Universal Asynchronous Receiver/Transmitter (UART)

This chapter describes the UART of the device.

Topic **Page**

19.1 Introduction **4316**
19.2 Integration **4318**
19.3 Functional Description **4322**
19.4 UART/IrDA/CIR Basic Programming Model **4365**
19.5 UART Registers **4374**

19.1 Introduction

19.1.1 UART Mode Features

The general features of the UART/IrDA module when operating in UART mode are:

- 16C750 compatibility
- Baud rate from 300 bps up to 3.6864 Mbps
- Auto-baud between 1200 bps and 115.2 Kbps
- 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, mark (always 1), space (always 0), or no parity (non-parity bit frame) bit generation and detection
  - 1, 1.5, or 2 stop bit generation
- False start bit detection
- Line break generation and detection
- Modem control functions (CTS, RTS, DSR, DTR, RI, and DCD)
- Fully prioritized interrupt system controls
- Internal test and loopback capabilities

19.1.2 IrDA Mode Features

The general features of the UART/IrDA when operating in IrDA mode are:

- Support of IrDA 1.4 slow infrared (SIR), medium infrared (MIR) and fast infrared (FIR) communications (very fast infrared (VFIR) is not supported)
- Frame formatting: addition of variable xBOF characters and EOF characters
- Uplink/downlink CRC generation/detection
- Asynchronous transparency (automatic insertion of break character)
- 8-entry status FIFO (with selectable trigger levels) available to monitor frame length and frame errors
- Framing error, cyclic redundancy check (CRC) error, illegal symbol (FIR), abort pattern (SIR, MIR) detection

19.1.3 CIR Mode Features

The general features of the UART/IrDA when operating in CIR mode are:

- Support of consumer infrared (CIR) for remote control applications
- Transmit and receive
- Free data format (supports any remote control private standards)
- Selectable bit rate
- Configurable carrier frequency
- 1/2, 5/12, 1/3 or 1/4 carrier duty cycle

19.1.4 Unsupported UART Features

The following UART/IrDA module features are not supported in this device.

Table 19-1. Unsupported UART Features

Feature	Reason
Full modem control on UART0	DCD, DSR, DTR, RI not pinned-out
Full modem control on UART2-5	DCD, DSR, DTR, RI not pinned-out
Device wake-up on UART1-5	Wake-up not supported - no SWake connection

19.2 Integration

This device contains 6 instantiations of the UART/IrDA (UARTIRDAOCP) peripheral. There are six UART modules called UART0 – UART5. UART0 provides wakeup capability. Only UART 1 provides full modem control signals. All UARTs support IrDA and CIR modes and RTS/CTS flow control (subject to pin muxing configuration). Figure 19-1 shows an example of system connectivity using UART communication with hardware handshake.

Figure 19-1. UART/IrDA Module — UART Application

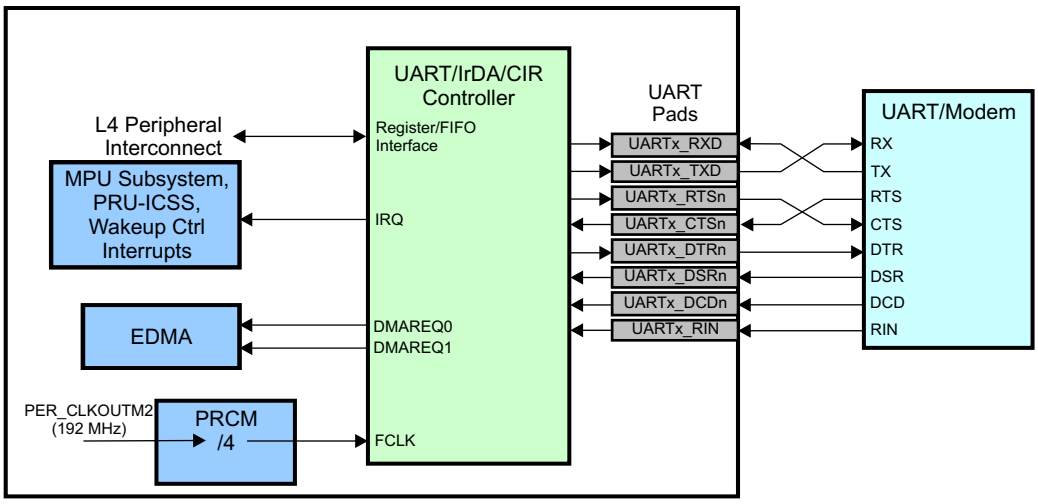
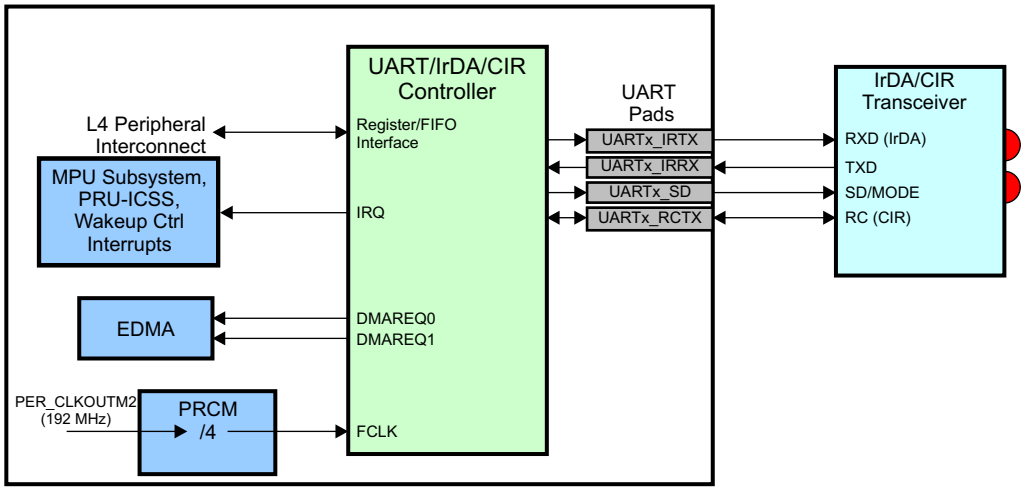


Figure 19-2 shows an example of system connectivity using infrared communication with remote control (consumer infrared).

Figure 19-2. UART/IrDA Module — IrDA/CIR Application



19.2.1 UART Connectivity Attributes

The general connectivity attributes for each of the UART modules are shown in Table 19-2 and Table 19-3.

Table 19-2. UART0 Connectivity Attributes

Attributes	Type
Power Domain	Wake-Up Domain
Clock Domain	PD_WKUP_L4_WKUP_GCLK (OCP) PD_WKUP_UART0_GFCLK (Func)

Table 19-2. UART0 Connectivity Attributes (continued)

Attributes	Type
Reset Signals	WKUP_DOM_RST_N
Idle/Wakeup Signals	Smart Idle / Wakeup
Interrupt Requests	1 interrupt to MPU Subsystem (UART0INT), PRU-ICSS (nirq) and WakeM3
DMA Requests	2 DMA requests to EDMA (TX – UTXEVT0, RX – URXEVT0)
Physical Address	L4 Wakeup slave port

Table 19-3. UART1–5 Connectivity Attributes

Attributes	Type
Power Domain	Peripheral Domain
Clock Domain	PD_PER_L4LS_GCLK (OCP) PD_PER_UART_GFCLK (Func)
Reset Signals	PER_DOM_RST_N
Idle/Wakeup Signals	Smart Idle
Interrupt Requests	UART1-2 1 interrupt per instance to MPU Subsystem (UART1INT, UART2INT) and PRU-ICSS (nirq) UART3-5 1 interrupt per instance to only MPU Subsystem (UART3INT, UART4INT, UART5INT)
DMA Requests	2 DMA requests per instance to EDMA (TX – UTXEVTx, RX – URXEVTx)
Physical Address	L4 Peripheral slave port

19.2.2 UART Clock and Reset Management

The UART modules use separate functional and bus interface clocks.

Table 19-4. UART0 Clock Signals

Clock Signal	Max Freq	Reference / Source	Comments
CLK Interface clock From PRCM	100 MHz	CORE_CLKOUTM4 / 2	pd_wkup_l4_wkup_gclk
FCLK Functional clock From PRCM	48 MHz	PER_CLKOUTM2 / 4	pd_wkup_uart0_gfclk

Table 19-5. UART1–5 Clock Signals

Clock Signal	Max Freq	Reference / Source	Comments
CLK Interface clock	100 MHz	CORE_CLKOUTM4 / 2	pd_per_l4ls_gclk From PRCM
FCLK Functional clock	48 MHz	PER_CLKOUTM2 / 4	pd_per_uart_gfclk From PRCM

For UART operation, the functional clock is used to produce a baud rate up to 3.6M bits/s. Table 19-6 lists the supported baud rates, the requested divider, and the corresponding error versus the standard baud rate.

Table 19-6. UART Mode Baud and Error Rates

Baud rate	Over sampling	Divisor	Error (%)
300	16	10000	0
600	16	5000	0
1200	16	2500	0
2400	16	1250	0
4800	16	625	0
9600	16	313	0.16
14400	16	208	0.16
19200	16	156	0.16
28800	16	104	0.16
38400	16	78	0.16
57600	16	52	0.16
115200	16	26	0.16
230400	16	13	0.16
460800	13	8	0.16
921600	13	4	0.16
1843200	13	2	0.16
3000000	16	1	0
3686400	13	1	0.16

For IrDA operation, the internal functional clock divisor allows generation of SIR, MIR, or FIR baud rates as shown in Table 19-7.

Table 19-7. IrDA Mode Baud and Error Rates

Baud rate	IR mode	Encoding	Divisor	Error (%)
2400	SIR	3/16	1250	0
9600	SIR	3/16	312	0.16
19200	SIR	3/16	156	0.16
38400	SIR	3/16	78	0.16
57600	SIR	3/16	52	0.16
115200	SIR	3/16	26	0.16
576000	MIR	1/4	2	0
1152000	MIR	1/4	1	0
4000000	FIR	4PPM	1	0

19.2.3 UART Pin List

The UART interface pins are listed in Table 19-8. Pin functionality depends on the selected operating mode of the module.

Table 19-8. UART Pin List

Pin	Type	Description
UARTx_RXD / IRRX / RCRX	I	UART / IrDA / CIR Receive Data
UARTx_TXD / IRTX / RCTX	OZ	UART / IrDA / CIR Transmit Data
UARTx_RTSn / SD	OZ	UART Request to Send / IrDA Mode
UARTx_CTSn	I	UART Clear to Send
UARTx_DTRn(1)	OZ	UART Data Terminal Ready
UARTx_DSRn(1)	I	UART Data Set Ready
UARTx_DCDn(1)	I	UART Data Carrier Detect
UARTx_RIn(1)	I	UART Ring Indicator

(1) UART1 only

The UART module can operate in three different modes based on the MODE_SELECT bits. The signal muxing based on these mode bits is shown in Table 19-9.

Table 19-9. UART Muxing Control

UARTx_TXD / IRTX / RCTX Function	UARTx_RXD / IRRX / RCRX Function	UARTx_RTSn / SD Function	UARTx_CTSn Function	Mode
TXD	RXD	RTSn	CTSn	UART
IRTX	IRRX	SD	not used	IrDA (SIR, MIR, FIR)
RCTX	RCRX	SD	not used	CIR

19.3 Functional Description

19.3.1 Block Diagram

The UART/IrDA/CIR module can be divided into three main blocks:

- FIFO management
- Mode selection
- Protocol formatting

FIFO management is common to all functions and enables the transmission and reception of data from the host processor point of view.

There are two modes:

- Function mode: Routes the data to the chosen function (UART, IrDA, or CIR) and enables the mechanism corresponding to the chosen function
- Register mode: Enables conditional access to registers

For more information about mode configuration, see Section 19.3.7, *Mode Selection*.

Protocol formatting has three subcategories:

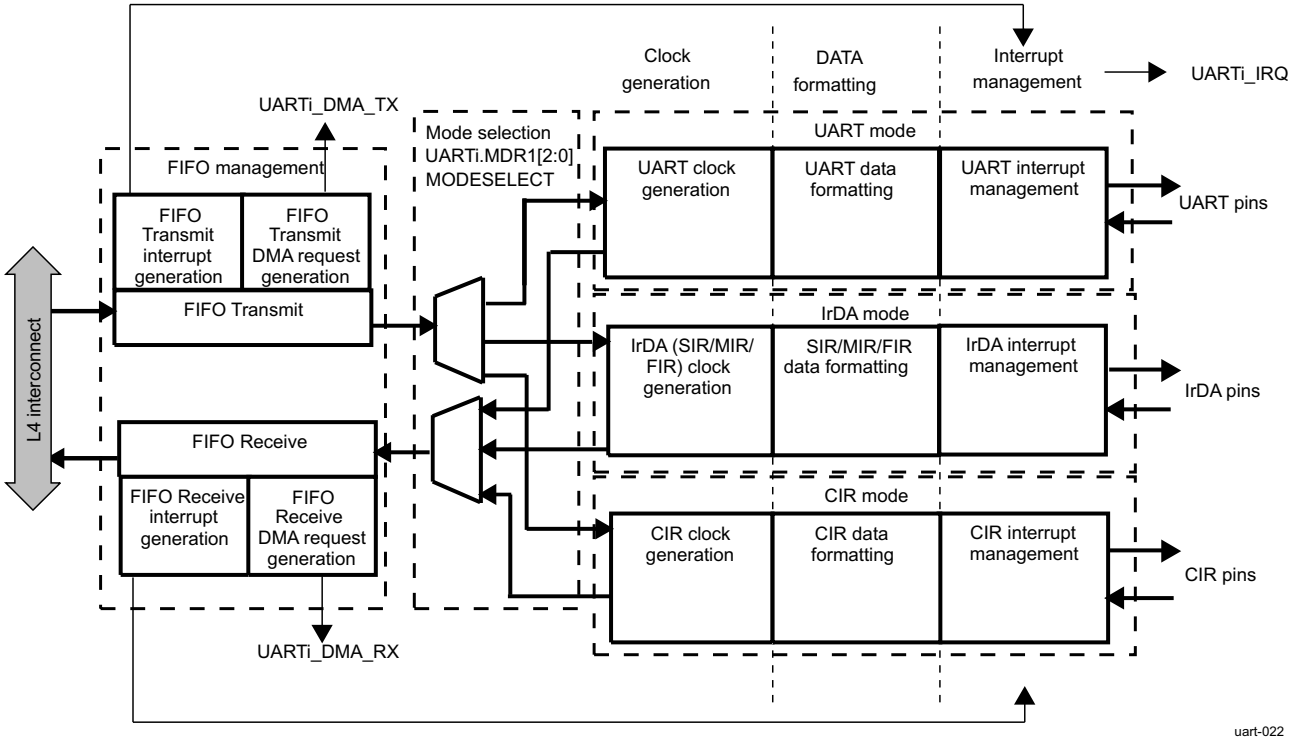
- Clock generation: The 48-MHz input clock generates all necessary clocks.
- Data formatting: Each function uses its own state-machine that is responsible for the transition between FIFO data and frame data associated with it.
- Interrupt management: Different interrupt types are generated depending on the chosen function:
  - UART mode interrupts: Seven interrupts prioritized in six different levels
  - IrDA mode interrupts: Eight interrupts. The interrupt line is activated when any interrupt is generated (there is no priority).
  - CIR mode interrupts: A subset of existing IrDA mode interrupts is used.

  In each mode, when an interrupt is generated, the UART_IIR register indicates the interrupt type.

In parallel with these functional blocks, a power-saving strategy exists for each function.

Figure 19-3 is the UART/IrDA/CIR block diagram.

Figure 19-3. UART/IrDA/CIR Functional Specification Block Diagram



19.3.2 Clock Configuration

Each UART uses a 48-MHz functional clock for its logic and to generate external interface signals. Each UART uses an interface clock for register accesses. The PRCM module generates and controls all these clocks (for more information, see *Clock Domain Module Attributes*, in Chapter 8, *Power, Reset, and Clock Management*).

The idle and wake-up processes use a handshake protocol between the PRCM and the UART (for a description of the protocol, see *Module-Level Clock Management* in Chapter 8, *Power, Reset, and Clock Management*). The UARTi.UART_SYSC[4:3] IDLEMODE bit field controls UART idle mode.

19.3.3 Software Reset

The UARTi.UART_SYSC[1] SOFTRESET bit controls the software reset; setting this bit to 1 triggers a software reset functionally equivalent to hardware reset.

19.3.4 Power Management

19.3.4.1 UART Mode Power Management

19.3.4.1.1 Module Power Saving

In UART modes, sleep mode is enabled by setting the UARTi.UART_IER[4] SLEEP_MODE bit to 1 (when the UARTi.UART_EFR[4] ENHANCED_EN bit is set to 1).

Sleep mode is entered when all the following conditions exist:

- The serial data input line, uarti_rx, is idle.
- The TX FIFO and TX shift register are empty.
- The RX FIFO is empty.
- The only pending interrupts are 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 to modem mode. Therefore, even if the UART has no key role functionally, 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 by these clocks, this greatly reduces power consumption. The module wakes up when a change is detected on the uarti_rx line, when data is written to the TX FIFO, and when there is a change in the state of the modem input pins.

An interrupt can be generated on a wake-up event by setting the UARTi.UART_SCR[4] RX_CTS_WU_EN bit to 1. To understand how to manage the interrupt, see Section 19.3.5.2, *Wake-Up Interrupt*.

NOTE: There must be no writing to the divisor latches, UARTi.UART_DLL and UARTi.UART_DLH, to set the baud clock (BCLK) while in sleep mode. It is advisable to disable sleep mode using the UARTi.UART_IER[4] SLEEP_MODE bit before writing to the UARTi.UART_DLL register or the UARTi.UART_DLH register.

19.3.4.1.2 System Power Saving

Sleep and auto-idle modes are embedded power-saving features. Power-reduction techniques can be applied at the system level by shutting down certain internal clock and power domains of the device.

The UART supports an idle req/idle ack handshaking protocol used at the system level to shut down the UART clocks in a clean and controlled manner and to switch the UART from interrupt-generation mode to wake-up generation mode for unmasked events (see the UARTi.UART_SYSC[2] ENAWAKEUP bit and the UARTi.UART_WER register).

For more information, see *Module Level Clock Management* in Chapter 8, *Power, Reset, and Clock Management*.

19.3.4.2 IrDA/CIR Mode Power Management

19.3.4.2.1 Module Power Saving

In IrDA/CIR modes, sleep mode is enabled by setting the UARTi.MDR[3] IR_SLEEP bit to 1.

Sleep mode is entered when all the following conditions exist:

- The serial data input line, uarti.rx_irrx, is idle.
- The TX FIFO and TX shift register are empty.
- The RX FIFO is empty.
- No interrupts are pending except THR interrupts.

The module wakes up when a change is detected on the uarti_rx_irrx line or when data is written to the TX FIFO.

19.3.4.2.2 System Power Saving

System power saving for the IrDA/CIR mode has the same function as for the UART mode (see Section 19.3.4.1.2, *System Power Saving*).

19.3.4.3 Local Power Management

Table 19-10 describes power-management features available for the UART.

NOTE: For information about source clock gating and sleep/wake-up transitions description, see *Module-Level Clock Management* in Chapter 8, *Power, Reset, and Clock Management*.

Table 19-10. Local Power-Management Features

Feature	Registers	Description
Clock autogating	UART_SYSC[0] AUTOIDLE	This bit allows local power optimization in the module by gating the UARTi_ICLK clock on interface activity or gating the UARTi_FCLK clock on internal activity.
Slave idle modes	UART_SYSC[4:3] IDLEMODE	Force-idle, no-idle, smart-idle, and smart-idle wakeup-capable modes are available
Clock activity	N/A	Feature not available
Master standby modes	N/A	Feature not available
Global wake-up enable	UART_SYSC[2] ENAWAKEUP	This bit enables the wake-up feature at module level.
Wake-Up sources enable	N/A	Feature not available

19.3.5 Interrupt Requests

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

19.3.5.1 UART Mode Interrupt Management

19.3.5.1.1 UART Interrupts

UART mode includes seven possible interrupts prioritized to six levels.

When an interrupt is generated, the interrupt identification register (UARTi.UART_IIR) sets the UARTi.UART_IIR[0] IT_PENDING bit to 0 to indicate that an interrupt is pending, and indicates the type of interrupt through the UARTi.UART_IIR[5:1] bit field. Table 19-11 summarizes the interrupt control functions.

Table 19-11. UART Mode Interrupts

UART_IIR[5:0]	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Method
000001	None	None	None	None
000110	1	Receiver line status	OE, FE, PE, or BI errors occur in characters in the RX FIFO.	FE, PE, BI: Read the UART_RHR register. OE: Read the UART_LSR register.
001100	2	RX time-out	Stale data in RX FIFO	Read the UART_RHR register.
000100	2	RHR interrupt	DRDY (data ready) (FIFO disable)	Read the UART_RHR register until the interrupt condition disappears.
			RX FIFO above trigger level (FIFO enable)	
000010	3	THR interrupt	TFE (UART_THR empty) (FIFO disable)	Write to the UART_THR until the interrupt condition disappears.
			TX FIFO below trigger level (FIFO enable)	
000000	4	Modem status	See the UART_MSR register.	Read the UART_MSR register.
010000	5	XOFF interrupt/special character interrupt	Receive XOFF characters/special character	Receive XON character(s), if XOFF interrupt/read of the UART_IIR register, if special character interrupt.
100000	6	CTS, RTS, DSR	RTS pin or CTS pin or DSR change state from active (low) to inactive (high).	Read the UART_IIR register.

For the receiver-line status interrupt, the RX_FIFO_STS bit (UARTi.UART_LSR[7]) generates the 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 UARTi.UART_IIR register.

19.3.5.2 Wake-Up Interrupt

Wake-up interrupt is a special interrupt that works differently from other interrupts. This interrupt is enabled when the UARTi.UART_SCR[4] RXCTSDSRWAKEUPENABLE bit is set to 1. The UARTi.UART_IIR register is not modified when this occurs; the UARTi.UART_SSR[1] RXCTSDSRWAKEUPSTS bit must be checked to detect a wake-up event.

When a wake-up interrupt occurs, it can be cleared only by resetting the UARTi.UART_SCR[4] RXCTSDSRWAKEUPENABLE bit. This bit must be re-enabled (set to 1) after the current wake-up interrupt event is processed to detect the next incoming wake-up event.

A wake-up interrupt can also occur if the WER[7] TXWAKEUPEN bit is set to 1 and one of the following occurs:

- THR interrupt occurred if it is enabled (omitted if TX DMA request is enabled).
- TX DMA request occurred if it is enabled.
- TX_STATUS_IT occurred if it is enabled (only IrDA and CIR modes). Cannot be used with THR interrupt.

CAUTION

Wake-Up interface implementation in IrDA mode is based on the UARTi_SIDLEACK low-to-high transition instead of the UARTi_SIDLEACK state.

This does not ensure wake-up event generation as expected when configured in smart-idle mode, and the system wakes up for a short period.

19.3.5.3 IrDA Mode Interrupt Management

19.3.5.3.1 IrDA Interrupts

The IrDA function generates interrupts. All interrupts can be enabled and disabled by writing to the appropriate bit in the interrupt enable register (UARTi.UART_IER). The interrupt status of the device can be checked by reading the interrupt identification register (UARTi.UART_IIR).

UART, IrDA, and CIR modes have different interrupts in the UART/IrDA/CIR module and, therefore, different UARTi.UART_IER and UARTi.UART_IIR mappings, depending on the selected mode.

IrDA modes have eight possible interrupts (see Table 19-12). The interrupt line is activated when any interrupt is generated (there is no priority).

Table 19-12. IrDA Mode Interrupts

UART_IIR Bit	Interrupt Type	Interrupt Source	Interrupt Reset Method
0	RHR interrupt	DRDY (data ready) (FIFO disable) RX FIFO above trigger level (FIFO enable)	Read the UART_RHR register until the interrupt condition disappears.
1	THR interrupt	TFE (UART_THR empty) (FIFO disable) TX FIFO below trigger level (FIFO enable)	Write to the UART_THR until the interrupt condition disappears.

Table 19-12. IrDA Mode Interrupts (continued)

UART_IIR Bit	Interrupt Type	Interrupt Source	Interrupt Reset Method
2	Last byte in RX FIFO	Last byte of frame in RX FIFO is available to be read at the RHR port.	Read the UART_RHR register.
3	RX overrun	Write to the UART_RHR register when the RX FIFO is full.	Read UART_RESUME register.
4	Status FIFO interrupt	Status FIFO triggers level reached.	Read STATUS FIFO.
5	TX status	1. UART_THR empty before EOF sent. Last bit of transmission of the IrDA frame occurred, but with an underrun error. OR 2. Transmission of the last bit of the IrDA frame completed successfully.	1. Read the UART_RESUME register. OR 2. Read the UART_IIR register.
6	Receiver line status interrupt	CRC, ABORT, or frame-length error is written into the STATUS FIFO.	Read the STATUS FIFO (read until empty - maximum of eight reads required).
7	Received EOF	Received end-of-frame	Read the UART_IIR register.

19.3.5.4 CIR Mode Interrupt Management

19.3.5.4.1 CIR Interrupts

The CIR function generates interrupts that can be enabled and disabled by writing to the appropriate bit in the interrupt enable register (UARTi.UART_IER). The interrupt status of the device can be checked by reading the interrupt identification register (UARTi.UART_IIR).

UART, IrDA, and CIR modes have different interrupts in the UART/IrDA/CIR module and, therefore, different UARTi.UART_IER and UARTi.UART_IIR mappings, depending on the selected mode.

Table 19-13 lists the interrupt modes to be maintained. In CIR mode, the sole purpose of the UARTi.UART_IIR[5] bit is to indicate that the last bit of infrared data was passed to the uart_cts_rctx pin.

Table 19-13. CIR Mode Interrupts

UART_IIR Bit Number	Interrupt Type	Interrupt Source	Interrupt Reset Method
0	RHR interrupt	DRDY (data ready) (FIFO disable) RX FIFO above trigger level (FIFO enable)	Read UART_RHR until interrupt condition disappears.
1	THR interrupt	TFE (UART_THR empty) (FIFO disable) TX FIFO below trigger level (FIFO enable)	Write to the UART_THR register until the interrupt condition disappears.
2	RX_STOP_IT	Receive stop interrupt (depending on value set in the BOF Length Register (UART_EBLR).	Read IIR
3	RX overrun	Write to RHR when RX FIFO is full.	Read RESUME register.
4	N/A for CIR mode	N/A for CIR mode	N/A for CIR mode
5	TX status	Transmission of the last bit of the frame is complete successfully.	Read the UART_IIR register.
6	N/A for CIR mode	N/A for CIR mode	N/A for CIR mode
7	N/A for CIR mode	N/A for CIR mode	N/A for CIR mode

19.3.6 FIFO Management

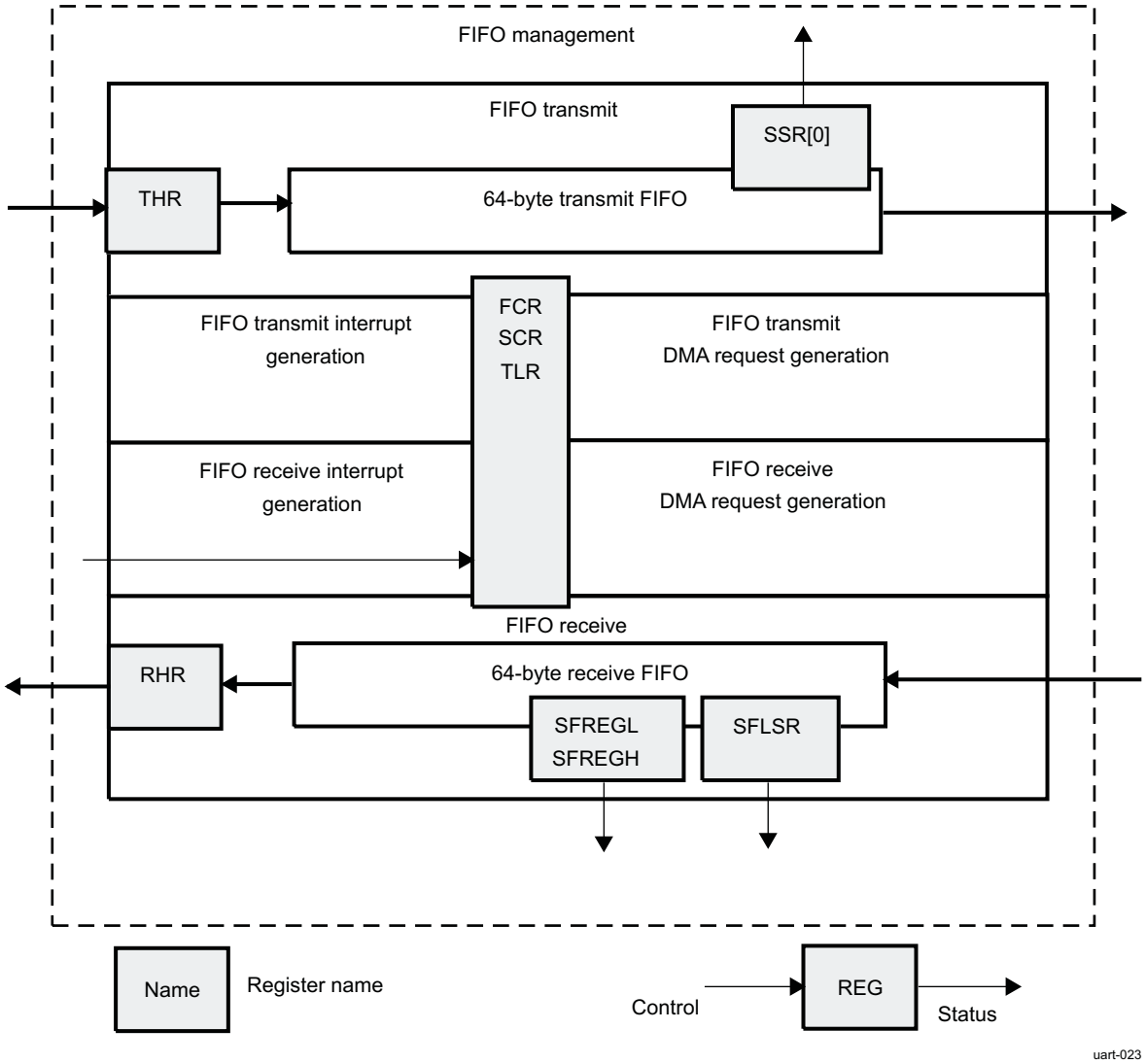
The FIFO is accessed by reading and writing the UARTi.UART_RHR and UARTi.UART_THR registers. Parameters are controlled using the FIFO control register (UARTi.UART_FCR) and supplementary control register (UARTi.UART_SCR). Reading the UARTi.UART_SSR[0] TX_FIFO_FULL bit at 1 means the FIFO is full.

