission. The XOFF interrupt is cleared by a read of the interrupt identification register (IIR). The special character is transferred to the RX FIFO.

### 5.10.1 RX

When software flow control operation is enabled, the UART compares incoming data with XOFF1/2 programmed characters (in certain cases XOFF1 and XOFF2 must be received sequentially). When the correct XOFF characters are received, transmission is halted after completing transmission of the current character. XOFF detection also sets IIR[4] (if enabled via IER[5]) and causes nIRQ to go low.

To resume transmission, an XON1/2 character must be received (in certain cases XON1 and XON2 must be received sequentially). When the correct XON characters are received, IIR[4] is cleared and the XOFF interrupt disappears.

If a parity, framing, or break error occurs while receiving a software flow control character, this character is treated as normal data and is written to the RX FIFO.

When XON-any and special character detect are disabled and software flow control is enabled, no valid XON or XOFF characters are written to the RX FIFO. For example, if EFR[1:0] = 10, then received XON1 and XOFF1 characters are not written to the RX FIFO.

When pairs of software flow characters are programmed to be received sequentially (EFR[1:0] = 11), they are not written to the RX FIFO if they are received sequentially. However, received XON1/XOFF1 characters must be written to the RX FIFO if the subsequent character is not XON2/XOFF2.

### 5.10.2 TX

With XOFF1, two characters are transmitted when the RX FIFO has passed the programmed trigger level TCR[3:0].

With XON1, two characters are transmitted when the RX FIFO reaches the trigger level programmed via TCR[7:4].

After an XOFF character has been sent, if software flow control is disabled, the module transmits XON characters automatically to enable normal transmission to proceed.

The transmission of XOFF/XON follows the exact same protocol as transmission of an ordinary byte from the FIFO. This means that even if the word length is set to be 5, 6, or 7 characters, the 5, 6, or 7 least significant bits of XOFF1,2/XON1,2 are transmitted. (The transmission of 5, 6, or 7 bits of a character is seldom done, but this functionality is included to maintain compatibility with earlier designs.)

It is assumed that software flow control and hardware flow control are never enabled simultaneously.

## 5.11 Autobauding Mode

In autobaud mode, the UART can extract transfer characteristics (speed, length and parity) from an attention (AT) command. These characteristics are used to receive data following an AT and to send data.

Here are valid AT commands:

```
AT  DATA <CR>
at DATA  <CR>
A  /a/
```

A line break during the acquisition of the sequence AT is not recognized and echo functionality is not implemented in the hardware.

A/ and a/ are not used to extract characteristics, but they must be recognized because of their special meaning. They instruct the software to repeat the last received AT command; therefore, an a/ (or A/) always comes after an AT and transfer characteristics are not expected to change between an AT and an a/ (or A/).

When a valid AT command is received, it and all subsequent data are saved into the FIFO, including the final CR (0x0D). Then the autobaud state machine waits for the next valid AT command. If an a/ (or A/) is received, the a/ (or A/) is saved into the FIFO and the state machine waits for next valid AT command.

The following settings are allowed in autobaud mode:

- Speed:

  115.2K baud, 57.6K baud, 38.4K baud, 28.8K baud, 19.2K baud, 14.4K baud, 9.6K baud, 4.8K baud, 2.4K baud, or 1.2K baud.

- Length: 7 or 8 bits

- Parity: Odd, even, or space

Combination 7-bit space parity is forbidden.

The method to identify the speed is:

1) Detect the transition 1->0 on the received data. This happens as soon as a stop-to-start-bit transition occurs. The transition is valid after a majority vote on three sampling periods.

2) Sample the start bit duration with 115 200 *16 Hz clock frequency as long as there is no rising edge. A transition 0->1 is considered as valid after a majority vote on three sampling periods.

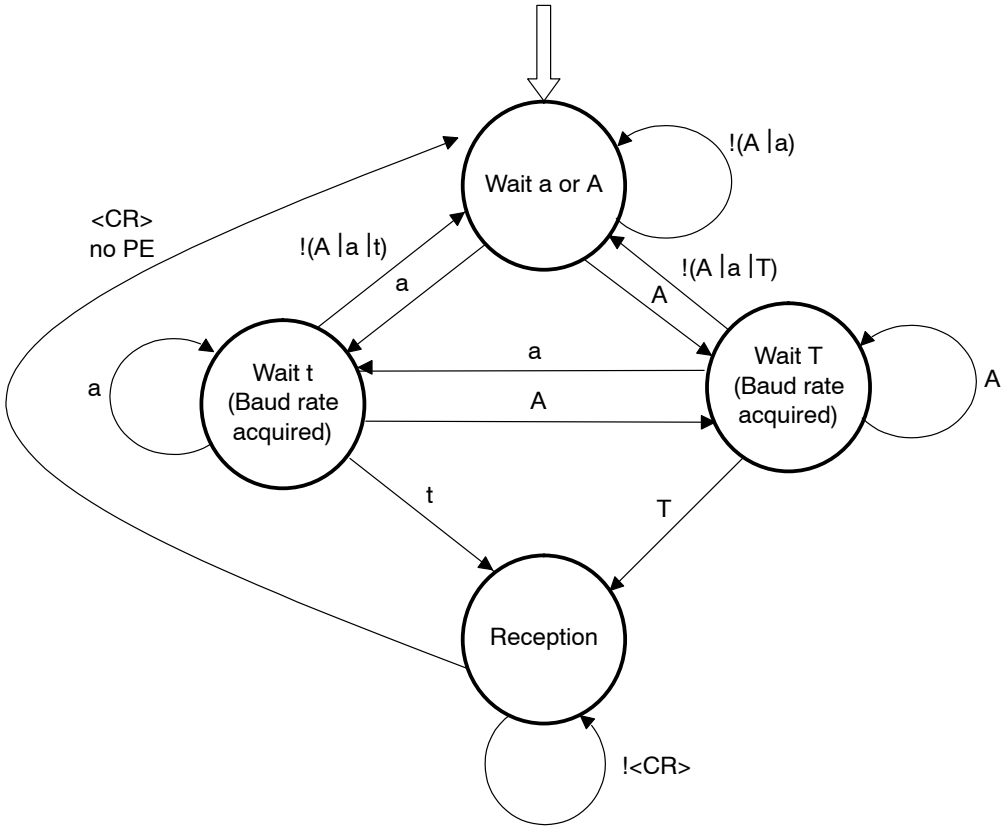
3) Compare the sampled value with a table. If the sampled value is outside a valid range, an error is reported (no speed identified), and the hardware goes back to the first state (1).

4) Otherwise, the first data bit in the received register (for serial to parallel conversion) is stored and goes to frame format identification.

5) The next received bits are sampled according to the programmed baud rate. After the reception of seven bits, the speed identification must be restarted, because several a or A characters before a valid t or T character may be received.

Autobaud mode is selected when MDR1[2:0] = 010. In the UART autobaud mode, DLL, DLH, and LCR[5:0] settings are not used. Instead, the UASR is updated with the configuration detected by the autobauding logic (see Figure 13).

*Figure 13. Autobaud State Machine*



# 6 UART/Autobaud Configuration Example

This section specifies the programming stages to operate one UART module with FIFO, interrupt, and no DMA capabilities. It is a three-step procedure that ensures quick start of these modules (it does not cover all UART module features and performance):

6) Software reset of the module (interrupts, status and controls
7) FIFO configuration and enable
8) Baud rate data and stop configuration

This procedure is programming-language agnostic.

## 6.1 UART SW Reset

The goal is to clear the IER and MCR registers, remove the UART breaks (LCR[6] = 0), and put the module in reset (MDR1[2:0] = 0x3).

1) To write into both the IER and MCR registers, set EFR[4] to 1.
2) To enable access to the EFR register, write 0xBF to the LCR register:
   - LCR = 0xBF
   - EFR[4] = 1
   - LCR = 0x80 (access to IER and MCR allowed)
   - IER = 0x00
   - MCR = 0x00
   - LCR[6] = 0 (UART breaks removed)
   - MDR1 = 0x03 (UART in reset)

## 6.2 UART FIFO Configuration

The goal is to set the trigger level for the halt/restore (TCR register), set the trigger level for the transmit/receive (TLR register), and configure the FIFO (FCR register).

Procedure:

1) To write into both the TLR and TCR registers, set the EFR[4] and MCR[6] to 1. To write into the FCR, set the EFR[4] to 1. Notice that EFR[4] = 1 has already been done in the software reset, so a simple write to MCR[6] is necessary.

2) Set the TCR TLR and FCR to the desired value.

3) Disable accesses to TCR, TLR, and FCR to avoid any further undesired writes to these registers:

   - LCR = 0xBF (provides EFR access)
   - EFR[4] = 0
   - LCR[7] = 0
   - MCR[6] = 0

## 6.3 Baud Rate Data and Stop Configurations

The goal is to configure the UART data, stop (LCR register) baud rate (DLH and DLL registers), and enable the UART operation. If needed, interrupt capability configuration can be added right before the UART is enabled.

1) Input clock is 12 MHz, so set OSC_12M_SEL to 1.
2) Set the LCR to the desired value.
3) Set the LCR[7] to 1 (access to DLH and DLL registers).
4) Set the DLH and DLL LCR[7] = 0 (remove access to the DLH and DLL registers).
5) Set the IER to the desired value (set interrupts).
6) MDR1[2:0] = 0
7) Enable the UART without autobauding.

# 7 UART/IrDA Control and Status Registers

Each register is selected using a combination of address and some LCR register bit settings, as shown in Table 43.

The UART3 is accessible as follows:

- MPU (32-bit aligned byte address) from the following base address:
  - UART3: 0xFFFB 9800
- DSP (16-bit aligned word address) from the following base address:
  - UART3: 0x00CC00

*Table 43. UART IrDA Register Program*

| MPU Byte Offset | DSP Byte Offset | Registers | | | | | |
|---|---|---|---|---|---|---|---|
| | | LCR[7] = 0 | | LCR[7] = 1 LCR[7:0] ≠ 0xBF | | LCR[7:0] = 0xBF | |
| | | Read | Write | Read | Write | Read | Write |
| 0x00 | 0x00 | RHR | THR | DLL | DLL | DLL | DLL |
| 0x04 | 0x02 | IER† | IER† | DLH | DLH | DLH | DLH |
| 0x08 | 0x04 | IIR | FCR‡ | IIR | FCR‡ | EFR | EFR |
| 0x0C | 0x06 | LCR | LCR | LCR | LCR | LCR | LCR |
| 0x10 | 0x08 | MCR‡ | MCR‡ | MCR‡ | MCR‡ | XON1/ADDR1 | XON1/ADDR1 |
| 0x14 | 0x0A | LSR | - | LSR | - | XON2/ADDR2 | XON2/ADDR2 |
| 0x18 | 0x0C | MSR/TCR§ | TCR§ | MSR/TCR§ | TCR§ | XOFF1/TCR§ | XOFF1/TCR§ |
| 0x1C | 0x0E | SPR/TLR§ | SPR/TLR§ | SPR/TLR§ | SPR/TLR§ | XOFF2/TLR§ | XOFF2/TLR§ |
| 0x20 | 0x10 | MDR1 | MDR1 | MDR1 | MDR1 | MDR1 | MDR1 |
| 0x24 | 0x12 | MDR2 | MDR2 | MDR2 | MDR2 | MDR2 | MDR2 |
| 0x28 | 0x14 | SFLSR | TXFLL | SFLSR | TXFLL | SFLSR | TXFLL |
| 0x2C | 0x16 | RESUME | TXFLH | RESUME | TXFLH | RESUME | TXFLH |
| 0x30 | 0x18 | SFREGL | RXFLL | SFREGL | RXFLL | SFREGL | RXFLL |
| 0x34 | 0x1A | SFREGH | RXFLH | SFREGH | RXFLH | SFREGH | RXFLH |
| 0x38 | 0x1C | BLR | BLR | - | - | - | - |

† In UART mode, IER[7:4] can only be written when EFR[4] = 1. In SIR mode, EFR[4] has no impact on the access to IER[7:4].
‡ MCR[7:5] and FCR[5:4] can only be written when EFR[4] = 1.
§ Transmission control register (TCR) and trigger level register (TLR) are accessible only when EFR[4] = 1 and MCR[6] = 1.

