Chapter 11

UART

The universal asynchronous receiver-transmitter (UART) module provides an interface for serial communication protocols.

11.1 UART Overview

11.1.1 Purpose of the Peripheral

This interface can be used transfer data between a MSPM0 device and another device with an asynchronous serial communication protocol like LIN (local interconnection network), ISO7816 (Smart card protocol), IrDA (infrared data association), hardware flow control (CTS/RTS) and multiprocessor communications are supported.

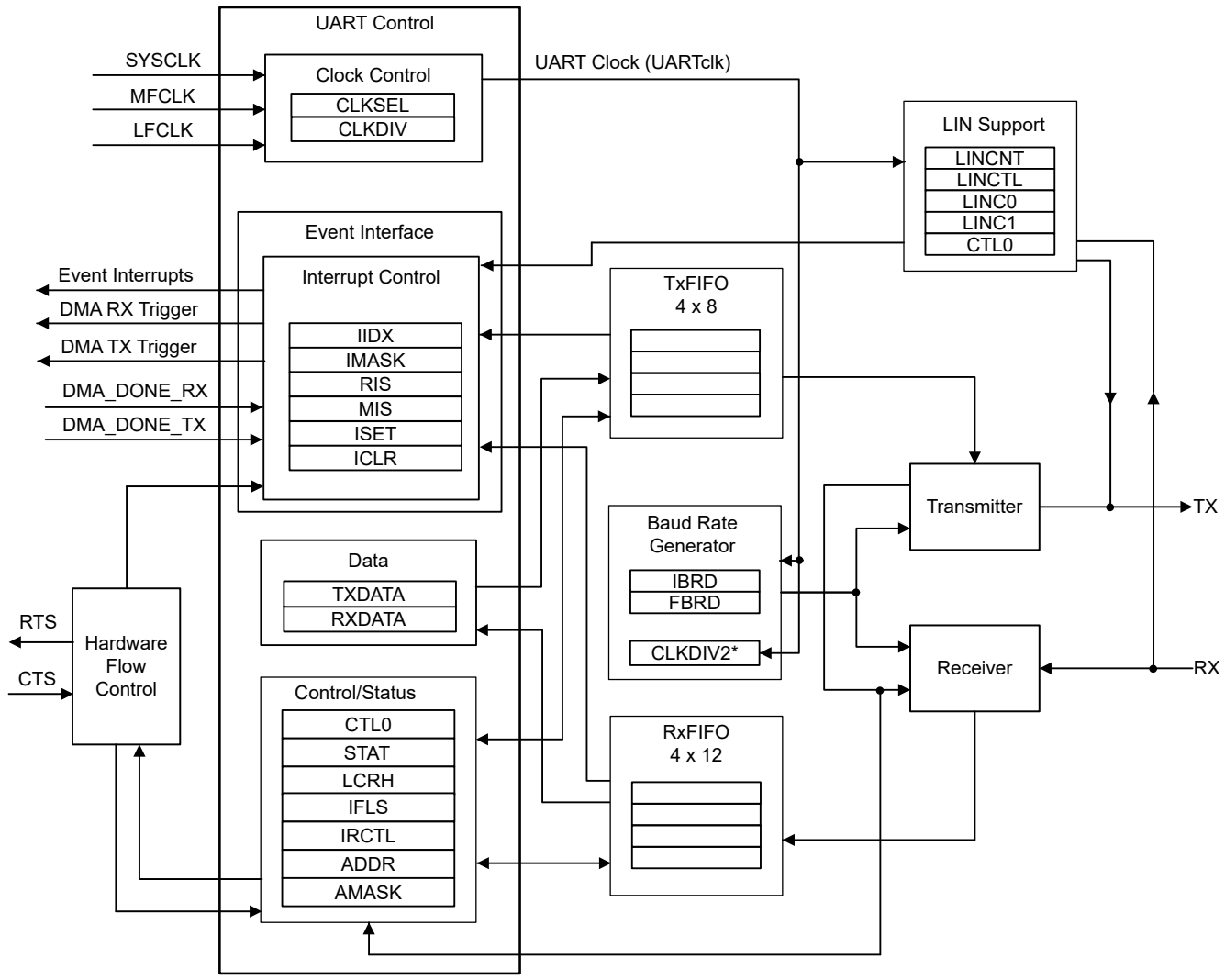
11.1.2 Features

The UART controller includes the following features:

- Fully programmable serial interface
  - 5, 6, 7 or 8 data bits
  - Even, odd, stick, or no-parity bit generation and detection
  - 1 or 2 stop bit generation
  - LSB-first or MSB-first data transmit and receive
  - Line-break detection
  - Glitch filter on the input signals
  - Programmable baud-rate generation with oversampling by 16, 8 or 3
- Separated transmit and receive 4 depth FIFOs reduce CPU interrupt service loading
- Supports DMA data transfer
- Standard FIFO-level and End-of-Transmission interrupts
- Active in all low-power mode including stop and standby mode
- Support for waking up SYSOSC via an asynchronous fast clock request upon start bit detection when operating in low power modes (supports up to 19200B rates when using SYSOSC in FCL mode (1% accuracy)
- Support loopback mode operation
- Support hardware flow control
- Support 9-bit multi-drop configuration
- Protocols supported:
  - Local Interconnect Network (LIN) support
  - DALI
  - IrDA
  - ISO7816 Smart card
  - RS485
  - Manchester coding
  - Idle-Line Multiprocessor

There are two types of UART instances: Extend and Main. The following table compares the features of UART Extend and UART Main.