The UARTi.UART_TLR register controls the FIFO trigger level, which enables DMA and interrupt generation. After reset, transmit (TX) and receive (RX) FIFOs are disabled; thus, the trigger level is the default value of 1 byte. Figure 19-4 shows the FIFO management registers.

NOTE: Data in the UARTi.UART_RHR register is not overwritten when an overflow occurs.

NOTE: The UARTi.UART_SFLSR, UARTi.UART_SFREGL, and UARTi.UART_SFREGH status registers are used in IrDA mode only. For use, see Section 19.3.8.2.6, *IrDA Data Formatting*.

Figure 19-4. FIFO Management Registers



19.3.6.1 FIFO Trigger

19.3.6.1.1 Transmit FIFO Trigger

Table 19-14 lists the TX FIFO trigger level settings.

Table 19-14. TX FIFO Trigger Level Setting Summary

UART_SCR[6]	UART_TLR[3:0]	TX FIFO Trigger Level
0	= 0x0	Defined by the UARTi.UART_FCR[5:4] TX_FIFO_TRIG bit field (8,16, 32, or 56 spaces)
0	!= 0x0	Defined by the UARTi.UART_TLR[3:0] TX_FIFO_TRIG_DMA bit field (from 4 to 60 spaces with a granularity of 4 spaces)
1	Value	Defined by the concatenated value of TX_FIFO_TRIG_DMA and TX_FIFO_TRIG (from 1 to 63 spaces with a granularity of 1 space) **Note:** The combination of TX_FIFO_TRIG_DMA = 0x0 and TX_FIFO_TRIG = 0x0 (all zeros) is not supported (minimum of one space required). All zeros result in unpredictable behavior.

19.3.6.1.2 Receive FIFO Trigger

Table 19-15 lists the RX FIFO trigger level settings.

Table 19-15. RX FIFO Trigger Level Setting Summary

UART_SCR[7]	UART_TLR[7:4]	RX FIFO Trigger Level
0	= 0x0	Defined by the UARTi.UART_FCR[7:6] RX_FIFO_TRIG bit field (8,16, 56, or 60 characters)
0	!= 0x0	Defined by the UARTi.UART_TLR[7:4] RX_FIFO_TRIG_DMA bit field (from 4 to 60 characters with a granularity of 4 characters)
1	Value	Defined by the concatenated value of RX_FIFO_TRIG_DMA and RX_FIFO_TRIG (from 1 to 63 characters with a granularity of 1 character) **Note:** The combination of RX_FIFO_TRIG_DMA = 0x0 and RX_FIFO_TRIG = 0x0 (all zeros) is not supported (minimum of one character required). All zeros result in unpredictable behavior.

The receive threshold is programmed using the UARTi.UART_TCR[7:4] RX_FIFO_TRIG_START and UARTi.UART_TCR[3:0] RX_FIFO_TRIG_HALT bit fields:

- Trigger levels from 0 to 60 bytes are available with a granularity of 4 (trigger level = 4 x [4-bit register value]).
- To ensure correct device operation, ensure that RX_FIFO_TRIG_HALT RX_FIFO_TRIG when auto-RTS is enabled.

  Delay = [4 + 16 x (1 + CHAR_LENGTH + Parity + Stop 0.5)] x Baud_rate + 4 x FCLK

  NOTE: The RTS signal is deasserted after the UART module receives the data over RX_FIFO_TRIG_HALT. Delay means how long the UART module takes to deassert the RTS signal after reaching RX_FIFO_TRIG_HALT.

- In FIFO interrupt mode with flow control, ensure that the trigger level to HALT transmission is greater than or equal to the RX FIFO trigger level (the UARTi.UART_TCR[7:4] RX_FIFO_TRIG_START bit field or the UARTi.UART_FCR[7:6] RX_FIFO_TRIG bit field); otherwise, FIFO operation stalls. In FIFO DMA mode with flow control, this concept does not exist, because a DMA request is sent when a byte is received.

19.3.6.2 FIFO Interrupt Mode

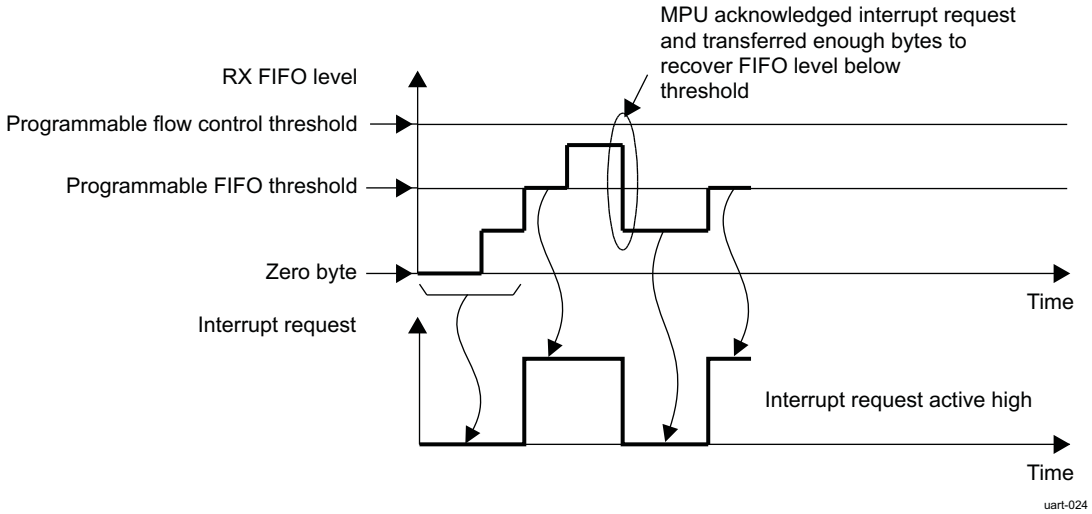
In FIFO interrupt mode (the FIFO control register UARTi.UART_FCR[0] FIFO_EN bit is set to 1 and relevant interrupts are enabled by the UARTi.UART_IER register), an interrupt signal informs the processor of the status of the receiver and transmitter. These interrupts are raised when the RX/TX FIFO threshold (the UARTi.UART_TLR[7:4] RX_FIFO_TRIG_DMA and UARTi.UART_TLR[3:0] TX_FIFO_TRIG_DMA bit fields or the UARTi.UART_FCR[7:6] RX_FIFO_TRIG and UARTi.UART_FCR[5:4] TX_FIFO_TRIG bit fields, respectively) is reached.

The interrupt signals instruct the MPU to transfer data to the destination (from the UART in receive mode and/or from any source to the UART FIFO in transmit mode).

When UART flow control is enabled with interrupt capabilities, the UART flow control FIFO threshold (the UARTi.UART_TCR[3:0] RX_FIFO_TRIG_HALT bit field) must be greater than or equal to the RX FIFO threshold.

Figure 19-5 shows the generation of the RX FIFO interrupt request.

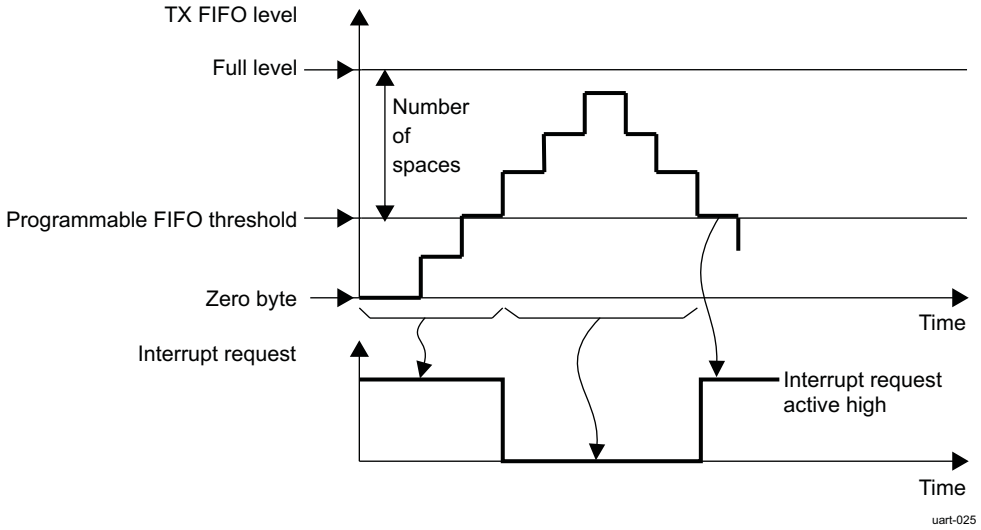
Figure 19-5. RX FIFO Interrupt Request Generation



In receive mode, no interrupt is generated until the RX FIFO reaches its threshold. Once low, the interrupt can be deasserted only when the MPU has handled enough bytes to put the FIFO level below threshold. The flow control threshold is set at a higher value than the FIFO threshold.

Figure 19-6 shows the generation of the TX FIFO interrupt request.

Figure 19-6. TX FIFO Interrupt Request Generation



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

19.3.6.3 FIFO Polled Mode Operation

In FIFO polled mode (the UARTi.UART_FCR[0] FIFO_EN bit is set to 0 and the relevant interrupts are disabled by the UARTi.UART_IER register), the status of the receiver and transmitter can be checked by polling the line status register (UARTi.UART_LSR).

This mode is an alternative to the FIFO interrupt mode of operation in which the status of the receiver and transmitter is automatically determined by sending interrupts to the MPU.

19.3.6.4 FIFO DMA Mode Operation

Although DMA operation includes four modes (DMA modes 0 through 3), assume that mode 1 is used. (Mode 2 and mode 3 are legacy modes that use only one DMA request for each module.)

In mode 2, the remaining DMA request is used for RX. In mode 3, the remaining DMA request is used for TX.

DMA requests in mode 2 and mode 3 use the following signals:

- S_DMA_48
- S_DMA_50
- S_DMA_52/D_DMA_10
- S_DMA_54

The following signals are not used by the module in mode 2 and mode 3:

- S_DMA_49
- S_DMA_51
- S_DMA_53/D_DMA_11
- S_DMA_55

These signals can be selected as follows:

- When the UARTi.UART_SCR[0] DMA_MODE_CTL bit is set to 0, setting the UARTi.UART_FCR[3]DMA_MODE bit to 0 enables DMA mode 0. Setting the DMA_MODE bit to 1 enables DMA mode 1.
- When the DMA_MODE_CTL bit is set to 1, the UARTi.UART_SCR[2:1]DMA_MODE_2 bit field determines DMA mode 0 to mode 3 based on the supplementary control register (UART_SCR) description.

For example:

- If no DMA operation is desired, set the DMA_MODE_CTL bit to 1 and the DMA_MODE_2 bit field to 0x0. (The DMA_MODE bit is discarded.)
- If DMA mode 1 is desired, set the DMA_MODE_CTL bit to 0 and the DMA_MODE bit to 1, or set the DMA_MODE_CTL bit to 1 and the DMA_MODE_2 bit field to 01. (The DMA_MODE bit is discarded.)

If the FIFOs are disabled (the UARTi.UART_FCR[0] FIFO_EN bit is set to 0), the DMA occurs in single-character transfers.

When DMA mode 0 is programmed, the signals associated with DMA operation are not active.

Depending on UART_MDR3[2] SET_DMA_TX_THRESHOLD, the threshold can be programmed different ways:

- SET_TX_DMA_THRESHOLD = 1:

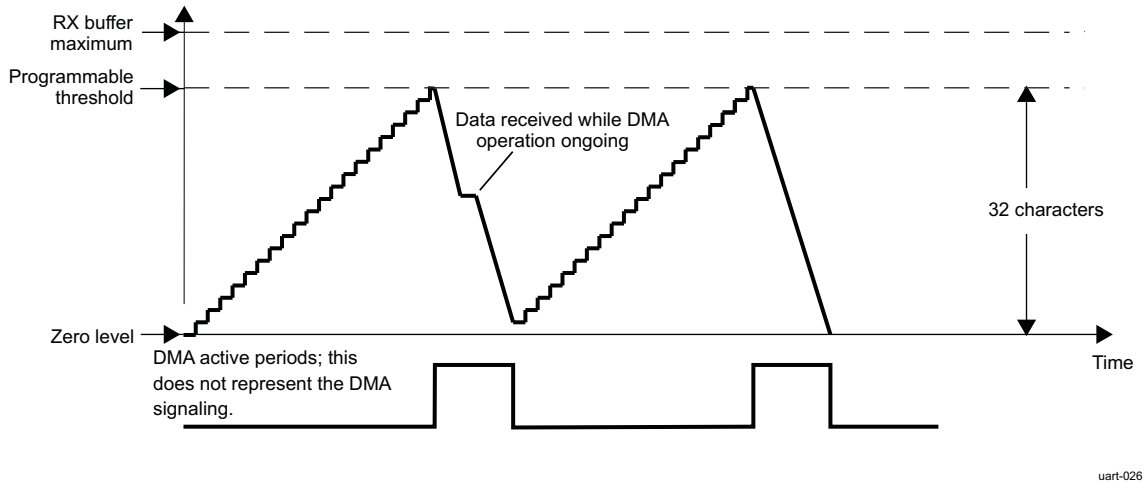
  The threshold value will be the value of the UART_TX_DMA_THRESHOLD register. If SET_TX_DMA_THRESHOLD + TX trigger spaces 64, then the default method of threshold is used: threshold value = TX FIFO size.
- SET_TX_DMA_THRESHOLD = 0:

  The threshold value = TX FIFO size - TX trigger space. The TX DMA line is asserted if the TX FIFO level is lower then the threshold. It remains asserted until TX trigger spaces number of bytes are written into the FIFO. The DMA line is then deasserted and the FIFO level is compared with the threshold value.

19.3.6.4.1 DMA Transfers (DMA Mode 1, 2, or 3)

Figure 19-7 through Figure 19-10 show the supported DMA operations.

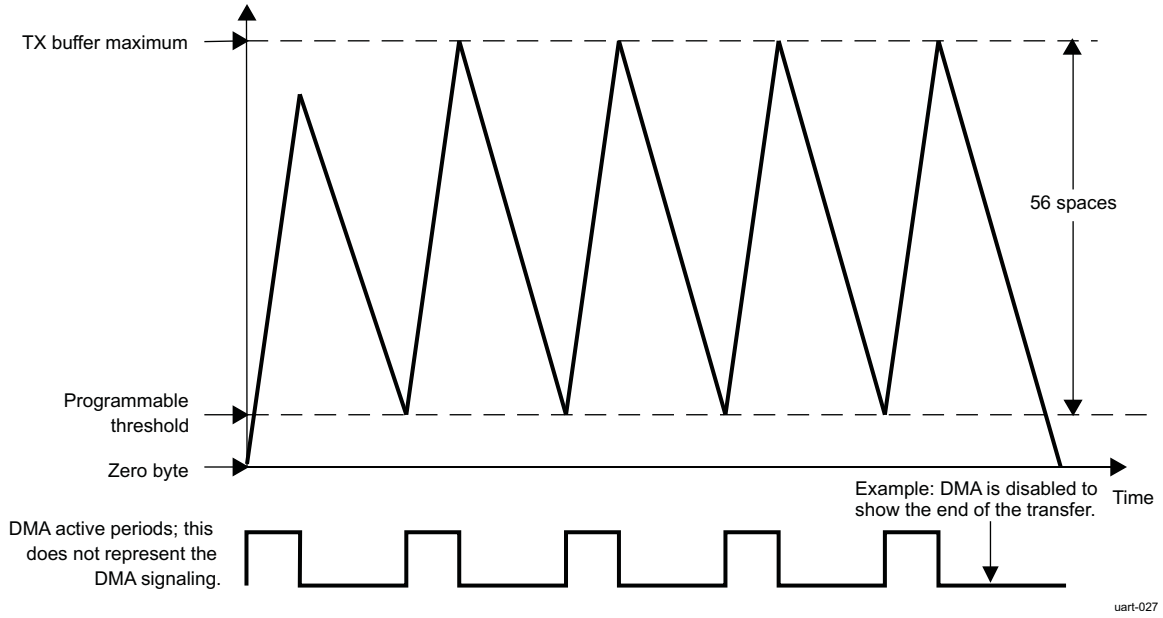
Figure 19-7. Receive FIFO DMA Request Generation (32 Characters)



In receive mode, a DMA request is generated when the RX FIFO reaches its threshold level defined in the trigger level register (UARTi.UART_TLR). This request is deasserted when the number of bytes defined by the threshold level is read by the sDMA.

In transmit mode, a DMA request is automatically asserted when the TX FIFO is empty. This request is deasserted when the number of bytes defined by the number of spaces in the UARTi.UART_TLR register is written by the sDMA. If an insufficient number of characters is written, the DMA request stays active.

Figure 19-8. Transmit FIFO DMA Request Generation (56 Spaces)



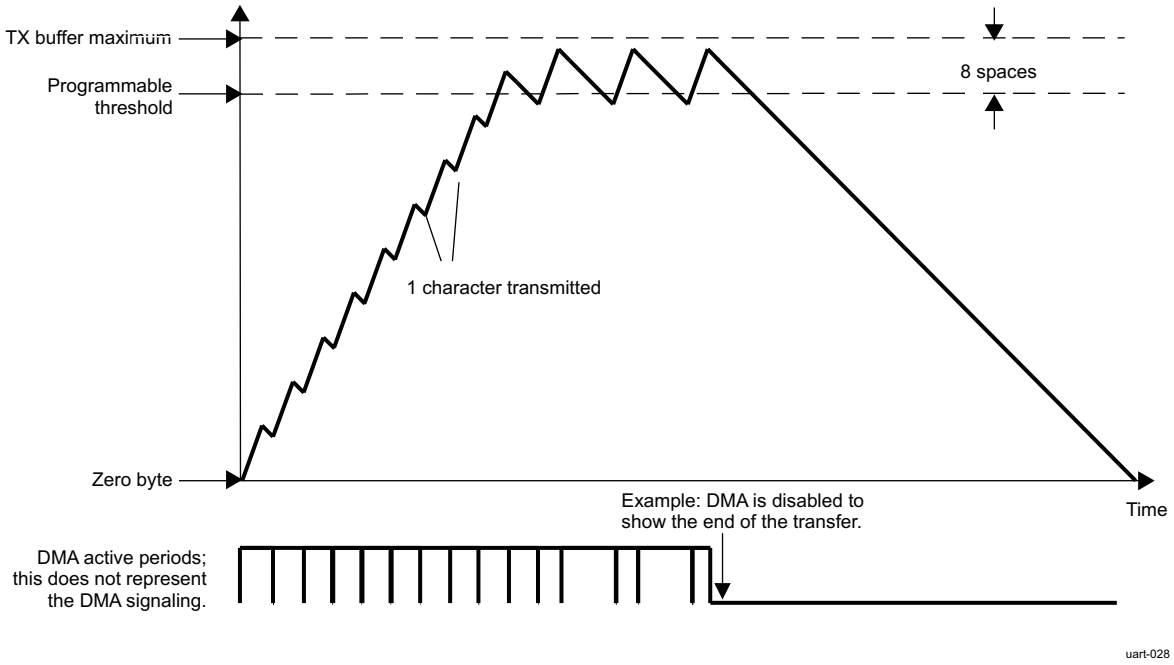
The DMA request is again asserted if the FIFO can receive the number of bytes defined by the UARTi.UART_TLR register.

The threshold can be programmed in a number of ways. Figure 19-8 shows a DMA transfer operating with a space setting of 56 that can arise from using the auto settings in the UARTi.UART_FCR[5:4] TX_FIFO_TRIG bit field or the UARTi.UART_TLR[3:0] TX_FIFO_TRIG_DMA bit field concatenated with the TX_FIFO_TRIG bit field.

The setting of 56 spaces in the UART/IrDA/CIR module must correlate with the settings of the sDMA so that the buffer does not overflow (program the DMA request size of the LH controller to equal the number of spaces in the UART/IrDA/CIR module).

Figure 19-9 shows an example with eight spaces to show the buffer level crossing the space threshold. The LH DMA controller settings must correspond to those of the UART/IrDA/CIR module.

Figure 19-9. Transmit FIFO DMA Request Generation (8 Spaces)

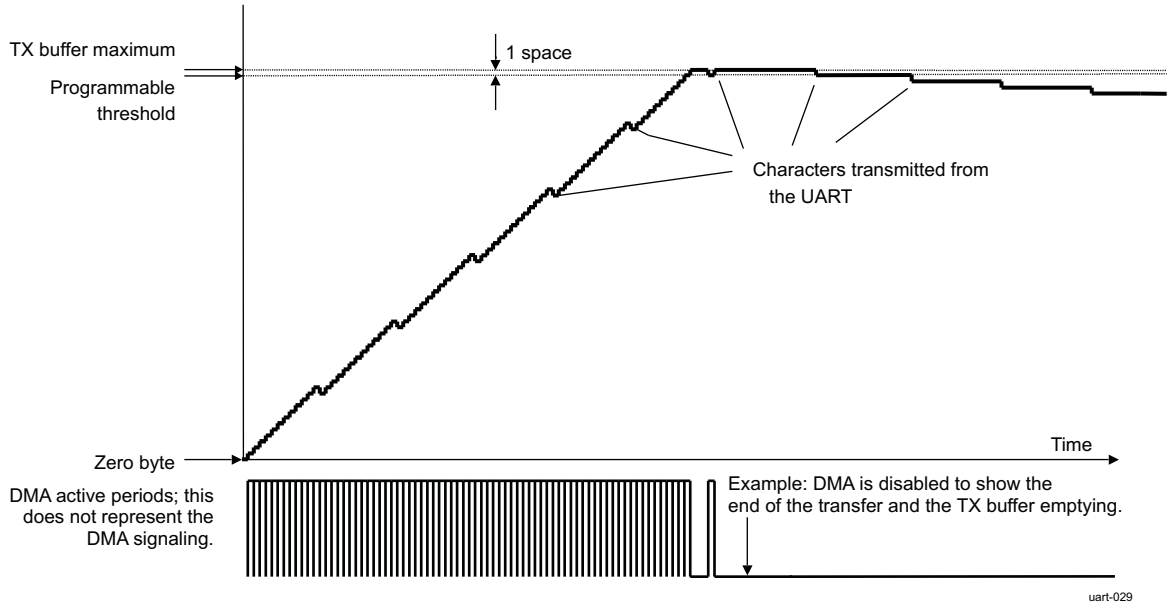


The next example shows the setting of one space that uses the DMA for each transfer of one character to the transmit buffer (see Figure 19-10). The buffer is filled faster than the baud rate at which data is transmitted to the TX pin. Eventually, the buffer is completely full and the DMA operations stop transferring data to the transmit buffer.

On two occasions, the buffer holds the maximum amount of data words; shortly after this, the DMA is disabled to show the slower transmission of the data words to the TX pin. Eventually, the buffer is emptied at the rate specified by the baud rate settings of the UARTi.UART_DLL and UARTi.UART_DLH registers.

The DMA settings must correspond to the system LH DMA controller settings to ensure correct operation of this logic.

Figure 19-10. Transmit FIFO DMA Request Generation (1 Space)



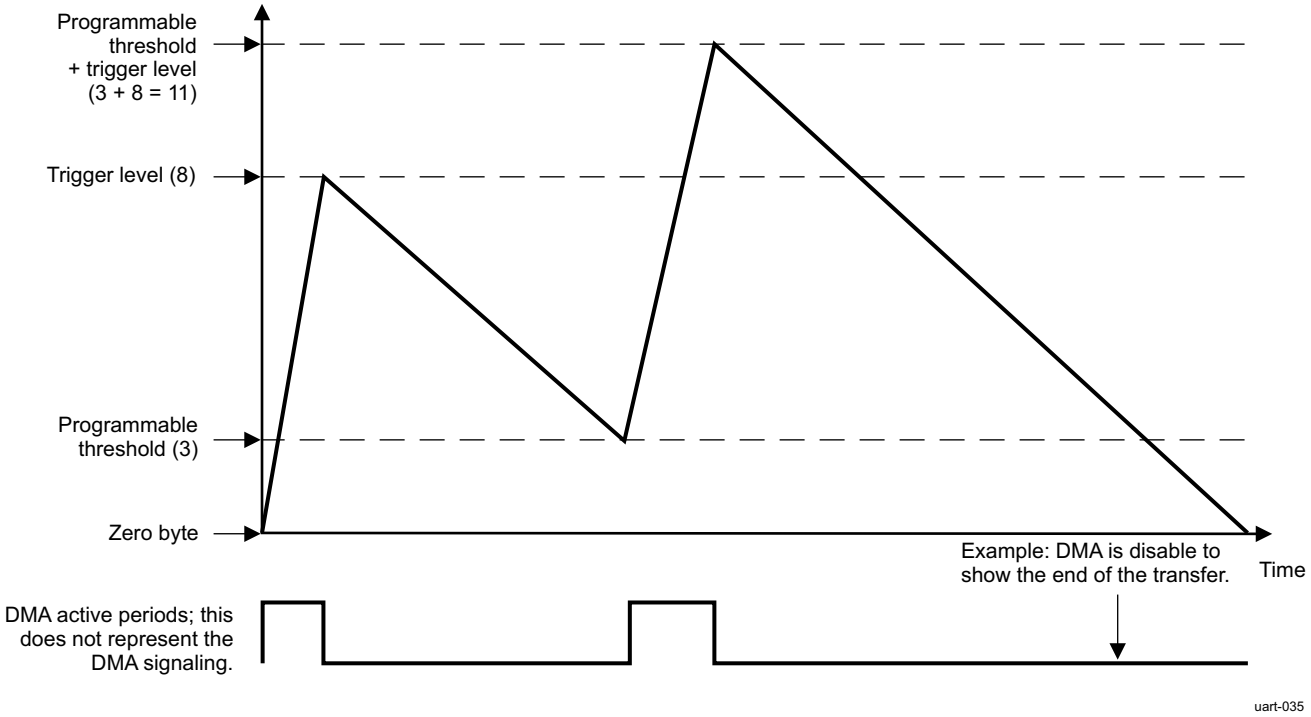
The final example shows the setting of eight spaces, but setting the TX DMA threshold directly by setting the UART_MDR3[1]SET_DMA_RX_THRESHOLD bit and the UART_TX_DMA_THRESHOLD register (see Figure 19-11). In the example, UART_TX_DMA_THRESHOLD[2:0]TX_DMA_THRESHOLD = 3 and the trigger level is 8. The buffer is filled at a faster rate than the baud rate transmits data to the TX pin. The buffer is filled with 8 bytes and the DMA operations stop transferring data to the transmit buffer. When the buffer is emptied to the threshold level by transmission, the DMA operation activates again to fill the buffer with 8 bytes.

Eventually, the buffer is emptied at the rate specified by the baud rate settings of the UART_DLL and UART_DLH registers.

If the selected threshold level plus the trigger level exceed the maximum buffer size, the original TX DMA threshold method is used to prevent TX overrun, regardless of the value of the UART_MDR3[1]SET_DMA_RX_THRESHOLD bit.

The DMA settings must correspond to the settings of the system local host DMA controller to ensure the correct operation of this logic.

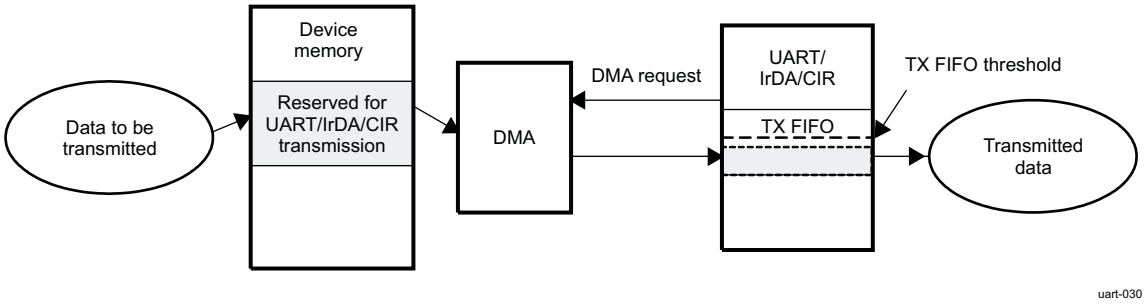
Figure 19-11. Transmit FIFO DMA Request Generation Using Direct TX DMA Threshold Programming. (Threshold = 3; Spaces = 8)



19.3.6.4.2 DMA Transmission

Figure 19-12 shows DMA transmission.