*Table 43. UART IrDA Register Program (Continued)*

| MPU Byte Off-set | DSP Byte Off-set | Registers | | | | | |
|---|---|---|---|---|---|---|---|
| | | LCR[7] = 0 | | LCR[7] = 1 LCR[7:0] ≠ 0xBF | | LCR[7:0] = 0xBF | |
| | | Read | Write | Read | Write | Read | Write |
| 0x3C | 0x1E | ACREG | ACREG | DIV1.6 | DIV1.6 | DIV1.6 | DIV1.6 |
| 0x40 | 0x20 | SCR | SCR | SCR | SCR | SCR | SCR |
| 0x44 | 0x22 | SSR | - | SSR | - | SSR | - |
| 0x48 | 0x24 | EBLR | EBLR | - | - | - | - |
| 0x4C | 0x26 | | OSC_12M_ SEL | - | - | - | - |
| 0x50 | 0x28 | MVR | - | MVR | - | MVR | - |

† In UART mode, IER[7:4] can only be written when EFR[4] = 1. In SIR mode, EFR[4] has no impact on the access to IER[7:4].
‡ MCR[7:5] and FCR[5:4] can only be written when EFR[4] = 1.
§ Transmission control register (TCR) and trigger level register (TLR) are accessible only when EFR[4] = 1 and MCR[6] = 1.

Table 44 lists the UART/IrDA registers. Table 45 through Table 87 describe the register bits.

*Table 44. UART/IrDA Registers*

| Register | Description | Access |
|---|---|---|
| RHR | Receive holding | 8 bits R |
| THR | Transmit holding | 8 bits W |
| FCR | FIFO control | 8 bits W |
| SCR | Supplementary control | 8 bits R/W |
| LCR | Line control | 8 bits R/W |
| LSR | UART mode (LSR) | 8 bits R |
| SSR | Supplementary status | 8 bits R |
| MCR | Modem control | 8 bits R/W |
| MSR | Modem status | 8 bits R |
| IER | Interrupt enable (IER) | 8 bits R/W |
| IIR | Interrupt identification (IIR) | 8 bits R |

*Table 44. UART/IrDA Registers (Continued)*

| Register | Description | Access |
|---|---|---|
| EFR | Enhanced feature | 8 bits R/W |
| XON1/ADDR1 | XON1/Address 1 | 8 bits R/W |
| XON2/ADDR2 | XON2/Address 2 | 8 bits R/W |
| XOFF1 | XOFF1 | 8 bits R/W |
| XOFF2 | XOFF2 | 8 bits R/W |
| SPR | Scratchpad | 8 bits R/W |
| DLL | Divisor latch low | 8 bits R/W |
| DLH | Divisor latch high | 8 bits R/W |
| TCR | Transmission control | 8 bits R/W |
| TLR | Trigger level | 8 bits R/W |
| MDR1 | Mode definition 1 | 8 bits R/W |
| MDR2 | Mode definition 2 | 8 bits R/W |
| TXFLL | Transmit frame length low | 8 bits W |
| TXFLH | Transmit frame length high | 8 bits W |
| RXFLL | Received frame length low | 8 bits W |
| RXFLH | Received frame length high | 8 bits W |
| SFLSR | Status FIFO line status | 8 bits R |
| RESUME | Resume | 8 bits R |
| SFREGL | Status FIFO low | 8 bits R |
| SFREGH | Status FIFO high | 8 bits R |
| BLR | BOF control | 8 bits R/W |
| EBLR | BOF length | 8 bits R/W |
| DIV16 | DIV1.6 | 8 bits R/W |
| ACREG | Auxiliary control | 8 bits R/W |
| OSC_12M_SEL | OSC 12-MHz select | 8 bits W |
| MVR | Module version | 8 bits R |

The receiver section consists of the receiver holding register (RHR) and the receiver shift register. The RHR is actually a 64-byte FIFO. The receiver shift register receives serial data from RX input. The data is converted to parallel data and moved to the RHR. If the FIFO is disabled, location zero of the FIFO is used to store the single data character. If an overflow occurs, data in the RHR is not overwritten.

*Table 45. Receive Holding Register (RHR) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–0 | RHR | Receive holding register | R | Undefined |

The transmitter section consists of the transmit holding register (THR) and the transmit shift register. The THR is actually a 64-byte FIFO. The host (MPU or DSP) writes data to the THR. The data is placed into the transmit shift register where it is shifted out serially on the TX output. If the FIFO is disabled, location 0 of the FIFO is used to store the data.

*Table 46. Transmit Holding Register (THR) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–0 | THR | Transmit holding register | W | Undefined |

*Table 47. FIFO Control (FCR) Register Field Descriptions*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 7–6 | RX_FIFO_TRIG | | Sets the trigger level for the RX FIFO:<br>If SCR[7} = 0 and TLR[7:4] = 0000: | W | 00 |
| | | 00 | 8 characters | | |
| | | 01 | 16 characters | | |
| | | 10 | 56 characters | | |
| | | 11 | 60 characters | | |
| | | | If SCR[7] = 0 and TLR[7:4] ≠ 0000, RX_FIFO_TRIG is not considered. | | |
| | | 1 | RX_FIFO_TRIG is two LSBs of the trigger level (1-63 on 6 bits) with granularity of 1. | | |
| 5–4 | TX_FIFO_TRIG | | Sets the trigger level for the TX FIFO:<br>If SCR6 = 0 and TLR3:0 = 0000:<br>00: 8 characters<br>01: 16 characters<br>10: 32 characters<br>11: 56 characters | W | 00 |
| | | 00 | 8 characters | | |
| | | 01 | 16 characters | | |
| | | 10 | 56 characters | | |
| | | 11 | 60 characters | | |
| | | | If SCR[6]=1, TX_FIFO_TRIG is two LSBs of the trigger level (1-63 on 6 bits) with granularity of 1. | | |

**Notes:** 1) Bits 4 and 5 can only be written when EFR[4] = 1.
2) Bits 0 to 3 can be changed only when baud clock is not running (DLL and DLH set to 0).
3) See Table 36 for FCR[5:4] setting restriction when SCR[6] = 1.
4) See Table 37 for FCR[7:6] setting restriction when SCR[7] = 1.

*Table 47. FIFO Control (FCR) Register Field Descriptions (Continued)*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 3 | DMA_MODE | 0 | DMA_MODE 0 (no DMA) | W | 0 |
| | | 1 | DMA_MODE 1 (UART_nDMA_REQ0 in TX, UART_nDMA_REQ1 in RX)<br>This register is considered if SCR[0] = 0. | | |
| 2 | TX_FIFO_CLEAR | 0 | No change | W | 0 |
| | | 1 | Clears the transmit FIFO and resets its counter logic to zero. Returns to zero after clearing FIFO. | | |
| 1 | RX_FIFO_CLEAR | 0 | No change | W | 0 |
| | | 1 | Clears the receive FIFO and resets its counter logic to zero. Returns to zero after clearing FIFO. | | |
| 0 | FIFO_EN | 0 | Disables the transmit and receive FIFOs | W | 0 |
| | | 1 | Enables the transmit and receive FIFOs | | |

**Notes:** 1) Bits 4 and 5 can only be written when EFR[4] = 1.
2) Bits 0 to 3 can be changed only when baud clock is not running (DLL and DLH set to 0).
3) See Table 36 for FCR[5:4] setting restriction when SCR[6] = 1.
4) See Table 37 for FCR[7:6] setting restriction when SCR[7] = 1.

*Table 48. Supplementary Control Register (SCR) Field Descriptions*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 7 | RX_TRIG_GRANU1 | 0 | Disables the granularity of 1 for the trigger RX level | R/W | 0 |
| | | 1 | Enables the granularity of 1 for the trigger RX level | | |
| 6 | TX_TRIG_GRANU1 | 0 | Disables the granularity of 1 for the trigger TX level | R/W | 0 |
| | | 1 | Enables the granularity of 1 for the trigger TX level | | |

*Table 48. Supplementary Control Register (SCR) Field Descriptions (Continued)*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 5 | DSR_IT | 0 | Disables the $\overline{DSR}$ interrupt | R/W | 0 |
| | | 1 | Enables the $\overline{DSR}$ interrupt | | |
| 4 | RX_CTS_DSR_WAKE_UP_ENABLE | 0 | Disables the wake up interrupt and clears SSR1 | R/W | 0 |
| | | 1 | Waits for a falling edge of pins RX, $\overline{CTS}$ or $\overline{DSR}$ to generate an interrupt | | |
| 3 | TX_EMPTY_CTL_IT | 0 | Normal mode for the THR interrupt (see Table 55) | R/W | 0 |
| | | 1 | The THR interrupt is generated when the TX FIFO and TX shift register are empty. | | |
| 2–1 | DMA_MODE_2 | | Used to specify the DMA mode, valid if SCR[0] = 1 | R/W | 00 |
| | | 00 | DMA mode 0 (no DMA) | | |
| | | 01 | DMA mode 1 (UART_nDMA_REQ0 in TX, UART_nDMA_REQ1 in RX) | | |
| | | 10 | DMA mode 2 (UART_nDMA_REQ0 in RX) | | |
| | | 11 | DMA mode 3 (UART_nDMA_REQ0 in TX) | | |
| 0 | DMA_MODE_CTL | 0 | The DMA_MODE is set with FCR[3]. | R/W | 0 |
| | | 1 | The DMA_MODE is set with SCR[2:1]. | | |

**Note:** Bit 4 enables the wake-up interrupt, but this interrupt is not mapped on the IIR register. Therefore, when an interrupt occurs and if there is no interrupt pending in IIR, the SSR[1] must be checked. To clear the wake-up interrupt, the SCR[4] must be reset to 0.

The line control register (LCR)[6:0] defines parameters of the transmission and reception.

*Table 49. Line Control Register (LCR) Field Descriptions*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 7 | DIV_EN | 0 | Normal operating condition | R/W | 0 |
| | | 1 | Divisor latch enable. Allows access to the DLL, DLH, and other registers (refer to the registers mapping). | | |
| 6 | BREAK_EN | | Break control bit. | R/W | 0 |
| | | 0 | Normal operating condition | | |
| | | 1 | Forces the transmitter output to go low to alert the communication terminal | | |
| 5 | PARITY_TYPE2 | | Selects the forced parity format (if LCR[3] = 1). | R/W | 0 |
| | | | If LCR[5] = 1 and LCR[4] = 0, the parity bit is forced to 1 in the transmitted and received data. | | |
| | | | If LCR[5] = 1 and LCR[4] = 1, the parity bit is forced to 0 in the transmitted and received data. | | |
| 4 | PARITY_TYPE1 | 0 | Odd parity is generated (if bit 3 = 1). | R/W | 0 |
| | | 1 | Even parity is generated (if bit 3 = 1). | | |
| 3 | PARITY_EN | 0 | No parity | R/W | 0 |
| | | 1 | Parity is generated during transmission, and the receiver checks for received parity. | | |
| 2 | NB_STOP | | Specifies the number of stop bits: | R/W | 0 |
| | | 0 | 1 stop bits (word length = 5, 6, 7, 8) | | |
| | | 1 | 1.5 stop bits (word length = 5) | | |
| | | 1 | 2 stop bits (word length = 6, 7, 8) | | |

**Note:** As soon as LCR[6] is set to 1, the RX line is forced to 0 and remains in this state as long as LCR[6] = 1.

*Table 49. Line Control Register (LCR) Field Descriptions (Continued)*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 1–0 | CHAR_LENGTH | | Specify the word length to be transmitted or received. | R/W | 00 |
| | | 00 | 5 bits | | |
| | | 01 | 6 bits | | |
| | | 10 | 7 bits | | |
| | | 11 | 8 bits | | |

**Note:** As soon as LCR[6] is set to 1, the RX line is forced to 0 and remains in this state as long as LCR[6] = 1.

*Table 50. UART Mode Line Status Register (UART_LSR) Field Descriptions*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 7 | RX_FIFO_STS | 0 | Normal operation. | R | 1 |
| | | 1 | At least one parity error, framing error, or break indication in the receiver FIFO. The bit is cleared when no more errors are present in FIFO. | | |
| 6 | TX_SR_E | 0 | Transmitter hold and shift registers are not empty. | R | 1 |
| | | 1 | Transmitter hold and shift registers are empty. | | |
| 5 | TX_FIFO_E | 0 | Transmit hold register is not empty | R | 1 |
| | | 1 | Transmit hold register is empty. The processor can now load up to 64 bytes of data into the THR if the TX FIFO is enabled. | | |
| 4 | RX_BI | 0 | No break condition | R | 0 |
| | | 1 | A break was detected while the data being read from the RX FIFO was being received (RX input was low for one character time frame). | | |

*Table 50. UART Mode Line Status Register (UART_LSR) Field Descriptions (Continued)*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 3 | RX_FE | 0 | No framing error exists in data being read from the RX FIFO | R | 0 |
| | | 1 | A framing error occurred in data being read from the RX FIFO (received data did not have a valid stop bit). | | |
| 2 | RX_PE | 0 | No parity error exists in data being read from the RX FIFO | R | 0 |
| | | 1 | A parity error exists in data being read from the RX FIFO | | |
| 1 | RX_OE | 0 | No overrun error exists | R | 0 |
| | | 1 | An overrun error has occurred. Set when the character held in the receive shift register is not transferred to the RX FIFO. This case can occur only when the receive FIFO is full. | | |
| 0 | RX_FIFO_E | 0 | No data is in the receive FIFO | R | 0 |
| | | 1 | At least one data character is in the RX_FIFO | | |

When the line status register (LSR) is read, LSR[4:2] reflects the error bits (BI, FE, PE) of the character at the top of the RX FIFO (next character to be read). Therefore, reading the LSR and then reading the RHR identifies the errors in a character.