11.1.3 Functional Block Diagram



*CLKDIV2 is for IrDA mode only

Figure 11-1. UART Functional Block Diagram

11.2 UART Operation

This section describes the operation of the UART peripheral.

11.2.1 Clock Control

The UART internal functional clock is selected and divided from the functional clock of the IP.

- Use UARTx.CLKSEL register to select the source of the UART functional clock.
  - BUSSCLK: the current bus clock is selected as the source for UART. The current bus clock depends on power domain. If the UART instance is in power domain 1 (PD1) refer to MCLK, if the UART instance is in power domain 0 (PD0) refer to ULPCLK.
  - MFCLK: MFCLK is selected as the source for UART, refer to MFCLK.
  - LFCLK: LFCLK is selected as the source for UART, refer to LFCLK
- Use UARTx.CLKDIV register to select the divide ratio of the UART function clock,options are from divide by 1 to 8. For UART Extend, there is a CLKDIV2 register to further divide the UART function clock to support the IrDA mode. When using IrDA mode, CLKDIV2.RATIO must be set to 1h for proper IrDA clocking.

The selected source clock is always available and the frequency depends on the power mode, for more information, see the Clock Module (CKM) section. After enabling the UART module by setting the ENABLE bit, the module will be ready to start receiving and transmitting data.

11.2.2 Signal Descriptions

UART communications require two pins: Receive Data (RX) and Transmit Data (TX):

- RX (Receive Data): RX is the serial data input. Glitch filter and oversampling techniques are used on the receive signal to ensure accurate incoming data.
- TX (Transmit Data): TX is the serial data output. When the transmitter is enabled and no data needs to be transmitted, the TX pin will be held high. In some bidirectional protocols like ISO7816 Smart card, this pin is also used to receive data.

In hardware flow control mode, the following pins are also used:

- CTS (Clear To Send): when driven high by external signal, this signal blocks the data transmission at the end of the current transfer.
- RTS (Request To Send): when low, this signal indicates that the UART is ready to receive data.

11.2.3 General Architecture and Protocol

UART transmits and receives characters at a bit rate that is asynchronous to another device. Timing for each character is based on the selected baud-rate. The transmit and receive functions use the same baud-rate frequency.

In general, control registers should only be programmed when the UART is disabled (ENABLE bit in the UARTx.CTL0 register is cleared). If the UART is operating and gets disabled during transmit or receive operation, the current transaction gets completed before the UART stops. The baud-rate divisor registers (UARTx.IBRD and UARTx.FBRD) can be modified without disabling the UART.

11.2.3.1 Transmit Receive Logic

The transmit logic performs parallel-to-serial conversion on the data read from the transmit FIFO. The control logic outputs the serial bit stream beginning with a start bit and followed by the data bits (LSB first), parity bit, and the stop bits according to the programmed configuration.

The receive logic performs serial-to-parallel conversion on the received bit stream after a valid start pulse has been detected. Overrun, parity, frame error checking, and line-break detection are also performed, and their status accompanies the data that is written to the receive FIFO, this is the reason for 12-bit receive FIFO.

UARTx.CTL0.ENABLE bit is used to enable and disable the UART module, UARTx.CTL0.TXE and RXE bits are used to enable the transmit and receive mode, UARTx.LCRH.WLEN bit is used to configure the number of data bits transmitted or received in a frame, UARTx.LCRH.PEN is used to enable parity and UARTx.LCRH.STP2 is used to send two stop bits.

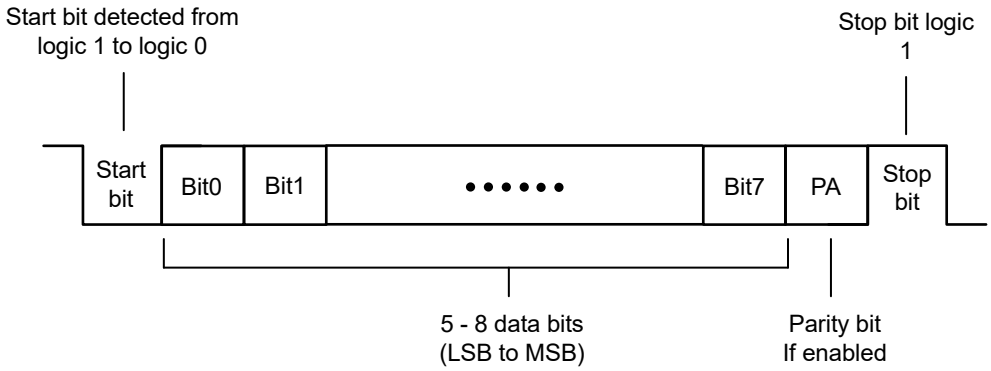


Figure 11-2. UART Character Frame

11.2.3.2 Bit Sampling

By default, UARTx.CTL0.HSE is set to 0 and 16 oversampling is selected and receiving bits are expected to have the length of 16 UART clock UARTclk cycles and is sampled on the 8th UARTclk cycle.