Figure 19-12. DMA Transmission



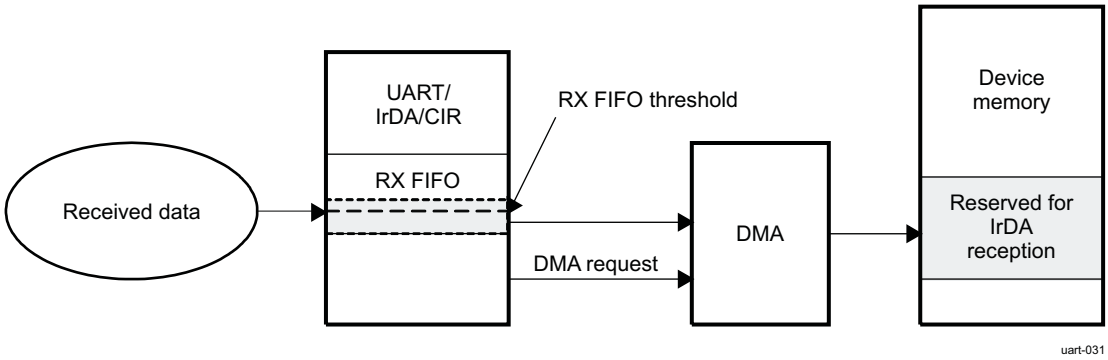
1. Data to be transmitted are put in the device memory reserved for UART/IrDA/CIR transmission by the DMA:
   (a) Until the TX FIFO trigger level is not reached, a DMA request is generated
   (b) An element (1 byte) is transferred from the SDRAM to the TX FIFO at each DMA request (DMA element synchronization).
2. Data in the TX FIFO are automatically transmitted.
3. The end of the transmission is signaled by the UARTi.UART_THR empty (TX FIFO empty).

NOTE: In IrDA mode, the transmission does not end immediately after the TX FIFO empties, at which point the last data byte, the CRC field, and the stop flag still must be transmitted; thus, the end of transmission occurs a few milliseconds after the UARTi.UART_THR register empties.

19.3.6.4.3 DMA Reception

Figure 19-13 shows DMA reception.

Figure 19-13. DMA Reception



1. Enable the reception.
2. Received data are put in the RX FIFO.
3. Data are transferred from the RX FIFO to the device memory by the DMA:
   (a) At each received byte, the RX FIFO trigger level (one character) is reached and a DMA request is generated.
   (b) An element (1 byte) is transferred from the RX FIFO to the SDRAM at each DMA request (DMA element synchronization).
4. The end of the reception is signaled by the EOF interrupt.

19.3.7 Mode Selection

19.3.7.1 Register Access Modes

19.3.7.1.1 Operational Mode and Configuration Modes

Register access depends on the register access mode, although register access modes are not correlated to functional mode selection. Three different modes are available:

- Operational mode
- Configuration mode A
- Configuration mode B

Operational mode is the selected mode when the function is active; serial data transfer can be performed in this mode.

Configuration mode A and configuration mode B are used during module initialization steps. These modes enable access to configuration registers, which are hidden in the operational mode. The modes are used when the module is inactive (no serial data transfer processed) and only for initialization or reconfiguration of the module.

The value of the UARTi.UART_LCR register determines the register access mode (see Table 19-16).

Table 19-16. UART/IrDA/CIR Register Access Mode Programming (Using UART_LCR)

Mode	Condition
Configuration mode A	UART_LCR[7] = 0x1 and UART_LCR[7:0] != 0xBF
Configuration mode B	UART_LCR[7] = 0x1 and UART_LCR[7:0] = 0xBF
Operational mode	UART_LCR[7] = 0x0

19.3.7.1.2 Register Access Submode

In each access register mode (operational mode or configuration mode A/B), some register accesses are conditional on the programming of a submode (MSR_SPR, TCR_TLR, and XOFF).

Table 19-17 through Table 19-19 summarize the register access submodes.

Table 19-17. Subconfiguration Mode A Summary

Mode	Condition
MSR_SPR	(UART_EFR[4] = 0x0 or UART_MCR[6] = 0x0)
TCR_TLR	UART_EFR[4] = 0x1 and UART_MCR[6] = 0x1

Table 19-18. Subconfiguration Mode B Summary

Mode	Condition
TCR_TLR	UART_EFR[4] = 0x1 and UART_MCR[6] = 0x1
XOFF	(UART_EFR[4] = 0x0 or UART_MCR[6] = 0x0)

Table 19-19. Suboperational Mode Summary

Mode	Condition
MSR_SPR	UART_EFR[4] = 0x0 or UART_MCR[6] = 0x0
TCR_TLR	UART_EFR[4] = 0x1 and UART_MCR[6] = 0x1

19.3.7.1.3 Registers Available for the Register Access Modes

Table 19-20 lists the names of the register bits in each access register mode. Gray shading indicates that the register does not depend on the register access mode (available in all modes).

Table 19-20. UART/IrDA/CIR Register Access Mode Overview

Address Offset	Registers					
	Configuration Mode A		Configuration Mode B		Operational Mode	
	Read	Write	Read	Write	Read	Write
0x000	UART_DLL	UART_DLL	UART_DLL	UART_DLL	UART_RHR	UART_THR
0x004	UART_DLH	UART_DLH	UART_DLH	UART_DLH	UART_IER	UART_IER
0x008	UART_IIR	UART_FCR	UART_EFR	UART_EFR	UART_IIR	UART_FCR
0x00C	UART_LCR	UART_LCR	UART_LCR	UART_LCR	UART_LCR	UART_LCR
0x010	UART_MCR	UART_MCR	UART_XON1_ADDR1	UART_XON1_ADDR1	UART_MCR	UART_MCR

Table 19-20. UART/IrDA/CIR Register Access Mode Overview (continued)

Address Offset	Registers					
	Configuration Mode A		Configuration Mode B		Operational Mode	
	Read	Write	Read	Write	Read	Write
0x014	UART_LSR	–	UART_XON2_ADDR2	UART_XON2_ADDR2	UART_LSR	–
0x018	UART_MSR (1)/UART_TCR (2)	UART_TCR (2)	UART_TCR (2)/UART_XOFF1 (3)	UART_TCR (2)/UART_XOFF1 (3)	UART_MSR (1)/UART_TCR (2)	UART_TCR (2)
0x01C	UART_SPR (1)/UART_TLR (2)	UART_SPR (1)/UART_TLR (2)	UART_TLR (2)/UART_XOFF2 (3)	UART_TLR (2)/UART_XOFF2 (3)	UART_SPR (1)/UART_TLR (2)	UART_SPR (1)/UART_TLR (2)
0x020	UART_MDR1	UART_MDR1	UART_MDR1	UART_MDR1	UART_MDR1	UART_MDR1
0x024	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2
0x028	UART_SFLSR	UART_TXFLL	UART_SFLSR	UART_TXFLL	UART_SFLSR	UART_TXFLL
0x02C	UART_RESUME	UART_TXFLH	UART_RESUME	UART_TXFLH	UART_RESUME	UART_TXFLH
0x030	UART_SFREGL	UART_RXFLL	UART_SFREGL	UART_RXFLL	UART_SFREGL	UART_RXFLL
0x034	UART_SFREGH	UART_RXFLH	UART_SFREGH	UART_RXFLH	UART_SFREGH	UART_RXFLH
0x038	UART_UASR	–	UART_UASR	–	UART_BLR	UART_BLR
0x03C	–	–	–	–	UART_ACREG	UART_ACREG
0x040	UART_SCR	UART_SCR	UART_SCR	UART_SCR	UART_SCR	UART_SCR
0x044	UART_SSR	–	UART_SSR	–	UART_SSR	–
0x048	–	–	–	–	UART_EBLR	UART_EBLR
0x050	UART_MVR	–	UART_MVR	–	UART_MVR	–
0x054	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC
0x058	UART_SYSS	–	UART_SYSS	–	UART_SYSS	–
0x05C	UART_WER	UART_WER	UART_WER	UART_WER	UART_WER	UART_WER
0x060	UART_CFPS	UART_CFPS	UART_CFPS	UART_CFPS	UART_CFPS	UART_CFPS
0x064	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL
0x068	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL
0x06C	UART_IER2	UART_IER2	UART_IER2	UART_IER2	UART_IER2	UART_IER2
0x070	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2
0x074	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL
0x080	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3
0x084	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD

(1) MSR_SPR mode is active (see Section 19.3.7.1.2, *Register Access Submode*)
(2) TCR_TLR mode is active (see Section 19.3.7.1.2, *Register Access Submode*)
(3) XOFF mode is active (see Section 19.3.7.1.2, *Register Access Submode*)

19.3.7.2 UART/IrDA (SIR, MIR, FIR)/CIR Mode Selection

To select a mode, set the UARTi.UART_MDR1[2:0] MODESELECT bit field (see Table 19-21).

Table 19-21. UART Mode Selection

Value	Mode
0x0:	UART 16x mode
0x1:	SIR mode
0x2:	UART 16x auto-baud
0x3:	UART 13x mode
0x4:	MIR mode
0x5:	FIR mode
0x6:	CIR mode

MODESELECT is effective when the module is in operational mode (see Section 19.3.7.1, *Register Access Modes*).

19.3.7.2.1 Registers Available for the UART Function

Only the registers listed in Table 19-22 are used for the UART function.

Table 19-22. UART Mode Register Overview(1) (2)

Address Offset	Registers					
	Configuration Mode A		Configuration Mode B		Operational Mode	
	Read	Write	Read	Write	Read	Write
0x000	UART_DLL	UART_DLL	UART_DLL	UART_DLL	UART_RHR	UART_THR
0x004	UART_DLH	UART_DLH	UART_DLH	UART_DLH	UART_IER(UART)	UART_IER(UART)
0x008	UART_IIR	UART_FCR	UART_EFR[4]	UART_EFR[4]	UART_IIR(UART)	UART_FCR(UART)
0x00C	UART_LCR	UART_LCR	UART_LCR	UART_LCR	UART_LCR	UART_LCR
0x010	UART_MCR	UART_MCR	UART_XON1_ADDR1	UART_XON1_ADDR1	UART_MCR	UART_MCR
0x014	UART_LSR(UART)	–	UART_XON2_ADDR2	UART_XON2_ADDR2	UART_LSR(UART)	–
0x018	UART_MSR/UART_TCR	UART_TCR	UART_XOFF1/UART_TCR	UART_XOFF1/UART_TCR	UART_MSR/UART_TCR	UART_TCR
0x01C	UART_TLR/UART_SPR	UART_TLR/UART_SPR	UART_TLR/UART_XOFF2	UART_TLR/UART_XOFF2	UART_TLR/UART_SPR	UART_TLR/UART_SPR
0x020	UART_MDR1	UART_MDR1[2:0]	UART_MDR1[2:0]	UART_MDR1[2:0]	UART_MDR1[2:0]	UART_MDR1[2:0]
0x024	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2
0x028	–	–	–	–	–	–
0x02C	–	–	–	–	–	–
0x030	–	–	–	–	–	–
0x034	–	–	–	–	–	–
0x038	UART_UASR	–	UART_UASR	–	–	–
0x03C	–	–	–	–	–	–
0x040	UART_SCR	UART_SCR	UART_SCR	UART_SCR	UART_SCR	UART_SCR
0x044	UART_SSR	–	UART_SSR	–	UART_SSR	–
0x048	–	–	–	–	–	–
0x050	UART_MVR	–	UART_MVR	–	UART_MVR	–
0x054	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC
0x058	UART_SYSS	–	UART_SYSS	–	UART_SYSS	–

(1) REGISTER_NAME(UART) notation indicates that the register exists for other functions (IrDA or CIR), but fields have different meanings for other functions.
(2) REGISTER_NAME[m:n] notation indicates that only register bits numbered m to n apply to the UART function.

Table 19-22. UART Mode Register Overview[1] [2] (continued)

Address Offset	Registers					
	Configuration Mode A		Configuration Mode B		Operational Mode	
	Read	Write	Read	Write	Read	Write
0x05C	UART_WER	UART_WER	UART_WER	UART_WER	UART_WER	UART_WER
0x060	–	–	–	–	–	–
0x064	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL
0x068	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL
0x06C	UART_IER2	UART_IER2	UART_IER2	UART_IER2	UART_IER2	UART_IER2
0x070	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2
0x074	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL
0x080	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3
0x084	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD

19.3.7.2.2 Registers Available for the IrDA Function

Only the registers listed in Table 19-23 are used for the IrDA function.

Table 19-23. IrDA Mode Register Overview[1] [2]

Address Offset	Registers					
	Configuration Mode A		Configuration Mode B		Operational Mode	
	Read	Write	Read	Write	Read	Write
0x000	UART_DLL	UART_DLL	UART_DLL	UART_DLL	UART_RHR	UART_THR
0x004	UART_DLH	UART_DLH	UART_DLH	UART_DLH	UART_IER(IrDA)	UART_IER(IrDA)
0x008	UART_IIR	UART_FCR	UART_EFR[4]	UART_EFR[4]	UART_IIR(IrDA)	UART_FCR(IrDA)
0x00C	UART_LCR[7]	UART_LCR[7]	UART_LCR[7]	UART_LCR[7]	UART_LCR[7]	UART_LCR[7]
0x010	–	–	UART_XON1_ADDR1	UART_XON1_ADDR1	–	–
0x014	UART_LSR(IrDA)	–	UART_XON2_ADDR2	UART_XON2_ADDR2	UART_LSR(IrDA)	–
0x018	UART_MSR/UART_TCR	UART_TCR	UART_TCR	UART_TCR	UART_MSR/UART_TCR	UART_TCR
0x01C	UART_TLR/UART_SPR	UART_TLR/UART_SPR	UART_TLR	UART_TLR	UART_TLR/UART_SPR	UART_TLR/UART_SPR
0x020	UART_MDR1	UART_MDR1	UART_MDR1	UART_MDR1	UART_MDR1	UART_MDR1
0x024	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2
0x028	UART_SFLSR	UART_TXFLL	UART_SFLSR	UART_TXFLL	UART_SFLSR	UART_TXFLL
0x02C	UART_RESUME	UART_TXFLH	UART_RESUME	UART_TXFLH	UART_RESUME	UART_TXFLH
0x030	UART_SFREGL	UART_RXFLL	UART_SFREGL	UART_RXFLL	UART_SFREGL	UART_RXFLL
0x034	UART_SFREGH	UART_RXFLH	UART_SFREGH	UART_RXFLH	UART_SFREGH	UART_RXFLH

(1) REGISTER_NAME(UART) notation indicates that the register exists for other functions (IrDA or CIR), but fields have different meanings for other functions.

(2) REGISTER_NAME[m:n] notation indicates that only register bits numbered m to n apply to the UART function.

Table 19-23. IrDA Mode Register Overview[1] [2] (continued)

Address Offset	Registers					
	Configuration Mode A		Configuration Mode B		Operational Mode	
	Read	Write	Read	Write	Read	Write
0x038	–	–	–	–	UART_BLR	UART_BLR
0x03C	–	–	–	–	UART_ACREG	UART_ACREG
0x040	UART_SCR	UART_SCR	UART_SCR	UART_SCR	UART_SCR	UART_SCR
0x044	UART_SSR	–	UART_SSR	–	UART_SSR	–
0x048	–	–	–	–	UART_EBLR	UART_EBLR
0x050	UART_MVR	–	UART_MVR	–	UART_MVR	–
0x054	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC
0x058	UART_SYSS	–	UART_SYSS	–	UART_SYSS	–
0x05C	UART_WER[6:4]	UART_WER[6:4]	UART_WER[6:4]	UART_WER[6:4]	UART_WER[6:4]	UART_WER[6:4]
0x060	–	–	–	–	–	–
0x064	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL
0x068	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL
0x06C	UART_IER2	UART_IER2	UART_IER2	UART_IER2	UART_IER2	UART_IER2
0x070	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2
0x074	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL
0x080	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3
0x084	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD

19.3.7.2.3 Registers Available for the CIR Function

Only the registers listed in Table 19-24 are used for the CIR function.

Table 19-24. CIR Mode Register Overview[1] [2]

Address Offset	Registers					
	Configuration Mode A		Configuration Mode B		Operational Mode	
	Read	Write	Read	Write	Read	Write
0x000	UART_DLL	UART_DLL	UART_DLL	UART_DLL	–	UART_THR
0x004	UART_DLH	UART_DLH	UART_DLH	UART_DLH	UART_IER(CIR)	UART_IER(CIR)
0x008	UART_IIR	UART_FCR	UART_EFR	UART_EFR	UART_IIR(CIR)	UART_FCR(CIR)
0x00C	UART_LCR	UART_LCR[7]	UART_LCR[7]	UART_LCR[7]	UART_LCR[7]	UART_LCR[7]
0x010	–	–	–	–	–	–
0x014	UART_LSR(IrDA)	–	–	–	UART_LSR(IrDA)	–
0x018	UART_MSR/UART_TCR	UART_TCR	UART_TCR	UART_TCR	UART_MSR/UART_TCR	UART_TCR
0x01C	UART_TLR/UART_SPR	UART_TLR/UART_SPR	UART_TLR	UART_TLR	UART_TLR/UART_SPR	UART_TLR/UART_SPR

(1) REGISTER_NAME(UART) notation indicates that the register exists for other functions (IrDA or CIR), but fields have different meanings for other functions.
(2) REGISTER_NAME[m:n] notation indicates that only register bits numbered m to n apply to the UART function.

Table 19-24. CIR Mode Register Overview[1] [2] (continued)

Address Offset	Registers					
	Configuration Mode A		Configuration Mode B		Operational Mode	
	Read	Write	Read	Write	Read	Write
0x020	UART_MDR1[3:0]	UART_MDR1[3:0]	UART_MDR1[3:0]	UART_MDR1[3:0]	UART_MDR1[3:0]	UART_MDR1[3:0]
0x024	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2	UART_MDR2
0x028	–	–	–	–	–	–
0x02C	UART_RESUME	–	UART_RESUME	–	UART_RESUME	–
0x030	–	–	–	–	–	–
0x034	–	–	–	–	–	–
0x038	–	–	–	–	–	–
0x03C	–	–	–	–	UART_ACREG	UART_ACREG
0x040	UART_SCR	UART_SCR	UART_SCR	UART_SCR	UART_SCR	UART_SCR
0x044	UART_SSR	–	UART_SSR	–	UART_SSR	–
0x048	–	–	–	–	UART_EBLR	UART_EBLR
0x050	UART_MVR	–	UART_MVR	–	UART_MVR	–
0x054	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC	UART_SYSC
0x058	UART_SYSS	–	UART_SYSS	–	UART_SYSS	–
0x05C	UART_WER[6:4]	UART_WER[6:4]	UART_WER[6:4]	UART_WER[6:4]	UART_WER[6:4]	UART_WER[6:4]
0x060	UART_CFPS	UART_CFPS	UART_CFPS	UART_CFPS	UART_CFPS	UART_CFPS
0x064	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL	UART_RXFIFO_LVL
0x068	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL	UART_TXFIFO_LVL
0x06C	UART_IER2	UART_IER2	UART_IER2	UART_IER2	UART_IER2	UART_IER2
0x070	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2	UART_ISR2
0x074	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL	UART_FREQ_SEL
0x080	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3	UART_MDR3
0x084	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD	UART_TX_DMA_THRESHOLD

19.3.8 Protocol Formatting

The UART/IRDA module can operate in seven different modes:

1. UART 16x mode (≤230.4 Kbits/s), UART16x ≤460Kbits/s if MDR3[1] is set
2. UART 16x mode with autobauding (≥1200 bits/s and ≤115.2 Kbits/s) if MDR3[1] is not set
3. UART 13x mode (≥460.8 Kbits/s) if MDR3[1] is not set
4. IrDA SIR mode (≤115.2 Kbits/s) if MDR3[1] is not set
5. IrDA MIR mode (0.576 and 1.152 Mbits/s) if MDR3[1] is not set
6. IrDA FIR mode (4 Mbits/s) if MDR3[1] is not set
7. CIR mode (programmable modulation rates specific to remote control applications) if MDR3[1] is not set

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

The module 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 LH and the module.

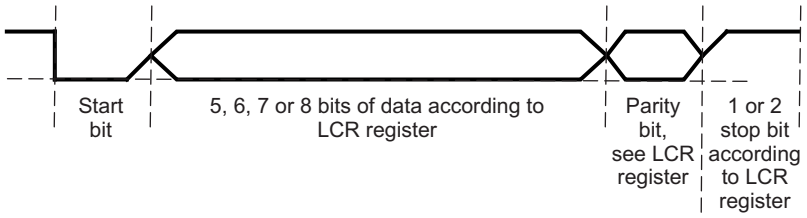
19.3.8.1 UART Mode

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

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

The UART modem module is enhanced with an autobauding functionality which in control mode allows to automatically set the speed, the number of bit per character, the parity selected.

Figure 19-14. UART Data Format

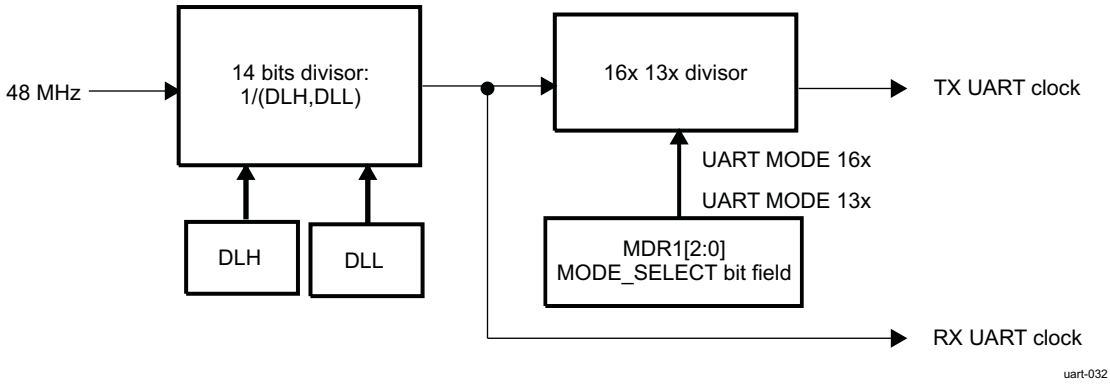


19.3.8.1.1 UART Clock Generation: Baud Rate Generation

The UART function contains a programmable baud generator and a set of fixed dividers that divide the 48-MHz clock input down to the expected baud rate.

Figure 19-15 shows the baud rate generator and associated controls.

Figure 19-15. Baud Rate Generation



CAUTION

Before initializing or modifying clock parameter controls (UARTi.UART_DLH, UARTi.UART_DLL), MODE_SELECT = DISABLE (UARTi.UART_MDR1[2:0]) must be set to 0x7. Failure to observe this rule can result in unpredictable module behavior.

19.3.8.1.2 Choosing the Appropriate Divisor Value

Two divisor values are:

- UART 16x mode: Divisor value = Operating frequency/(16x baud rate)
- UART 13x mode: Divisor value = Operating frequency/(13x baud rate)

Table 19-25 describes the UART baud rate settings.

Table 19-25. UART Baud Rate Settings (48-MHz Clock)

Baud Rate	Baud Multiple	DLH,DLL (Decimal)	DLH,DLL (Hex)	Actual Baud Rate	Error (%)
0.3 kbps	16x	10000	0x27, 0x10	0.3 kbps	0
0.6 kbps	16x	5000	0x13, 0x88	0.6 kbps	0
1.2 kbps	16x	2500	0x09, 0xC4	1.2 kbps	0
2.4 kbps	16x	1250	0x04, 0xE2	2.4 kbps	0
4.8 kbps	16x	625	0x02, 0x71	4.8 kbps	0
9.6 kbps	16x	312	0x01, 0x38	9.6153 kbps	+0.16
14.4 kbps	16x	208	0x00, 0xD0	14.423 kbps	+0.16
19.2 kbps	16x	156	0x00, 0x9C	19.231 kbps	+0.16
28.8 kbps	16x	104	0x00, 0x68	28.846 kbps	+0.16
38.4 kbps	16x	78	0x00, 0x4E	38.462 kbps	+0.16
57.6 kbps	16x	52	0x00, 0x34	57.692 kbps	+0.16
115.2 kbps	16x	26	0x00, 0x1A	115.38 kbps	+0.16
230.4 kbps	16x	13	0x00, 0x0D	230.77 kbps	+0.16
460.8 kbps	13x	8	0x00, 0x08	461.54 kbps	+0.16
921.6 kbps	13x	4	0x00, 0x04	923.08 kbps	+0.16
1.843 Mbps	13x	2	0x00, 0x02	1.846 Mbps	+0.16
3.6884 Mbps	13x	1	0x00, 0x01	3.6923 Mbps	+0.16

19.3.8.1.3 UART Data Formatting

The UART can use hardware flow control to manage transmission and reception. Hardware flow control significantly reduces software overhead and increases system efficiency by automatically controlling serial data flow using the RTS output and CTS input signals.

The UART is enhanced with the autobauding function. In control mode, autobauding lets the speed, the number of bits per character, and the parity selected be set automatically.

19.3.8.1.3.1 Frame Formatting

When autobauding is not used, frame format attributes must be defined in the UARTi.UART_LCR register.

Character length is specified using the UARTi.UART_LCR[1:0] CHAR_LENGTH bit field.

The number of stop-bits is specified using the UARTi.UART_LCR[2] NB_STOP bit.

The parity bit is programmed using the UARTi.UART_LCR[5:3] PARITY_EN, PARITY_TYPE_1, and PARITY_TYPE_2 bit fields (see Table 19-26).

Table 19-26. UART Parity Bit Encoding

PARITY_EN	PARITY_TYPE_1	PARITY_TYPE_2	Parity
0	N/A	N/A	No parity
1	0	0	Odd parity
1	1	0	Even parity
1	0	1	Forced 1
1	1	1	Forced 0

19.3.8.1.3.2 Hardware Flow Control

Hardware flow control is composed of auto-CTS and auto-RTS. Auto-CTS and auto-RTS can be enabled and disabled independently by programming the UARTi.UART_EFR[7:6] AUTO_CTS_EN and AUTO_RTS_EN bit fields, respectively.

With auto-CTS, uarti_cts must be active before the module can transmit data.

Auto-RTS activates the uarti_rts output only when there is enough room in the RX FIFO to receive data. It deactivates the uarti_rts output when the RX FIFO is sufficiently full. The HALT and RESTORE trigger levels in the UARTi.UART_TCR register determine the levels at which uarti_rts is activated and deactivated.

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

- Auto-RTS:

  Auto-RTS data flow control originates in the receiver block. The RX FIFO trigger levels used in auto-RTS are stored in the UARTi.UART_TCR register. uarti_rts is active if the RX FIFO level is below the HALT trigger level in the UARTi.UART_TCR[3:0] RX_FIFO_TRIG_HALT bit field. When the RX FIFO HALT trigger level is reached, uarti_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 RTS until it begins sending the additional byte.

  uarti_rts is automatically reasserted when the RX FIFO reaches the RESUME trigger level programmed by the UARTi.UART_TCR[7:4] RX_FIFO_TRIG_START bit field. This reassertion requests the sending device to resume transmission.

  In this case, uarti_rts is an active-low signal.

- Auto-CTS:

  The transmitter circuitry checks uarti_cts before sending the next data byte. When uarti_cts is active, the transmitter sends the next byte. To stop the transmitter from sending the next byte, uarti_cts must be deasserted before the middle of the last stop-bit currently sent.

  The auto-CTS function reduces interrupts to the host system. When auto-CTS flow control is enabled, the uarti_cts state changes do not have to trigger host interrupts because the device automatically controls its own transmitter. Without auto-CTS, the transmitter sends any data present in the transmit FIFO, and a receiver overrun error can result.

  In this case, uarti_cts is an active-low signal.

19.3.8.1.3.3 Software Flow Control

Software flow control is enabled through the enhanced feature register (UARTi.UART_EFR) and the modem control register (UARTi.UART_MCR). Different combinations of software flow control can be enabled by setting different combinations of the UARTi.UART_EFR[3:0] bit field (see Table 19-27).

Two other enhanced features relate to software flow control:

- XON any function (UARTi.UART_MCR[5]): Operation resumes after receiving any character after the XOFF character is recognized. If special character detect is enabled and special character is received after XOFF1, it does not resume transmission. The special character is stored in the RX FIFO.

  NOTE: The XON-any character is written into the RX FIFO even if it is a software flow character.

- Special character (UARTi.UART_EFR[5]): Incoming data is compared to XOFF2. When the special character is detected, the XOFF interrupt (UARTi.UART_IIR[4]) is set, but it does not halt transmission. The XOFF interrupt is cleared by a read of UARTi.UART_IIR. The special character is transferred to the RX FIFO. Special character does not work with XON2, XOFF2, or sequential XOFFs.