LSR[7] is set when there is an error anywhere in the RX FIFO and is cleared only when there are no more errors remaining in the FIFO.

**Note:**

Reading the LSR does not cause an increment of the RX FIFO read pointer. The RX FIFO read pointer is incremented by reading the RHR.

*Table 51. SIR Mode Line Status Register (SIR_LSR) Field Descriptions*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 7 | THR_EMPTY | 0 | Transmit hold register is not empty. | R | 1 |
| | | 1 | Transmit hold register is empty. The processor can now load up to 64 bytes of data into the THR if the TX FIFO is enabled. | | |
| 6 | STS_FIFO_FULL | 0 | Status FIFO is not full | R | 0 |
| | | 1 | Status FIFO is full | | |
| 5 | RX_LAST_BYTE | 0 | Did not receive last byte of a frame from the FIFO | R | 0 |
| | | 1 | Received last byte from the FIFO. This bit is set when the last byte of a frame is read. Used to determine the frame boundary. Cleared by first reading the last received byte, then reading the SIR_LSR register. | | |
| 4 | FRAME_TOO_LONG | 0 | No frame-too-long error is in the frame | R | 0 |
| | | 1 | A frame-too-long error is in the frame at the top of the STATUS FIFO next character to be read. This bit is set to 1 when a frame exceeding the maximum length (set by RXFLH and RXFLL registers) has been received. When this error is detected, the current frame reception is terminated. Reception is stopped until the next START flag is detected. | | |
| 3 | ABORT | 0 | No abort pattern error is in the frame | R | 0 |
| | | 1 | Abort pattern received | | |
| 2 | CRC | 0 | No CRC error is in the frame | R | 0 |
| | | 1 | CRC error is in the frame at the top of the STATUS FIFO (next character to be read) | | |

*Table 51. SIR Mode Line Status Register (SIR_LSR) Field Descriptions (Continued)*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 1 | STS_FIFO_E | 0 | Status FIFO is not empty | R | 1 |
| | | 1 | Status FIFO is empty | | |
| 0 | RX_FIFO_E | 0 | At least one data character is in the RX_FIFO | R | 1 |
| | | 1 | No data is in the receive FIFO | | |

When the LSR is read, LSR[4:2] reflects the error bits [FL, CRC, ABORT] of the frame at the top of the STATUS FIFO (next frame status to be read). In SIR mode, the LSR bits [4:2] reflect the same values as the SFLSR bits [3:1].

*Table 52. Supplementary Status Register (SSR) Field Descriptions*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 7–2 | – | | Reserved | R | 000000 |
| 1 | RX_CTS_DSR_WAKE_UP_STS | 0 | No falling edge event is on RX, $\overline{CTS}$ and $\overline{DSR}$ | R | 0 |
| | | 1 | A falling edge occurred on RX, $\overline{CTS}$ or $\overline{DSR}$. | | |
| 0 | TX_FIFO_FULL | 0 | TX FIFO is not full | R | 0 |
| | | 1 | TX FIFO is full | | |

**Note:** Bit 1 is reset only when SCR[4] is reset to 0.

The modem control register (MCR)[3:0] controls the interface with the modem, data set, or peripheral device that is emulating the modem.

*Table 53. Modem Control Register (MCR) Field Descriptions*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 7 | CLKSEL | 0 | No action | R/W | 0 |
| | | 1 | Divide clock input by 4. | | |

**Note:** Bits 5, 6, and 7 can be written only when EFR[4] = 1.

*Table 53. Modem Control Register (MCR) Field Descriptions (Continued)*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 6 | TCR_TLR | 0 | No action | R/W | 0 |
| | | 1 | Enables access to the TCR and TLR registers | | |
| 5 | XON_EN | 0 | Disable XON for any function | R/W | 0 |
| | | 1 | Enable XON for any function | | |
| 4 | LOOPBACK_EN | 0 | Normal operating mode | R/W | 0 |
| | | 1 | Enable local loopback mode (internal). In this mode the MCR[3:0] signals are looped back into MSR[7:4]. The transmit output is looped back to the receive input internally. | | |
| 3 | RESERVED | | Reserved. This bit should always be written as 0. | R/W | 0 |
| | | 1 | In loopback mode, forces the IRQ outputs to inactive state | | |
| 2 | RESERVED | | Reserved. This bit should always be written as 0. | R/W | 0 |
| 1 | RTS | 0 | Forces the $\overline{\text{RTS}}$ output to inactive (high) | R/W | 0 |
| | | 1 | Forces the $\overline{\text{RTS}}$ output to active (low) | | |
| | | | In loopback mode controls MSR[4] | | |
| | | | If automatic RTS is enabled, the $\overline{\text{RTS}}$ output is controlled by hardware flow control. | | |
| 0 | DTR | 0 | Forces the $\overline{\text{DTR}}$ output to inactive (high) | R/W | 0 |
| | | 1 | Forces the $\overline{\text{DTR}}$ output to active (low) | | |

**Note:** Bits 5, 6, and 7 can be written only when EFR[4] = 1.

The modem status register (MSR) provides information about the current state of the control lines from the modem, data set, or peripheral device to the host (MPU or DSP). It also indicates when a control input from the modem changes state.

*Table 54. Modem Status Register (MSR) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–6 | RESERVED | Reserved | R | |
| 5 | NDSR_STS | This bit is the complement of the $\overline{DSR}$ input.<br>In loopback mode, it is equivalent to MCR[0]. | R | Input signal |
| 4 | NCTS_STS | This bit is the complement of the $\overline{CTS}$ input.<br>In loopback mode, it is equivalent to MCR[1]. | R | Input signal |
| 3–2 | RESERVED | Reserved | R | 0 |
| 1 | DSR_STS | 1: Indicates that $\overline{DSR}$ input (or MCR[0] in loopback) has changed state. Cleared on a read. | R | 0 |
| 0 | CTS_STS | 1: Indicates that $\overline{CTS}$ input (or MCR[1] in loopback) has changed state. Cleared on a read. | R | 0 |

The interrupt enable register (IER) in UART mode can be programmed to enable/disable any of the following interrupts:

- ❑ Receiver error
- ❑ RHR
- ❑ THR
- ❑ XOFF
- ❑ $\overline{CTS}$/$\overline{RTS}$ change of state from low to high

Each of these interrupts can be enabled/disabled individually. There is also a sleep mode enable bit.

*Table 55. UART Mode Interrupt Enable Register (UART_IER) Field Descriptions*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 7 | CTS_IT | 0 | Disables the $\overline{CTS}$ interrupt | R/W | 0 |
| | | 1 | Enables the $\overline{CTS}$ interrupt | | |
| 6 | RTS_IT | 0 | Disables the $\overline{RTS}$ interrupt | R/W | 0 |
| | | 1 | Enables the $\overline{RTS}$ interrupt | | |
| 5 | XOFF_IT | 0 | Disables the XOFF interrupt | R/W | 0 |

**Note:** Bits 4, 5, 6, and 7 can only be written when EFR[4] = 1.

*Table 55. UART Mode Interrupt Enable Register (UART_IER) Field Descriptions (Continued)*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| | | 1 | Enables the XOFF interrupt | | |
| 4 | SLEEP_MODE | 0 | Disables sleep mode | R/W | 0 |
| | | 1 | Enables sleep mode (stop baud rate clock when the module is inactive) | | |
| 3 | MODEM_STS_IT | 0 | Disables the modem status register interrupt | R/W | 0 |
| | | 1 | Enables the modem status register interrupt | | |
| 2 | LINE_STS_IT | 0 | Disables the receiver line status interrupt | R/W | 0 |
| | | 1 | Enables the receiver line status interrupt | | |
| 1 | THR_IT | 0 | Disables the THR interrupt | R/W | 0 |
| | | 1 | Enables the THR interrupt | | |
| 0 | RHR_IT | 0 | Disables the RHR interrupt and time out interrupt | R/W | 0 |
| | | 1 | Enables the RHR interrupt and time out interrupt | | |

**Note:** Bits 4, 5, 6, and 7 can only be written when EFR[4] = 1.

The interrupt enable register (IER) in SIR mode can be programmed to enable/disable any of the following interrupts:

- ❏ Received error
- ❏ LSR
- ❏ TX underrun
- ❏ Status FIFO
- ❏ RX overrun
- ❏ Last byte in RX FIFO
- ❏ THR
- ❏ RHR

Each of these interrupts can be enabled/disabled individually. There is also a sleep mode enable bit.

*Table 56. SIR Mode Interrupt Enable Register (SIR_IER) Field Descriptions*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 7 | EOF_IT | 0 | Disables the received EOF interrupt | R/W | 0 |
| | | 1 | Enables the received EOF interrupt | | |
| 6 | LINE_STS_IT | 0 | Disables the receiver line status interrupt | R/W | 0 |
| | | 1 | Enables the receiver line status interrupt | | |
| 5 | TX_UNDERRUN_IT | 0 | Disables the TX underrun interrupt | R/W | 0 |
| | | 1 | Enables the TX underrun interrupt | | |
| 4 | STS_FIFO_TRIG_IT | 0 | Disables the status FIFO trigger level interrupt | R/W | 0 |
| | | 1 | Enables the status FIFO trigger level interrupt | | |
| 3 | RX_OVERRUN_IT | 0 | Disables the RX overrun interrupt | R/W | 0 |
| | | 1 | Enables the RX overrun interrupt | | |
| 2 | LAST_RX_BYTE_IT | 0 | Disables the last byte of the frame in the RX FIFO interrupt | R/W | 0 |
| | | 1 | Enables the last byte of the frame in the RX FIFO interrupt | | |
| 1 | THR_IT | 0 | Disables the THR interrupt | R/W | 0 |
| | | 1 | Enables the THR interrupt | | |
| 0 | RHR_IT | 0 | Disables the RHR interrupt | R/W | 0 |
| | | 1 | Enables the RHR interrupt | | |

The interrupt identification register (IIR) is a read-only register, which provides the source of the interrupt in a prioritized manner.

*Table 57. UART Mode Interrupt Identification Register (UART_IIR) Field Descriptions*

| Bits | Field | Value | Description | | | | | | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|---|---|
| 7–6 | FCR_MIRROR | | Mirror the contents of FCR(0) on both bits. | | | | | | R | 00 |
| 5–1 | IT_TYPE | | **Priority** | **5** | **4** | **3** | **2** | **1** | **Source** | R | 00000 |
| | | | 1 | 0 | 0 | 0 | 1 | 1 | Receiver line status error | | |
| | | | 2 | 0 | 0 | 1 | 1 | 0 | RX time-out | | |
| | | | 2 | 0 | 0 | 0 | 1 | 0 | RHR interrupt | | |
| | | | 3 | 0 | 0 | 0 | 0 | 1 | THR interrupt | | |
| | | | 4 | 0 | 0 | 0 | 0 | 0 | Modem interrupt | | |
| | | | 5 | 0 | 1 | 0 | 0 | 0 | XOFF/special character | | |
| | | | 6 | 1 | 0 | 0 | 0 | 0 | CTS, RTS, DSR change state from active (low) to inactive (high) | | |
| 0 | IT_PENDING | 0 | An interrupt is pending (nIRQ active). | | | | | | | R | 1 |
| | | 1 | No interrupt is pending (nIRQ inactive). | | | | | | | | |

The $\overline{IRQ}$ output is activated whenever one of the 8 interrupts is active.