Setting the UARTx.CTL0.HSE bit to 1 selects the 8 oversampling where the receiving bits are expected to have the length of 8 UART clock UARTclk cycles and is sampled on the 4th UARTclk cycle.

Setting the UARTx.CTL0.HSE bit to 2 selects the 3 oversampling, the receiving bits are expected to have the length of 3 UART clock UARTclk cycles and are sampled on the 2nd UARTclk cycle.

The aforementioned scenarios assume an IBRD =1 and FBRD =0.

Depending on the application:

- Select oversampling by 3 or 8 to achieve higher speed with UARTclk/8 or UARTclk/3. In this case the receiver tolerance to clock deviation is reduced.
- Select oversampling by 16 to increase the tolerance of the receiver to clock deviations. The maximum speed is limited to UARTclk/16.

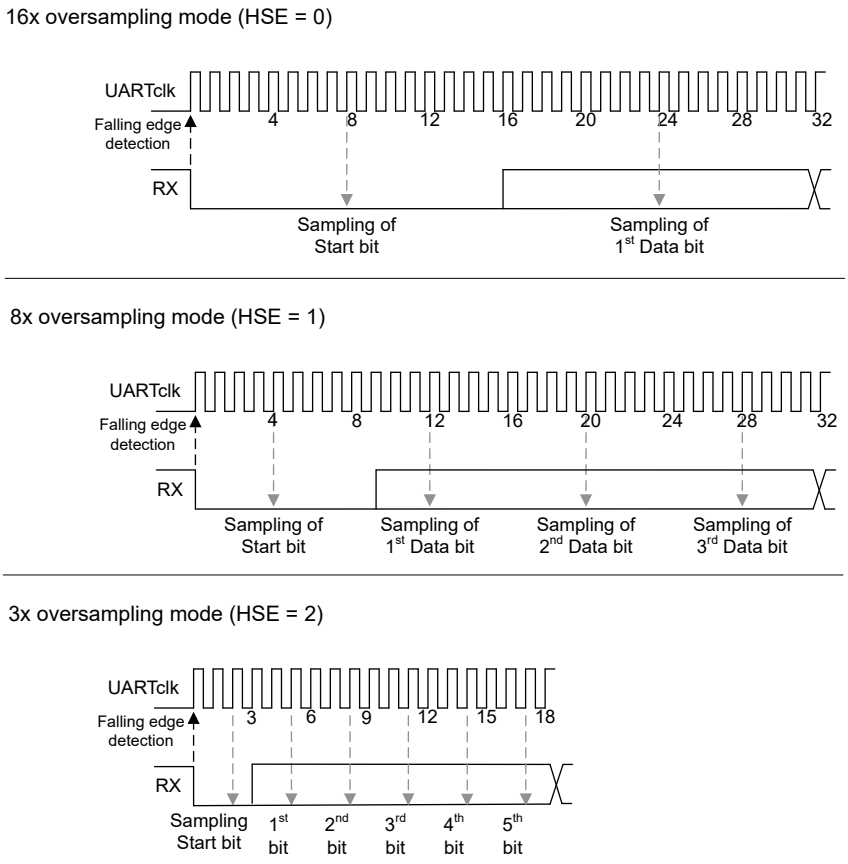


Figure 11-3. UART Oversampling mode

11.2.3.3 Majority Voting Feature

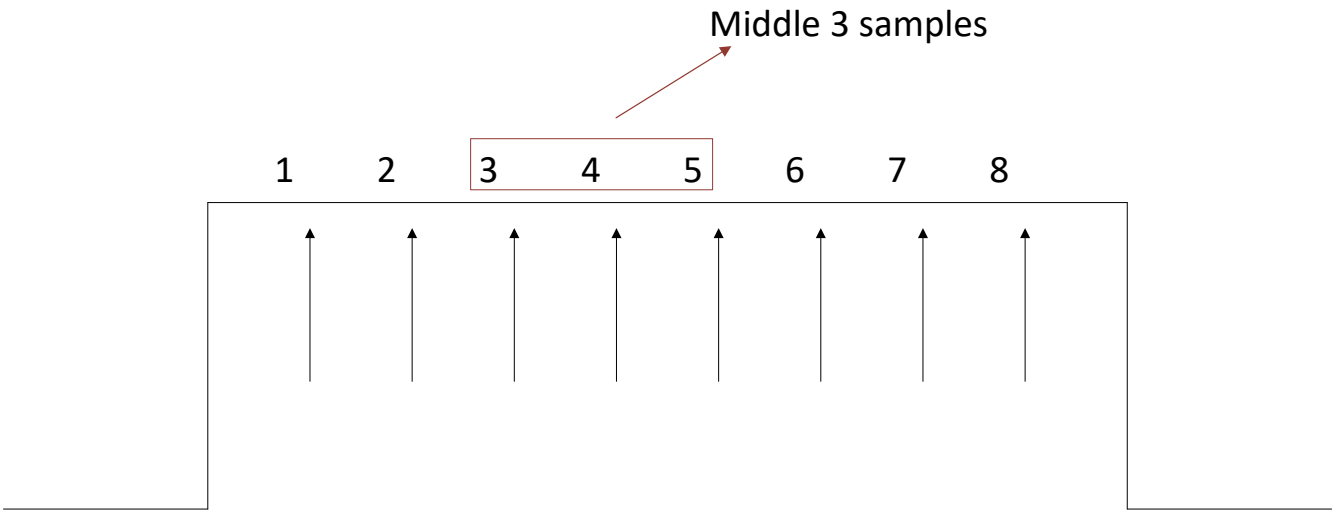


Figure 11-4. Majority voting for 8 oversampling

The majority voting feature of the UART provides noise immunity by sampling each bit 3 times in the center of the bit period. For example, in the case when the UARTx.CTL0.HSE is set to 0 with 16 oversampling; the 7th, 8th and 9th bit are sampled and the majority value is considered as final value to be sampled. When the UARTx.CTL0.HSE is set to 1 with 8 oversampling; then the 3rd ,4th and 5th bits are sampled and majority value is considered as final value to be sampled. The oversampling is only applicable for 16 and 8 oversampling. When the 3 samples used for majority vote are not equal; the RIS.NE (noise error bit) is set. The received data is transferred despite the noise error. The NE bit will get appended to the received data before storing it in the RXDATA register at bit position 12. Please note that the majority voting feature is implemented only for data bits. One can select majority voting or single sample using the MAJVOTE control bit in the CTL0 register.

Note

Even though UART instances have noise filters, this feature provides an extra layer of noise immunity for longer glitches

11.2.3.4 Baud Rate Generation

The baud-rate divisor is a 22-bit number consisting of a 16-bit integer and a 6-bit fractional part. The number formed by these two values is used by the baud-rate generator to determine the bit sample period. Having a fractional baud-rate divisor allows the UART to generate all of the standard baud-rates very accurately

The 16-bit integer is loaded through the UART Integer Baud-Rate Divisor UARTx.IBRD register and the 6-bit fractional part is loaded with the UART Fractional Baud-Rate Divisor UARTx.FBRD register.