Table 19-27. UART_EFR[3:0] 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(1)
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(1)

(1) 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 is defined in the UARTi.UART_XON1_ADDR1[7:0] XON_WORD1 bit field. XON2 is defined in the UARTi.UART_XON2_ADDR2[7:0] XON_WORD2 bit field.
XOFF1 is defined in the UARTi.UART_XOFF1[7:0] XOFF_WORD1 bit field. XOFF2 is defined in the UARTi.UART_XOFF2[7:0] XOFF_WORD2 bit field.

19.3.8.1.3.3.1 Receive (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 stops after transmission of the current character completes. Detection of XOFF also sets the UARTi.UART_IIR[4] bit (if enabled by UARTi.UART_IER[5]) and causes the interrupt line 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, the UARTi.UART_IIR[4] bit is cleared and the XOFF interrupt disappears.

NOTE: When 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 UARTi.UART_EFR[1:0] = 0x2, if XON1 and XOFF1 characters are received, they are not written to the RX FIFO.

When pairs of software flow characters are programmed to be received sequentially (UARTi.UART_EFR[1:0] = 0x3), 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.

19.3.8.1.3.3.2 Transmit (TX)

Two XOFF1 characters are transmitted when the RX FIFO passes the trigger level programmed by UARTi.UART_TCR[3:0]. As soon as the RX FIFO reaches the trigger level programmed by UARTi.UART_TCR[7:4], two XON1 characters are sent, so the data transfer recovers.

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

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

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

19.3.8.1.3.4 Autobauding Modes

In autobauding mode, the UART can extract transfer characteristics (speed, length, and parity) from an "at" (AT) command (ASCII code). These characteristics are used to receive data after an AT and to send data.

The following AT commands are valid:

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

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

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

When a valid AT is received, AT and all subsequent data, including the final <CR> (0x0D), are saved to the RX FIFO. The autobaud state-machine waits for the next valid AT command. If an a/ (A/) is received, the a/ (A/) is saved in the RX FIFO and the state-machine waits for the next valid AT command.

On the first successful detection of the baud rate, the UART activates an interrupt to signify that the AT (upper or lower case) sequence is detected. The UARTi.UART_UASR register reflects the correct settings for the baud rate detected. Interrupt activity can continue in this fashion when a subsequent character is received. Therefore, it is recommended that the software enable the RHR interrupt when using the autobaud mode.

The following settings are detected in autobaud mode with a module clock of 48 MHz:

- 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
  - 1.2K baud
- Length: 7 or 8 bits
- Parity: Odd, even, or space

NOTE: The combination of 7-bit character plus space parity is not supported.

Autobauding mode is selected when the UARTi.UART_MDR1[2:0] MODE_SELECT bit field is set to 0x2. In UART autobauding mode, UARTi.UART_DLL, UARTi.UART_DLH, and UARTi.UART_LCR[5:0] bit field settings are not used; instead, UART_UASR is updated with the configuration detected by the autobauding logic.

UART_UASR Autobauding Status Register Use

This register is used to set up transmission according to the characteristics of the previous reception instead of the UARTi.UART_LCR, UARTi.UART_DLL, and UARTi.UART_DLH registers when the UART is in autobauding mode.

To reset the autobauding hardware (to start a new AT detection) or to set the UART in standard mode (no autobaud), the UARTi.UART_MDR1[2:0] MODE_ SELECT bit field must be set to reset state (0x7) and then to the UART in autobauding mode (0x2) or to the UART in standard mode (0x0).

Use limitation:

- Only 7- and 8-bit characters (5- and 6-bit not supported)
- 7-bit character with space parity not supported
- Baud rate between 1200 and 115,200 bps (10 possibilities)

19.3.8.1.3.5 Error Detection

When the UARTi.UART_LSR register is read, the UARTi.UART_LSR[4:2] bit field reflects the error bits (BI: break condition, FE: framing error, PE: parity error) of the character at the top of the RX FIFO (the next character to be read). Therefore, reading the UARTi.UART_LSR register and then reading the UARTi.UART_RHR register identifies errors in a character.

Reading the UARTi.UART_RHR register updates the BI, FE, and PE bits (see Table 19-11 for the UART mode interrupts).

The UARTi.UART_LSR[7] RX_FIFO_STS bit is set when there is an error in the RX FIFO and is cleared only when no errors remain in the RX FIFO.

NOTE: Reading the UARTi.UART_LSR register does not cause an increment of the RX FIFO read pointer. The RX FIFO read pointer is incremented by reading the UARTi.UART_RHR register.

Reading the UARTi.UART_LSR register clears the OE bit if it is set (see Table 19-11 for the UART mode interrupts).

19.3.8.1.3.6 Overrun During Receive

Overrun during receive occurs 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 MPU with the UARTi.UART_IIR[5:1] IT_TYPE bit field set to 0x3 (receiver line status error) 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 MPU must:

- Reset the RX FIFO.
- Read the UARTi.UART_RESUME register, which clears the internal flag.

19.3.8.1.3.7 Time-Out and Break Conditions

19.3.8.1.3.7.1 Time-Out Counter

An RX idle condition is detected when the receiver line (uarti_rx) is high for a time that equals 4x the programmed word length + 12 bits. uarti_rx is sampled midway through each bit.

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

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

19.3.8.1.3.7.2 Break Condition

When a break condition occurs, uarti_tx is pulled low. A break condition is activated by setting the UARTi.UART_LCR[6] BREAK_EN bit. The break condition is not aligned on word stream (a break condition can occur in the middle of a character). The only way to send a break condition on a full character is:

1. Reset the TX FIFO (if enabled).
2. Wait for the transmit shift register to empty (the UARTi.UART_LSR[6] TX_SR_E bit is set to 1).

3. Take a guard time according to stop-bit definition.
4. Set the BREAK_EN bit to 1.

The break condition is asserted while the BREAK_EN bit is set to 1.

The time-out counter and break condition apply only to UART modem operation and not to IrDA/CIR mode operation.

19.3.8.2 IrDA Mode

19.3.8.2.1 Slow Infrared (SIR) Mode

In SIR mode, data transfers take place between the LH and peripheral devices at speeds of up to 115200 bauds. A SIR transmit frame starts with start flags (either a single 0xC0, multiple 0xC0, or a single 0xC0 preceded by a number of 0xFF flags), followed by frame data, CRC-16, and ends with a stop flag (0xC1). The bit format for a single word uses a single start bit, eight data bits, and one stop bit. The format is unaffected by the use and settings of the LCR register.

Note that BLR[6] is used to select whether to use 0xC0 or 0xFF start patterns 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 0xC0, 0xC1, or 0x7D. If one of these is about to be transmitted, then the SIR state machine sends an escape character (0x7D) first, then inverts the fifth bit of the real data to be sent and sends this data immediately after the 0x7D character.

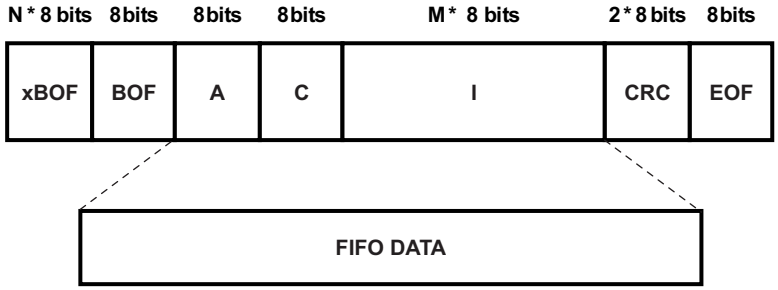
The SIR receive state machine recovers the receive clock, removes the start flags, removes any transparency from the incoming data, and determines frame boundary with reception of the stop flag. It also checks for errors, such as frame abort (0x7D character followed immediately by a 0xC1 stop flag, without transparency), CRC error, and frame-length error. At the end of a frame reception, the LH reads the line status register (LSR) to find out possible errors of the received frame.

Data can be transferred both ways by the module, but when the device is transmitting the IR RX circuitry is automatically disabled by hardware. See bit 5 in the auxiliary control register (ACREG) for a description of the logical operation. **Note:** This applies to all three modes SIR, MIR, and FIR.

The infrared output in SIR mode can either be 1.6µs or 3/16 encoding, selected by the PULSETYPE bit of the Auxiliary Control Register (ACREG[7]). In 1.6µs encoding, the infrared pulse width is 1.6µs and in 3/16 encoding the infrared pulse width is 3/16 of a bit duration (1/baud-rate). The receiver supports both 3/16 and 1.6µs pulse duration by default. 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.

19.3.8.2.1.1 Frame Format

Figure 19-16. IrDA SIR 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.

19.3.8.2.1.2 Asynchronous Transparency

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

In transmission

1. Inserts a control escape (CE) byte preceding the byte.
2. Complements bit 5 of the byte (i.e., exclusive OR's 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 field:

1. Compare the byte with CE byte, and if not equal send it to the CRC detector and store it in the RX FIFO.
2. If equal to CE, discard the CE byte.
3. Complements the bit 5 of the byte following the CE.
4. 4. Send the complemented byte to the CRC detector and store it in the RX FIFO.

19.3.8.2.1.3 Abort Sequence

The transmitter may decide to prematurely close a frame. The transmitter aborts by sending the following 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 the ABORTEN bit of the Auxiliary Control Register (ACREG[1]). When this bit is set to 1, 0x7D and 0xC1 are transmitted and the frame is not terminated with CRC or stop flags. The receiver treats a frame as an aborted frame when a 0x7D character, followed immediately by a 0xC1 character, has been received without transparency.

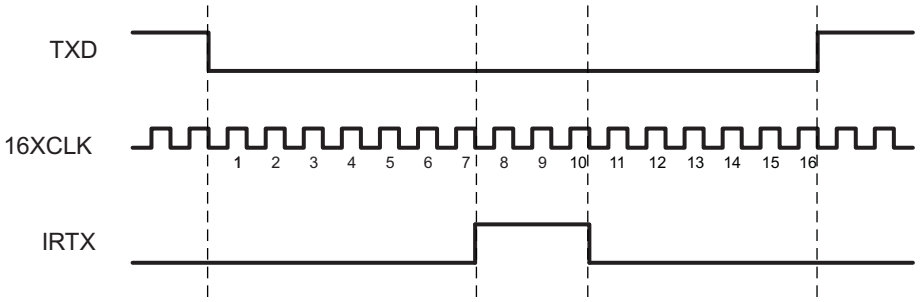
19.3.8.2.1.4 Pulse Shaping

In SIR mode, both the 3/16th and the 1.6μs pulse duration methods are supported in receive and transmit. ACREG[7] selects the pulse width method in transmit mode.

19.3.8.2.1.5 Encoder

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

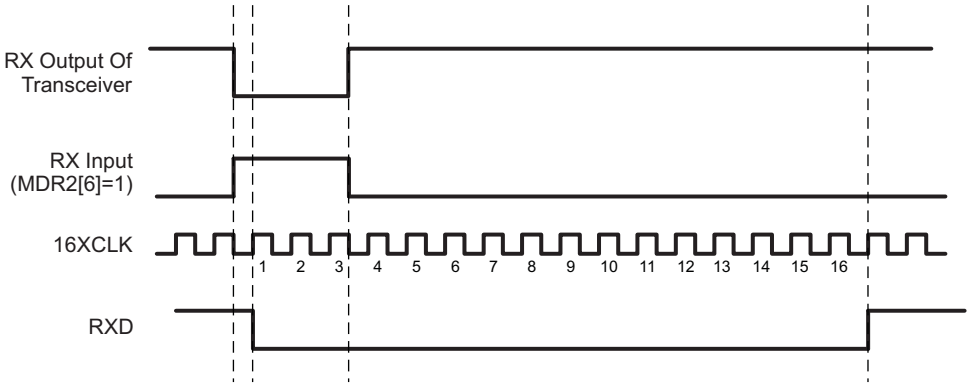
Figure 19-17. IrDA Encoding Mechanism



19.3.8.2.1.6 Decoder

After reset, RXD is high and the 4-bit counter is cleared. When a rising edge is detected on RX, RXD falls on the next rising edge of 16XCLK with sufficient setup time. RXD stays 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 RX, RXD remains high.

Figure 19-18. IrDA Decoding Mechanism



The operation of the RX input can be disabled with DISIRRX bit of the Auxiliary Control Register (ACREG[5]). Furthermore, the MDR2[6] can be used to invert the signal from the transceiver (RX ouput) pin to the IRRX logic inside the UART.

19.3.8.2.1.7 IR Address Checking

In all IR modes, if address checking has been enabled, only frames intended for the device are written to the RX FIFO. This is to avoid receiving frames not meant for this device in a multi-point infrared environment. It is possible to program two frame addresses that the UART IrDA receives with 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 private and public addresses. If address checking is disabled, then all received frames are written into the reception FIFO.

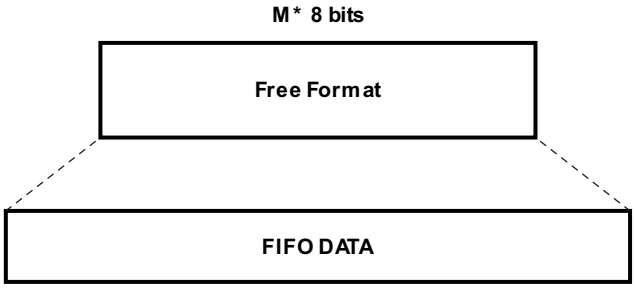
19.3.8.2.1.8 SIR Free Format Mode

To allow complete software flexibility in the transmission and reception of Infrared data packets, the SIR free format mode is a sub-function of the existing SIR mode such that all frames going to and from the FIFO buffers are untouched with respect to appending and removing control characters and CRC values. In the transmission phase, the mode uses the IRTX pin, as in SIR mode.

This mode corresponds to a UART mode with a pulse modulation of 3/16 of baud-rate pulse width.

For example, a normal SIR packet has BOF control and CRC error checking data appended (transmitting) or removed (receiving) from the data going to and from the FIFOs. In SIR free format mode, only the data termed the FIFO DATA area, illustrated in Figure 19-19, would be transmitted and received.

Figure 19-19. SIR Free Format Mode



In this mode, the entire FIFO data packet is to be constructed (encoded and decoded) by the LH software.

The SIR free format mode is selected by setting the module in UART mode (MDR1[2:0] = 000) and the MDR2[3] register bit to one to allow the pulse shaping. As the bit format is to remain the same, some UART mode configuration registers need to be set at specific value:

- LCR[1:0] = “11” (8 data bits)
- LCR[2] = 0 (1 stop bit)
- LCR[3] = 0 (no parity)
- ACREG[7] = 0 (3/16 of baud-rate pulse width)

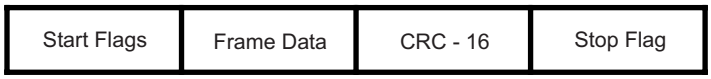
The features defined through MDR2[6] and ACREG[5] are also supported.

Note: - All other configuration registers need to be at the reset value. The UART mode interrupts are used for the SIR free format mode, but many of them are not relevant (e.g., XOFF, RTS, CTS, Modem status register).

19.3.8.2.2 Medium Infrared (MIR) Mode

In MIR mode, data transfers take place between LH and peripheral devices at 0.576 or 1.152 Mbits/s speed. A MIR transmit frame starts with start flags (at least two), followed by a frame data, CRC-16, and ends with a stop flag.

Figure 19-20. MIR Transmit Frame Format



On transmit, the MIR state machine attaches start flags, CRC-16, and stop flags. It also looks for five consecutive values of 1 in the frame data and automatically inserts a zero after five consecutive values of one (this is called bit stuffing).

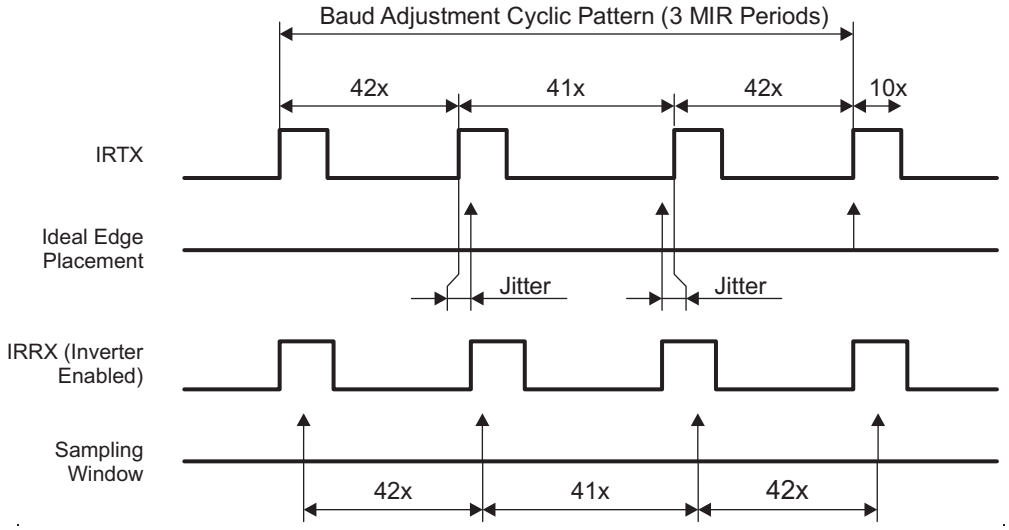
On receive, the MIR receive state machine recovers the receive clock, removes the start flags, de-stuffs the incoming data, and determines frame boundary with reception of the stop flag. It also checks for errors, such as frame abort, CRC error, or frame-length error. At the end of a frame reception, the LH reads the line status register (LSR) to find possible errors of received frame.

Data can be transferred both ways by the module but when the device is transmitting, the IR RX circuitry is automatically disabled by hardware. See bit 5 in the auxiliary control register (ACREG) for a description of the logical operation. **Note:** This applies to all three modes SIR, MIR and FIR.

19.3.8.2.2.1 MIR Encoder/Decoder

In order to meet MIR baud-rate tolerance of +/-0.1% with a 48-MHz clock input, a 42-41-42 encoding/decoding adjustment is performed. The reference start point is the first start flag and the 42-41-42 cyclic pattern is repeated until the stop flag is sent or detected. The jitter created this way is within MIR tolerances. The pulse width is not exactly 1/4 but within tolerances defined by the IrDA specifications.

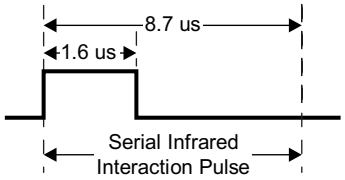
Figure 19-21. MIR BAUD Rate Adjustment Mechanism



19.3.8.2.2.2 SIP Generation

In MIR and FIR operation modes, the transmitter needs to send a serial infrared interaction pulse (SIP) at least once every 500 ms. The purpose of the SIP is to let slow devices (operating in SIR mode) know that the medium is currently occupied. The SIP pulse is shown in Figure 19-22

Figure 19-22. SIP Pulse



When the SIPMODE bit of Mode Definition Register 1 (MDR1[6]) equals 1, the TX state machine will always send one SIP at the end of a transmission frame. But when MDR1[6] = 0, the transmission of the SIP depends on the SENDSIP bit of the Auxiliary Control Register (ACREG[3]). The system (LH) can set ACREG[3] at least once every 500ms. The advantage of this approach over the default approach is that the TX state machine does not need to send the SIP at the end of each frame which may reduce the overhead required

19.3.8.2.3 Fast Infrared (FIR) Mode

In FIR mode, data transfers take place between LH and peripheral devices at 4 Mbits/s speed. A FIR transmit frame starts with a preamble, followed by a start flag, frame data, CRC-32, and ends with a stop flag.

Figure 19-23. FIR Transmit Frame Format

Preamble (16x)	Start Flag	Frame Data	CRC - 32	Stop Flag

On transmit, the FIR transmit state machine attaches the preamble, start flag, CRC-32, and stop flag. It also encodes the transmit data into 4PPM format. It also generates the serial infrared interaction pulse (SIP).

On receive, the FIR receive state machine recovers the receive clock, removes the start flag, decodes the 4PPM incoming data, and determines frame boundary with a reception of the stop flag. It also checks for errors such as an illegal symbol, a CRC error, and a frame-length error. At the end of a frame reception, the LH reads the line status register (LSR) to find out possible errors of the received frame.

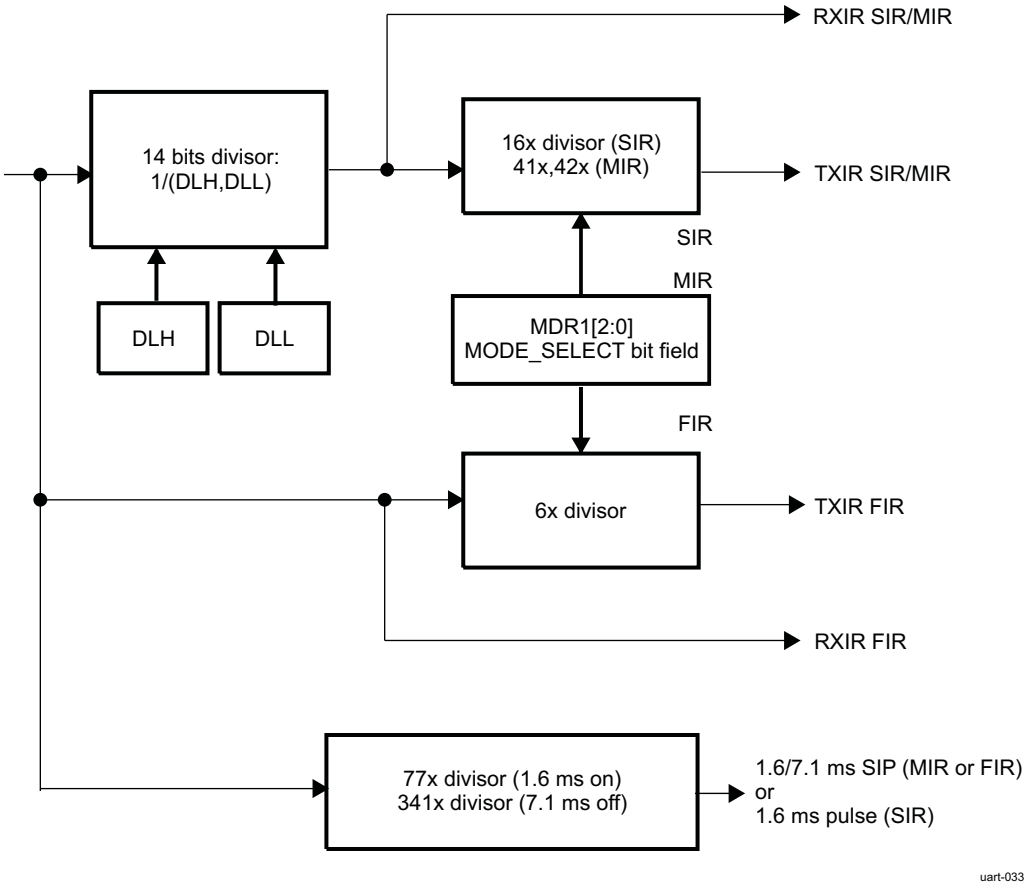
Data can be transferred both ways by the module but when the device is transmitting, the IR RX circuitry is automatically disabled by hardware. See bit 5 in the auxiliary control register (ACREG) for a description of the logical operation. **Note:** This applies to all three modes of SIR, MIR, and FIR.

19.3.8.2.4 IrDA Clock Generation: Baud Generator

The IrDA function contains a programmable baud generator and a set of fixed dividers that divide the 48-MHz clock input down to the expected baud rate.

Figure 19-24 shows the baud rate generator and associated controls.

Figure 19-24. Baud Rate Generator



CAUTION

Before initializing or modifying clock parameter controls (UARTi.UART_DLH, UARTi.UART_DLL), MODE_SELECT=DISABLE (UARTi.UART_MDR1[2:0]) must be set to 0x7). Failure to observe this rule can result in unpredictable module behavior.

19.3.8.2.5 Choosing the Appropriate Divisor Value

Three divisor values are:

- SIR mode: Divisor value = Operating frequency/(16x baud rate)
- MIR mode: Divisor value = Operating frequency/(41x/42x baud rate)
- FIR mode: Divisor value = None

Table 19-28 lists the IrDA baud rate settings.

Table 19-28. IrDA Baud Rate Settings

Baud Rate	IR Mode	Baud Multiple	Encoding	DLH, DLL (Decimal)	Actual Baud Rate	Error (%)	Source Jitter (%)	Pulse Duration
2.4 kbps	SIR	16x	3/16	1250	2.4 kbps	0	0	78.1 µs
9.6 kbps	SIR	16x	3/16	312	9.6153 kbps	+0.16	0	19.5 µs
19.2 kbps	SIR	16x	3/16	156	19.231 kbps	+0.16	0	9.75 µs
38.4 kbps	SIR	16x	3/16	78	38.462 kbps	+0.16	0	4.87 µs
57.6 kbps	SIR	16x	3/16	52	57.692 kbps	+0.16	0	3.25 µs
115.2 kbps	SIR	16x	3/16	26	115.38 kbps	+0.16	0	1.62 µs
0.576 Mbps	MIR	41x/42x	1/4	2	0.5756 Mbps(1)	0	+1.63/–0.80	416 ns
1.152 Mbps	MIR	41x/42x	1/4	1	1.1511 Mbps(1)	0	+1.63/–0.80	208 ns
4 Mbps	FIR	6x	4 PPM	–	4 Mbps	0	0	125 ns

(1) Average value

NOTE: Baud rate error and source jitter table values do not include 48-MHz reference clock error and jitter.

19.3.8.2.6 IrDA Data Formatting

The methods described in this section apply to all IrDA modes (SIR, MIR, and FIR).

19.3.8.2.6.1 IR RX Polarity Control

The UARTi.UART_MDR2[6] IRRXINVERT bit provides the flexibility to invert the uarti_rx_irrx pin in the UART to ensure that the protocol at the output of the transceiver has the same polarity at module level. By default, the uarti_rx_irrx pin is inverted because most transceivers invert the IR receive pin.

19.3.8.2.6.2 IrDA Reception Control

The module can transmit and receive data, but when the device is transmitting, the IR RX circuitry is automatically disabled by hardware.

Operation of the uarti_rx_irrx input can be disabled by the UARTi.UART_ACREG[5] DIS_IR_RX bit.

19.3.8.2.6.3 IR Address Checking

In all IR modes, when address checking is enabled, only frames intended for the device are written to the RX FIFO. This restriction avoids receiving frames not meant for this device in a multipoint infrared environment. It is possible to program two frame addresses that the UART IrDA receives, with the UARTi.UART_XON1_ADDR1[7:0] XON_WORD1 and UARTi.UART_XON2_ADDR2[7:0] XON_WORD2 bit fields.

Setting the UART_EFR[0] bit to 1 selects address1 checking. Setting the UART_EFR[1] bit to 1 selects address2 checking. Setting the UART_EFR[1:0] bit field to 0 disables all address checking operations. If both bits are set, the incoming frame is checked for private and public addresses.

If address checking is disabled, all received frames write to the RX FIFO.

19.3.8.2.6.4 Frame Closing

A transmission frame can be terminated in two ways:

- Frame-length method: Set the UARTi.UART_MDR1[7] FRAME_END_MODE bit to 0. The MPU writes the value of the frame length to the UARTi.UART_TXFLH and UARTi.UART_TXFLL registers. The device automatically attaches end flags to the frame when the number of bytes transmitted equals the value of the frame length.
- Set-EOT bit method: Set the FRAME_END_MODE bit to 1. The MPU writes 1 to the UARTi.UART_ACREG[0] EOT bit just before it writes the last byte to the TX FIFO. When the MPU writes the last byte to the TX FIFO, the device internally sets the tag bit for that 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 correctly terminate the frame.

19.3.8.2.6.5 Store and Controlled Transmission

In store and controlled transmission (SCT) mode, the MPU starts writing data to the TX FIFO. Then, after writing a part of a frame (for a bigger frame) or an entire frame (a small frame; that is, a supervisory frame), the MPU writes 1 to the UARTi.UART_ACREG[2] SCTX_EN bit (deferred TX start) to start transmission.

SCT mode is enabled by setting the UARTi.UART_MDR1[5] SCT bit to 1. This transmission method differs from normal mode, in which data transmission starts immediately after data is written to the TX FIFO. SCT mode is useful for sending short frames without TX underrun.

19.3.8.2.6.6 Error Detection

When the UARTi.UART_LSR register is read, the UARTi.UART_LSR[4:2] bit field reflects the error bits [FL, CRC, ABORT] of the frame at the top of the STATUS FIFO (the next frame status to be read).

The error is triggered by an interrupt (for IrDA mode interrupts, see Table 19-12). The STATUS FIFO must be read until empty (a maximum of eight reads is required).

19.3.8.2.6.7 Underrun During Transmission

Underrun during transmission occurs when the TX FIFO is 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 additional transmissions. Before the next frame can be transmitted, the MPU must:

- Reset the TX FIFO.
- Read the UARTi.UART_RESUME register, which clears the internal flag.

This function can be disabled by the UARTi.UART_ACREG[4] DIS_TX_UNDERRUN bit, compensated by the extension of the stop-bit in transmission if the TX FIFO is empty.

19.3.8.2.6.8 Overrun During Receive

Overrun during receive for the IrDA mode has the same function as that for the UART mode (see Section 19.3.8.1.3.6, *Overrun During Receive*).

19.3.8.2.6.9 Status FIFO

In IrDA modes, a status FIFO records 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 to the status FIFO.

Reading the UARTi.UART_SFREGH[3:0] MSB and UARTi.UART_SFREGL[3:0] (LSB) bit fields obtains the frame length. The frame error status is read in the UARTi.UART_SFLSR register. Reading the UARTi.UART_SFLSR register increments the status FIFO read pointer. Because the status FIFO is eight entries deep, it can hold the status of eight frames.