*Table 58. SIR Mode Interrupt Identification Register (SIR_IIR) Field Descriptions*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 7 | EOF_IT | 0 | Received EOF interrupt is inactive | R | 0 |
| | | 1 | Received EOF interrupt is active | | |
| 6 | LINE_STS_IT | 0 | Receiver line status interrupt is inactive | R | 0 |
| | | 1 | Receiver line status interrupt is active | | |
| 5 | TX_UE_IT | 0 | TX underrun interrupt is inactive | R | 0 |
| | | 1 | TX underrun interrupt is active | | |
| 4 | STS_FIFO_IT | 0 | Status FIFO trigger level interrupt is inactive | R | 0 |
| | | 1 | Status FIFO trigger level interrupt is active | | |
| 3 | RX_OE_IT | 0 | RX overrun interrupt is inactive | R | 0 |
| | | 1 | RX overrun interrupt is active | | |

*Table 58. SIR Mode Interrupt Identification Register (SIR_IIR) Field Descriptions (Continued)*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 2 | RX_FIFO_LAST_ BYTE_IT | 0 | Last byte of the frame in the RX FIFO interrupt is inactive | R | 0 |
| | | 1 | Last byte of the frame in the RX FIFO interrupt is active | | |
| 1 | THR_IT | 0 | THR interrupt is inactive | R | 0 |
| | | 1 | THR interrupt is active | | |
| 0 | RHR_IT | 0 | RHR interrupt is inactive | R | 0 |
| | | 1 | RHR interrupt is active | | |

The enhanced feature register (EFR) enables or disables the enhanced features, most of which only apply to the UART mode. But EFR[4] enables write accesses to FCR[5:4], the TX trigger level, which is also used in SIR mode.

*Table 59. Enhanced Feature Register (EFR) Field Descriptions*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 7 | AUTO_CTS_EN | | Automatic CTS enable bit | R/W | 0 |
| | | 0 | Normal operation | | |
| | | 1 | Automatic CTS flow control is enabled; that is, transmission is halted when the $\overline{CTS}$ pin is high (inactive). | | |
| 6 | AUTO_RTS_EN | | Automatic RTS enable bit | R/W | 0 |
| | | 0 | Normal operation | | |
| | | 1 | Automatic RTS flow control is enabled; that is, the $\overline{RTS}$ pin goes high (inactive) when the receiver FIFO HALT trigger level, TCR[3:0], is reached, and goes low (active) when the receiver FIFO restore transmission trigger level is reached. | | |

*Table 59. Enhanced Feature Register (EFR) Field Descriptions (Continued)*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 5 | SPECIAL_CHAR_ DETECT | 0 | Normal operation | R/W | 0 |
| | | 1 | Special character detect enable bit. Received data is compared with XOFF2 data. If a match occurs, the received data is transferred to the FIFO and the IIR bit 4 is set to 1 to indicate that a special character has been detected. | | |
| 4 | ENHANCED_EN | | Enhanced functions write enable bit. | R/W | 0 |
| | | 0 | Disables writing to IER bits 4-7, FCR bits 4-5, and MCR bits 5-7 | | |
| | | 1 | Enables writing to IER bits 4-7, FCR bits 4-5, and MCR bits 5-7 | | |
| 3–0 | SW_FLOW_CONTROL | | Combinations of software flow control can be selected by programming bits 3:0. See Table 60. | R/W | 0 |

*Table 60. EFR[0:3]: Software Flow Control Options*

| Bit 3 | Bit 2 | Bit 1 | Bit 0 | TX, RX Software Flow Controls |
|---|---|---|---|---|
| 0 | 0 | X | X | No transmit flow control |
| 1 | 0 | X | X | Transmit XON1, XOFF1 |
| 0 | 1 | X | X | Transmit XON2, XOFF2 |
| 1 | 1 | X | X | Transmit XON1, XON2: XOFF1, XOFF2† |
| X | X | 0 | 0 | No receive flow control |
| X | X | 1 | 0 | Receiver compares XON1, XOFF1 |
| X | X | 0 | 1 | Receiver compares XON2, XOFF2 |
| X | X | 1 | 1 | Receiver compares XON1, XON2: XOFF1, XOFF2† |

† In these cases, the XON1 and XON2 characters or the XOFF1 and XOFF2 characters must be transmitted/received sequentially with XON1/XOFF1 followed by XON2/XOFF2.

XON1 and XON2 must be set to different values if the software flow control is enabled.

*Table 61. XON1/Address Register 1 (XON1/ADDR1) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–0 | XON_WORD1 | Used to store the 8-bit XON1 character in UART mode and ADDR1 address 1 for SIR mode. | R/W | 0x00 |

*Table 62. XON2/Address Register 2 (XON2/ADDR2) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–0 | XON_WORD2 | Used to store the 8-bit XON2 character in UART mode and ADDR2 address 2 for SIR mode. | R/W | 0x00 |

*Table 63. XOFF1 Register (XOFF1) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–0 | XOFF_WORD1 | Used to store the 8-bit XOFF1 character in used in UART modes. | R/W | 0x00 |

*Table 64. XOFF2 Register (XOFF2) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–0 | XOFF_WORD2 | Used to store the 8-bit XOFF2 character in used in UART mode. | R/W | 0x00 |

The scratchpad register (SPR) does not control the module in anyway. It is a scratchpad register to be used by the programmer to hold temporary data.

*Table 65. Scratchpad Register (SPR) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–0 | SPR_WORD | Scratchpad register | R/W | 0x00 |

The two divisor latch registers (DLL and DLH) store the 16-bit divisor for generation of the baud clock in the baud rate generator. DLL stores the least significant part of the divisor. DLH stores the most significant part of the divisor.

DLL and DLH can only be written to before sleep mode is enabled (that is, before IER[4] is set).

*Table 66. Divisor Latch Low Register (DLL) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–0 | CLOCK_LSB | Used to store the 8-bit LSB divisor value | R/W | 0x00 |

*Table 67. Divisor Latch High Register (DLH) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–0 | CLOCK_MSB | Used to store the 8-bit MSB divisor value | R/W | 0x00 |

To achieve the required baud rate, the DLL/DLH latch control registers must be programmed with the integer part of the divisor value.

Choosing the appropriate divisor value:

UART: Divisor value = Operating Frequency/(16 x baud rate).

The input frequency of the UART IrDA must be fixed to the operating frequency (here 12 MHz; no CLKSEL bit setting), and the the OSC_12M_SEL bit must be set to be able to reach the desired baud rate. Setting OSC_12M_SEL to 1 enables turning on the 6.5 division factor. For instance, 12 MHz/16/6.5 = 115200 bps; if OSC_12M_SEL is not set, the reached baud rate is either 12 MHz/16/6 or 12 MHz/16/7, which are outside permitted tolerance.

The transmission control register (TCR) stores the receive FIFO threshold levels to start/stop transmission during hardware/software flow control.

*Table 68. Transmission Control Register (TCR) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–4 | RX_FIFO_TRIG_START | RCV FIFO trigger level to restore transmission (0-60) | R/W | 0000 |
| 3–0 | RX_FIFO_TRIG_HALT | RCV FIFO trigger level to halt transmission (0–60) | R/W | 1111 |

**Note:** Trigger levels from 0–60 bytes are available with a granularity of four (trigger level = 4 x [4-bit register value]).

The programmer must ensure that TCR[3:0] is greater than TCR[7:4] whenever automatic RTS or software flow control is enabled to avoid spurious operation of the device.

In FIFO interrupt mode with flow control, the programmer must also ensure that the trigger level to halt transmission is greater than or equal to the receive FIFO trigger level (either TLR[7:4] or FCR[7:6]); otherwise, FIFO operation stalls. In FIFO DMA mode with flow control, this concept does not exist because a DMA request is sent each time a byte is received.

The trigger level register (TLR) stores the programmable transmit and receive FIFO trigger levels used for the DMA and IRQ generation.

*Table 69. Trigger Level Register (TLR) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–4 | RX_FIFO_TRIG_DMA | RCV FIFO trigger level | R/W | 0000 |
| 3–0 | TX_FIFO_TRIG_DMA | Transmit FIFO trigger level | R/W | 0000 |

Table 70 and Table 71 summarize the different ways that can be used to set the trigger levels for the transmit FIFO and the receive FIFO, respectively.

*Table 70. Transmit FIFO Trigger Level Setting Summary*

| SCR[6] | TLR[3:0] | TX FIFO Trigger Level |
|---|---|---|
| 0 | 0000 | Defined by FCR[5:4] (either 8, 16, 32, and 56 characters) |
| 0 | ≠0000 | Defined by TLR[3:0] (from 4 to 60 characters with a granularity of 4 characters) |
| 1 | value | Defined by the concatenated value of TLR[3:0] and FCR[5:4] (from 1 to 63 characters with a granularity of 1 character).<br>The combination of TLR[3:0] = 0000 and FCR[5:4] = 00 (all zeros) is not supported (minimum one character required). All zeros result in unpredictable behavior. |

**Note:** The protocol to set the concatenation of TLR and FCR is:
- Set SCR[6] = 0
- Set the value of threshold into FCR and TLR
- Set SCR[6] = 1

*Table 71. Receive FIFO Trigger Level Setting Summary*

| SCR[7] | TLR[7:4] | RX FIFO Trigger Level |
|---|---|---|
| 0 | 0000 | Defined by FCR[7:6] (either 8, 16, 56, and 60 characters) |
| 0 | ≠0000 | Defined by TLR[7:4] (from 4 to 60 characters with a granularity of 4 characters) |
| 1 | value | Defined by the concatenated value of TLR[7:4] and FCR[7:6] (from 1 to 63 characters with a granularity of 1 character).<br>The combination of TLR[7:4] = 0000 and FCR[7:6] = 00 (all zeros) is not supported (minimum one character required). All zeros result in unpredictable behavior. |

**Note:** The protocol to set the concatenation of TLR and FCR is:
- Set SCR[7] = 0
- Set the value of threshold into FCR and TLR
- Set SCR[7] = 1

The mode of operation can be programmed by writing to MDR1[2:0]; therefore the mode definition 1 register (MDR1) must be programmed on start-up after configuration of the configuration registers (DLL, DLH, LCR…). The value of MDR1[2:0] must not be changed again during normal operation.

*Table 72. Mode Definition 1 Register (MDR1) Field Descriptions*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 7 | FRAME_END_MODE | 0 | Frame-length method | R/W | 0 |
| | | 1 | Set EOT bit method | | |
| 6 | – | | Reserved | R/W | 0 |
| 5 | SCT | | Stores and controls the transmission | R/W | 0 |
| | | 0 | Starts the SIR transmission as soon as a value is written to THR | | |
| | | 1 | Starts the SIR transmission with the control of ACREG2 | | |
| 4 | – | | Reserved | R | 0 |
| 3 | IR_SLEEP | 0 | SIR sleep mode disabled | R/W | 0 |
| | | 1 | SIR sleep mode enabled | | |
| 2–0 | MODE_SELECT† | 000 | UART mode | R/W | 111 |

† The MODE_SELECT = 0x7 setting disables the UART module by disabling the FIFO and the state machine. It does not gate the functional clock to the module. The lowest power state is not achieved by setting MODE_SELECT = 0x7, but by putting the UART into sleep mode. The lowest power state is achieved when in sleep mode witht DLL = 0xFFFF and DLH = 0xFFFF.

*Table 72. Mode Definition 1 Register (MDR1) Field Descriptions (Continued)*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| | | 001 | SIR mode | | |
| | | 111 | Disable UART/default state | | |
| | | | All the other values are reserved | | |

† The MODE_SELECT = 0x7 setting disables the UART module by disabling the FIFO and the state machine. It does not gate the functional clock to the module. The lowest power state is not achieved by setting MODE_SELECT = 0x7, but by putting the UART into sleep mode. The lowest power state is achieved when in sleep mode witht DLL = 0xFFFF and DLH = 0xFFFF.

The mode definition 2 register (MDR2) sets the trigger level for the frame status FIFO (8 entries) and must be programmed before the mode is programmed in MDR1[2:0].

*Table 73. Mode Definition Register 2 (MDR2) Field Descriptions*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 7–5 | – | | Reserved | R/W | 000 |
| 4–3 | DIV_1.6M | | MSB part of DIV_1.6 | R/W | 00 |
| 2–1 | STS_FIFO_TRIG | | Frame status FIFO threshold select: | R/W | 00 |
| | | 00 | 1 entry | | |
| | | 01 | 4 entry | | |
| | | 10 | 7 entry | | |
| | | 11 | 8 entry | | |
| 0 | – | | Reserved | R/W | 0 |

The transmit frame length registers (TXFLL and TXFLH) hold the 13-bit transmit frame length. TXFLL holds the least significant bits, and TXFLH holds the most significant bits. The frame length value is used if the frame length method of frame closing is used.

In terms of the IrDA frame format (see Figure 14), the value stored in the TXFLH/TXFLL registers is the byte length from A to I.

*Table 74. Transmit Frame Length Low Register (TXFLL) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–0 | TXFLL | LSB register used to specify the frame length | W | 00000000 |

*Table 75. Transmit Frame Length High Register (TXFLH) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–5 | – | Reserved | W | 000 |
| 4–0 | TXFLH | MSB register used to specify the frame length | W | 00000 |

The received frame length registers (RXFLL and RXFLH) hold the 12-bit receive maximum frame length. RXFLL holds the least significant bits, and RXFLH holds the most significant bits. If the intended maximum receive frame length is n bytes, then program RXFLL and RXFLH to n + 3 in SIR mode (+3 is due to frame format with CRC and stop flag).