The baud-rate divisor can be calculated by using the following formula:

$$BRD = UART\ Clock / (Oversampling \times Baud\ rate) \quad (10)$$

UART clock is the clock output of the UART clock control logic, configured by CLKSEL and CLKDIV. Oversampling is selected by the HSE bit in the UARTx.CTL0 register and it can be 16, 8 or 3.

- UARTx.IBRD = INT(BRD), integer part of the BRD
- UARTx.FBRD = BRD % 64 , fractional part

The integer part of BRD is loaded into UARTx.IBRD register. The 6-bit fractional number must be loaded into the UARTx.FBRD register.

Note

When IBRD = 0, FBRD is ignored and no data gets transferred by the UART. Similarly, when IBRD = 65535 (that is 0xFFFF), then FBRD must not be greater than zero. If this is exceeded it results in an aborted transmission or reception.

The following example shows a simple method to calculate IBRD.DIVINT and FBRD.DIVFRAC for a baud rate of 19200 bit/s:

UART Clock = 40 MHz
Oversampling = 16
Baudrate = 19200 bit/s

$$BRD = \frac{UARTclk}{OVS \times Baudrate} = \frac{40MHz}{16 \times 19200\ bit/s} = 130.2083333$$

→ UARTx.IBRD.DIVINT= 130 (=82h)

UARTx.FBRRD.DIFRAC
= INT((.2083333 x 64) + 0.5)
= INT(13.833333)
= 13d (= Dh)

Note: The adder '+0.5' ensures rounding to the closest integer value to keep the rounding error as small as possible

Figure 11-5. Baud Rate Configuration

When updating the baud-rate divisor (UARTx.IBRD or UARTx.IFRD), the UART.LCRH register must also be written, so any changes to the baud-rate divisor must be followed by a write to the LCRH register for the changes to take effect. The contents of the UART.IBRD and UART.FBRD registers are not updated until transmission or reception of the current character is complete.

11.2.3.5 Data Transmission

Data received or transmitted is stored in two FIFOs, though the receive FIFO has an extra four bits per character for status information.

Transmit data:

For transmission, data is written into the transmit FIFO. If the UART is enabled, it causes a data frame to start transmitting with the parameters indicated in the UARTx.LCRH register. Data continues to be transmitted until there is no data left in the transmit FIFO. The BUSY bit in the UARTx.STAT register is asserted as soon as data is written to the transmit FIFO (that is, if the FIFO is nonempty) and remains asserted while data is being transmitted. The BUSY bit is negated only when the transmit FIFO is empty, and the last character has been transmitted to the shift register, including the stop bits. The UART can indicate that it is busy even though the UART can no longer be enabled. BUSY also is set during the generation of a BREAK signal.