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

This status FIFO can be used effectively in DMA mode because the MPU must be interrupted only when the programmed status FIFO trigger level is reached, not each time a frame is received.

19.3.8.2.7 SIR Mode Data Formatting

This section provides specific instructions for SIR mode programming.

19.3.8.2.7.1 Abort Sequence

The transmitter can prematurely close a frame (abort) by sending the sequence 0x7DC1. The abort pattern closes the frame without a CRC field or an ending flag.

A transmission frame can be aborted by setting the UARTi.UART_ACREG[1] ABORT_EN bit to 1. When this bit is set to 1, 0x7D and 0xC1 are transmitted and the frame is not terminated with CRC or stop flags.

When a 0x7D character followed immediately by a 0xC1 character is received without transparency, the receiver treats a frame as an aborted frame.

CAUTION

When the TX FIFO is not empty and the UARTi.UART_MDR1[5] SCT bit is set to 1, the UART IrDA starts a new transfer with data of a previous frame when the aborted frame is sent. Therefore, the TX FIFO must be reset before sending an aborted frame.

19.3.8.2.7.2 Pulse Shaping

SIR mode supports the 3/16 or the 1.6-µs pulse duration methods in receive and transmit. The UARTi.UART_ACREG[7] PULSE_TYPE bit selects the pulse width method in the transmit mode.

19.3.8.2.7.3 SIR Free Format Programming

The SIR FF mode is selected by setting the module in the UART mode (UARTi.UART_MDR1[2:0] MODE_SELECT = 0x0) and the UARTi.UART_MDR2[3] UART_PULSE bit to 1 to allow pulse shaping.

Because the bit format stays the same, some UART mode configuration registers must be set at specific values:

- UARTi.UART_LCR[1:0] CHAR_LENGTH bit field = 0x3 (8 data bits)
- UARTi.UART_LCR[2] NB_STOP bit = 0x0 (1 stop-bit)
- UARTi.UART_LCR[3] PARITY_EN bit = 0x0 (no parity)

The UART mode interrupts are used for the SIR FF mode, but many are not relevant (XOFF, RTS, CTS, modem status register, etc.).

19.3.8.2.8 MIR and FIR Mode Data Formatting

This section describes common instructions for FIR and MIR mode programming.

At the end of a frame reception, the MPU reads the line status register (UARTi.UART_LSR) to detect errors in the received frame.

When the UARTi.UART_MDR1[6] SIP_MODE bit is set to 1, the TX state-machine always sends one SIP at the end of a transmission frame. However, when the SIP_MODE bit is set to 0, SIP transmission depends on the UARTi.UART_ACREG[3] SEND_SIP bit.

The MPU can set the SEND_SIP bit at least once every 500 ms. The advantage of this approach over the default approach is that the TX state-machine does not have to send the SIP at the end of each frame, thus reducing the overhead required.

19.3.8.3 CIR Mode

In consumer infrared mode, the infrared operation is designed to function as a programmable (universal) remote control. By setting the MDR1 register, the UART can be set to CIR mode in the same way as the other IrDA modes are set using the MDR1 register.

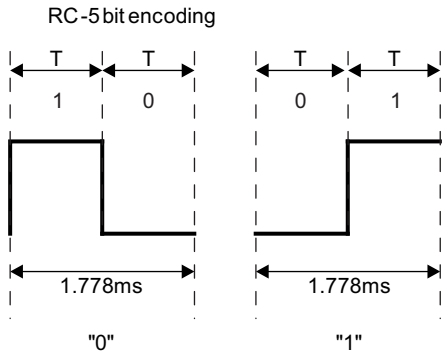
The CIR mode uses a variable pulse width modulation technique (based on multiples of a programmable T period) to encompass the various formats of infrared encoding for remote control applications. The CIR logic is to transmit and receive data packets according to the user definable frame structure and packet content.

19.3.8.3.1 Consumer IR Encoding

There are two distinct methods of encoding for remote control applications. The first uses time extended bit forms i.e. a variable pulse distance (or duration) whereby the difference between a logic one and logic zero is the length of the pulse width; and the second is the use of a bi-phase where the encoding of the logic zero and one is in the change of signal level from 1→0 or 0→1 respectively. Japanese manufacturers tend to favor the use of pulse duration encoding whereas European manufacturers favor the use of bi-phase encoding.

The CIR mode is designed to use a completely flexible free format encoding where a digit ‘1’ from the TX/RX FIFO is to be transmitted/received as a modulated pulse with duration T. Equally, a ‘0’ is to be transmitted/received as a blank duration T. The protocol of the data is to be constructed and deciphered by the host CPU. For example, the RC-5 protocol using Manchester encoding can be emulated as using a “01” pair for one and “10” pair for a zero.

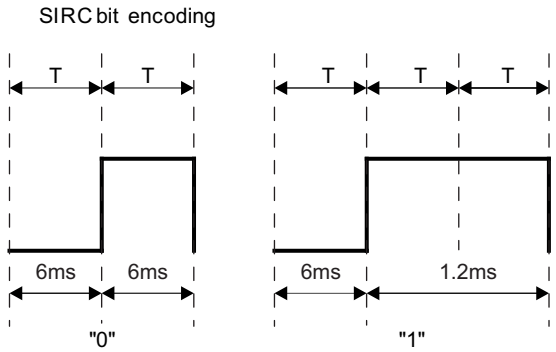
Figure 19-25. RC-5 Bit Encoding



Since the CIR mode logic does not impose a fixed format for infrared packets of data, the CPU software is at liberty to define the format through the use of simple data structures that will then be modulated into an industry standard, such as RC5 or SIRC. To send a sequence of “0101” in RC5, the host software must write an eight bit binary character of “10011001” to the data TX FIFO of the UART.

For SIRC, the modulation length (i.e. multiples of T) is the method to distinguish between a “1” or a “0”. The following SIRC digits show the difference in encoding between this and RC5 for example. Note: the pulse width is extended for “1” digits.

Figure 19-26. SIRC Bit Encoding



To construct comprehensive packets that constitute remote control commands, the host software must combine a number of eight bit data characters in a sequence that follows one of the universally accepted formats. For illustrative purposes, a standard RC5 frame is described below (the SIRC format follows this). Each of the above fields in RC-5 can be considered as two T pulses (digital bits) from the TX and RX FIFOs.

The standard RC5 format as seen by the UART_IrDA in CIR mode.

Figure 19-27. RC-5 Standard Packet Format

S1	S2	T	A4	A3	A2	A1	A0	C5	C4	C3	C2	C1	C0

Where:

S1, S2: Start bits (always 1)

T: Toggle bit

A4–A0: Address (or system) bits

C5–C0: Command bits

The toggle bit T changes each time a new command is transmitted to allow detection of pressing the same key twice (or effectively receiving the same data from the host consecutively). Since a code is being sent as long as the CPU transmits characters to the UART for transmission, a brief delay in the transmission of the same command would be detected by the use of the toggle bit. The address bits define the machine or device that the Infrared transmission is intended for and the command defines the operation.

To accommodate an extended RC5 format, the S2 bit is replaced by a further command bit (C6) that allows the command range to increase to 7-bits. This format is known as the extended RC-5 format.

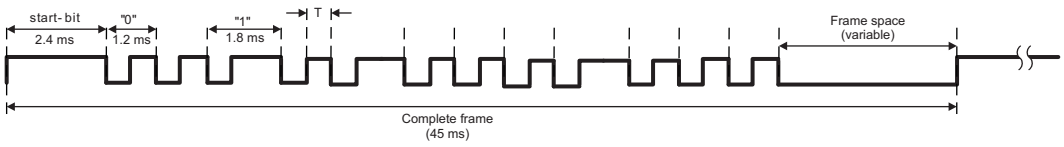
The SIRC encoding uses the duration of modulation for mark and space; hence the duration of data bits inside the standard frame length will vary depending upon the logic 1 content. The packet format and bit encoding is illustrated below. There is one start bit of two milliseconds and control codes followed by data that constitute the whole frame.

Figure 19-28. SIRC Packet Format

S	C0	C1	C2	C3	C4	C5	C6	D0	D1	D2	D3	D4

It should be noted that the encoding must take a standard duration but the contents of the data may vary. This implies that the control software for emitting and receiving data packets must exercise a scheme of inter-packet delay, where the emission of successive packets can only be done after a real time delay has expired.

Figure 19-29. SIRC Bit Transmission Example

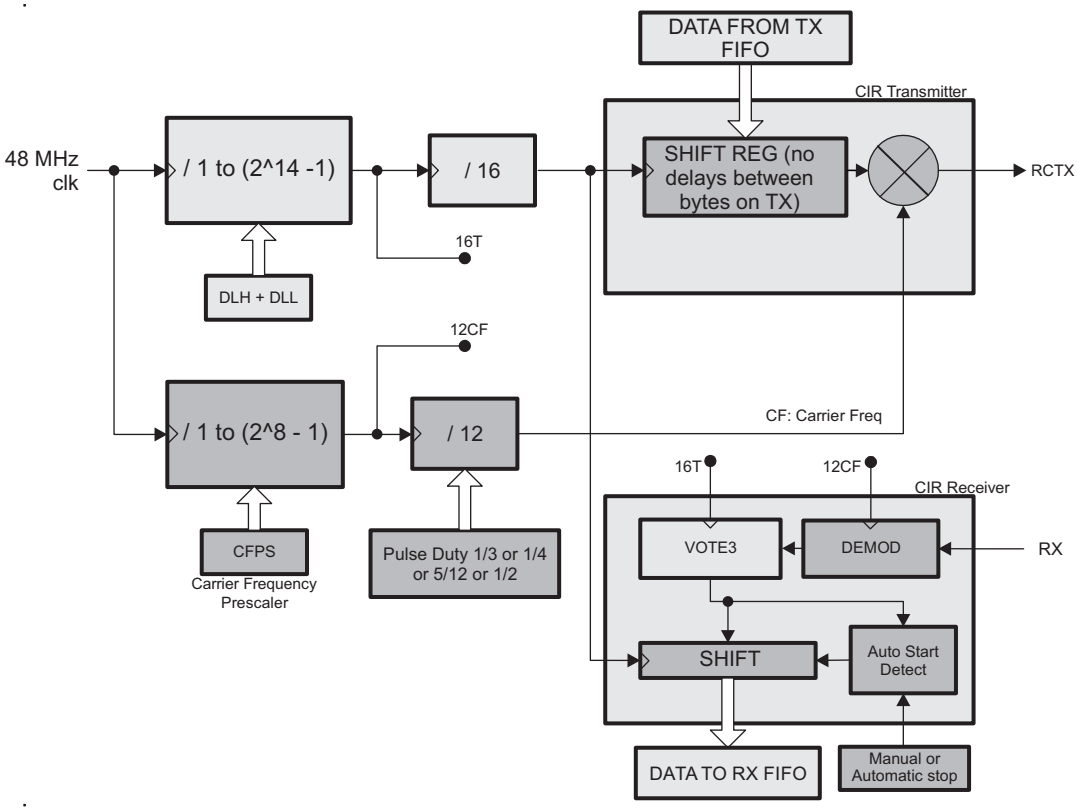


It is beyond the scope of this document to describe all encoding methods and techniques, the previous information was provided to illustrate the consideration required to employ different encoding methods for different industry standard protocols. The user should refer to industry standard documentation for specific methods of encoding and protocol usage.

19.3.8.3.2 CIR Mode Operation

Depending on the encoding method (variable pulse distance / bi-phase), the LH should develop a data structure that combines the 1 and 0 with a T period in order to encode the complete frame to transmit. This can then be transmitted to the infrared output with a method of modulation shown in the following diagram.

Figure 19-30. CIR Mode Block Components



In transmission, the LH software must exercise an element of real time control for transmitting data packets; they must each be emitted at a constant delay from the start bits of each of the individual packets which means when sending a series of packets, the packet to packet delay must respect a specific delay. To control this delay 2 methods can be used:

- By filling the TX FIFO with a number of zero bit which is transmitted with a T period.
- By using an external system timer which controls the delay either between each start of frame or between the end of a frame and the start of the next one. This can be performed:
  - By controlling the start of the frame through the configuration register MDR1[5] and ACREG[2] depending on the timer status (in case of control the delay between each start of frame).
  - By using the TX_STATUS interrupt IIR[5] to pre-load the next frame in the TX FIFO and to control the start of the timer (in case of control the delay between end of frame and start of next frame).

In reception, there are two ways to stop it :

- The LH can disable the reception by setting the ACREG[5] to 1 when it considers that the reception is finished because a large number of 0 has been received. To receive a new frame, the ACREG[5] must be set to 0.
- A specific mechanism, depending on the value set in the BOF length register (EBLR), allows for automatically stopping the reception. If the value set in the EBLR register is different than 0, this features is enabled and count a number of bit received at 0. When the counter achieves the value defined in the EBLR register, the reception is automatically stopped and RX_STOP_IT (IIR[2]) is set. When a 1 is detected on the RCRX pin, the reception is automatically enabled.

Note: There's a limitation when receiving data in UART CIR mode. The IrDA transceivers on the market have a common characteristic that shrinks the hold time of the received modulation pulse. The UART filtering schema on receiving is based on the same encoding mechanism used in transmission.

For the following scenario:

- Shift register period: 0.9 us
- Modulation frequency: 36 Khz
- Duty cycle: 1/4 of a modulation frequency period

The data sent in these conditions would look like 7us pulses within 28us period. The UART expects to receive similar incoming data on receive, but available transceiver timing characteristics typically send 2us modulated pulses. Those will be filtered out and RX FIFO won't receive any data.

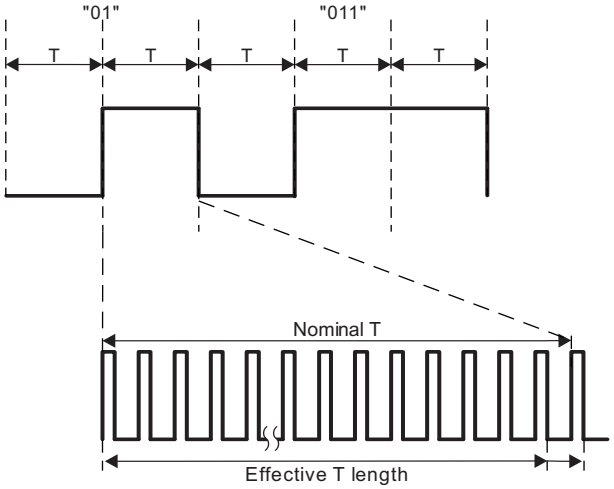
This does not affect UART CIR mode in transmission.

Note: The CIR RX demodulation can be bypassed by setting the MDR3[0] register bit.

19.3.8.3.3 Carrier Modulation

Looking closer at the actual modulation pulses of the infrared data stream, it should be noted that each modulated pulse that constitutes a digit is in fact a train of on/off pulses.

Figure 19-31. CIR Pulse Modulation



A minimum of 4 modulation pulses per bit is required by the module.

Based on the requested modulation frequency, the CFPS register must be set with the correct dividing value to provide the more accurate pulse frequency:

Dividing value = (FCLK/12)/MODfreq

Where FCLK = System clock frequency (48 MHz)

12 = real value of BAUD multiple

MODfreq = Effective frequency of the modulation (MHz)

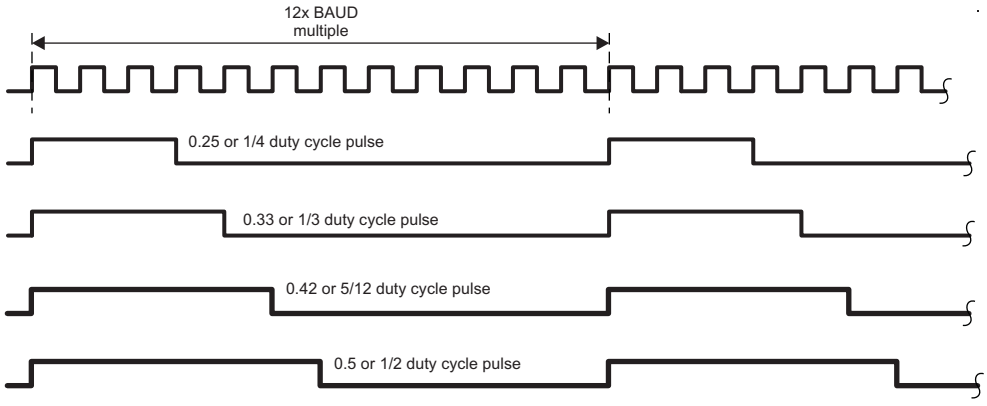
Example: For a targeted modulation frequency of 36 kHz, the CFPS value must be set to 111 in decimal which provide an modulation frequency of 36.04 kHz.

Note: The CFPS register is to start with a reset value of 105 (decimal) which translates to a frequency of 38.1 kHz.

The duty cycle of these pulses is user defined by the pulse duty register bits in the MDR2 configuration register.

MDR2[5:4]	Duty Cycle (High Level)
00	1/4
01	1/3
10	5/12
11	1/2

Figure 19-32. CIR Modulation Duty Cycle



The transmission logic ensures that all pulses are transmitted completely; i.e., there is no cut off of any pulses during its transmission. Furthermore, while transmitting continuous bytes back-to-back, no delay is inserted between two transmitted bytes. **Note:** The CIR RX demodulation can be bypassed by setting the MDR3[0] register bit. This bit will not affect the transmission modulation.

19.3.8.3.4 Frequency Divider Values

The data transferred is a succession of pulse with a T period. Depending on the standards used, the T period is defined through the DLL and DLH registers which defined the value to divide the functional clock (48 MHz):

Dividing value = (FCLK/16)/Tfreq

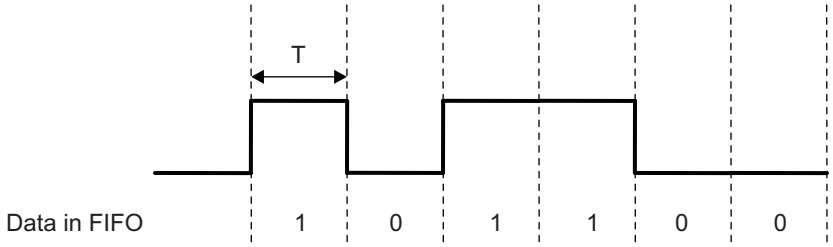
Where FCLK = System clock frequency (48 MHz)

16 = real value of BAUD multiple

Tfreq = Effective frequency of the T pulse (MHz)

In an example case using a variable pulse duration definitions:

Figure 19-33. Variable Pulse Duration Definitions



For a logical “1”, the pulse duration is equal to 2T and for a logical “0”, it’s equal to 4T.

If T =0.56 ms, the value coded into the DLH and DLL register must be 1680 in decimal.

19.4 UART/IrDA/CIR Basic Programming Model

19.4.1 UART Programming Model

19.4.1.1 Quick Start

This section describes the procedure for operating the UART with FIFO and DMA or interrupts. This three-part procedure ensures the quick start of the UART. It does not cover every UART feature.

The first programming model covers software reset of the UART. The second programming model describes FIFO and DMA configuration. The last programming model describes protocol, baud rate, and interrupt configuration.

NOTE: Each programming model can be used independently of the other two; for instance, reconfiguring the FIFOs and DMA settings only.

Each programming model can be executed starting from any UART register access mode (register modes, submodes, and other register dependencies). However, if the UART register access mode is known before executing the programming model, some steps that enable or restore register access are optional. For more information, see Section 19.3.7.1, *Register Access Modes*.

19.4.1.1.1 Software Reset

To clear the UART registers, perform the following steps:

1. Initiate a software reset:
   Set the UARTi.UART_SYSC[1] SOFTRESET bit to 1.
2. Wait for the end of the reset operation:
   Poll the UARTi.UART_SYSS[0] RESETDONE bit until it equals 1.

19.4.1.1.2 FIFOs and DMA Settings

To enable and configure the receive and transmit FIFOs and program the DMA mode, perform the following steps:

1. Switch to register configuration mode B to access the UARTi.UART_EFR register:
   (a) Save the current UARTi.UART_LCR register value.
   (b) Set the UARTi.UART_LCR register value to 0x00BF.
2. Enable register submode TCR_TLR to access the UARTi.UART_TLR register (part 1 of 2):
   (a) Save the UARTi.UART_EFR[4] ENHANCED_EN value.
   (b) Set the UARTi.UART_EFR[4] ENHANCED_EN bit to 1.
3. Switch to register configuration mode A to access the UARTi.UART_MCR register:
   Set the UARTi.UART_LCR register value to 0x0080.
4. Enable register submode TCR_TLR to access the UARTi.UART_TLR register (part 2 of 2):
   (a) Save the UARTi.UART_MCR[6] TCR_TLR value.
   (b) Set the UARTi.UART_MCR[6] TCR_TLR bit to 1.
5. Enable the FIFO; load the new FIFO triggers (part 1 of 3) and the new DMA mode (part 1 of 2):
   Set the following bits to the desired values:
   - UARTi.UART_FCR[7:6] RX_FIFO_TRIG
   - UARTi.UART_FCR[5:4] TX_FIFO_TRIG
   - UARTi.UART_FCR[3] DMA_MODE
   - UARTi.UART_FCR[0] FIFO_ENABLE (0: Disable the FIFO; 1: Enable the FIFO)

NOTE: The UARTi.UART_FCR register is not readable.

6. Switch to register configuration mode B to access the UARTi.UART_EFR register:
   Set the UARTi.UART_LCR register value to 0x00BF.
7. Load the new FIFO triggers (part 2 of 3):
   Set the following bits to the desired values:
   - UARTi.UART_TLR[7:4] RX_FIFO_TRIG_DMA
   - UARTi.UART_TLR[3:0] TX_FIFO_TRIG_DMA
8. Load the new FIFO triggers (part 3 of 3) and the new DMA mode (part 2 of 2):
   Set the following bits to the desired values:
   - UARTi.UART_SCR[7] RX_TRIG_GRANU1
   - UARTi.UART_SCR[6] TX_TRIG_GRANU1
   - UARTi.UART_SCR[2:1] DMA_MODE_2
   - UARTi.UART_SCR[0] DMA_MODE_CTL
9. Restore the UARTi.UART_EFR[4] ENHANCED_EN value saved in Step 2a.
10. Switch to register configuration mode A to access the UARTi.UART_MCR register:
    Set the UARTi.UART_LCR register value to 0x0080.
11. Restore the UARTi.UART_MCR[6] TCR_TLR value saved in Step 4a.
12. Restore the UARTi.UART_LCR value saved in Step 1a.

Triggers are used to generate interrupt and DMA requests. See Section 19.3.6.1.1, *Transmit FIFO Trigger,* to choose the following values:

- UARTi.UART_FCR[5:4] TX_FIFO_TRIG
- UARTi.UART_TLR[3:0] TX_FIFO_TRIG_DMA
- UARTi.UART_SCR[6] TX_TRIG_GRANU1

Triggers are used to generate interrupt and DMA requests. See Section 19.3.6.1.2, *Receive FIFO Trigger*, to choose the following values:

- UARTi.UART_FCR[7:6] RX_FIFO_TRIG
- UARTi.UART_TLR[7:4] RX_FIFO_TRIG_DMA
- UARTi.UART_SCR[7] RX_TRIG_GRANU1

DMA mode enables DMA requests. See Section 19.3.6.4, *FIFO DMA Mode Operation*, to choose the following values:

- UARTi.UART_FCR[3] DMA_MODE
- UARTi.UART_SCR[2:1] DMA_MODE_2
- UARTi.UART_SCR[0] DMA_MODE_CTL

19.4.1.1.3 Protocol, Baud Rate, and Interrupt Settings

To program the protocol, baud rate, and interrupt settings, perform the following steps:

1. Disable UART to access the UARTi.UART_DLL and UARTi.UART_DLH registers:
   Set the UARTi.UART_MDR1[2:0] MODE_SELECT bit field to 0x7.
2. Switch to register configuration mode B to access the UARTi.UART_EFR register:
   Set the UARTi.UART_LCR register value to 0x00BF.
3. Enable access to the UARTi.UART_IER[7:4] bit field:
   (a) Save the UARTi.UART_EFR[4] ENHANCED_EN value.
   (b) Set the UARTi.UART_EFR[4] ENHANCED_EN bit to 1.
4. Switch to register operational mode to access the UARTi.UART_IER register:
   Set the UARTi.UART_LCR register value to 0x0000.

5. Clear the UARTi.UART_IER register (set the UARTi.UART_IER[4] SLEEP_MODE bit to 0 to change the UARTi.UART_DLL and UARTi.UART_DLH registers). Set the UARTi.UART_IER register value to 0x0000.
6. Switch to register configuration mode B to access the UARTi.UART_DLL and UARTi.UART_DLH registers:

   Set the UARTi.UART_LCR register value to 0x00BF.
7. Load the new divisor value:

   Set the UARTi.UART_DLL[7:0] CLOCK_LSB and UARTi.UART_DLH[5:0] CLOCK_MSB bit fields to the desired values.
8. Switch to register operational mode to access the UARTi.UART_IER register:

   Set the UARTi.UART_LCR register value to 0x0000.
9. Load the new interrupt configuration (0: Disable the interrupt; 1: Enable the interrupt):

   Set the following bits to the desired values:
   - UARTi.UART_IER[7] CTS_IT
   - UARTi.UART_IER[6] RTS_IT
   - UARTi.UART_IER[5] XOFF_IT
   - UARTi.UART_IER[4] SLEEP_MODE
   - UARTi.UART_IER[3] MODEM_STS_IT
   - UARTi.UART_IER[2] LINE_STS_IT
   - UARTi.UART_IER[1] THR_IT
   - UARTi.UART_IER[0] RHR_IT
10. Switch to register configuration mode B to access the UARTi.UART_EFR register:

    Set the UARTi.UART_LCR register value to 0x00BF.
11. Restore the UARTi.UART_EFR[4] ENHANCED_EN value saved in Step 3a.
12. Load the new protocol formatting (parity, stop-bit, character length) and switch to register operational mode:

    Set the UARTi.UART_LCR[7] DIV_EN bit to 0.

    Set the UARTi.UART_LCR[6] BREAK_EN bit to 0.

    Set the following bits to the desired values:
    - UARTi.UART_LCR[5] PARITY_TYPE_2
    - UARTi.UART_LCR[4] PARITY_TYPE_1
    - UARTi.UART_LCR[3] PARITY_EN
    - UARTi.UART_LCR[2] NB_STOP
    - UARTi.UART_LCR[1:0] CHAR_LENGTH
13. Load the new UART mode:

    Set the UARTi.UART_MDR1[2:0] MODE_SELECT bit field to the desired value.

See Section 19.3.8.1.2, *Choosing the Appropriate Divisor Value*, to choose the following values:

- UARTi.UART_DLL[7:0] CLOCK_LSB
- UARTi.UART_DLH[5:0] CLOCK_MSB
- UARTi.UART_MDR1[2:0] MODE_SELECT

See Section 19.3.8.1.3.1, *Frame Formatting*, to choose the following values:

- UARTi.UART_LCR[5] PARITY_TYPE_2
- UARTi.UART_LCR[4] PARITY_TYPE_1
- UARTi.UART_LCR[3] PARITY_EN
- UARTi.UART_LCR[2] NB_STOP
- UARTi.UART_LCR[1:0] CHAR_LENGTH

19.4.1.2 Hardware and Software Flow Control Configuration

This section describes the programming steps to enable and configure hardware and software flow control. Hardware and software flow control cannot be used at the same time.

NOTE: Each programming model can be executed starting from any UART register access mode (register modes, submodes, and other register dependencies). However, if the UART register access mode is known before executing the programming model, some steps that enable or restore register access are optional. For more information, see Section 19.3.7.1, *Register Access Modes*.

19.4.1.2.1 Hardware Flow Control Configuration

To enable and configure hardware flow control, perform the following steps:

1. Switch to register configuration mode A to access the UARTi.UART_MCR register:
   (a) Save the current UARTi.UART_LCR register value.
   (b) Set the UARTi.UART_LCR register value to 0x0080.
2. Enable register submode TCR_TLR to access the UARTi.UART_TCR register (part 1 of 2):
   (a) Save the UARTi.UART_MCR[6] TCR_TLR value.
   (b) Set the UARTi.UART_MCR[6] TCR_TLR bit to 1.
3. Switch to register configuration mode B to access the UARTi.UART_EFR register:
   Set the UARTi.UART_LCR register value to 0x00BF.
4. Enable register submode TCR_TLR to access the UARTi.UART_TCR register (part 2 of 2):
   (a) Save the UARTi.UART_EFR[4] ENHANCED_EN value.
   (b) Set the UARTi.UART_EFR[4] ENHANCED_EN bit to 1.
5. Load the new start and halt trigger values for hardware flow control:
   Set the following bits to the desired values:
   - UARTi.UART_TCR[7:4] AUTO_RTS_START
   - UARTi.UART_TCR[3:0] AUTO_RTS_HALT
6. Enable or disable receive and transmit hardware flow control mode and restore the UARTi.UART_EFR[4] ENHANCED_EN value saved in Step 4a.
   Set the following bits to the desired values:
   - UARTi.UART_EFR[7] AUTO_CTS_EN (0: Disable; 1: Enable)
   - UARTi.UART_EFR[6] AUTO_RTS_EN (0: Disable; 1: Enable)

   Restore the UARTi.UART_EFR[4] ENHANCED_EN bit to the saved value.
7. Switch to register configuration mode A to access the UARTi.UART_MCR register:
   Set the UARTi.UART_LCR register value to 0x0080.
8. Restore the UARTi.UART_MCR[6] TCR_TLR value saved in Step 2a.
9. Restore the UARTi.UART_LCR value saved in Step 1a.