In terms of the IrDA frame format (see Figure 14), the value stored in the RXFLH/RXFLL registers is the byte length from A to EOF.

*Table 76. Received Frame Length Low Register (RXFLL) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–0 | RXFLL | LSB register used to specify the frame length in reception | W | 00000000 |

Offset Address (hex): 0x0D x Start Address

*Table 77. Received Frame Length High Register (RXFLH) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–4 | – | Reserved | W | 0000 |
| 3–0 | RXFLH | MSB register used to specify the frame length in reception | W | 0000 |

The status FIFO line status line register (SFLSR) reads frame status information from the status FIFO (this register does not physically exist). Reading this register increments the status FIFO read pointer (SFREGL and SFREGH must be read first).

*Table 78. Status FIFO Line Status Register (SFLSR) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–5 | – | Reserved | R | 000 |
| 4 | OE_ERROR | 1: Overrun error in RX FIFO when frame at top of FIFO was received | R | 0 |
| 3 | FRAME_LENGTH_ ERROR | 1: Frame-length error in frame at top of FIFO | R | 0 |
| 2 | ABORT_DETECT | 1: Abort pattern detected in frame at top of FIFO | R | 0 |

*Table 78. Status FIFO Line Status Register (SFLSR) Field Descriptions (Continued)*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 1 | CRC_ERROR | 1: CRC error in frame at top of FIFO | R | 0 |
| 0 | – | Reserved | R | 0 |

The resume register (RESUME) is used to clear internal flags which halt transmission/reception when an underrun/overrun error occurs. Reading this register resumes the halted operation. This register does not physically exist and reads always as 0x00.

*Table 79. Resume Register (RESUME) Field Descriptions*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–0 | DI | Dummy read to restart the TX or RX | R | 00000000 |

The frame lengths of received frames are written into the status FIFO. This information can be read by reading the status FIFO registers (SFREGL and SFREGH—these registers do not physically exist). The least significant bits are read from SFREGL, and the most significant bits are read from SFREGH. Reading these registers does not alter the status FIFO read pointer. These registers must be read before the pointer is incremented by reading the SFLSR.

In terms of the IrDA frame format (see Figure 14), the value read in the SFREGH/SFREGL registers is the byte length from A to CRC.

*Table 80. Status FIFO Register Low (SFREGL)*

| Bits | Field | Description | R/W | Reset Value |
|---|---|---|---|---|
| 7–0 | SFREGL | LSB part of the frame length | R | Undefined |

*Table 81. Status FIFO Register High (SFREGH) Field Descriptions*

| Bit | Name | Function | R/W | Reset Value |
|---|---|---|---|---|
| 7–4 | – | Reserved | R | 0000 |
| 3–0 | SFREGH | MSB part of the frame length | R | Undefined |

The beginning of frame control register (BLR) [6] selects whether 0xC0 or 0xFF start patterns are to be used and when multiple start flags are required in SIR mode. If only one start flag is required, this is always 0xC0. If n start flags are required, then either (n-1) C0x0 or (n-1) 0xFF flags are sent, followed by a single 0xC0 flag (immediately preceding the first data byte).

*Table 82. BOF Control Register (BLR)*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 7 | STS_FIFO_RESET | | Status FIFO reset. This bit is self-clearing | R/W | 0 |
| 6 | XBOF_TYPE | | SIR XBOF select | R/W | 1 |
| | | 0 | 0xFF | | |
| | | 1 | 0xC0 | | |
| 5–0 | – | | Reserved | R/W | 000000 |

The beginning of the frame length register (EBLR) specifies the number of BOF + XBOFs to transmit in IrDA SIR operations. The value set into this register must take into account the BOF character; to send one BOF with no XBOF, this register must be set to 1. To send one BOF with N XBOFs, this register must be set to N+1. Furthermore, the value 0 sends 1 BOF plus 255 XBOF.

*Table 83. BOF Length Register (EBLR) Field Descriptions*

| Bit | Name | Function | R/W | Reset Value |
|---|---|---|---|---|
| 7–0 | EBLR | This register allows definition of up to 176 XBOFs, the maximum required by IrDA specification. | W | 00000000 |

*Table 84. DIV1.6 Register (DIV16) Field Descriptions*

| Bit | Name | Function | R/W | Reset Value |
|---|---|---|---|---|
| 7–0 | DIV_1.6L | Used to generate the 1.6-µs pulse | R/W | 00000000 |

In SIR, the DIV1.6 register (DIV16) is used to generate 1.6-µs pulse encoding instead of 3/16 encoding when selected by using ACREG[7]. The value of DIV_1.6 is coded on ten bits by MDR2[4:3] for its MSB and DIV_1.6[7:0] for its MSB.

In SIR mode, DIV1.6 must be programmed as follows:

DIV1.6 = [(3/(16 * baud rate)) - 1.6E-6] * FCLK_frequency

DIV1.6 = 0 is forbidden. If the calculated value Div_1.6 is between 0 and 1 the rounding must be done to 1.

With an input frequency of 13 MHz:

| | | | | |
|---|---|---|---|---|
| At 115200 baud | DLH = 0x00 | DLL = 0x07 | MDR24:3 = 0x00 | DIV_1.6 = 0x01 |
| At 57600 baud | DLH = 0x00 | DLL = 0x0E | MDR24:3 = 0x00 | DIV_1.6 = 0x16 |
| At 38400 baud | DLH = 0x00 | DLL = 0x15 | MDR24:3 = 0x00 | DIV_1.6 = 0x2B |
| At 19200 baud | DLH = 0x00 | DLL = 0x2A | MDR24:3 = 0x00 | DIV_1.6 = 0x6A |
| At 9600 baud | DLH = 0x00 | DLL = 0x55 | MDR24:3 = 0x00 | DIV_1.6 = 0xEB |
| At 2400 baud | DLH = 0x01 | DLL = 0x53 | MDR24:3 = 0x03 | DIV_1.6 = 0x96E5 |

With an input frequency of 12 MHz:

| | | | | |
|---|---|---|---|---|
| At 115200 baud | DLH = 0x00 | DLL = 0x01 | MDR24:3 = 0x00 | DIV_1.6 = 0x01 |
| At 57600 baud | DLH = 0x00 | DLL = 0x02 | MDR24:3 = 0x00 | DIV_1.6 = 0x14 |
| At 38400 baud | DLH = 0x00 | DLL = 0x03 | MDR24:3 = 0x00 | DIV_1.6 = 0x27 |
| At 19200 baud | DLH = 0x00 | DLL = 0x06 | MDR24:3 = 0x00 | DIV_1.6 = 0x62 |
| At 9600 baud | DLH = 0x00 | DLL = 0x0C | MDR24:3 = 0x00 | DIV_1.6 = 0xD7 |
| At 2400 baud | DLH = 0x01 | DLL = 0x30 | MDR24:3 = 0x03 | DIV_1.6 = 0x96 |

*Table 85. Auxiliary Control Register (ACREG) Field Descriptions*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 7 | PULSE_TYPE | | SIR pulse-width select: | R/W | 0 |
| | | 0 | 3/16 of baud-rate pulse width | | |
| | | 1 | 1.6-µs | | |
| 6 | SD_MOD | | Primary output used to configure transceivers. Connected to the SD/MODE input of transceivers. | R/W | 0 |
| | | 0 | SD_MODE pin is set to high. | | |
| | | 1 | SD_MODE pin is set to low. | | |
| 5 | DIS_IR_RX | 0 | Enables RXIR input | R/W | 0 |
| | | 1 | Disables RXIR input for half-duplex purpose | | |

*Table 85. Auxiliary Control Register (ACREG) Field Descriptions (Continued)*

| Bits | Field | Value | Description | R/W | Reset Value |
|---|---|---|---|---|---|
| 4 | DIS_TX_UNDERRUN | 0 | Long stop bits cannot be transmitted, TX underrun is enabled. | R/W | 0 |
| | | 1 | Long stop bits can be transmitted, TX underrun is disabled. | | |
| 3 | – | | Reserved | R | 0 |
| 2 | SCTX_EN | | Store and controlled TX start. When MDR[15] = 1 and the host writes 1 to this bit, the TX state machine starts frame transmission. This bit is self-clearing. | R/W | 0 |
| 1 | ABORT_EN | | Frame abort. The host can intentionally abort transmission of a frame by writing 1 to this bit. Neither the end flag nor the CRC bits are appended to the frame. | R/W | 0 |
| 0 | EOT_EN | | EOT (end of transmission) bit. The host writes 1 to this bit just before it writes the last byte to the TX FIFO in the set-EOT bit frame closing method. This bit automatically gets cleared when the host writes to the THR (TX FIFO). | R/W | 0 |

*Table 86. OSC 12-MHz Select Register (OSC_12M_SEL) Field Descriptions*

| Bit | Name | Function | R/W | Reset Value |
|---|---|---|---|---|
| 7–1 | – | Reserved | R | 0000000 |
| 0 | OSC_12M_SEL† | When 1, selects 6.5 division factor with a 12-MHz system clock. | W | 0 |

† This register is write-only and cannot be read.

*Table 87. Module Version Register (MVR) Field Descriptions*

| Bit | Name | Function | R/W | Reset Value |
|---|---|---|---|---|
| 7–4 | MAJOR_REV | Major revision number of the module | R | † |
| 3–0 | MINOR_REV | Minor revision number of the module | R | |

† For example: MVR = 0x11: Version 1.1

# 8 UART/IrDA Modes of Operation

The UART/IrDA module can operate in two different modes: UART mode and slow infrared (SIR) mode.

The modules perform serial-to-parallel conversion on data characters received and parallel-to-serial conversion on data characters transmitted by the processor. The complete status of each channel of the modules and each received character/frame can be read at any time during functional operation via the line control register (LSR).

The modules can be placed in an alternate mode (FIFO mode) to relieve the processor of excessive software overhead by buffering received/transmitted characters. Both the receiver and transmitter FIFOs can store up to 64 bytes of data (plus three additional bits of error status per byte for the receiver FIFO) and have selectable trigger levels.

Both interrupts and DMA are available to control the data-flow between the host (MPU or DSP) and the module.

## 8.1 UART Mode

The UART uses a wired interface for serial communication with a remote device.

UART modules are functionally compatible to the TL16C750 UART and are also functionally compatible with earlier designs such as the TL16C550.

UART modules can use hardware or software flow controls to manage transmission/reception. Hardware flow control significantly reduces software overhead and increases system efficiency by automatically controlling serial data flow using the $\overline{\text{RTS}}$ output and $\overline{\text{CTS}}$ input signals. Software flow control automatically controls data flow by using programmable XON/XOFF characters.

## 8.2 SIR Mode

In slow infrared (SIR) mode, data transfer takes place between the host (MPU or DSP) and peripheral devices at speeds of up to 115200 baud. An SIR transmit frame starts with start flags (either a single C0h, multiple C0hs, or a single C0h preceded by a number of FFh flags), followed by frame data, CRC-16, and a stop flag (C1h).

BLR[6] selects whether C0h or FFh start patterns are to be used when multiple start flags are required.

The SIR transmit state machine attaches start flags, CRC-16, and stop flags. It checks the outgoing data to establish if data transparency is required.

SIR transparency is carried out if the outgoing data (between the start and stop flags) contains C0h, C1h, or 7Dh. If one of these is about to be transmitted, then the SIR state machine sends an escape character [7Dh] first, then inverts the fifth bit of the real data to be sent, and sends this data immediately after the 7Dh character.

The SIR receive state machine recovers the receive clock, removes the start flags, removes any transparency from the incoming data, and determines frame boundary with reception of the stop flag. It also checks for errors such as frame aborts (7Dh character followed immediately by a C1h stop flag, without transparency), CRC errors, and frame-length errors. At the end of a frame reception, the host (MPU or DSP) reads the line status register (LSR) to find the errors, if any, of the received frame.

Data can be transferred both ways simultaneously by the module, but transmit and receive must not take place at the same time according to the standard.

The infrared output in SIR mode can either be 1.6-µs or 3/16 encoding, selected by ACREG[7]. In 1.6-µs encoding the infrared pulse width is 1.6-µs, and in 3/16 encoding the infrared pulse width is 3/16 of a bit duration (1/baudrate).

The transmitting device must send at least two start flags at the start of each frame for back-to-back frames.

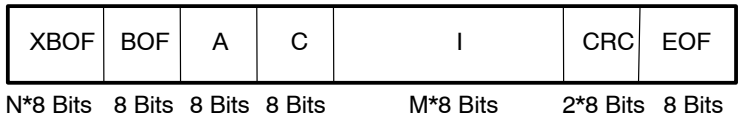
**Note:**

Reception supports variable-length stop bits.