Receive data:

When the receiver is idle (the RX signal is continuously 1), and the data input goes low (a start bit has been received), the receive counter begins running and data is sampled on the different cycle based on the oversampling setting of the HSE bit in UARTx.CTL0 register .The start bit is valid and recognized if the RX signal is still low after certain number for cycles based on the oversampling setting. After a valid start bit is detected, successive data bits are sampled according to the programmed length of the data characters. The parity bit is then checked if parity mode is enabled. Data length and parity are defined in the UART.LCRH register. Oversampling is explained in Section 11.2.3.2.

Lastly, a valid stop bit is confirmed if the RXD signal is high, otherwise a framing error has occurred. When a full word is received, the data is stored in the receive FIFO along with any error bits associated with that word.

11.2.3.6 Error and Status

For received data, the data byte and the 4-bit status (break, frame, parity, and overrun) is pushed onto the 12-bit wide receive FIFO. The error and status can be can be retrieved by reading UARTx.RXDATA register as showing in Table 11-1.

Table 11-1. UART Error and Conditions

Error Condition(1)	Bit Field	Description
Framing error	FRMERR	A framing error occurs when a low stop bit is detected. When two stop bits are used, both stop bits are checked for framing error. When a framing error is detected, the FRMERR bit is set.
Parity error	PARERR	A parity error is a mismatch between the number of 1s in a character and the value of the parity bit. When an address bit is included in the character, it is included in the parity calculation. When a parity error is detected, the PARERR bit is set.
Receive overrun	OVRERR	An overrun error occurs when a character is loaded into RXDATA/FIFO before the prior character has been read. When an overrun occurs, the OVRERR bit is set.
Break condition	BRKERR	A break is detected when all received data, parity, and stop bits are 0. When a break condition is detected, the BRKERR bit is set. A break condition can also set the interrupt flag RXINT if the break interrupt enable IMASK.BRKERR bit is set.

(1) Framing error and break condition are not set when LIN mode is enabled, this pattern is used to signal a Sync frame for LIN. Break detection is not available for IRDA, Manchester and Dali mode.

UART module flag status can also be checked by reading UARTx.STAT register as showing in Table 11-2.

Table 11-2. UART Flag Status

Bit Field	Description
BUSY	This bit is set as soon as the transmit FIFO becomes nonempty (regardless of whether UART is enabled). In IDLE_Line mode the busy signal also stays set during the idle time generation.
RXFE	This bit is set when receive FIFO is empty. If the FIFO is disabled (FEN is 0), the receive holding register is full. If the FIFO is enabled (FEN is 1), the receive FIFO is full.
RXFF	This bit is set when receive FIFO is full. If the FIFO is disabled (FEN is 0), the receive holding register is empty. If the FIFO is enabled (FEN is 1), the receive FIFO is empty.
TXFE	This bit is set when transmit FIFO is empty. If the FIFO is disabled (FEN is 0), the transmit holding register is full. If the FIFO is enabled (FEN is 1), the transmit FIFO is full.
TXFF	This bit is set when transmit FIFO is full. If the FIFO is disabled (FEN is 0), the transmit holding register is empty. If the FIFO is enabled (FEN is 1), the transmit FIFO is empty.
CTS	This bit is set when CTS signal is asserted (low) and cleared when CTS signal is not asserted (high).
IDLE	IDLE mode has been detected in idle line multiprocessor mode. The IDLE bit is used as an address tag for each block of characters. In idle line multiprocessor format, this bit is set when a received character is an address.

11.2.3.7 Local Interconnect Network (LIN) Support

This section is only relevant for UART Extend, which supports LIN mode. Refer to the device data sheet for the device-specific configuration of UART extend and UART main.

For supporting local interconnect network (LIN) protocol, the following hardware enhancements are implemented in the UART module:

- 16 bit up-counter (LINCNT) clocked by the UART clock.
- Interrupt capability on counter overflow (CPU_INT.IMASK.LINOVF).
- 16 bit capture register (LINC0) with two configurable modes
  - Capture of LINCNT value on RXD falling edge. Interrupt capability on capture.
  - Compare of LINCNT with interrupt capability on match.
- 16 bit capture register (LINC1) can be configured:
  - Capture LINCNT value on RXD rising edge. Interrupt capability on capture.

LIN transmission

Sending the break signal can be done by setting the BRK bit in UARTx.LCRH register. This bit needs to be set before the data is written into the FIFO or transmit data register TXDATA.

To generate LIN responder signals such as wake signals, the TX pin can be configured by TXD_OUT and TXD_CTL_EN bits in register UARTx.CTL0 to be software controlled. By setting TXD_CTL_EN bit to 1, the TX output pin can be controlled by the TXD_OUT bit if the UART transmit is disabled (CTL0.TXE is cleared).

LIN reception

LIN commander issues a break field and sync field at the start of every frame. Hardware must be added such that the LIN responder software driver can reasonably detect BREAK-SYNC and measure the necessary timing parameters to adjust the baud rate or determine an error.