See Section 19.3.8.1.3.2, *Hardware Flow Control*, to choose the following values:

- UARTi.UART_EFR[7] AUTO_CTS_EN
- UARTi.UART_EFR[6] AUTO_RTS_EN
- UARTi.UART_TCR[7:4] AUTO_RTS_START
- UARTi.UART_TCR[3:0] AUTO_RTS_HALT

19.4.1.2.2 Software Flow Control Configuration

To enable and configure software flow control, perform the following steps:

1. Switch to register configuration mode B to access the UARTi.UART_EFR register.

(a) Save the current UARTi.UART_LCR register value.

(b) Set the UARTi.UART_LCR register value to 0x00BF.

2. Enable register submode XOFF to access the UARTi.UART_XOFF1 and UARTi.UART_XOFF2 registers:

   (a) Save the UARTi.UART_EFR[4] ENHANCED_EN value.

   (b) Set the UARTi.UART_EFR[4] ENHANCED_EN bit to 0.

3. Load the new software flow control characters:

   Set the following bits to the desired values:

   - UARTi.UART_XON1_ADDR1[7:0] XON_WORD1
   - UARTi.UART_XON2_ADDR2[7:0] XON_WORD2
   - UARTi.UART_XOFF1[7:0] XOFF_WORD1
   - UARTi.UART_XOFF2[7:0] XOFF_WORD2

4. Enable access to the UARTi.UART_MCR[7:5] bit field and enable register submode TCR_TLR to access the UARTi.UART_TCR register (part 1 of 2):

   Set the UARTi.UART_EFR[4] ENHANCED_EN bit to 1.

5. Switch to register configuration mode A to access the UARTi.UART_MCR register:

   Set the UARTi.UART_LCR register value to 0x0080.

6. Enable register submode TCR_TLR to access the UARTi.UART_TCR register (part 2 of 2) and enable or disable XON any function:

   (a) Save the UARTi.UART_MCR[6] TCR_TLR value.

   (b) Set the UARTi.UART_MCR[6] TCR_TLR bit to 1.

   (c) Set the UARTi.UART_MCR[5] XON_EN bit to the desired value (0: Disable; 1: Enable).

7. Switch to register configuration mode B to access the UARTi.UART_EFR register:

   Set the UARTi.UART_LCR register value to 0x00BF.

8. Load the new start and halt trigger values for software flow control:

   Set the following bits to the desired values:

   - UARTi.UART_TCR[7:4] AUTO_RTS_START
   - UARTi.UART_TCR[3:0] AUTO_RTS_HALT

9. Enable or disable special character function and load the new software flow control mode and restore the UARTi.UART_EFR[4] ENHANCED_EN value saved in Step 2a:

   Set the following bits to the desired values:

   - UARTi.UART_EFR[5] SPEC_CHAR (0: Disable; 1: Enable)
   - UARTi.UART_EFR[3:0] SW_FLOW_CONTROL

   Restore the UARTi.UART_EFR[4] ENHANCED_EN bit to the saved value.

10. Switch to register configuration mode A to access the UARTi.UART_MCR register:

    Set the UARTi.UART_LCR register value to 0x0080.

11. Restore the UARTi.UART_MCR[6] TCR_TLR bit value saved in Step 6a.

12. Restore the UARTi.UART_LCR value saved in Step 1a.

See Section 19.3.8.1.3.3, *Software Flow Control*, to choose the following values:

- UARTi.UART_EFR[5] SPEC_CHAR
- UARTi.UART_EFR[3:0] SW_FLOW_CONTROL
- UARTi.UART_TCR[7:4] AUTO_RTS_START
- UARTi.UART_TCR[3:0] AUTO_RTS_HALT
- UARTi.UART_XON1_ADDR1[7:0] XON_WORD1
- UARTi.UART_XON2_ADDR2[7:0] XON_WORD2
- UARTi.UART_XOFF1[7:0] XOFF_WORD1

- UARTi.UART_XOFF2[7:0] XOFF_WORD2

19.4.2 IrDA Programming Model

19.4.2.1 SIR Mode

19.4.2.1.1 Receive

The following programming model explains how to program the module to receive an IrDA frame with parity forced to 1, baud rate = 112.5KB, FIFOs disabled, 2 stop-bits, and 8-bit word length:

1. Disable the UART before accessing the UARTi.UART_DLL and UARTi.UART_DLH registers:
   Set the UARTi.UART_MDR1[2:0] MODE_SELECT bit field to 0x7.
2. Grant access to the UART_DLL and UART_DLH registers (the UART_LCR[7] DIV_EN bit = 1):
   UARTi.UART_LCR = 0x80 (Data format is unaffected by the use and settings of the UARTi.UART_LCR register in IrDA mode.)
3. Load the new baud rate (115.2 kbps):
   UARTi.UART_DLL = 0x1A
   UARTi.UART_DLH = 0x00
4. Set SIR mode:
   UARTi.UART_MDR1[2:0] MODE_SELECT = 0x1
5. Disable access to the UART_DLL and UART_DLH registers and switch to register operational mode:
   UARTi.UART_LCR = 0x00.
6. Optional: Enable the RHR interrupt:
   UARTi.UART_IER[0] RHR_IT = 0x1

19.4.2.1.2 Transmit

The following programming model explains how to program the module to transmit an IrDA 6-byte frame with no parity, baud rate = 112.5 kbps, FIFOs disabled, 3/16 encoding, 2 stop-bits, and 7-bit word length:

1. Disable the UART before accessing the UARTi.UART_DLL and UARTi.UART_DLH registers:
   Set the UART_MDR1[2:0] MODE_SELECT bit field to 0x7.
2. Grant access to the UART_EFR register:
   UARTi.UART_LCR = 0xBF
3. Enable the enhanced features (the UART_EFR[4] ENAHNCED_EN bit = 1):
   Set the UARTi.UART_EFR register value to 0x10.
4. Grant access to the UART_DLL and UART_DLH registers (the UART_LCR[7] DIV_EN bit = 1):
   UARTi.UART_LCR = 0x80 (Data format is unaffected by the use and settings of the UARTi.UART_LCR register in IrDA mode.)
5. Load the new baud rate (115.2 kbps):
   UARTi.UART_DLL = 0x1A
   UARTi.UART_DLH = 0x00
6. Set SIR mode (the UART_MDR1[2:0] MODE_SELECT bit field = 0x1):
   UARTi.UART_MDR1 = 0x01
7. Disable access to the UART_DLL and UART_DLH registers and switch to register operational mode:
   UARTi.UART_LCR = 0x00.
8. Force DTR output to active:
   UARTi.UART_MCR[0] DTR = 1
9. Optional: Enable the THR interrupt:
   UARTi.UART_IER[1] THR_IT = 1
10. Set transmit frame length to 6 bytes:
    UARTi.UART_TXFLL = 0x06
11. Set 7 starts of frame transmission:
    UARTi.UART_EBLR = 0x08
12. Optional: Set SIR pulse width to be 1.6 us:
    UARTi.UART_ACREG[7] PULSE_TYPE = 1
13. Load the UART_THR register with the data to be transmitted.

19.4.2.2 MIR Mode

19.4.2.2.1 Receive

The following programming model explains how to program the module to receive an IrDA frame with no parity, baud rate = 1.152 Mpbs, and FIFOs disabled.

1. Disable the UART before accessing the UARTi.UART_DLL and UARTi.UART_DLH registers:
   Set the UARTi.UART_MDR1[2:0] MODE_SELECT bit field to 0x7.
2. Grant access to the UART_DLL and UART_DLH registers (UART_LCR[7] DIV_EN bit = 1):
   UARTi.UART_LCR = 0x80 (Data format is unaffected by the use and settings of the UARTi.UART_LCR register in IrDA mode.)
3. Load the new baud rate (1.152 Mpbs):
   UARTi.UART_DLL = 0x01
   UARTi.UART_DLH = 0x00
4. Set MIR mode:
   UARTi.UART_MDR1[2:0] MODE_SELECT = 0x4
5. Disable access to the UART_DLL and UART_DLH registers and switch to register operational mode:
   UARTi.UART_LCR = 0x00
6. Force DTR output to active (UART_MCR[0] DTR = 1):
   Force RTS output to active (UART_MCR[1] RTS = 1).
   UARTi.UART_MCR = 0x3
7. Optional: Enable the RHR interrupt:
   UARTi.UART_IER[0] RHR_IT = 1

19.4.2.2.2 Transmit

The following programming model explains how to program the module to transmit an IrDA 60-byte frame with no parity, baud rate = 1.152 Mpbs, and FIFOs disabled:

1. Disable the UART before accessing the UARTi.UART_DLL and UARTi.UART_DLH registers:
   Set the UARTi.UART_MDR1[2:0] MODE_SELECT bit field to 0x7.
2. Grant access to the UART_DLL and UART_DLH registers (UART_LCR[7] DIV_EN bit = 1):
   UARTi.UART_LCR = 0x80 (Data format is unaffected by the use and settings of the UARTi.UART_LCR register in IrDA mode.)
3. Load the new baud rate (1.152 Mpbs):
   UARTi.UART_DLL = 0x01
   UARTi.UART_DLH = 0x00
4. Set MIR mode:
   UARTi.UART_MDR1[2:0] MODE_SELECT = 0x4
5. Disable access to the UART_DLL and UART_DLH registers and switch to register operational mode:
   UARTi.UART_LCR = 0x00
6. Force DTR output to active:
   UARTi.UART_MCR[0] DTR = 1
7. Optional: Enable the THR interrupt:
   UARTi.UART_IER[1] THR_IT = 1
8. Set the frame length to 60 bytes:
   UARTi.UART_TXFLL = 0x3C
9. Optional: Transmit eight additional starts of frame (MIR mode requires two starts):
   UARTi.UART_EBLR = 0x08
10. SIP is sent at the end of transmission:
    UARTi.UART_ACREG[3] = 1
11. Load the UART_THR register with the data to be transmitted.

19.4.2.3 FIR Mode

19.4.2.3.1 Receive

The following programming model explains how to program the module to receive the IrDA frame with no parity, baud rate = 4 Mbps, FIFOs enabled, 8-bit word length.

1. Disable the UART before accessing the UARTi.UART_DLL and UARTi.UART_DLH registers:
   Set the UARTi.UART_MDR1[2:0] MODE_SELECT bit field to 0x7.
2. Grant access to the UART_DLL and UART_DLH registers (UART_LCR[7] DIV_EN bit = 1):
   UARTi.UART_LCR = 0x80 (Data format is unaffected by the use and settings of the UARTi.UART_LCR register in IrDA mode.)
3. FIFO clear and enable:
   UARTi.UART_FCR = 0x7 (TX/RX FIFO trigger: UART_FCR[7:6] and UART_FCR[5:4])
   UARTi.UART_LCR[7] = 0
4. Set FIR mode:
   UARTi.UART_MDR1[2:0] MODE_SELECT = 0x5
5. Set frame length:
   UARTi.UART_RXFLL = 0xA (Data + CRC + STOP)
6. Disable access to the UARTi.UART_DLL registers and UARTi.UART_DLH and switch to register operational mode:
   UARTi.UART_LCR[7] DIV_EN = 0x0
7. Optional: Enable the RHR interrupt:
   UARTi.UART_IER[0] RHR_IT = 1

19.4.2.3.2 Transmit

The following programming model explains how to program the module to transmit an IrDA 4-byte frame with no parity, baud rate = 4 Mbps, FIFOs enabled, and 8-bit word length.

1. Disable the UART before accessing the UARTi.UART_DLL and UARTi.UART_DLH registers:
   Set the UARTi.UART_MDR1[2:0] MODE_SELECT bit field to 0x7.
2. Grant access to EFR_REG:
   UARTi.UART_LCR = 0xBF
3. Enable the enhanced features (EFR_REG[4] ENAHNCED_EN = 0x1):
   UARTi.UART_EFR = 0x10
4. FIFO clear and enable:
   UARTi.UART_FCR = 0x7 (TX/RX FIFO trigger: UART_FCR[7:6] and UART_FCR[5:4]).
   UARTi.UART_LCR[7] = 0
5. Set FIR mode and enable auto-SIP mode:
   UARTi.UART_MDR1 = 0x45
6. Set frame length:
   UARTi.UART_TXFLL = 0x4
   UARTi.UART_TXFLH = 0x0
   UARTi.UART_RXFLL = 0xA (Data + CRC + STOP)
   UARTi.UART_RXFLH = 0x0
7. Force DTR output to active:
   UARTi.UART_MCR[0] DTR = 0x1
8. Optional: Enable the THR interrupt:
   UARTi.UART_IER[1] THR_IT = 0x1
9. Optional: Transmit eight additional starts of frame (MIR mode requires two starts):
   UARTi.UART_EBLR = 0x08
10. SIP is sent at the end of transmission:
    UARTi.UART_ACREG[3] = 1
11. Load the UART_THR register with the data to be transmitted.

19.5 UART Registers

19.5.1 UART Registers

Table 19-29 lists the memory-mapped registers for the UART. All register offset addresses not listed in Table 19-29 should be considered as reserved locations and the register contents should not be modified.

Table 19-29. UART Registers

Offset	Acronym	Register Name	Section
0h	THR	Transmit Holding Register	Section 19.5.1.1
0h	RHR	Receiver Holding Register	Section 19.5.1.2
0h	DLL	Divisor Latches Low Register	Section 19.5.1.3
4h	IER_IRDA	Interrupt Enable Register (IrDA)	Section 19.5.1.4
4h	IER_CIR	Interrupt Enable Register (CIR)	Section 19.5.1.5
4h	IER_UART	Interrupt Enable Register (UART)	Section 19.5.1.6
4h	DLH	Divisor Latches High Register	Section 19.5.1.7
8h	EFR	Enhanced Feature Register	Section 19.5.1.8
8h	IIR_UART	Interrupt Identification Register (UART)	Section 19.5.1.9
8h	IIR_CIR	Interrupt Identification Register (CIR)	Section 19.5.1.10
8h	FCR	FIFO Control Register	Section 19.5.1.11
8h	IIR_IRDA	Interrupt Identification Register (IrDA)	Section 19.5.1.12
Ch	LCR	Line Control Register	Section 19.5.1.13
10h	MCR	Modem Control Register	Section 19.5.1.14
10h	XON1_ADDR1	XON1/ADDR1 Register	Section 19.5.1.15
14h	XON2_ADDR2	XON2/ADDR2 Register	Section 19.5.1.16
14h	LSR_CIR	Line Status Register (CIR)	Section 19.5.1.17
14h	LSR_IRDA	Line Status Register (IrDA)	Section 19.5.1.18
14h	LSR_UART	Line Status Register (UART)	Section 19.5.1.19
18h	TCR	Transmission Control Register	Section 19.5.1.20
18h	MSR	Modem Status Register	Section 19.5.1.21
18h	XOFF1	XOFF1 Register	Section 19.5.1.22
1Ch	SPR	Scratchpad Register	Section 19.5.1.23
1Ch	TLR	Trigger Level Register	Section 19.5.1.24
1Ch	XOFF2	XOFF2 Register	Section 19.5.1.25
20h	MDR1	Mode Definition Register 1	Section 19.5.1.26
24h	MDR2	Mode Definition Register 2	Section 19.5.1.27
28h	TXFLL	Transmit Frame Length Low Register	Section 19.5.1.28
28h	SFLSR	Status FIFO Line Status Register	Section 19.5.1.29
2Ch	RESUME	RESUME Register	Section 19.5.1.30
2Ch	TXFLH	Transmit Frame Length High Register	Section 19.5.1.31
30h	RXFLL	Received Frame Length Low Register	Section 19.5.1.32
30h	SFREGL	Status FIFO Register Low	Section 19.5.1.33
34h	SFREGH	Status FIFO Register High	Section 19.5.1.34
34h	RXFLH	Received Frame Length High Register	Section 19.5.1.35
38h	BLR	BOF Control Register	Section 19.5.1.36
38h	UASR	UART Autobauding Status Register	Section 19.5.1.37
3Ch	ACREG	Auxiliary Control Register	Section 19.5.1.38
40h	SCR	Supplementary Control Register	Section 19.5.1.39
44h	SSR	Supplementary Status Register	Section 19.5.1.40
48h	EBLR	BOF Length Register	Section 19.5.1.41

Table 19-29. UART Registers (continued)

Offset	Acronym	Register Name	Section
50h	MVR	Module Version Register	Section 19.5.1.42
54h	SYSC	System Configuration Register	Section 19.5.1.43
58h	SYSS	System Status Register	Section 19.5.1.44
5Ch	WER	Wake-Up Enable Register	Section 19.5.1.45
60h	CFPS	Carrier Frequency Prescaler Register	Section 19.5.1.46
64h	RXFIFO_LVL	Received FIFO Level Register	Section 19.5.1.47
68h	TXFIFO_LVL	Transmit FIFO Level Register	Section 19.5.1.48
6Ch	IER2	IER2 Register	Section 19.5.1.49
70h	ISR2	ISR2 Register	Section 19.5.1.50
74h	FREQ_SEL	FREQ_SEL Register	Section 19.5.1.51
80h	MDR3	Mode Definition Register 3	Section 19.5.1.52
84h	TX_DMA_THRESHOLD	TX DMA Threshold Register	Section 19.5.1.53

19.5.1.1 THR Register (offset = 0h) [reset = 0h]

THR is shown in Figure 19-34 and described in Table 19-30.

The transmit holding register (THR) is selected with the register bit setting of LCR[7] = 0. The transmitter section consists of the transmit holding register and the transmit shift register. The transmit holding register is a 64-byte FIFO. The MPU 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 zero of the FIFO is used to store the data.

Figure 19-34. THR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
THR							
W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-30. THR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7-0	THR	W	0h	Transmit holding register. Value 0 to FFh.

19.5.1.2 RHR Register (offset = 0h) [reset = 0h]

RHR is shown in Figure 19-35 and described in Table 19-31.

The receiver holding register (RHR) is selected with the register bit setting of LCR[7] = 0. The receiver section consists of the receiver holding register 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, the data in the RHR is not overwritten.

Figure 19-35. RHR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RHR							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-31. RHR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7-0	RHR	R	0h	Receive holding register. Value 0 to FFh.

19.5.1.3 DLL Register (offset = 0h) [reset = 0h]

DLL is shown in Figure 19-36 and described in Table 19-32.

The divisor latches low register (DLL) is selected with a register bit setting of LCR[7] not equal to BFh or LCR[7] = BFh. The divisor latches low register (DLL) with the DLH register stores the 14-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. DLL and DLH can be written to only before sleep mode is enabled (before IER[4] is set).

Figure 19-36. DLL Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
CLOCK_LSB							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-32. DLL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7-0	CLOCK_LSB	R/W	0h	Divisor latches low. Stores the 8 LSB divisor value. Value 0 to FFh.

19.5.1.4 IER_IRDA Register (offset = 4h) [reset = 0h]

IER_IRDA is shown in Figure 19-37 and described in Table 19-33.

The following interrupt enable register (IER) description is for IrDA mode. The IrDA IER is selected with a register bit setting of LCR[7] = 0. In IrDA mode, EFR[4] has no impact on the access to IER[7:4]. The IrDA interrupt enable register (IER) can be programmed to enable/disable any interrupt. There are 8 types of interrupt in these modes, received EOF, LSR interrupt, TX status, status FIFO interrupt, RX overrun, last byte in RX FIFO, THR interrupt, and RHR interrupt. Each interrupt can be enabled/disabled individually. The TXSTATUSIT interrupt reflects two possible conditions. The MDR2[0] bit should be read to determine the status in the event of this interrupt.

Figure 19-37. IER_IRDA Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
EOFIT	LINESTSIT	TXSTATUSIT	STSFIFOTRIGIT	RXOVERRUNIT	LASTRXBYTEIT	THRIT	RHRIT
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-33. IER_IRDA Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7	EOFIT	R/W	0h	EOFIT 0h = Disables the received EOF interrupt. 1h = Enables the received EOF interrupt.
6	LINESTSIT	R/W	0h	LINESTSIT 0h = Disables the receiver line status interrupt. 1h = Enables the receiver line status interrupt.
5	TXSTATUSIT	R/W	0h	TXSTATUSIT 0h = Disables the TX status interrupt. 1h = Enables the TX status interrupt.
4	STSFIFOTRIGIT	R/W	0h	STSFIFOTRIGIT 0h = Disables status FIFO trigger level interrupt. 1h = Enables status FIFO trigger level interrupt.
3	RXOVERRUNIT	R/W	0h	RXOVERRUNIT 0h = Disables the RX overrun interrupt. 1h = Enables the RX overrun interrupt.
2	LASTRXBYTEIT	R/W	0h	LASTRXBYTEIT 0h = Disables the last byte of frame in RX FIFO interrupt. 1h = Enables the last byte of frame in RX FIFO interrupt.
1	THRIT	R/W	0h	THRIT 0h = Disables the THR interrupt. 1h = Enables the THR interrupt.
0	RHRIT	R/W	0h	RHRIT 0h = Disables the RHR interrupt. 1h = Enables the RHR interrupt.

19.5.1.5 IER_CIR Register (offset = 4h) [reset = 0h]

IER_CIR is shown in Figure 19-38 and described in Table 19-34.

The following interrupt enable register (IER) description is for CIR mode. The CIR IER is selected with a register bit setting of LCR[7] = 0. In IrDA mode, EFR[4] has no impact on the access to IER[7:4]. The CIR interrupt enable register (IER) can be programmed to enable/disable any interrupt. There are 5 types of interrupt in these modes, TX status, RX overrun, RX stop interrupt, THR interrupt, and RHR interrupt. Each interrupt can be enabled/disabled individually. In CIR mode, the TXSTATUSIT bit has only one meaning corresponding to the case MDR2[0] = 0. The RXSTOPIT interrupt is generated based on the value set in the BOF Length register (EBLR).

Figure 19-38. IER_CIR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED		TXSTATUSIT	RESERVED	RXOVERRUNIT	RXSTOPIT	THRIT	RHRIT
R-0h		R/W-0h	R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-34. IER_CIR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-6	RESERVED	R	0h	
5	TXSTATUSIT	R/W	0h	TXSTATUSIT. 0h = Disables the TX status interrupt. 1h = Enables the TX status interrupt.
4	RESERVED	R	0h	
3	RXOVERRUNIT	R/W	0h	RXOVERRUNIT. 0h = Disables the RX overrun interrupt. 1h = Enables the RX overrun interrupt.
2	RXSTOPIT	R/W	0h	RXSTOPIT. 0h = Disables the RX stop interrupt. 1h = Enables the RX stop interrupt.
1	THRIT	R/W	0h	THRIT. 0h = Disables the THR interrupt. 1h = Enables the THR interrupt.
0	RHRIT	R/W	0h	RHRIT. 0h = Disables the RHR interrupt. 1h = Enables the RHR interrupt.

19.5.1.6 IER_UART Register (offset = 4h) [reset = 0h]

IER_UART is shown in Figure 19-39 and described in Table 19-35.

The following interrupt enable register (IER) description is for UART mode. The UART IER is selected with a register bit setting of LCR[7] = 0. In UART mode, IER[7:4] can only be written when EFR[4] = 1 The interrupt enable register (IER) can be programmed to enable/disable any interrupt. There are seven types of interrupt in this mode: receiver error, RHR interrupt, THR interrupt, XOFF received and CTS (active-low)/RTS (active-low) change of state from low to high. Each interrupt can be enabled/disabled individually. There is also a sleep mode enable bit. The UART interrupt enable register (IER) is shown in and described in .

Figure 19-39. IER_UART Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
CTSIT	RTSIT	XOFFIT	SLEEPMODE	MODEMSTSIT	LINESTSIT	THRIT	RHRIT
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-35. IER_UART Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7	CTSIT	R/W	0h	Can be written only when EFR[4] = 1. 0h = Disables the CTS (active-low) interrupt. 1h = Enables the CTS (active-low) interrupt.
6	RTSIT	R/W	0h	Can be written only when EFR[4] = 1. 0h = Disables the RTS (active-low) interrupt. 1h = Enables the RTS (active-low) interrupt.
5	XOFFIT	R/W	0h	Can be written only when EFR[4] = 1. 0h = Disables the XOFF interrupt. 1h = Enables the XOFF interrupt.
4	SLEEPMODE	R/W	0h	Can be only written when EFR[4] = 1. 0h = Disables sleep mode. 1h = Enables sleep mode (stop baud rate clock when the module is inactive).
3	MODEMSTSIT	R/W	0h	MODEMSTSIT. 0h = Disables the modem status register interrupt. 1h = Enables the modem status register interrupt
2	LINESTSIT	R/W	0h	LINESTSIT. 0h = Disables the receiver line status interrupt. 1h = Enables the receiver line status interrupt.
1	THRIT	R/W	0h	THRIT. 0h = Disables the THR interrupt. 1h = Enables the THR interrupt.
0	RHRIT	R/W	0h	RHRIT. 0h = Disables the RHR interrupt and time out interrupt. 1h = Enables the RHR interrupt and time out interrupt.

19.5.1.7 DLH Register (offset = 4h) [reset = 0h]

DLH is shown in Figure 19-40 and described in Table 19-36.

The divisor latches high register (DLH) is selected with a register bit setting of LCR[7] not equal to BFh or LCR[7] = BFh. The divisor latches high register (DLH) with the DLL register stores the 14-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. DLL and DLH can be written to only before sleep mode is enabled (before IER[4] is set).

Figure 19-40. DLH Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED		CLOCK_MSB					
R-0h		R/W-0h					

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-36. DLH Register Field Descriptions

Bit	Field	Type	Reset	Description
15-6	RESERVED	R	0h	
5-0	CLOCK_MSB	R/W	0h	Divisor latches high. Stores the 6 MSB divisor value. Value 0 to 3Fh.

19.5.1.8 EFR Register (offset = 8h) [reset = 0h]

EFR is shown in Figure 19-41 and described in Table 19-37.

The enhanced feature register (EFR) is selected with a register bit setting of LCR[7] = BFh. The enhanced feature register (EFR) enables or disables enhanced features. Most enhanced functions apply only to UART modes, but EFR[4] enables write accesses to FCR[5:4], the TX trigger level, which is also used in IrDA modes.

Figure 19-41. EFR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
AUTOCTSEN	AUTORTSEN	SPECIALCHAR DETECT	ENHANCEDEN	SWFLOWCONTROL			
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-37. EFR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7	AUTOCTSEN	R/W	0h	Auto-CTS enable bit (UART mode only). 0h = Normal operation. 1h = Auto-CTS flow control is enabled; transmission is halted when the CTS (active-low) pin is high (inactive).
6	AUTORTSEN	R/W	0h	Auto-RTS enable bit (UART mode only). 0h = Normal operation. 1h = Auto-RTS flow control is enabled; RTS (active-low) 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.
5	SPECIALCHARDETECT	R/W	0h	Special character detect (UART mode only). 0h = Normal operation. 1h = Special character detect enable. Received data is compared with XOFF2 data. If a match occurs, the received data is transferred to RX FIFO and the IIR[4] bit is set to 1 to indicate that a special character was detected.
4	ENHANCEDEN	R/W	0h	Enhanced functions write enable bit. 0h = Disables writing to IER[7:4], FCR[5:4], and MCR[7:5]. 1h = Enables writing to IER[7:4], FCR[5:4], and MCR[7:5].

Table 19-37. EFR Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
3-0	SWFLOWCONTROL	R/W	0h	Combinations of software flow control can be selected by programming this bit. XON1 and XON2 should be set to different values if the software flow control is enabled. The TX and RX software flow control options are as follows. EFR[3] = 0, EFR[2] = 0, EFR[1] = X, and EFR[0] = X, then: No transmit flow control. EFR[3] = 1, EFR[2] = 0, EFR[1] = X, and EFR[0] = X, then: Transmit XON1, XOFF1. EFR[3] = 0, EFR[2] = 1, EFR[1] = X, and EFR[0] = X, then: Transmit XON2, XOFF2. EFR[3] = 1, EFR[2] = 1, EFR[1] = X, and EFR[0] = X, then: Transmit XON1, XON2 or XOFF1, XOFF2. The XON1 and XON2 characters or the XOFF1 and XOFF2 characters must be transmitted/received sequentially with XON1/XOFF1 followed by XON2/XOFF2. EFR[3] = X, EFR[2] = X, EFR[1] = 0, and EFR[0] = 0, then: No receive flow control. EFR[3] = X, EFR[2] = X, EFR[1] = 1, and EFR[0] = 0, then: Receiver compares XON1, XOFF1. EFR[3] = X, EFR[2] = X, EFR[1] = 0, and EFR[0] = 1, then: Receiver compares XON2, XOFF2. EFR[3] = X, EFR[2] = X, EFR[1] = 1, and EFR[0] = 1, then: Receiver compares XON1, XON2 or XOFF1, XOFF2. The XON1 and XON2 characters or the XOFF1 and XOFF2 characters must be transmitted/received sequentially with XON1/XOFF1 followed by XON2/XOFF2. In IrDA mode, EFR[1] and EFR[0] select the IR address to check (see IR Address Checking).

19.5.1.9 IIR_UART Register (offset = 8h) [reset = 1h]

IIR_UART is shown in Figure 19-42 and described in Table 19-38.