### 8.2.1 CRC Generation

Figure 14 shows the IrDA frame format.

*Figure 14. IrDA Frame Format*




The CRC is applied on the address (A), control (C), and information (I) bytes.

**Note:**

The two words of CRC are written in the FIFO in reception.

### 8.2.2 Asynchronous Transparency

Before transmitting a byte, the UART IrDA controller examines each byte in the payload and the CRC field (between BOF and EOF). For each byte equal to 0xC0 (BOF), 0xC1 (EOF), 0x7D (control escape), the controller does the following.

❏ In transmission:

1) Inserts a control escape (CE) byte preceding the byte.

2) Complements bit 5 of the byte (that is, exclusive ORs the byte with 0x20).

The byte sent for the CRC computation is the initial byte written in the TX FIFO (before the XOR with 0x20).

❏ In reception:

For the A, C, I, CRC fields:

1) Compares the byte with the CE byte; if not equal, sends it to the CRC detector and stores it in the RX FIFO.

2) If equal to the CE byte, discards the CE byte.

3) Complements the bit 5 of the byte following the CE.

4) Send the complemented byte to the CRC detector and stores it in the RX FIFO.

### 8.2.3 Abort Sequence

The transmitter may decide to prematurely close a frame. The transmitting station aborts by sending the sequence 0x7dc1. The abort pattern closes the frame without a CRC field or an ending flag.

It is possible to abort a transmission frame by programming ACREG [1].

When this bit is set to 1, 7Dh and C1h are transmitted and the frame is not terminated with CRC or stop flags.

The receiver treats a frame as an aborted frame when a 7Dh character followed immediately by a C1h character has been received without transparency.

❏ When UART3 receives an abort sequence (0x7DC1), the abort bit (LSR[3]) is set to 1.

❏ When the UART3 FIFO is empty or the SFSLR register is read, the abort bit is cleared to 0 (If SFSLR register is read, abort actually reflects the status of the next frame).

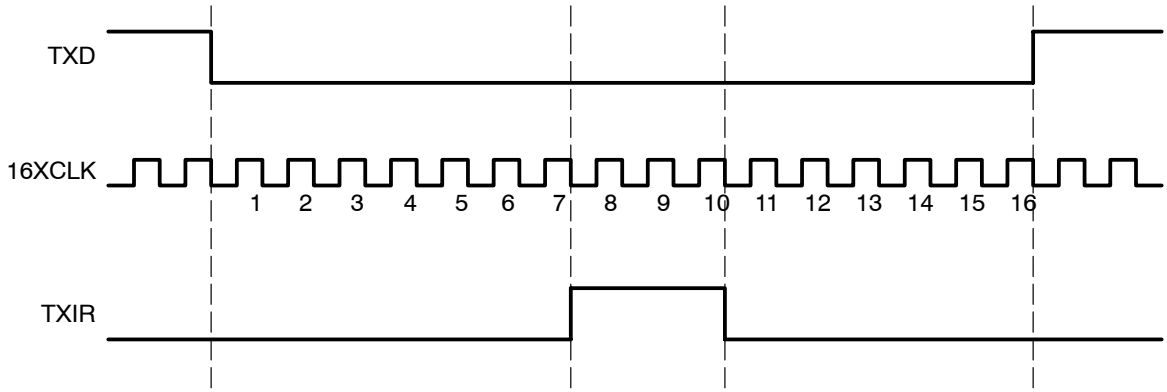
### 8.2.4 Pulse Shaping

In SIR mode both the 3/16$^{th}$ or the 1.6-µs pulse duration methods are supported. ACREG[7] selects the pulse-width method in transmit mode.

### 8.2.5 Encoder

Serial data from the transmit state machine is encoded to transmit data to the optoelectronics (see Figure 15). While the serial data input to (TXD) is high, the output (TXIR) is always low and the counter used to form a pulse on TXIR is continuously cleared. After TXD resets to 0, TXIR rises on the falling edge of the 7$^{th}$ 16XCLK. On the falling edge of the 10$^{th}$ 16XCLK pulse, TXIR falls, creating a 3-clock-wide pulse. While TXD remains low, a pulse is transmitted during the 7$^{th}$ to the 10$^{th}$ clocks of each 16-clock bit cycle.

*Figure 15. IrDA Encoder Mechanism*

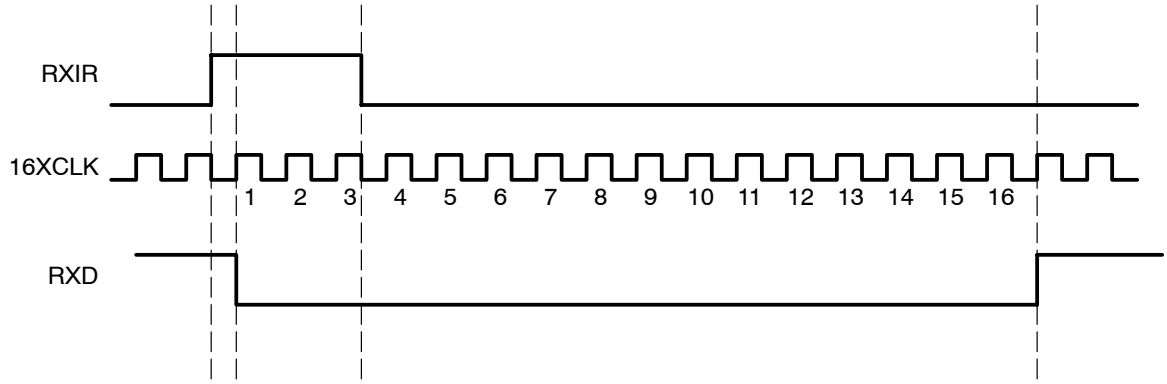



### 8.2.6 Decoder

After reset, RXD is high and the 4-bit counter is cleared (see Figure 16). When a rising edge is detected on RXIR, RXD falls on the next rising edge of 16XCLK with sufficient setup time. RXD remains low for 16 cycles (16XCLK) and then returns to high as required by the IrDA specification. As long as no pulses (rising edges) are detected on the RXIR, RXD remains high.

The reception of RXIR input can be disabled with DIS_IR_RX bits of the auxiliary control register (ACREG[5]).

*Figure 16. IrDA Decoder Mechanism*




### 8.2.7 Address Checking

In SIR mode, only frames intended for the device are written to the RX FIFO, if address checking has been enabled. This avoids receiving frames not meant for this device in a multipoint infrared environment. The two frame addresses the UART IrDA receives can be programmed with the XON1/ADDR1 and XON2/ADDR2 registers.

Selecting address1 checking is done by setting EFR[0] to 1. Address2 checking is done by setting EFR[1] to 1. Setting EFR[1:0] to 0 disables all address checking operations. If both bits are set, then the incoming frame is checked for both the private and public addresses.

If address checking is disabled, then all received frames are written into the reception FIFO.

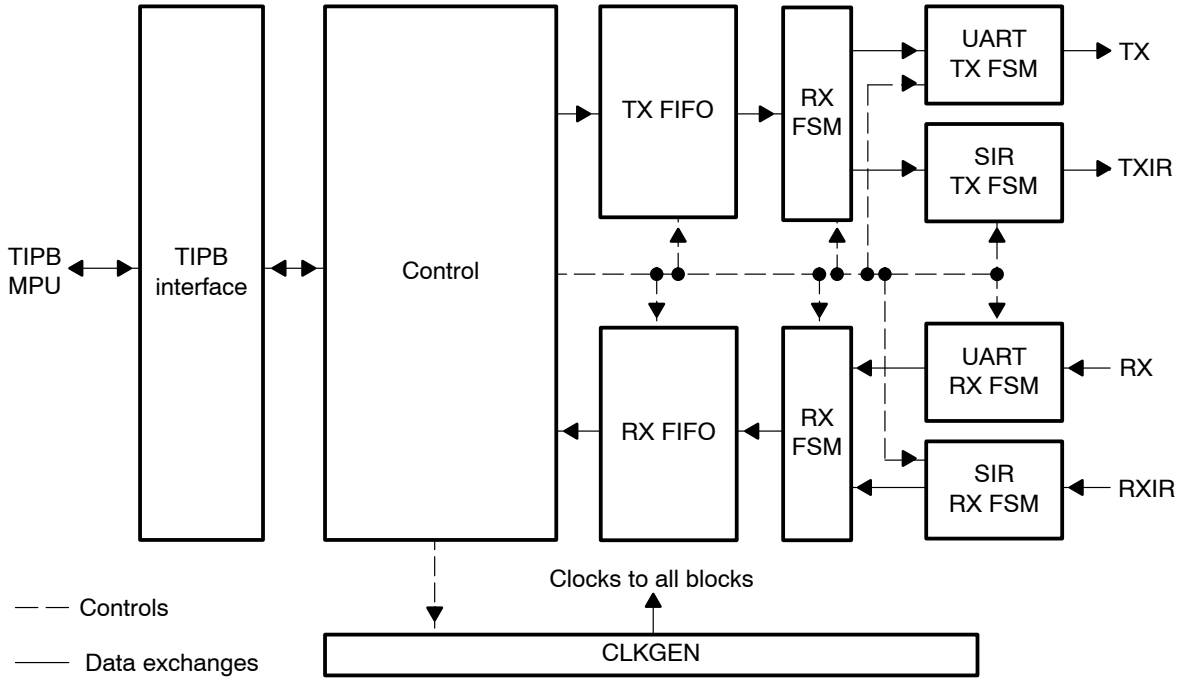
# 9 UART/IrDA Functional Description

This section provides a functional description of the UART IrDA.

## 9.1 UART/IrDA Functional Block Diagram

Figure 17 shows the UART/IrDA (FSM stands for finite state machine).

*Figure 17. Functional Block Diagram*




## 9.2 Trigger Levels

The UART provides programmable trigger levels for both receiver and transmitter DMA and interrupt generation. After reset, both transmitter and receiver FIFOs are disabled (in effect, the trigger level is the default value of one byte). The programmable trigger levels are an enhanced feature available via the trigger level register (TLR).

## 9.3 Interrupts

The UART generates interrupts on the UART_nIRQ output pin. All interrupts can be enabled/disabled by writing to the appropriate bit in the interrupt enable register (IER). The interrupt status of the device can be checked at any time by reading the interrupt identification register (IIR).

The UART and IrDA modes have different interrupts in the UART/IrDA module and therefore different IER and IIR mappings according to the selected mode.

### 9.3.1 Interrupts in MODEM Mode

There are seven possible interrupts, prioritized to six different levels. When an interrupt is generated, the interrupt identification register (IIR) indicates a pending interrupt by bringing IIR[0] to logic 0, and it specifies the type of interrupt through IIR[5-1]. Table 88 summarizes the interrupt control functions.

*Table 88. Generic Interrupt Functions in Modem Mode*

| IIR[5-0] | Priority Level | Interrupt Type | Interrupt Source | Interrupt Reset Method |
|---|---|---|---|---|
| 0 0 0 0 0 1 | None | None | None | None |
| 0 0 0 1 1 0 | 1 | Receiver line status | OE, FE, PE, or BI errors occur in characters in the RX FIFO | FE,PE,BI: All erroneous characters are read form the RX FIFO. OE: Read LSR |
| 0 0 1 1 0 0 | 2 | RX time-out | Stale data in RX FIFO | Read RHR |
| 0 0 0 1 0 0 | 2 | RHR interrupt | DRDY (data ready) (FIFO disable)<br>RX FIFO above trigger level (FIFO enable) | Read RHR until interrupt condition disappears. |
| 0 0 0 0 1 0 | 3 | THR interrupt | TFE (THR empty) (FIFO disable)<br>TX FIFO below trigger level (FIFO enable) | Write to THR until interrupt condition disappears. |
| 0 0 0 0 0 0 | 4 | Modem status | MSR[1:0] = 0 | Read MSR |
| 0 1 0 0 0 0 | 5 | XOFF interrupt/ special character interrupt | Receive XOFF characters(s)/special character | Receive XON character(s), if XOFF interrupt/read of IIR, if special character interrupt |
| 1 0 0 0 0 0 | 6 | $\overline{\text{CTS}}$,$\overline{\text{RTS}}$, $\overline{\text{DSR}}$ | $\overline{\text{RTS}}$ pin, $\overline{\text{CTS}}$ pin or $\overline{\text{DSR}}$ pin change state from active (low) to inactive (high). | Read IIR |

**Note:** Once LSR[7] (RX_FIFO_STS) is set on FIFO disable (FCR[0]=0), this bit bit cannot be cleared by reading LSR. First, FCR[1] (RX_FIFO_CLERA) must be set to 1, then LSR[7] can be cleared.