For LIN reception, break field detection and compare mode are needed. To configure these features:

1. Initialize LIN counter to 0 (UARTx.LINCNT = 0)
2. Enable counter compare match mode (UARTx.LINCTL.LINC0_MATCH = 1)
3. Load UARTx.LINC0 (counter capture 0 register) with counter value corresponding to 9.5 x T_{bit} (refer to LIN Specification for details on this timing).
4. Enable LINC0 match interrupt (CPU_INT.IMASK.LINC0 = 1)
5. Setup LIN count control (UARTx.LINCTL):
   - Enable count while low signal on RXD (LINCTL.CNTRXLOW = 1)
   - Enable LIN counter clearing on RXD falling edge (LINCTL.ZERONE = 1)
   - Enable LIN counter (LINCTL.CTRENA = 1)

   Optional:
   - User can enable the rising edge on UART TX signal interrupt (CPU_INT.IMASK.RXPE = 1), when the RXPE interrupt fires, the software can read the LINCTR directly to see the BREAK field timing.
   - User can enable the LIN counter overflow interrupt (CPU_INT.IMASK.LINOVF = 1) to detect the BREAK field is too long and overflows 16bit counter. The timeout can be calculated as $t_{Timeout}$= 2^{16}/ UART clock.

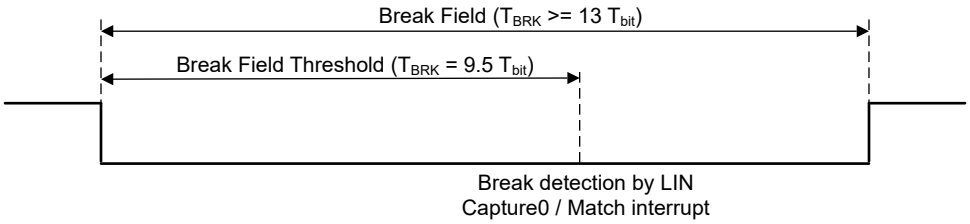


Figure 11-6. LIN Break Field Detection

Sync field validation is required to ensure accuracy of this LIN header part and to calculate the commander baud rate. The synch field consists of the data 0x55 inside a byte field (see Figure 11-7). After software detects a valid BREAK, it can then set the counter to measure the SYNC field. Both capture registers in LIN counter are used so that software sees fewer interrupts. Figure 11-7 shows the SYNC byte format. The LINCTR should be set to 0 on the start bit falling edge and count continuously. The LINC0 capture or RX falling edge interrupts fire at the falling edges of the RX line. During the interrupt processing, software can measure the individual HIGH-LOW times of the bits themselves using the values in the LINC0 and LINC1 registers to make sure all of the timings are valid.

The following flow describes a possible LIN sync field validation procedure:

1. Initialize LIN counter to 0 (UARTx.LINCNT = 0) after detecting a valid break field.
2. Enable interrupt on RX falling edge (CPU_INT.IMASK.RXNE = 1)
3. Setup LIN count control (LINCTL):
   - Enable LIN counter capture on raising RX edge (LINCTL.LINC1CAP = 1)
   - Enable LIN counter capture on falling RX edge (LINCTL.LINC0CAP = 1)
   - Enable LIN counter clearing on RX falling edge (LINCTL.ZERONE = 1)
   - Enable LIN counter (LINCTL.CTRENA = 1)

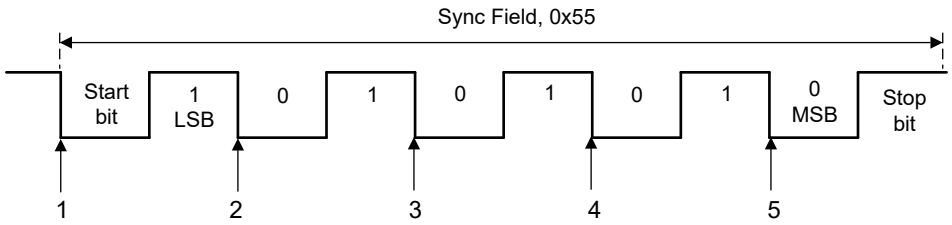


Figure 11-7. LIN SYNC Field Detection

Actions at each falling edge of the RX line for the sync field as showing below:

1. LIN counter is set to 0 and start counting on the falling RX edge. (LINCTL.ZERONE = 1)
2. RX falling edge interrupt trigger (RXNE):
   - Read capture register LINC0 (falling edge) and LINC1 (rising edge) values
   - Verify bit times
3. RX falling edge interrupt trigger (RXNE):
   - Read capture register LINC0 (falling edge) and LINC1 (rising edge) values
   - Verify bit times
4. RX falling edge interrupt trigger (RXNE):
   - Read capture register LINC0 (falling edge) and LINC1 (rising edge) values
   - Verify bit times
5. RX falling edge interrupt trigger (RXNE):
   - Read capture register LINC0 (falling edge) and LINC1 (rising edge) values
   - Verify bit times
   - Calculate the proper baud rate to set. Software must set the baud rate before the start bit of the PID field after sync field.

On each interrupt occurrence, the capture registers must be read and the bit times need to be validated by the application software. In case of a bit time verification error, the sync field analysis process must be aborted and the application software must switch back to break detection.