The following interrupt identification register (IIR) description is for UART mode. The UART IIR is selected with a register bit setting of LCR[7] = 0 or LCR[7] not equal to BFh. The UART interrupt identification register (IIR) is a read-only register that provides the source of the interrupt. An interrupt source can be flagged only if enabled in the IER register.

Figure 19-42. IIR_UART Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
FCR_MIRROR		IT_TYPE					IT_PENDING
R-0h		R-0h					R-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 19-38. IIR_UART Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7-6	FCR_MIRROR	R	0h	Mirror the contents of FCR[0] on both bits.
5-1	IT_TYPE	R	0h	Seven possible interrupts in UART mode. Other combinations never occur: 0h = Modem interrupt. Priority = 4. 1h = THR interrupt. Priority = 3. 2h = RHR interrupt. Priority = 2. 3h = Receiver line status error. Priority = 1. 4h = Reserved 5h = Reserved 6h = Rx timeout. Priority = 2. 7h = Reserved 8h = Xoff/special character. Priority = 5. 9h = Reserved, from 9h to Fh. 10h = CTS (active-low), RTS (active-low), DSR (active-low) change state from active (low) to inactive (high). Priority = 6. 11h = Reserved, from 11 to 1Fh.
0	IT_PENDING	R	1h	Interrupt pending. 0h = An interrupt is pending. 1h = No interrupt is pending.

19.5.1.10 IIR_CIR Register (offset = 8h) [reset = 0h]

IIR_CIR is shown in Figure 19-43 and described in Table 19-39.

The following interrupt identification register (IIR) description is for CIR mode. The CIR IIR is selected with a register bit setting of LCR[7] = 0 or LCR[7] not equal to BFh. The CIR interrupt identification register (IIR) is a read-only register that provides the source of the interrupt. An interrupt source can be flagged only if enabled in the IER register.

Figure 19-43. IIR_CIR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED		TXSTATUSIT	RESERVED	RXOEIT	RXSTOPIT	THRIT	RHRIT_
R-0h		R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 19-39. IIR_CIR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-6	RESERVED	R	0h	
5	TXSTATUSIT	R	0h	TXSTATUSIT 0h = TX status interrupt inactive 1h = TX status interrupt active
4	RESERVED	R	0h	
3	RXOEIT	R	0h	RXOEIT 0h = RX overrun interrupt inactive 1h = RX overrun interrupt active
2	RXSTOPIT	R	0h	RXSTOPIT 0h = Receive stop interrupt is inactive 1h = Receive stop interrupt is active
1	THRIT	R	0h	THRIT 0h = THR interrupt inactive 1h = THR interrupt active
0	RHRIT_	R	0h	RHRIT 0h = RHR interrupt inactive 1h = RHR interrupt active

19.5.1.11 FCR Register (offset = 8h) [reset = 0h]

FCR is shown in Figure 19-44 and described in Table 19-40.

The FIFO control register (FCR) is selected with a register bit setting of LCR[7] = 0 or LCR[7] not equal to BFh. FCR[5:4] can only be written when EFR[4] = 1.

Figure 19-44. FCR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RX_FIFO_TRIG		TX_FIFO_TRIG		DMA_MODE	TX_FIFO_CLEAR	RX_FIFO_CLEAR	FIFO_EN
W-0h		W-0h		W-0h	W-0h	W-0h	W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-40. FCR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7-6	RX_FIFO_TRIG	W	0h	Sets the trigger level for the RX FIFO: If SCR[7] = 0 and TLR[7] to TLR[4] not equal to 0000, RX_FIFO_TRIG is not considered. If SCR[7] = 1, RX_FIFO_TRIG is 2 LSB of the trigger level (1 to 63 on 6 bits) with the granularity 1. If SCR[7] = 0 and TLR[7] to TLR[4] = 0000, then: 0h = 8 characters 1h = 16 characters 2h = 56 characters 3h = 60 characters
5-4	TX_FIFO_TRIG	W	0h	Can be written only if EFR[4] = 1. Sets the trigger level for the TX FIFO: If SCR[6] = 0 and TLR[3] to TLR[0] not equal to 0000, TX_FIFO_TRIG is not considered. If SCR[6] = 1, TX_FIFO_TRIG is 2 LSB of the trigger level (1 to 63 on 6 bits) with a granularity of 1. If SCR[6] = 0 and TLR[3] to TLR[0] = 0000, then: 0h = 8 characters 1h = 16 characters 2h = 32 characters 3h = 56 characters
3	DMA_MODE	W	0h	Can be changed only when the baud clock is not running (DLL and DLH cleared to 0). If SCR[0] = 0, this register is considered. 0h = DMA_MODE 0 (No DMA). 1h = DMA_MODE 1 (UART_NDMA_REQ[0] in TX, UART_NDMA_REQ[1] in RX).
2	TX_FIFO_CLEAR	W	0h	TX_FIFO_CLEAR. 0h = No change. 1h = Clears the transmit FIFO and resets its counter logic to 0. Returns to 0 after clearing FIFO.
1	RX_FIFO_CLEAR	W	0h	RX_FIFO_CLEAR. 0h = No change. 1h = Clears the receive FIFO and resets its counter logic to 0. Returns to 0 after clearing FIFO.
0	FIFO_EN	W	0h	Can be changed only when the baud clock is not running (DLL and DLH cleared to 0). 0h = Disables the transmit and receive FIFOs. The transmit and receive holding registers are 1-byte FIFOs. 1h = Enables the transmit and receive FIFOs. The transmit and receive holding registers are 64-byte FIFOs.

19.5.1.12 IIR_IRDA Register (offset = 8h) [reset = 0h]

IIR_IRDA is shown in Figure 19-45 and described in Table 19-41.

Figure 19-45. IIR_IRDA Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
EOF_IT	LINE_STS_IT	TX_STATUS_IT	STS_FIFO_IT	RX_OE_IT	RX_FIFO_LAST_BYTE_IT	THR_IT	RHR_IT
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-41. IIR_IRDA Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7	EOF_IT	R	0h	EOF_IT 0h = Received EOF interrupt inactive. 1h = Received EOF interrupt active.
6	LINE_STS_IT	R	0h	LINE_STS_IT 0h = Receiver line status interrupt inactive. 1h = Receiver line status interrupt active.
5	TX_STATUS_IT	R	0h	TX_STATUS_IT 0h = TX status interrupt inactive. 1h = TX status interrupt active.
4	STS_FIFO_IT	R	0h	STS_FIFO_IT 0h = Status FIFO trigger level interrupt inactive. 1h = Status FIFO trigger level interrupt active.
3	RX_OE_IT	R	0h	RX_OE_IT 0h = RX overrun interrupt inactive. 1h = RX overrun interrupt active.
2	RX_FIFO_LAST_BYTE_IT	R	0h	RX_FIFO_LAST_BYTE_IT 0h = Last byte of frame in RX FIFO interrupt inactive. 1h = Last byte of frame in RX FIFO interrupt active.
1	THR_IT	R	0h	THR_IT 0h = THR interrupt inactive. 1h = THR interrupt active.
0	RHR_IT	R	0h	RHR_IT 0h = RHR interrupt inactive. 1h = RHR interrupt active.

19.5.1.13 LCR Register (offset = Ch) [reset = 0h]

LCR is shown in Figure 19-46 and described in Table 19-42.

The line control register (LCR) is selected with a bit register setting of LCR[7] = 0, LCR[7] not equal to BFh, or LCR[7] = BFh. As soon as LCR[6] is set to 1, the TX line is forced to 0 and remains in this state as long as LCR[6] = 1.

Figure 19-46. LCR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
DIV_EN	BREAK_EN	PARITY_TYPE2	PARITY_TYPE1	PARITY_EN	NB_STOP	CHAR_LENGTH	
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-42. LCR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7	DIV_EN	R/W	0h	Divisor latch enable. 0h = Normal operating condition. 1h = Divisor latch enable. Allows access to DLL and DLH.
6	BREAK_EN	R/W	0h	Break control bit. Note: When LCR[6] is set to 1, the TX line is forced to 0 and remains in this state as long as LCR[6] = 1. 0h = Normal operating condition. 1h = Forces the transmitter output to go low to alert the communication terminal.
5	PARITY_TYPE2	R/W	0h	If LCR[3] = 1, then: 0h = If LCR[5] = 0, LCR[4] selects the forced parity format. 1h = 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	R/W	0h	If LCR[3] = 1, then: 0h = Odd parity is generated. 1h = Even parity is generated.
3	PARITY_EN	R/W	0h	Parity bit. 0h = No parity. 1h = A parity bit is generated during transmission, and the receiver checks for received parity.
2	NB_STOP	R/W	0h	Specifies the number of stop bits. 0h = 1 stop bit (word length = 5, 6, 7, 8). 1h = 1.5 stop bits (word length = 5) or 2 stop bits (word length = 6, 7, 8).
1-0	CHAR_LENGTH	R/W	0h	Specifies the word length to be transmitted or received. 0h = 5 bits 1h = 6 bits 2h = 7 bits 3h = 8 bit

19.5.1.14 MCR Register (offset = 10h) [reset = 0h]

MCR is shown in Figure 19-47 and described in Table 19-43.

The modem control register (MCR) is selected with a register bit setting of LCR[7] = 0 or LCR[7] not equal to BFh. MCR[7:5] can only be written when EFR[4] = 1. Bits 3-0 control the interface with the modem, data set, or peripheral device that is emulating the modem.

Figure 19-47. MCR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED	TCRTLR	XONEN	LOOPBACKEN	CDSTSCH	RISTSCH	RTS	DTR
R-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-43. MCR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-7	RESERVED	R	0h	
6	TCRTLR	R/W	0h	Can be written only when EFR[4] = 1. 0h = No action. 1h = Enables access to the TCR and TLR registers.
5	XONEN	R/W	0h	Can be written only when EFR[4] = 1. 0h = Disable XON any function. 1h = Enable XON any function.
4	LOOPBACKEN	R/W	0h	Loopback mode enable. 0h = Normal operating mode. 1h = 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	CDSTSCH	R/W	0h	CDSTSCH. 0h = In loopback mode, forces DCD (active-low) input high and IRQ outputs to INACTIVE state. 1h = In loopback mode, forces DCD (active-low) input low and IRQ outputs to INACTIVE state.
2	RISTSCH	R/W	0h	RISTSCH. 0h = In loopback mode, forces RI (active-low) input inactive (high). 1h = In loopback mode, forces RI (active-low) input active (low).
1	RTS	R/W	0h	In loopback mode, controls MSR[4]. If auto-RTS is enabled, the RTS (active-low) output is controlled by hardware flow control. 0h = Force RTS (active-low) output to inactive (high). 1h = Force RTS (active-low) output to active (low).
0	DTR	R/W	0h	DTR. 0h = Force DTR (active-low) output (used in loopback mode) to inactive (high). 1h = Force DTR (active-low) output (used in loopback mode) to active (low).

19.5.1.15 XON1_ADDR1 Register (offset = 10h) [reset = 0h]

XON1_ADDR1 is shown in Figure 19-48 and described in Table 19-44.

The XON1/ADDR1 registers are selected with a register bit setting of LCR[7] = BFh. In UART mode, XON1 character; in IrDA mode, ADDR1 address 1.

Figure 19-48. XON1_ADDR1 Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
XONWORD1							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-44. XON1_ADDR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7-0	XONWORD1	R/W	0h	Stores the 8 bit XON1 character in UART modes and ADDR1 address 1 in IrDA modes.

19.5.1.16 XON2_ADDR2 Register (offset = 14h) [reset = 0h]

XON2_ADDR2 is shown in Figure 19-49 and described in Table 19-45.

The XON2/ADDR2 registers are selected with a register bit setting of LCR[7] = BFh. In UART mode, XON2 character; in IrDA mode, ADDR2 address 2.

Figure 19-49. XON2_ADDR2 Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
XONWORD2							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-45. XON2_ADDR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7-0	XONWORD2	R/W	0h	Stores the 8 bit XON2 character in UART modes and ADDR2 address 2 in IrDA modes.

19.5.1.17 LSR_CIR Register (offset = 14h) [reset = 81h]

LSR_CIR is shown in Figure 19-50 and described in Table 19-46.

The following line status register (LSR) description is for CIR mode. The CIR LSR is selected with a register bit setting of LCR[7] = 0 or LCR[7] not equal to BFh.

Figure 19-50. LSR_CIR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
THREMPTY	RESERVED	RXSTOP	RESERVED				RXFIFOE
R-1h	R-0h	R-0h	R-0h				R-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-46. LSR_CIR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7	THREMPTY	R	1h	THREMPTY. 0h = Transmit holding register (TX FIFO) is not empty. 1h = Transmit hold register (TX FIFO) is empty. The transmission is not necessarily completed.
6	RESERVED	R	0h	
5	RXSTOP	R	0h	The RXSTOP is generated based on the value set in the BOF Length register (EBLR). 0h = Reception is on going or waiting for a new frame. 1h = Reception is completed. It is cleared on a single read of the LSR register.
4-1	RESERVED	R	0h	
0	RXFIFOE	R	1h	RXFIFOE. 0h = At least one data character in the RX FIFO. 1h = No data in the receive FIFO.

19.5.1.18 LSR_IRDA Register (offset = 14h) [reset = A3h]

LSR_IRDA is shown in Figure 19-51 and described in Table 19-47.

The following line status register (LSR) description is for IrDA mode. The IrDA LSR is selected with a register bit setting of LCR[7] = 0 or LCR[7] not equal to BFh. When the IrDA line status register (LSR) is read, LSR[4:2] reflect the error bits (FL, CRC, ABORT) of the frame at the top of the status FIFO (next frame status to be read).

Figure 19-51. LSR_IRDA Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
THR_EMPTY	STS_FIFO_FULL	RX_LAST_BYTE	FRAME_TOO_LONG	ABORT	CRC	STS_FIFO_E	RX_FIFO_E
R-1h	R-0h	R-1h	R-0h	R-0h	R-0h	R-1h	R-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-47. LSR_IRDA Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7	THR_EMPTY	R	1h	THR_EMPTY. 0h = Transmit holding register (TX FIFO) is not empty. 1h = Transmit hold register (TX FIFO) is empty. The transmission is not necessarily completed.
6	STS_FIFO_FULL	R	0h	STS_FIFO_FULL. 0h = Status FIFO is not full. 1h = Status FIFO is full.
5	RX_LAST_BYTE	R	1h	RX_LAST_BYTE. 0h = The RX FIFO (RHR) does not contain the last byte of the frame to be read. 1h = The RX FIFO (RHR) contains the last byte of the frame to be read. This bit is set to 1 only when the last byte of a frame is available to be read. It is used to determine the frame boundary. It is cleared on a single read of the LSR register.
4	FRAME_TOO_LONG	R	0h	FRAME_TOO_LONG. 0h = No frame-too-long error in frame. 1h = Frame-too-long error 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) is received. When this error is detected, current frame reception is terminated. Reception is stopped until the next START flag is detected.
3	ABORT	R	0h	ABORT. 0h = No abort pattern error in frame. 1h(Read) = Abort pattern received. SIR and MIabort pattern. FIR: Illegal symbol.
2	CRC	R	0h	CRC. 0h = No CRC error in frame. 1h = CRC error in the frame at the top of the status FIFO (next character to be read).
1	STS_FIFO_E	R	1h	STS_FIFO_E. 0h = Status FIFO is not empty. 1h = Status FIFO is empty.

Table 19-47. LSR_IRDA Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	RX_FIFO_E	R	1h	RX_FIFO_E. 0h = At least one data character in the RX FIFO. 1h = No data in the receive FIFO.

19.5.1.19 LSR_UART Register (offset = 14h) [reset = 60h]

LSR_UART is shown in Figure 19-52 and described in Table 19-48.

The following line status register (LSR) description is for UART mode. The UART LSR is selected with a register bit setting of LCR[7] = 0 or LCR[7] not equal to BFh. When the UART line status register (LSR) is read, LSR[4:2] reflect 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 errors in a character. Reading RHR updates BI, FE, and PE. 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 RX FIFO. 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 set.

Figure 19-52. LSR_UART Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RXFIFOSTS	TXSRE	TXFIFOE	RXBI	RXFE	RXPE	RXOE	RXFIFOE
R-0h	R-1h	R-1h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-48. LSR_UART Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7	RXFIFOSTS	R	0h	RXFIFOSTS. 0h = Normal operation. 1h = At least one parity error, framing error, or break indication in the RX FIFO. Bit 7 is cleared when no errors are present in the RX FIFO.
6	TXSRE	R	1h	TXSRE. 0h = Transmitter hold (TX FIFO) and shift registers are not empty. 1h = Transmitter hold (TX FIFO) and shift registers are empty.
5	TXFIFOE	R	1h	TXFIFOE. 0h = Transmit hold register (TX FIFO) is not empty. 1h = Transmit hold register (TX FIFO) is empty. The transmission is not necessarily completed.
4	RXBI	R	0h	RXBI. 0h = No break condition. 1h = A break was detected while the data being read from the RX FIFO was being received (RX input was low for one character + 1 bit time frame).
3	RXFE	R	0h	RXFE. 0h = No framing error in data being read from RX FIFO. 1h = Framing error occurred in data being read from RX FIFO (received data did not have a valid stop bit).
2	RXPE	R	0h	RXPE. 0h = No parity error in data being read from RX FIFO. 1h = Parity error in data being read from RX FIFO.
1	RXOE	R	0h	RXOE. 0h = No overrun error. 1h = Overrun error occurred. Set when the character held in the receive shift register is not transferred to the RX FIFO. This case occurs only when receive FIFO is full.

Table 19-48. LSR_UART Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	RXFIFOE	R	0h	RXFIFOE. 0h = No data in the receive FIFO. 1h = At least one data character in the RX FIFO.

19.5.1.20 TCR Register (offset = 18h) [reset = 0h]

TCR is shown in Figure 19-53 and described in Table 19-49.

The transmission control register (TCR) is selected with a register bit setting of LCR[7] = 0, LCR[7] not equal to BFh, or LCR[7] = BFh. The TCR is accessible only when EFR[4] = 1 and MCR[6] = 1. The transmission control register (TCR) stores the receive FIFO threshold levels to start/stop transmission during hardware flow control. Trigger levels from 0-60 bytes are available with a granularity of 4. Trigger level = 4 x [4-bit register value]. You must ensure that TCR[3:0] > TCR[7:4], whenever auto-RTS or software flow control is enabled to avoid a misoperation of the device. In FIFO interrupt mode with flow control, you have to also ensure that the trigger level to HALT transmission is greater or equal to 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 the DMA request is sent each time a byte is received.

Figure 19-53. TCR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RXFIFOTRIGSTART				RXFIFOTRIGHALT			
R/W-0h				R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-49. TCR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7-4	RXFIFOTRIGSTART	R/W	0h	RX FIFO trigger level to RESTORE transmission (0 to 60).
3-0	RXFIFOTRIGHALT	R/W	0h	RX FIFO trigger level to HALT transmission (0 to 60).

19.5.1.21 MSR Register (offset = 18h) [reset = 0h]

MSR is shown in Figure 19-54 and described in Table 19-50.

The modem status register (MSR) is selected with a register bit setting of LCR[7] = 0 or LCR[7] not equal to BFh. 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 Local Host. It also indicates when a control input from the modem changes state.

Figure 19-54. MSR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
NCD_STS	NRI_STS	NDSR_STS	NCTS_STS	DCD_STS	RI_STS	DSR_STS	CTS_STS
R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 19-50. MSR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7	NCD_STS	R	0h	This bit is the complement of the DCD (active-low) input. In loopback mode, it is equivalent to MCR[3].
6	NRI_STS	R	0h	This bit is the complement of the RI (active-low) input. In loopback mode, it is equivalent to MCR[2].
5	NDSR_STS	R	0h	This bit is the complement of the DSR (active-low) input. In loopback mode, it is equivalent to MCR[0].
4	NCTS_STS	R	0h	This bit is the complement of the CTS (active-low) input. In loopback mode, it is equivalent to MCR[1].
3	DCD_STS	R	0h	DCD_STS. 0h = No change. 1h = Indicates that DCD (active-low) input (or MCR[3] in loopback mode) has changed. Cleared on a read.
2	RI_STS	R	0h	RI_STS. 0h = No change. 1h = Indicates that RI (active-low) input (or MCR[2] in loopback mode) changed state from low to high. Cleared on a read.
1	DSR_STS	R	0h	DSR_STS. 0h = No change. 1h = Indicates that DSR (active-low) input (or MCR[0] in loopback mode) changed state. Cleared on a read.
0	CTS_STS	R	0h	CTS_STS. 0h = No change. 1h = Indicates that CTS (active-low) input (or MCR[1] in loopback mode) changed state. Cleared on a read.

19.5.1.22 XOFF1 Register (offset = 18h) [reset = 0h]

XOFF1 is shown in Figure 19-55 and described in Table 19-51.

The XOFF1 register is selected with a register bit setting of LCR[7] = BFh. In UART mode, XOFF1 character.

Figure 19-55. XOFF1 Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
XOFFWORD1							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-51. XOFF1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7-0	XOFFWORD1	R/W	0h	Stores the 8 bit XOFF1 character in UART modes.

19.5.1.23 SPR Register (offset = 1Ch) [reset = 0h]

SPR is shown in Figure 19-56 and described in Table 19-52.

The scratchpad register (SPR) is selected with a register bit setting of LCR[7] = 0 or LCR[7] not equal to BFh. The scratchpad register (SPR) is a read/write register that does not control the module. It is a scratchpad register used to hold temporary data.

Figure 19-56. SPR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
SPR_WORD							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-52. SPR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7-0	SPR_WORD	R/W	0h	Scratchpad register.

19.5.1.24 TLR Register (offset = 1Ch) [reset = 0h]

TLR is shown in Figure 19-57 and described in Table 19-53.

The trigger level register is selected with a register bit setting of LCR[7] = 0, LCR[7] not equal to BFh, or LCR[7] = BFh. The TLR is accessible only when EFR[4] = 1 and MCR[6] = 1. This register stores the programmable transmit and receive FIFO trigger levels used for DMA and IRQ generation.

Figure 19-57. TLR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RX_FIFO_TRIG_DMA				TX_FIFO_TRIG_DMA			
R/W-0h				R/W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-53. TLR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7-4	RX_FIFO_TRIG_DMA	R/W	0h	Receive FIFO trigger level. Following is a summary of settings for the RX FIFO trigger level. SCR[7] = 0, and TLR[7] to TLR[4]=0, then: Defined by FCR[7] and FCR[6] (either 8, 16, 56, 60 characters). SCR[7] = 0, and TLR[7] to TLR[4] not equal to 0000, then: Defined by TLR[7] to TLR[4] (from 4 to 60 characters with a granularity of 4 characters). SCR[7] = 1, and TLR[7] to TLR[4] = any value, then: Defined by the concatenated value of TLR[7] to TLR[4] and FCR[7] and FCR[6] (from 1 to 63 characters with a granularity of 1 character). Note: the combination of TLR[7] to TLR[4] = 0000 and FCR[7] and FCR[6] = 00 (all zeros) is not supported (minimum of 1 character is required). All zeros results in unpredictable behavior.
3-0	TX_FIFO_TRIG_DMA	R/W	0h	Transmit FIFO trigger level. Following is a summary of settings for the TX FIFO trigger level. SCR[6] = 0, and TLR[3] to TLR[0] = 0, then: Defined by FCR[5] and FCR[4] (either 8, 16, 32, 56 characters). SCR[6] = 0, and TLR[3] to TLR[0] not equal to 0000, then: Defined by TLR[3] to TLR[0] (from 4 to 60 characters with a granularity of 4 characters). SCR[6] = 1, and TLR[3] to TLR[0] = any value, then: Defined by the concatenated value of TLR[3] and TLR[0] and FCR[5] and FCR[4] (from 1 to 63 characters with a granularity of 1 character). Note: the combination of TLR[3] to TLR[0] = 0000 and FCR[5] and FCR[4] = 00 (all zeros) is not supported (minimum of 1 character is required). All zeros results in unpredictable behavior.

19.5.1.25 XOFF2 Register (offset = 1Ch) [reset = 0h]

XOFF2 is shown in Figure 19-58 and described in Table 19-54.

The XOFF2 register is selected with a register bit setting of LCR[7] = BFh. In UART mode, XOFF2 character.

Figure 19-58. XOFF2 Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
XOFFWORD2							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-54. XOFF2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7-0	XOFFWORD2	R/W	0h	Stores the 8 bit XOFF2 character in UART modes.

19.5.1.26 MDR1 Register (offset = 20h) [reset = 7h]

MDR1 is shown in Figure 19-59 and described in Table 19-55.

The mode definition register 1 (MDR1) is selected with a register bit setting of LCR[7] = 0, LCR[7] not equal to BFh, or LCR[7] = BFh. The mode of operation is programmed by writing to MDR1[2:0]; therefore, the mode definition register 1 (MDR1) must be programmed on startup after configuration of the configuration registers (DLL, DLH, and LCR). The value of MDR1[2:0] must not be changed again during normal operation. If the module is disabled by setting the MODESELECT field to 7h, interrupt requests can still be generated unless disabled through the interrupt enable register (IER). In this case, UART mode interrupts are visible. Reading the interrupt identification register (IIR) shows the UART mode interrupt flags.

Figure 19-59. MDR1 Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
FRAMEENDMODE	SIPMODE	SCT	SETTXIR	IRSLEEP	MODESELECT		
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-7h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-55. MDR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7	FRAMEENDMODE	R/W	0h	IrDA mode only. 0h = Frame-length method. 1h = Set EOT bit method.
6	SIPMODE	R/W	0h	MIR/FIR modes only. 0h = Manual SIP mode: SIP is generated with the control of ACREG[3]. 1h = Automatic SIP mode: SIP is generated after each transmission.
5	SCT	R/W	0h	Store and control the transmission. 0h = Starts the infrared transmission when a value is written to the THR register. 1h = Starts the infrared transmission with the control of ACREG[2]. Note: Before starting any transmission, there must be no reception ongoing.
4	SETTXIR	R/W	0h	Used to configure the infrared transceiver. 0h = If MDR2[7] = 0, no action; if MDR2[7] = 1, TXIR pin output is forced low. 1h = TXIR pin output is forced high (not dependant of MDR2[7] value).
3	IRSLEEP	R/W	0h	IrDA/CIR sleep mode. 0h = IrDA/CIR sleep mode disabled. 1h = IrDA/CIR sleep mode enabled.
2-0	MODESELECT	R/W	7h	UART/IrDA/CIR mode selection. 0h = UART 16x mode. 1h = SIR mode. 2h = UART 16x auto-baud. 3h = UART 13x mode. 4h = MIR mode. 5h = FIR mode. 6h = CIR mode. 7h = Disable (default state).

19.5.1.27 MDR2 Register (offset = 24h) [reset = 0h]

MDR2 is shown in Figure 19-60 and described in Table 19-56.

The mode definition register 2 (MDR2) is selected with a register bit setting of LCR[7] = 0, LCR[7] not equal to BFh, or LCR[7] = BFh. The MDR2[0] bit describes the status of the TX status interrupt in IIR[5]. The IRTXUNDERRUN bit must be read after a TX status interrupt occurs. The MDR2[2:1] bits set the trigger level for the frame status FIFO (8 entries) and must be programmed before the mode is programmed in MDR1[2:0]. The MDR2[6] bit gives the flexibility to invert the RX pin inside the UART module to ensure that the protocol at the input of the transceiver module has the same polarity at module level. By default, the RX pin is inverted because most of transceiver invert the IR receive pin.

Figure 19-60. MDR2 Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
SETTXIRALT	IRRXINVERT	CIRPULSEMODE		UARTPULSE	STSFIFOTRIG		IRTXUNDERRUN
R/W-0h	R/W-0h	R/W-0h		R/W-0h	R/W-0h		R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-56. MDR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7	SETTXIRALT	R/W	0h	Provides alternate functionality for MDR1[4]. 0h = Normal mode 1h = Alternate mode for SETTXIR
6	IRRXINVERT	R/W	0h	Only for IR mode (IrDA and CIR). Invert RX pin in the module before the voting or sampling system logic of the infrared block. This does not affect the RX path in UART modem modes. 0h = Inversion is performed. 1h = No inversion is performed.
5-4	CIRPULSEMODE	R/W	0h	CIR pulse modulation definition. Defines high level of the pulse width associated with a digit: 0h = Pulse width of 3 from 12 cycles. 1h = Pulse width of 4 from 12 cycles. 2h = Pulse width of 5 from 12 cycles. 3h = Pulse width of 6 from 12 cycles.
3	UARTPULSE	R/W	0h	UART mode only. Used to allow pulse shaping in UART mode. 0h = Normal UART mode. 1h = UART mode with pulse shaping.
2-1	STSFIFOTRIG	R/W	0h	Only for IrDA mode. Frame status FIFO threshold select: 0h = 1 entry 1h = 4 entries 2h = 7 entries 3h = 8 entries
0	IRTXUNDERRUN	R	0h	IrDA transmission status interrupt. When the TX status interrupt (IIR[5]) occurs, the meaning of the interrupt is: 0h = The last bit of the frame was transmitted successfully without error. 1h = An underrun occurred. The last bit of the frame was transmitted but with an underrun error. The bit is reset to 0 when the RESUME register is read.

19.5.1.28 TXFLL Register (offset = 28h) [reset = 0h]

TXFLL is shown in Figure 19-61 and described in Table 19-57.