The RX_FIFO_STS bit (LSR[7]) generates the interrupt for the receiver line status interrupt.

For the XOFF interrupt, if a XOFF flow character detection causes the interrupt, the interrupt is cleared by a XON flow character detection. If special character detection causes the interrupt, the interrupt is cleared by a read of the IIR.

### 9.3.2 Interrupts in SIR Mode

In the IrDA modes there are eight possible interrupts. The UART_nIRQ output is activated when any of the eight interrupts is generated (there is no priority).

Table 89 summarizes the interrupt control functions in SIR mode.

*Table 89. Generic Interrupt Functions in SIR Mode*

| IIR Bit No. | Interrupt Type | Interrupt Source | Interrupt Reset Method |
|---|---|---|---|
| 7 | Received EOF | Received end-of-frame | Read IIR |
| 6 | Receiver line status interrupt | CRC, ABORT or frame-length error is written into STATUS FIFO. | Read STATUS FIFO. Read until empty—maximum eight reads required. |
| 5 | TX underrun | THR empty before EOF sent | Read RESUME register |
| 4 | Status FIFO interrupt | Status FIFO triggers level reached | Read STATUS FIFO. |
| 3 | RX overrun | Write to RHR when RX FIFO full. | Read RESUME register |
| 2 | Last byte in RX FIFO | Last byte of frame in RX FIFO | Read IIR |
| 1 | THR interrupt | TFE (THR empty) (FIFO disable)<br>TX FIFO below trigger level (FIFO enable) | Write to THR until interrupt condition disappears. |
| 0 | RHR interrupt | DRDY (data ready) (FIFO disable)<br>RX FIFO above trigger level (FIFO enable) | Read RHR until interrupt condition disappears. |

### 9.3.3 Wake-Up Interrupt

A wake-up interrupt is a uniquely designed interrupt, enabled when SCR[4] is set to 1. The IIR register is not modified when it occurs; SSR[1] must be checked to detect a wake-up event. When a wake-up interrupt occurs, the only way to clear it is to reset SCR[4] to 0.

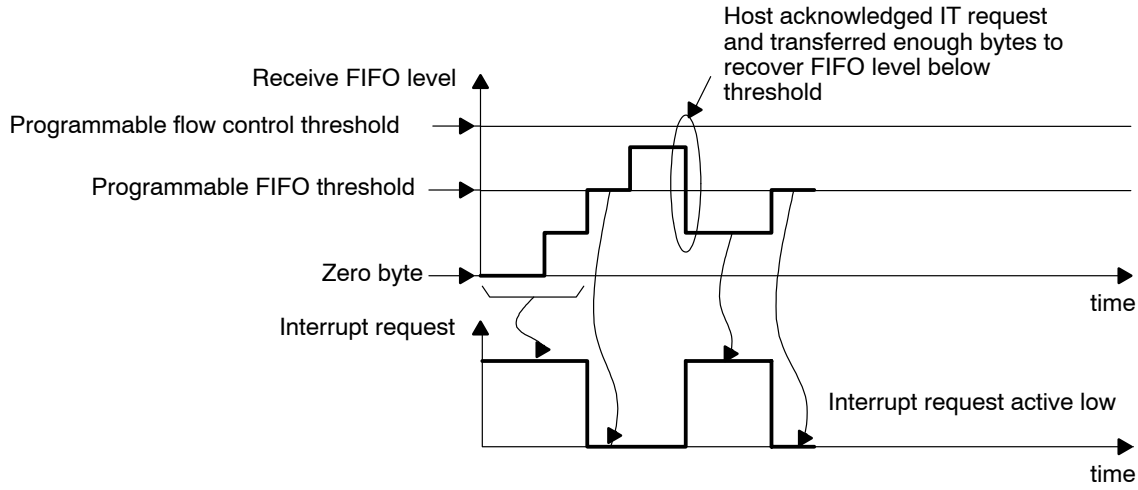
## 9.4 FIFO Interrupt Mode

In FIFO interrupt mode (FCR[0] = 1), relevant interrupts enabled via IER), the processor is informed of the status of the receiver and transmitter by an interrupt signal, $\overline{IRQ}$. These interrupts are raised when receive/transmit FIFO threshold (respectively TLR[7:4] and TLR[3:0] or FCR[7:6] and FCR[5:4]) are reached; they ask the host (MPU or DSP) to transfer data to destination (from UART module in receive mode and from any source to UART FIFO in transmit mode).

When UART flow control is enabled along with interrupt capabilities, ensure that the UART flow control FIFO threshold (TCR[3:0]) is greater than or equal to the receive FIFO threshold.

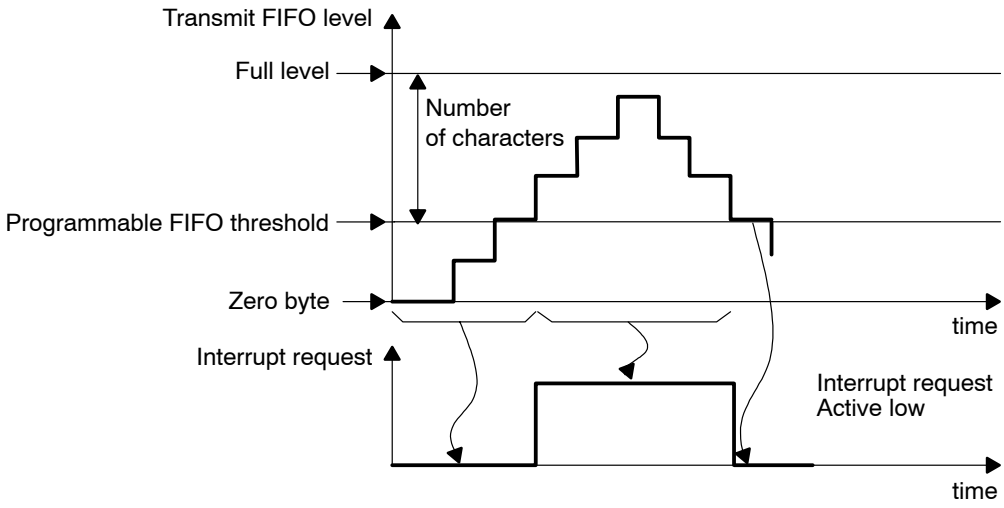
Figure 18 and Figure 19 show the receive and transmit IT operations, respectively.

*Figure 18. Receive FIFO IT Request Generation*




In receive mode, no interrupt is generated until the receive FIFO reaches its threshold. Once low, the interrupt can only be deasserted when the host (MPU or DSP) has handled enough bytes to make the FIFO level below threshold. Notice that the flow control threshold is set at a higher value than the FIFO threshold.

*Figure 19. Transmit FIFO IT Request Generation*




In transmit mode, an interrupt request is automatically asserted when FIFO is empty. This request is deasserted when the FIFO crossed the threshold level. The interrupt line is deasserted until a sufficient number of elements has been transmitted to go below FIFO threshold.

## 9.5 FIFO Polled Mode Operation

In FIFO polled mode (FCR[0] = 0 with relevant interrupts disabled via interrupt enable register (IER)), the status of the receiver and transmitter can then be checked by polling the line status register (LSR). This mode is an alternative to the FIFO interrupt mode of operation, where the status of the receiver and transmitter is automatically known by means of interrupts sent to the host (MPU or DSP).

## 9.6 FIFO DMA Mode Operation

### 9.6.1 DMA Signaling

There are four modes of DMA operation: DMA mode 0, DMA mode 1, DMA mode 2, and DMA mode 3. They can be selected as follows.

- When SCR[0] = 0:
  - Setting FCR[3] to 0 enables DMA mode 0.
  - Setting FCR[3] to 1 enables DMA mode 1.
- When SCR[0] = 1:SCR[2:1] determine DMA mode 0 to 3 according to supplementary control register (SCR) description.

So for instance:

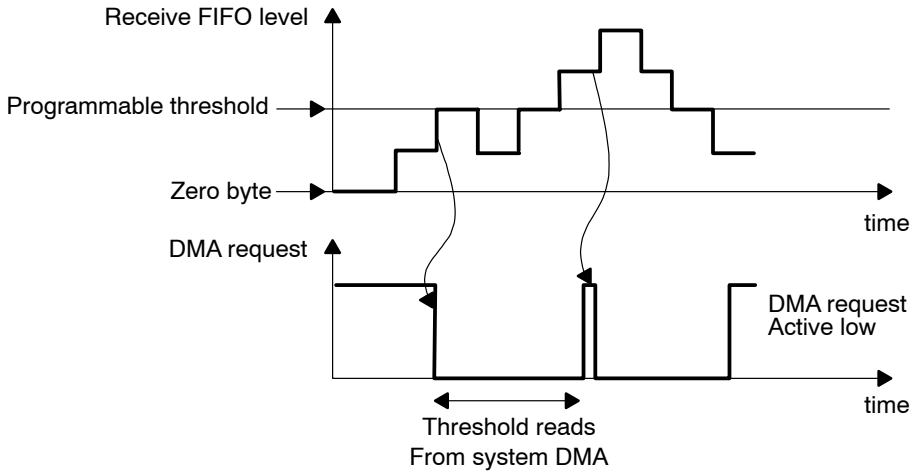
- ❏ If no DMA operation is desired, set SCR[0] to 1 and SCR[2:1] to 00 (FCR[3] is disregarded).
- ❏ If DMA mode 1 is desired, either set SCR[0] to 0 and FCR[3] to 1 or set SCR[0] to 1 SCR[2:1] to 01 (FCR[3] is disregarded).

If the FIFOs are disabled (FCR[0] = 0), DMA occurs in single character transfers. When DMA mode 0 has been programmed, the signals associated with DMA operation are not active.

### 9.6.2 DMA Transfers (DMA Mode 1, 2, or 3)

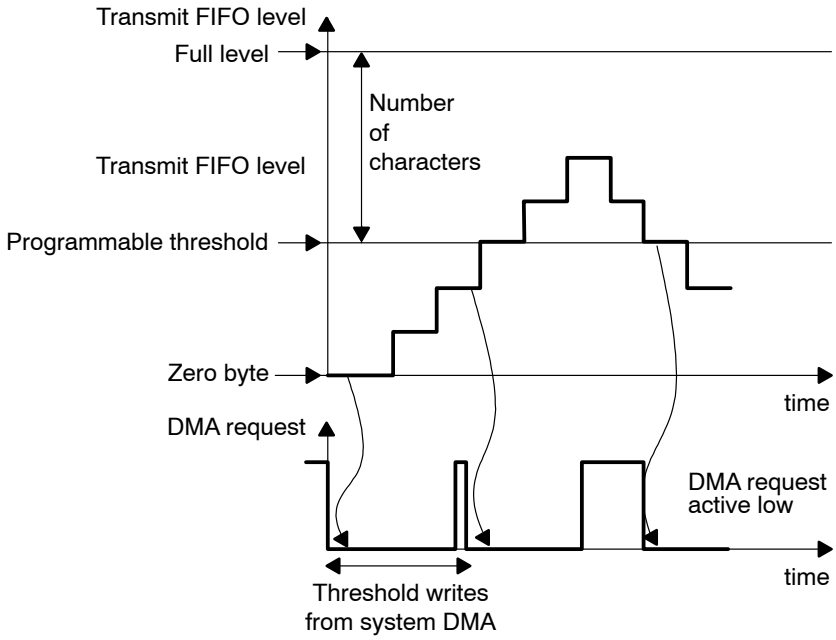
Figure 20 shows DMA operations at receive; Figure 21 shows DMA operations at transmit.

*Figure 20. Receive FIFO DMA Request Generation*




In receive mode, a DMA request is generated as soon as the receive FIFO reaches its threshold. This request is deasserted when the number of bytes defined by the threshold level has been read by the system DMA.

*Figure 21. Transmit FIFO DMA Request Generation*




In transmit mode, a DMA request is automatically asserted when FIFO is empty. This request is deasserted when the number of bytes defined by the threshold level has been written by the system DMA. The DMA request is again asserted if the FIFO is able to receive the number of bytes defined by the threshold.

## 9.7 Sleep Mode

### 9.7.1 UART Mode

Sleep mode is a low-power, enhanced feature of the UART that can be enabled by writing a 1 to IER[4] (when EFR[4] = 1).

Sleep mode is entered when:

- ❏ Serial RX data input line is idle.
- ❏ TX FIFO and TX shift register are empty.
- ❏ RX FIFO is empty.
- ❏ No interrupts are pending except THR interrupts.

Sleep mode is a good way to lower UART power consumption, but this state can be achieved only when the UART is set in modem mode. Therefore, even if UART does not have a functional key role, it must be initialized in a functional mode to take advantage of sleep mode.