In case of errors like a breaking commander communication during sync field detection, a timeout interrupt can be generated by enabling the LIN counter overflow (IMASK.LINOVF = 1). When the counter overflows, the interrupt handler can abort the sync field analysis and switch back to break detection. The time the counter overflow interrupt occurs can be calculated as $t_{Timeout}= 2^{16}$/ UART clock.

Note

The sync field is automatically stored in the RX FIFO and can be misread as the PID. Therefore, flush the RX FIFO after the sync field is received and before the PID is received.

11.2.3.7.1 LIN Responder Transmission Delay

The interrupt RXINT for starting the transmission on the responder line always occurs at the mid-point of the commander STOP bit. Depending on the BUSCLK and baud-rate; there might not be enough response space between the STOP bit of commander and START bit of the responder; and the data transmission can start before end of STOP bit.

To overcome this, a delay of half the STOP bit period can be added as response space time to make sure that there is sufficient delay time between STOP and START bit, before start of responder data transmission.

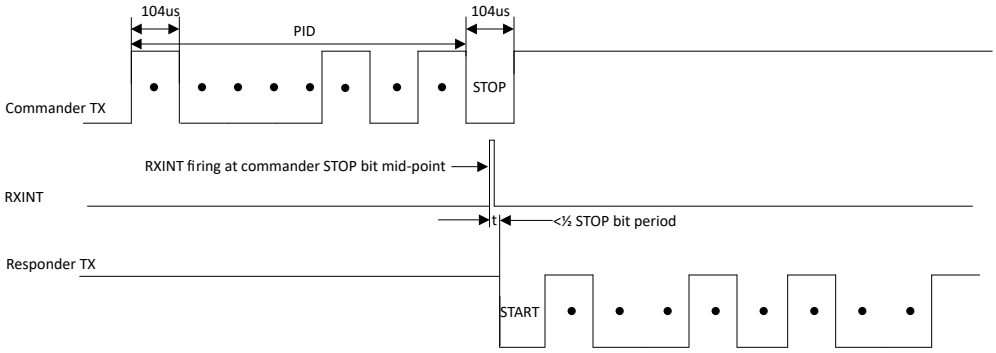


Figure 11-8. LIN Responder Transmission Delay

11.2.3.8 Flow Control

Flow control can be accomplished by hardware and the following sections describe the implementation method.

In UART mode (CTL0.MODE set to 0), hardware flow control between two devices is accomplished by connecting the RTS output to the CTS input on the receiving device, and connecting the RTS output on the receiving device to the CTS input. The RTS output signal is low active, the CTS input expects a low signal on a send request as shown in Figure 11-9.

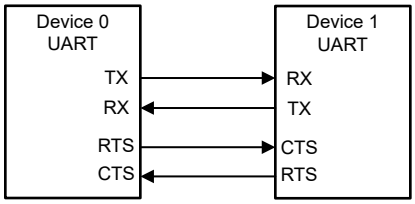


Figure 11-9. Flow Control

The CTS input controls the transmitter, the Device 0 and Device 1 transmitter can only transmit data when their CTS input is asserted low. When RTS flow control is enabled, the RTS output signal indicates the state of the receive FIFO. For example, the CTS of the Device 1 remains asserted low until the preprogrammed RX FIFO level of Device 0 is reached, indicating that the receive FIFO of Device 0 has no space to store additional characters.

The CTSEN and RTSEN bits in the UART.CTL0 register specify the flow control mode as shown in Table 11-3.

Table 11-3. Flow Control Enable

CTSEN	RTSEN	Description
1	1	RTS and CTS flow control enabled
1	0	Only CTS flow control enabled
0	1	Only RTS flow control enabled
0	0	Both RTS and CTS flow control disabled

When RTSEN is set to 1, the value of the CTL0.RTS bit is ignored and the RTS output signal is generated by the hardware trigger levels as described below. When RTSEN bit is cleared, the RTS signal output is controlled by the CTL0.RTS bit for SW control.

RTS flow control:

The RTS flow control logic is linked to the programmable receive FIFO watermark levels, it can be configured using UARTx.IFLS register. When RTS flow control is enabled, the RTS is asserted (low) until the receive FIFO is filled up to the watermark level. When the receive FIFO watermark level is reached, the RTS signal is de-asserted (high), indicating that there is no more room to receive any more data. The transmission of data is

expected to cease after the current character has been transmitted. The RTS signal is reasserted (low) when data has been read out of the receive FIFO so that it is filled to less than the watermark level. If RTS flow control is disabled and the UART is still enabled, then data is received until the receive FIFO is full, or no more data is transmitted to it.

As the RTS signal is de-asserted when the FIFO watermark level is reached by putting the last received character into the FIFO. This means on a back to back transmit another character transfer could already been started by the sender. Therefore, in such cases the watermark level should be set to one level lower to ensure all data can be received and put into the FIFO.

CTS flow control:

If CTS flow control is enabled, then the transmitter checks the CTS signal before transmitting the next byte. If the CTS signal is asserted (low), it transmits the byte otherwise transmission does not occur. The data continues to be transmitted while CTS is asserted (low), and the transmit FIFO is not empty. If the transmit FIFO is empty and the CTS signal is asserted (low) no data is transmitted. If the CTS signal is de-asserted (high) and CTS flow control is enabled, then the current character transmission is completed before stopping. If CTS flow control is disabled and the UART is enabled, then the data continues to be transmitted until the transmit FIFO is empty.