The transmit frame length low register (TXFLL) is selected with a register bit setting of LCR[7] = 0, LCR[7] not equal to BFh, or LCR[7] = BFh. The transmit frame length low register (TXFLL) and the TXFLH register hold the 13-bit transmit frame length (expressed in bytes). TXFLL holds the LSBs and TXFLH holds the MSBs. The frame length value is used if the frame length method of frame closing is used.

Figure 19-61. TXFLL Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
TXFLL							
W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-57. TXFLL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7-0	TXFLL	W	0h	LSB register used to specify the frame length.

19.5.1.29 SFLSR Register (offset = 28h) [reset = 0h]

SFLSR is shown in Figure 19-62 and described in Table 19-58.

The status FIFO line status register (SFLSR) is selected with a register bit setting of LCR[7] = 0, LCR[7] not equal to BFh, or LCR[7] = BFh. Reading the status FIFO line status register (SFLSR) effectively 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). Top of RX FIFO = Next frame to be read from RX FIFO.

Figure 19-62. SFLSR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			OE_ERROR	FRAME_TOO_LONG_ERROR	ABORT_DETECT	CRC_ERROR	RESERVED
R-0h			R-0h	R-0h	R-0h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-58. SFLSR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-5	RESERVED	R	0h	
4	OE_ERROR	R	0h	OE_ERROR. 0h = No error 1h = Overrun error in RX FIFO when frame at top of RX FIFO was received.
3	FRAME_TOO_LONG_ERROR	R	0h	FRAME_TOO_LONG_ERROR. 0h = No error 1h = Frame-length too long error in frame at top of RX FIFO.
2	ABORT_DETECT	R	0h	ABORT_DETECT. 0h = No error 1h = Abort pattern detected in frame at top of RX FIFO.
1	CRC_ERROR	R	0h	CRC_ERROR. 0h = No error 1h = CRC error in frame at top of RX FIFO.
0	RESERVED	R	0h	

19.5.1.30 RESUME Register (offset = 2Ch) [reset = 0h]

RESUME is shown in Figure 19-63 and described in Table 19-59.

The RESUME register is selected with a register bit setting of LCR[7] = 0, LCR[7] not equal to BFh, or LCR[7] = BFh. The RESUME register 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 always reads as 00.

Figure 19-63. RESUME Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESUME							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -*n* = value after reset

Table 19-59. RESUME Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7-0	RESUME	R	0h	Dummy read to restart the TX or RX, value 0 to FFh.

19.5.1.31 TXFLH Register (offset = 2Ch) [reset = 0h]

TXFLH is shown in Figure 19-64 and described in Table 19-60.

The transmit frame length high register (TXFLH) is selected with a register bit setting of LCR[7] = 0, LCR[7] not equal to BFh, or LCR[7] = BFh. The transmit frame length high register (TXFLH) and the TXFLL register hold the 13-bit transmit frame length (expressed in bytes). TXFLL holds the LSBs and TXFLH holds the MSBs. The frame length value is used if the frame length method of frame closing is used.

Figure 19-64. TXFLH Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			TXFLH				
R-0h			W-0h				

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-60. TXFLH Register Field Descriptions

Bit	Field	Type	Reset	Description
15-5	RESERVED	R	0h	
4-0	TXFLH	W	0h	MSB register used to specify the frame length, value 0 to 1Fh.

19.5.1.32 RXFLL Register (offset = 30h) [reset = 0h]

RXFLL is shown in Figure 19-65 and described in Table 19-61.

The received frame length low register (RXFLL) is selected with a register bit setting of LCR[7] = 0, LCR[7] not equal to BFh, or LCR[7] = BFh. The received frame length low register (RXFLL) and the RXFLH register hold the 12-bit receive maximum frame length. RXFLL holds the LSBs and RXFLH holds the MSBs. If the intended maximum receive frame length is n bytes, program RXFLL and RXFLH to be n + 3 in SIR or MIR modes and n + 6 in FIR mode (+3 and +6 are the result of frame format with CRC and stop flag; two bytes are associated with the FIR stop flag).

Figure 19-65. RXFLL Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RXFLL							
W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-61. RXFLL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7-0	RXFLL	W	0h	LSB register used to specify the frame length in reception, value 0 to FFh.

19.5.1.33 SFREGL Register (offset = 30h) [reset = 0h]

SFREGL is shown in Figure 19-66 and described in Table 19-62.

The status FIFO register low (SFREGL) is selected with a register bit setting of LCR[7] = 0, LCR[7] not equal to BFh, or LCR[7] = BFh. The frame lengths of received frames are written into the status FIFO. This information can be read by reading the status FIFO register low (SFREGL) and the status FIFO register high (SFREGH). These registers do not physically exist. The LSBs are read from SFREGL and the MSBs 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.

Figure 19-66. SFREGL Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
SFREGL							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-62. SFREGL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7-0	SFREGL	R	0h	LSB part of the frame length, value 0 to FFh.

19.5.1.34 SFREGH Register (offset = 34h) [reset = 0h]

SFREGH is shown in Figure 19-67 and described in Table 19-63.

The status FIFO register high (SFREGH) is selected with a register bit setting of LCR[7] = 0, LCR[7] not equal to BFh, or LCR[7] = BFh. The frame lengths of received frames are written into the status FIFO. This information can be read by reading the status FIFO register low (SFREGL) and the status FIFO register high (SFREGH). These registers do not physically exist. The LSBs are read from SFREGL and the MSBs 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.

Figure 19-67. SFREGH Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				SFREGH			
R-0h				R-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-63. SFREGH Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	
3-0	SFREGH	R	0h	MSB part of the frame length, value 0 to Fh.

19.5.1.35 RXFLH Register (offset = 34h) [reset = 0h]

RXFLH is shown in Figure 19-68 and described in Table 19-64.

The received frame length high register (RXFLH) is selected with a register bit setting of LCR[7] = 0, LCR[7] not equal to BFh, or LCR[7] = BFh. The received frame length high register (RXFLH) and the RXFLL register hold the 12-bit receive maximum frame length. RXFLL holds the LSBs and RXFLH holds the MSBs. If the intended maximum receive frame length is n bytes, program RXFLL and RXFLH to be n + 3 in SIR or MIR modes and n + 6 in FIR mode (+3 and +6 are the result of frame format with CRC and stop flag; two bytes are associated with the FIR stop flag).

Figure 19-68. RXFLH Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED				RXFLH			
R-0h				W-0h			

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-64. RXFLH Register Field Descriptions

Bit	Field	Type	Reset	Description
15-4	RESERVED	R	0h	
3-0	RXFLH	W	0h	MSB register used to specify the frame length in reception, value 0 to Fh.

19.5.1.36 BLR Register (offset = 38h) [reset = 40h]

BLR is shown in Figure 19-69 and described in Table 19-65.

The BOF control register (BLR) is selected with a register bit setting of LCR[7] = 0. The BLR[6] bit is used to select whether C0h or FFh start patterns are to be used, when multiple start flags are required in SIR mode. If only one start flag is required, this is always C0h. If n start flags are required, either (n - 1) C0h or (n -1) FFh flags are sent, followed by a single C0h flag (immediately preceding the first data byte).

Figure 19-69. BLR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
STSFIFORESET	XBOFTYPE	RESERVED					
R/W-0h	R/W-1h	R-0h					

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-65. BLR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7	STSFIFORESET	R/W	0h	Status FIFO reset. This bit is self-clearing.
6	XBOFTYPE	R/W	1h	SIR xBOF select. 0h = FFh start pattern is used. 1h = C0h start pattern is used.
5-0	RESERVED	R	0h	

19.5.1.37 UASR Register (offset = 38h) [reset = 0h]

UASR is shown in Figure 19-70 and described in Table 19-66.

The UART autobauding status register (UASR) is selected with a register bit setting of LCR[7] not equal to BFh or LCR[7] = BFh. The UART autobauding status register (UASR) returns the speed, the number of bits by characters, and the type of parity in UART autobauding mode. In autobauding mode, the input frequency of the UART modem must be fixed to 48 MHz. Any other module clock frequency results in incorrect baud rate recognition. This register is used to set up transmission according to characteristics of previous reception, instead of LCR, DLL, and DLH registers when UART is in autobauding 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 7h (reset state), then set to 2h (UART in autobaud mode) or cleared to 0 (UART in standard mode). Usage limitation: Only 7 and 8 bits character (5 and 6 bits not supported). 7 bits character with space parity not supported. Baud rate between 1200 and 115 200 bp/s (10 possibilities).

Figure 19-70. UASR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
PARITYTYPE		BITBYCHAR	SPEED				
R-0h		R-0h	R-0h				

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-66. UASR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7-6	PARITYTYPE	R	0h	Type of the parity in UART autobauding mode. 0h = No parity identified 1h = Parity space 2h = Even parity 3h = Odd parity
5	BITBYCHAR	R	0h	Number of bits by characters. 0h = 7-bit character identified. 1h = 8-bit character identified.
4-0	SPEED	R	0h	Speed. 0h = No speed identified. 1h = 115 200 baud 2h = 57 600 baud 3h = 38 400 baud 4h = 28 800 baud 5h = 19 200 baud 6h = 14 400 baud 7h = 9600 baud 8h = 4800 baud 9h = 2400 baud Ah = 1200 baud Bh = Reserved from Bh to 1Fh.

19.5.1.38 ACREG Register (offset = 3Ch) [reset = 0h]

ACREG is shown in Figure 19-71 and described in Table 19-67.

The auxiliary control register (ACREG) is selected with a register bit setting of LCR[7] = 0. If transmit FIFO is not empty and MDR1[5] = 1, IrDA starts a new transfer with data of previous frame as soon as abort frame has been sent. Therefore, TX FIFO must be reset before sending an abort frame. It is recommended to disable TX FIFO underrun capability by masking corresponding underrun interrupt. When disabling underrun by setting ACREG[4] = 1, unknown data is sent over TX line.

Figure 19-71. ACREG Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
PULSETYPE	SDMOD	DISIRRX	DISTXUNDERRUN	SENDSIP	SCTXEN	ABORTEN	EOTEN
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-67. ACREG Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7	PULSETYPE	R/W	0h	SIR pulse-width select: 0h = 3/16 of baud-rate pulse width 1h = 1.6 microseconds
6	SDMOD	R/W	0h	Primary output used to configure transceivers. Connected to the SD/MODE input pin of IrDA transceivers. 0h = SD pin is set to high. 1h = SD pin is set to low.
5	DISIRRX	R/W	0h	Disable RX input. 0h = Normal operation (RX input automatically disabled during transmit, but enabled outside of transmit operation). 1h = Disables RX input (permanent state; independent of transmit).
4	DISTXUNDERRUN	R/W	0h	Disable TX underrun. 0h = Long stop bits cannot be transmitted. TX underrun is enabled. 1h = Long stop bits can be transmitted.
3	SENDSIP	R/W	0h	MIR/FIR modes only. Send serial infrared interaction pulse (SIP). If this bit is set during an MIR/FIR transmission, the SIP is sent at the end of it. This bit is automatically cleared at the end of the SIP transmission. 0h = No action. 1h = Send SIP pulse.
2	SCTXEN	R/W	0h	Store and control TX start. When MDR1[5] = 1 and the LH writes 1 to this bit, the TX state-machine starts frame transmission. This bit is self-clearing.
1	ABORTEN	R/W	0h	Frame abort. The LH 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.
0	EOTEN	R/W	0h	EOT (end-of-transmission) bit. The LH 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 is automatically cleared when the LH writes to the THR (TX FIFO).

19.5.1.39 SCR Register (offset = 40h) [reset = 0h]

SCR is shown in Figure 19-72 and described in Table 19-68.

The supplementary control register (SCR) is selected with a register bit setting of LCR[7] = 0, LCR[7] not equal to BFh, or LCR[7] = BFh. Bit 4 enables the wake-up interrupt, but this interrupt is not mapped into the IIR register. Therefore, when an interrupt occurs and there is no interrupt pending in the IIR register, the SSR[1] bit must be checked. To clear the wake-up interrupt, bit SCR[4] must be reset to 0.

Figure 19-72. SCR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RXTRIGGRANU1	TXTRIGGRANU1	DSRIT	RXCTSDSRWAKEUPENABLE	TXEMPTYCTLIT	DMAMODE2		DMAMODECTL
R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h	R/W-0h		R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-68. SCR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7	RXTRIGGRANU1	R/W	0h	RXTRIGGRANU1 0h = Disables the granularity of 1 for trigger RX level. 1h = Enables the granularity of 1 for trigger RX level.
6	TXTRIGGRANU1	R/W	0h	TXTRIGGRANU1 0h = Disables the granularity of 1 for trigger TX level. 1h = Enables the granularity of 1 for trigger TX level.
5	DSRIT	R/W	0h	DSRIT 0h = Disables DSR (active-low) interrupt. 1h = Enables DSR (active-low) interrupt.
4	RXCTSDSRWAKEUPENABLE	R/W	0h	RX CTS wake-up enable. 0h = Disables the WAKE UP interrupt and clears SSR[1]. 1h = Waits for a falling edge of RX, CTS (active-low), or DSR (active-low) pins to generate an interrupt.
3	TXEMPTYCTLIT	R/W	0h	TXEMPTYCTLIT 0h = Normal mode for THR interrupt. 1h = THR interrupt is generated when TX FIFO and TX shift register are empty.
2-1	DMAMODE2	R/W	0h	Specifies the DMA mode valid if SCR[0] = 1, then: 0h = DMA mode 0 (no DMA). 1h = DMA mode 1 (UARTnDMAREQ[0] in TX, UARTnDMAREQ[1] in RX) 2h = DMA mode 2 (UARTnDMAREQ[0] in RX) 3h = DMA mode 3 (UARTnDMAREQ[0] in TX)
0	DMAMODECTL	R/W	0h	DMAMODECT 0h = The DMAMODE is set with FCR[3]. 1h = The DMAMODE is set with SCR[2:1].

19.5.1.40 SSR Register (offset = 44h) [reset = 4h]

SSR is shown in Figure 19-73 and described in Table 19-69.

The supplementary status register (SSR) is selected with a register bit setting of LCR[7] = 0, LCR[7] not equal to BFh, or LCR[7] = BFh. Bit 1 is reset only when SCR[4] is reset to 0.

Figure 19-73. SSR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED					DMACOUNTERRST	RXCTSDSRWAKEUPSTS	TXFIFOFULL
R-0h					R/W-1h	R-0h	R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-69. SSR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-3	RESERVED	R	0h	
2	DMACOUNTERRST	R/W	1h	DMACOUNTERRST. 0h = The DMA counter will not be reset, if the corresponding FIFO is reset (via FCR[1] or FCR[2]). 1h = The DMA counter will be reset, if the corresponding FIFO is reset (via FCR[1] or FCR[2]).
1	RXCTSDSRWAKEUPSTS	R	0h	Pin falling edge detection: Reset only when SCR[4] is reset to 0. 0h = No falling-edge event on RX, CTS (active-low), and DSR (active-low). 1h = A falling edge occurred on RX, CTS (active-low), or DSR (active-low).
0	TXFIFOFULL	R	0h	TXFIFOFULL. 0h = TX FIFO is not full. 1h = TX FIFO is full.

19.5.1.41 EBLR Register (offset = 48h) [reset = 0h]

EBLR is shown in Figure 19-74 and described in Table 19-70.

The BOF length register (EBLR) is selected with a register bit setting of LCR[7] = 0. In IrDA SIR operation, the BOF length register (EBLR) specifies the number of BOF + xBOFs to transmit. The value set into this register must consider the BOF character; therefore, to send only 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 XBOFs. In IrDA MIR mode, the BOF length register (EBLR) specifies the number of additional start flags (MIR protocol mandates a minimum of 2 start flags). In CIR mode, the BOF length register (EBLR) specifies the number of consecutive zeros to be received before generating the RXSTOP interrupt (IIR[2]). All the received zeros are stored in the RX FIFO. When the register is cleared to 0, this feature is deactivated and always in reception state, which is disabled by setting the ACREG[5] bit to 1. If the RX_STOP interrupt occurs before a byte boundary, the remaining bits of the last byte are filled with zeros and then passed into the RX FIFO.

Figure 19-74. EBLR Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
EBLR							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-70. EBLR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7-0	EBLR	R/W	0h	IrDA mode: This register allows definition of up to 176 xBOFs, the maximum required by IrDA specification. CIR mode: This register specifies the number of consecutive zeros to be received before generating the RXSTOP interrupt (IIR[2]). 0h = Feature disabled. 1h = Generate RXSTOP interrupt after receiving 1 zero bit. FFh = Generate RXSTOP interrupt after receiving 255 zero bits.

19.5.1.42 MVR Register (offset = 50h) [reset = 0h]

MVR is shown in Figure 19-75 and described in Table 19-71.

The module version register (MVR) is selected with a register bit setting of LCR[7] = 0, LCR[7] not equal to BFh, or LCR[7] = BFh. The reset value is fixed by hardware and corresponds to the RTL revision of this module. A reset has no effect on the value returned.

Figure 19-75. MVR Register

15	14	13	12	11	10	9	8
RESERVED					MAJORREV		
R-0h					R-0h		
7	6	5	4	3	2	1	0
RESERVED		MINORREV_					
R-0h		R-0h					

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-71. MVR Register Field Descriptions

Bit	Field	Type	Reset	Description
15-11	RESERVED	R	0h	
10-8	MAJORREV	R	0h	Major revision number of the module.
7-6	RESERVED	R	0h	
5-0	MINORREV_	R	0h	Minor revision number of the module.

19.5.1.43 SYSC Register (offset = 54h) [reset = 0h]

SYSC is shown in Figure 19-76 and described in Table 19-72.

The system configuration register (SYSC) is selected with a register bit setting of LCR[7] = 0, LCR[7] not equal to BFh, or LCR[7] = BFh. The AUTOIDLE bit controls a power-saving technique to reduce the logic power consumption of the module interface; that is, when the feature is enabled, the interface clock is gated off until the module interface is accessed. When the SOFTRESET bit is set high, it causes a full device reset.

Figure 19-76. SYSC Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			IDLEMODE		ENAWAKEUP	SOFTRESET	AUTOIDLE
R-0h			R/W-0h		R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-72. SYSC Register Field Descriptions

Bit	Field	Type	Reset	Description
15-5	RESERVED	R	0h	
4-3	IDLEMODE	R/W	0h	Power management req/ack control. 0h = Force idle: Idle request is acknowledged unconditionally. 1h = No-idle: Idle request is never acknowledged. 2h = Smart idle: Acknowledgement to an idle request is given based in the internal activity of the module. 3h = Smart idle Wakeup: Acknowledgement to an idle request is given based in the internal activity of the module. The module is allowed to generate wakeup request. Only available on UART0.
2	ENAWAKEUP	R/W	0h	Wakeup control. 0h = Wakeup is disabled. 1h = Wakeup capability is enabled.
1	SOFTRESET	R/W	0h	Software reset. Set this bit to 1 to trigger a module reset. This bit is automatically reset by the hardware. Read returns 0. 0h = Normal mode. 1h = Module is reset.
0	AUTOIDLE	R/W	0h	Internal interface clock-gating strategy. 0h = Clock is running. 1h = Reserved.

19.5.1.44 SYSS Register (offset = 58h) [reset = 0h]

SYSS is shown in Figure 19-77 and described in Table 19-73.

The system status register (SYSS) is selected with a register bit setting of LCR[7] = 0, LCR[7] not equal to BFh, or LCR[7] = BFh.

Figure 19-77. SYSS Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED							RESETDONE
R-0h							R-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-73. SYSS Register Field Descriptions

Bit	Field	Type	Reset	Description
15-1	RESERVED	R	0h	
0	RESETDONE	R	0h	Internal reset monitoring. 0h = Internal module reset is ongoing. 1h = Reset complete.

19.5.1.45 WER Register (offset = 5Ch) [reset = FFh]

WER is shown in Figure 19-78 and described in Table 19-74.

The wake-up enable register (WER) is selected with a register bit setting of LCR[7] = 0, LCR[7] not equal to BFh, or LCR[7] = BFh. The wake-up enable register (WER) is used to mask and unmask a UART event that subsequently notifies the system. An event is any activity in the logic that can cause an interrupt and/or an activity that requires the system to wake up. Even if wakeup is disabled for certain events, if these events are also an interrupt to the UART, the UART still registers the interrupt as such.

Figure 19-78. WER Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
TXWAKEUPEN	RLS__INTERRUPT	RHR__INTERRUPT	RX__ACTIVITY	DCD_ACTIVITY	RI__ACTIVITY	DSR_ACTIVITY	CTS__ACTIVITY
R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h	R/W-1h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-74. WER Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7	TXWAKEUPEN	R/W	1h	Wake-up interrupt. 0h = Event is not allowed to wake up the system. 1h = Event can wake up the system: Event can be: THRIT or TXDMA request and/or TXSATUSIT.
6	RLS__INTERRUPT	R/W	1h	Receiver line status interrupt. 0h = Event is not allowed to wake up the system. 1h = Event can wake up the system.
5	RHR__INTERRUPT	R/W	1h	RHR interrupt. 0h = Event is not allowed to wake up the system. 1h = Event can wake up the system.
4	RX__ACTIVITY	R/W	1h	RX_ACTIVITY. 0h = Event is not allowed to wake up the system. 1h = Event can wake up the system.
3	DCD_ACTIVITY	R/W	1h	DCD_ACTIVITY. 0h = Event is not allowed to wake up the system. 1h = Event can wake up the system.
2	RI__ACTIVITY	R/W	1h	RI_ACTIVITY. 0h = Event is not allowed to wake up the system. 1h = Event can wake up the system.
1	DSR_ACTIVITY	R/W	1h	DSR_ACTIVITY. 0h = Event is not allowed to wake up the system. 1h = Event can wake up the system.
0	CTS__ACTIVITY	R/W	1h	CTS_ACTIVITY. 0h = Event is not allowed to wake up the system. 1h = Event can wake up the system.

19.5.1.46 CFPS Register (offset = 60h) [reset = 69h]

CFPS is shown in Figure 19-79 and described in Table 19-75.

The carrier frequency prescaler register (CFPS) is selected with a register bit setting of LCR[7] = 0, LCR[7] not equal to BFh, or LCR[7] = BFh. Since the consumer IR (CIR) works at modulation rates of 30-56.8 kHz, the 48 MHz clock must be prescaled before the clock can drive the IR logic. The carrier frequency prescaler register (CFPS) sets the divisor rate to give a range to accommodate the remote control requirements in BAUD multiples of 12x. The value of the CFPS at reset is 105 decimal (69h), which equates to a 38.1 kHz output from starting conditions. The 48 MHz carrier is prescaled by the CFPS that is then divided by the 12x BAUD multiple.

Figure 19-79. CFPS Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
CFPS							
R/W-69h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-75. CFPS Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7-0	CFPS	R/W	69h	System clock frequency prescaler at (12x multiple). CFPS = 0 is not supported. Examples for CFPS values follow. Target Frequency (kHz) = 30, CFPS (decimal) = 133, Actual Frequency (kHz) = 30.08. Target Frequency (kHz) = 32.75, CFPS (decimal) = 122, Actual Frequency (kHz) = 32.79. Target Frequency (kHz) = 36, CFPS (decimal) = 111, Actual Frequency (kHz) = 36.04. Target Frequency (kHz) = 36.7, CFPS (decimal) = 109, Actual Frequency (kHz) = 36.69. Target Frequency (kHz) = 38, CFPS (decimal) = 105, Actual Frequency (kHz) = 38.1. Target Frequency (kHz) = 40, CFPS (decimal) = 100, Actual Frequency (kHz) = 40. Target Frequency (kHz) = 56.8, CFPS (decimal) = 70, Actual Frequency (kHz) = 57.14.

19.5.1.47 RXFIFO_LVL Register (offset = 64h) [reset = 0h]

RXFIFO_LVL is shown in Figure 19-80 and described in Table 19-76.

Figure 19-80. RXFIFO_LVL Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RXFIFO_LVL							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-76. RXFIFO_LVL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7-0	RXFIFO_LVL	R	0h	Level of the RX FIFO

19.5.1.48 TXFIFO_LVL Register (offset = 68h) [reset = 0h]

TXFIFO_LVL is shown in Figure 19-81 and described in Table 19-77.

Figure 19-81. TXFIFO_LVL Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
TXFIFO_LVL							
R-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-77. TXFIFO_LVL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7-0	TXFIFO_LVL	R	0h	Level of the TX FIFO

19.5.1.49 IER2 Register (offset = 6Ch) [reset = 0h]

IER2 is shown in Figure 19-82 and described in Table 19-78.

The IER2 enables RX/TX FIFOs empty corresponding interrupts.

Figure 19-82. IER2 Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED						EN_TXFIFO_EMPTY	EN_RXFIFO_EMPTY
R-0h						R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-78. IER2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-2	RESERVED	R	0h	
1	EN_TXFIFO_EMPTY	R/W	0h	EN_TXFIFO_EMPTY. 0h = Disables EN_TXFIFO_EMPTY interrupt. 1h = Enables EN_TXFIFO_EMPTY interrupt.
0	EN_RXFIFO_EMPTY	R/W	0h	Number of bits by characters. 0h = Disables EN_RXFIFO_EMPTY interrupt. 1h = Enables EN_RXFIFO_EMPTY interrupt.

19.5.1.50 ISR2 Register (offset = 70h) [reset = 0h]

ISR2 is shown in Figure 19-83 and described in Table 19-79.

The interrupt status register 2 (ISR2) is selected with a register bit setting of LCR[7] = 0, LCR[7] not equal to BFh, or LCR[7] = BFh. The ISR2 displays the status of RX/TX FIFOs empty corresponding interrupts.

Figure 19-83. ISR2 Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED						TXFIFO_EMPTY_STS	RXFIFO_EMPTY_STS
R-0h						R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-79. ISR2 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-2	RESERVED	R	0h	
1	TXFIFO_EMPTY_STS	R/W	0h	TXFIFO_EMPTY_STS. 0h = TXFIFO_EMPTY interrupt not pending. 1h = TXFIFO_EMPTY interrupt pending.
0	RXFIFO_EMPTY_STS	R/W	0h	RXFIFO_EMPTY_STS. 0h = RXFIFO_EMPTY interrupt not pending. 1h = RXFIFO_EMPTY interrupt pending.

19.5.1.51 FREQ_SEL Register (offset = 74h) [reset = 0h]

FREQ_SEL is shown in Figure 19-84 and described in Table 19-80.

Figure 19-84. FREQ_SEL Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
FREQ_SEL							
R/W-0h							

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-80. FREQ_SEL Register Field Descriptions

Bit	Field	Type	Reset	Description
15-8	RESERVED	R	0h	
7-0	FREQ_SEL	R/W	0h	Sets the sample per bit if non default frequency is used. MDR3[1] must be set to 1 after this value is set. Must be equal or higher then 6.

19.5.1.52 MDR3 Register (offset = 80h) [reset = 0h]

MDR3 is shown in Figure 19-85 and described in Table 19-81.

The mode definition register 3 (MDR3) is selected with a register bit setting of LCR[7] = 0, LCR[7] not equal to BFh, or LCR[7] = BFh. The DISABLE_CIR_RX_DEMOD register bit will force the CIR receiver to bypass demodulation of received data if set. See the CIR Mode Block Components. The NONDEFAULT_FREQ register bit allows the user to set sample per bit by writing it into FREQ_SEL register. Set it if non-default (48 MHz) fclk frequency is used to achieve a less than 2% error rate. Changing this bit (to any value) will automatically disable the device by setting MDR[2:0] to 111 .

Figure 19-85. MDR3 Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED					SET_DMA_TX_THRESHOLD	NONDEFAULT_FREQ	DISABLE_CIR_RX_DEMOD
R-0h					R/W-0h	R/W-0h	R/W-0h

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-81. MDR3 Register Field Descriptions

Bit	Field	Type	Reset	Description
15-3	RESERVED	R	0h	
2	SET_DMA_TX_THRESHOLD	R/W	0h	SET_DMA_TX_THRESHOLD. 0h = Disable use of TX DMA Threshold register. Use 64-TX trigger as DMA threshold. 1h = Enable to set different TX DMA threshold in the TX DMA Threshold register.
1	NONDEFAULT_FREQ	R/W	0h	NONDEFAULT_FREQ. 0h = Disables using NONDEFAULT fclk frequencies. 1h = Enables using NONDEFAULT fclk frequencies (set FREQ_SEL and DLH/DLL).
0	DISABLE_CIR_RX_DEMOD	R/W	0h	DISABLE_CIR_RX_DEMOD. 0h = Enables CIR RX demodulation. 1h = Disables CIR RX demodulation.

19.5.1.53 TX_DMA_THRESHOLD Register (offset = 84h) [reset = 0h]

TX_DMA_THRESHOLD is shown in Figure 19-86 and described in Table 19-82.

The TX DMA threshold register is selected with a register bit setting of LCR[7] = 0, LCR[7] not equal to BFh, or LCR[7] = BFh.

Figure 19-86. TX_DMA_THRESHOLD Register

15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED		TX_DMA_THRESHOLD					
R-0h		R/W-0h					

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; *-n* = value after reset

Table 19-82. TX_DMA_THRESHOLD Register Field Descriptions

Bit	Field	Type	Reset	Description
15-6	RESERVED	R	0h	
5-0	TX_DMA_THRESHOLD	R/W	0h	Used to manually set the TX DMA threshold level. UART_MDR3[2] SET_TX_DMA_THRESHOLD must be 1 and must be value + tx_trigger_level = 64 (TX FIFO size). If not, 64_tx_trigger_level will be used without modifying the value of this register.