In sleep mode, the module clock and baud rate clock are stopped internally. Because most registers are clocked using these clocks, the power consumption is greatly reduced. The module wakes up when any change is detected on the RX line, when data is written to the TX FIFO, or when there is any change in the state of the modem input pins. An interrupt can be generated on a wake up event by setting SCR[4] to 1.

**Note:**

Writing to the divisor latches, DLL and DLH, to set the baud clock, BCLK, must not be done during sleep mode. Disable sleep mode using IER[4] before writing to DLL or DLH.

### 9.7.2 IrDA Mode

In IrDA modes, sleep mode is enabled by writing a 1 to MDR1[3].

Sleep mode is entered when:

- Serial RXIR data input line is idle.
- TX FIFO and TX shift register are empty.
- RX FIFO is empty.
- No interrupts are pending except THR interrupts.

The module wakes up when any change is detected on the RXIR line, if data is written to the TX FIFO.

## 9.8 Break and Time-Out Conditions

- Time-out counter

  An RX idle condition is detected when the receiver line, RX, has been high for a time equivalent to 4 X programmed word length + 12 bits. The receiver line is sampled midway through each bit.

  For sleep mode, the counter is reset when there is activity on the RX line.

  For the time-out interrupt, the counter only counts when there is data in the RX FIFO and the count is reset when there is activity on the RX line or when the RHR is read.

❑ Break condition

When a break condition occurs, the TX line is pulled low. A break condition is activated by setting LCR[6]. The break condition is not aligned on word stream; that is, a break condition can occur in the middle of a character. The only way to send a break condition on a full character, is:

- Reset transmit FIFO (if enabled).
- Wait for transmit shift register becomes empty (LSR[6] = 1).
- Take a guard time according to stop bit definition.
- Set LCR[6] to 1.

The break condition is asserted as long as LCR[6] is set to 1.

## 9.9 Programmable Baud Rate Generator

The programmable baud generator takes any clock input and divides it by a divisor between 1 and ($2^{16-1}$). The CLKSEL register bit MCR[7] can be used to select the 1X or 1X/4 clock for the internal baud rate generator. The output frequency of the baud rate generator is 16x the baud rate.

To program the baud rate, a write must be issued to the DLL register (least significant bytes) and DLH register (most significant bytes) of the baud rate divisor to program the baud rate.

Writing to these registers may result in wait states being inserted during the write access while the baud rate generator is loaded with the new value. If both registers are 0, the module is effectively disabled, and no baud clock is generated.

**Note:**

The programmable baud rate generator selects both the transmit and receive clock rates.

## 9.10 Hardware Flow Control

Hardware flow control is composed of automatic CTS and automatic RTS. Automatic CTS and automatic RTS can be enabled/disabled independently by programming EFR[7:6]. With automatic CTS, $\overline{\text{CTS}}$ must be active before the module can transmit data.

Automatic RTS only activates the $\overline{\text{RTS}}$ output when there is enough room in the FIFO to receive data and deactivates the $\overline{\text{RTS}}$ output when the RX FIFO is sufficiently full. The HALT and RESTORE trigger levels in the TCR determine the levels at which $\overline{\text{RTS}}$ is activated/deactivated.

If both automatic CTS and automatic RTS are enabled, data transmission does not occur unless the receiver FIFO has empty space. Thus, overrun errors are eliminated during hardware flow control. If not enabled, overrun errors occur if the transmit data rate exceeds the receive FIFO latency.

- Automatic RTS

  Automatic RTS data flow control originates in the receiver block (see Figure 17). The receiver FIFO trigger levels used in automatic RTS are stored in the TCR. $\overline{RTS}$ is active if the RX FIFO level is below the HALT trigger level in TCR[3:0]. When the receiver FIFO HALT trigger level is reached, $\overline{RTS}$ is deasserted. The sending device (for example, another UART) can send an additional byte after the trigger level is reached because it may not recognize the deassertion of $\overline{RTS}$ until it has begun sending the additional byte. $\overline{RTS}$ is automatically reasserted once the receiver FIFO reaches the RESUME trigger level programmed via TCR(7:4). This reassertion requests the sending device to resume transmission.

- Automatic CTS

  The transmitter circuitry checks $\overline{CTS}$ before sending the next data byte. When $\overline{CTS}$ is active, the transmitter sends the next byte. To stop the transmitter from sending the following byte, $\overline{CTS}$ must be deasserted before the middle of the last stop bit that is currently being sent. The automatic CTS function reduces interrupts to the host system. When automatic CTS flow control is enabled, the $\overline{CTS}$ state changes need not trigger host interrupts because the device automatically controls its own transmitter. Without automatic CTS, the transmitter sends any data present in the transmit FIFO and a receiver overrun error can result.

## 9.11 Software Flow Control

Software flow control is enabled through the enhanced feature register (EFR) and the modem control register (MCR). Different combinations of software flow control can be enabled by setting different combinations of EFR[3-0].

There are two other enhanced features relating to software flow control:

- XON Any function (MCR[5]): Operation resumes after receiving any character after recognizing the XOFF character. The XON-Any character is written into the RX FIFO even if it is a software flow character.
- Special character (EFR[5]): Incoming data is compared to XOFF2. Detection of the special character sets the XOFF interrupt (IR[4]) but does not halt transmission. The XOFF interrupt is cleared by a read of the IIR. The special character is transferred to the RX FIFO.

### 9.11.1 RX

When software flow control operation is enabled, the UART compares incoming data with XOFF1/2 programmed characters (in certain cases XOFF1 and XOFF2 must be received sequentially). When the correct XOFF characters are received, transmission is halted after completing transmission of the current character. XOFF detection also sets IIR[4] (if enabled via IER[5]) and causes nIRQ to go low.

To resume transmission, an XON1/2 character must be received (in certain cases XON1 and XON2 must be received sequentially). When the correct XON characters are received, IIR[4] is cleared and the XOFF interrupt disappears.

If a parity, framing or break error occurs while receiving a software flow control character, this character is treated as normal data and is written to the RX FIFO.

When XON-Any and special character detect are disabled and software flow control is enabled, no valid XON or XOFF characters are written to the RX FIFO. For example, when EFR[1:0] = 10, if XON1 and XOFF1 characters are received they do not get written to the RX FIFO.

When pairs of software flow characters are programmed to be received sequentially (EFR[1:0] = 11), the software flow characters are not written to the RX FIFO if they are received sequentially. However, received XON1/XOFF1 characters must be written to the RX FIFO if the subsequent character is not XON2/XOFF2.

### 9.11.2 TX

XOFF1: Two characters are transmitted when the RX FIFO has passed the programmed trigger level TCR[3:0].

XON1: Two characters are transmitted when the RX FIFO reaches the trigger level programmed via TCR[7:4].

If, after an XOFF character has been sent, software flow control is disabled, the module transmits XON characters automatically to enable normal transmission to proceed.

The transmission of XOFF/XON(s) follows the exact same protocol as transmission of an ordinary byte from the FIFO. This means that even if the word length is set to be 5, 6, or 7 characters, the 5, 6, or 7 least significant bits of XOFF1,2/XON1,2 are transmitted. The transmission of 5, 6, or 7 bits of a character is seldom done, but this functionality is included to maintain compatibility with earlier designs.

It is assumed that software flow control and hardware flow control are never enabled simultaneously.

## 9.12 Frame Closing

There are two methods by which a transmission-frame can be properly terminated.

1) The frame-length method is selected when MDR1[7] = 0. The host (MPU or DSP) writes the frame-length value to TXFLH and TXFLL registers. The device automatically attaches end flags to the frame once the number of bytes transmitted becomes equal to the frame-length value.

2) The set-EOT bit method is selected when MDR1[7] = 1. The host writes 1 to ACREG[0] (EOT bit) just before it writes the last byte to the TX FIFO. When the host writes the last byte to the TX FIFO, the device internally sets the tag bit for that particular character in the TX FIFO. As the TX state machine reads data from the TX FIFO, it uses this tag-bit information to attach end flags and properly terminate the frame.

## 9.13 Store and Controlled Transmission

In a store and controlled transmission (SCT), the host (MPU or DSP) first starts writing data into the TX FIFO. Then, after it writes a part of a frame (for a bigger frame) or a whole frame (a small frame, that is, supervisory frame), it writes a 1 to ACREG[2] (deferred TX start) to start transmission. SCT is enabled when MDR1[5] = 1. This method of transmission is different from the normal mode, where transmission of data starts immediately after data is written to the TX FIFO. SCT is useful to send short frames without TX underrun.

## 9.14 Underrun During Transmission

Underrun in transmission occurs when the TX FIFO becomes empty before the end of the frame is transmitted. When underrun occurs, the device closes the frame with end-flags but attaches an incorrect CRC value. The receiving device detects a CRC error and discards the frame; it can then ask for a retransmission. Underrun also causes an internal flag to be set which disables further transmission. Before the next frame can be transmitted the system (host) must:

- ❏ Reset the TX FIFO.
- ❏ Read the RESUME register—this clears the internal flag.

This functionality can be disabled or compensated for by the extension of the stop bit in transmission, in case the TX FIFO is empty.

## 9.15 Overrun During Receive

Overrun occurs during receive if the RX state machine tries to write data into the RX FIFO when it is already full. When overrun occurs, the device interrupts the host (MPU or DSP) with IIR[3] and discards the remaining portion of the frame. Overrun also causes an internal flag to be set, which disables further reception. Before the next frame can be received the system (host) must:

- ❏ Reset the RX FIFO.
- ❏ Read the resume register—this clears the internal flag.

## 9.16 Status FIFO

In SIR mode, a status FIFO is used to record the received frame status. When a complete frame is received, the length of the frame and the error bits associated with the frame are written into the status FIFO.

The frame length and error status can be read by reading SFREGL/H and SFLSR. Reading the SFLSR causes the read pointer to be incremented. The status FIFO is eight entries deep and therefore can hold the status of eight frames.

The host (MPU or DSP) uses the frame-length information to locate the frame-boundary in the received frame data. The host can screen bad frames using the error-status information and later request the sender to resend only the bad frames.

This status FIFO can be used very effectively in DMA as the host does not need to be interrupted every time a frame is received, but only whenever the programmed status FIFO trigger level is reached.

## 10 UART/IrDA Configuration Example

This section outlines the programming stages to operate one UART module with FIFO, interrupt, and no DMA capabilities. This is a three-step procedure that ensures quick start of these modules (obviously it does not cover every UART module feature). The first stage covers software reset of the module (interrupts, status, and controls); the second stage deals with FIFO configuration and enable; and the last stage deals with baud rate data and stop configuration. The procedure below is programming language agnostic.

## 11 UART Software Reset

The goal of the UART software reset is to clear the IER and MCR registers, remove the UART breaks (LCR[6]=0), and put module in reset (MDR1[2:0]=0x3).

The procedure of the UART is as follows:

1) Write into both the IER and MCR register, (EFR[4] must first be set to 1).

2) Access the EFR register.

3) 0xBF must first be written to LCR register as follows:

   - LCR=0xBF
   - EFR[4]=1
   - LCR=0x80 (access to IER and MCR is allowed)
   - IER=0x00
   - MCR=0x00; LCR[6]=0 (UART breaks removed)
   - MDR1=0x03 (UART in reset)

## 12 UART FIFO Configuration

The goal of the UART FIFO configuration is to set trigger level for halt/restore (TCR register), set the trigger level for the transmit/receive (TLR register), and configure the FIFO (FCR register).

The procedure of the UART FIFO configuration is as follows:

4) Write into both the TLR and TCR registers

   - Set EFR[4] to 1
   - Set MCR[6] to 1.

5) Write into FCR.

   - Set EFR[4] to 1.

     EFR[4] = 1 has already been done in the previous section, so a simple write to MCR[6] is necessary.

6) Set TCR TLR and FCR to the desired value.

   Here accesses to TCR TLR and FCR must be disabled to avoid any further undesired write to these registers:

   - LCR=0xBF (provides access to EFR)
   - EFR[4]=0
   - LCR[7]=0
   - MCR[6]=0

## 12.1 Baud Rate Data and Stop Configuration

The goals of the baud rate and stop configuration are to configure the UART data, stop (LCR register) baud rate (DLH and DLL registers), and enable UART operation. If interrupt capability is added, the configuration must be added right before UART enable.

The procedure to accomplish these goals is as follows:

1) Input clock is 12 MHz, so set OSC_12M_SEL to 1.
2) Set LCR to desired value.
3) LCR[7] to 1 (access to DLH and DLL registers).
4) Set DLH and DLL.
5) LCR[7]=0 (removes access to DLH and DLL registers)
6) Set IER to desired value (sets interrupts).
7) MDR1[2:0]=0 (enables UART)

   The UART module is operational.

# Index

## O



## P



## R



## S



## T



## U