11.2.3.9 Idle-Line Multiprocessor

When IDLELINE is set in the CTL0.MODE register bits, the idle-line multiprocessor format is selected. Blocks of data are separated by an idle time on the transmit or receive lines (Figure 11-10). An idle receive line is detected when ten or more continuous ones (marks) are received after the one or two stop bits of a character. The baud-rate generator is switched off after reception of an idle line until the next start edge is detected. When an idle line is detected, the IDLE bit in UARTx.STAT is set. In Idle-Line mode the UART receiver operates in no parity mode and the UART word length (UARTx.LCRH.WLEN) must be set to 8bit.

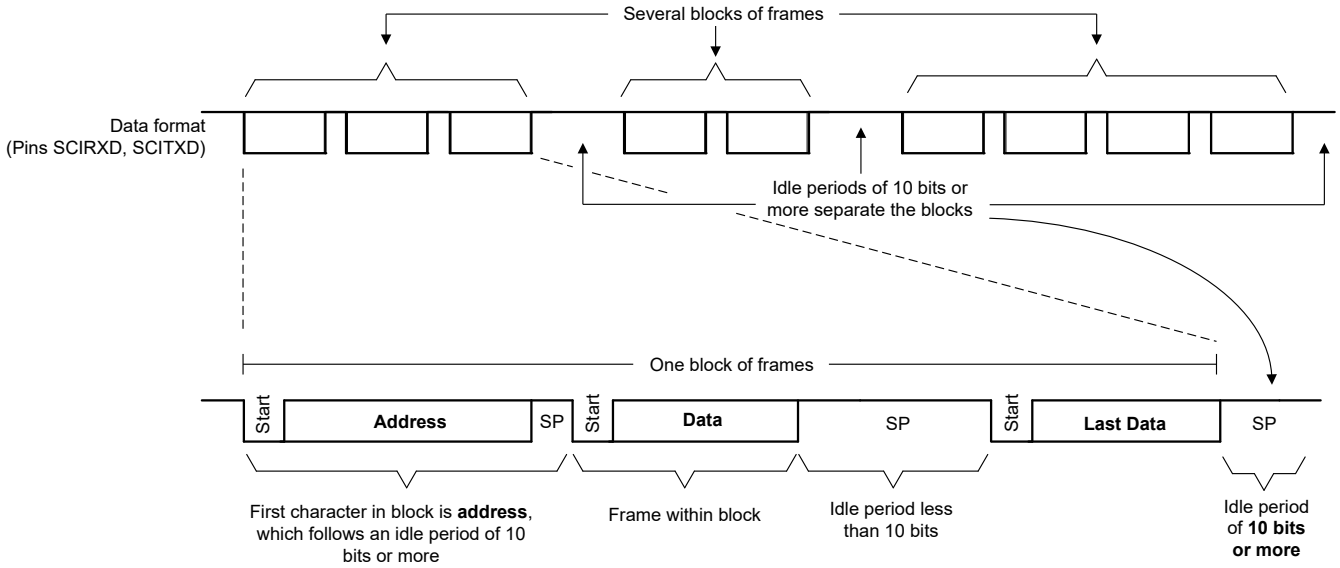


Figure 11-10. Idle-Line Multiprocessor

The first character received after an idle period is an address character. The IDLE bit in UARTx.STAT register is used as an address tag for each block of characters. In idle-line multiprocessor format, this bit is set when a received character is an address.

If an address character is received it is compared against the ADDR register with the AMASK applied. If the received character matches, the address character and all following received characters are transferred into the RXDATA buffer and interrupts/flags are generated until the next address without a match is received. The IDLE bit in UARTx.STAT register is automatically cleared when the address does match; otherwise the IDLE bit is set until the address is matched.

When the SENDIDLE bit in UARTx.LCRH register is set the UART inserts an idle period of 11 bit times on the bus, an ongoing transfer is finished first. The next transfer will be delayed till this idle period has finished. Then the next transfer can start with an address character.

The following procedure sends out an idle frame to indicate an address character followed by associated data:

1. Set SENDIDLE then write the address character to TXDATA. TXDATA must be ready for new data (TXINT interrupt must be 1). This generates an idle period of exactly 11 bits followed by the address character. SENDIDLE is reset automatically when the address character has been transferred (all bits are sent out of shift register).
2. Write desired data characters to TXDATA. TXDATA must be ready for new data (TXINT interrupt must be 1). The data written to TXDATA is transferred to the shift register and transmitted as soon as the shift register is ready for new data.

The idle-line time must not be exceeded between address and data transmission or between data transmissions. Otherwise, the transmitted data is misinterpreted as an address.

BUSY bit in IDLELINE mode:

- TX: BUSY is set during the generation of an IDLELINE signal and while the address and data bytes are sent
- RX: The BUSY signal set during the receive of data and till first 10 idle bits are received.

11.2.3.10 9-Bit UART Mode

9-bit mode is enabled by setting MODE bit to ADDR9BIT in the UARTx.CTL0 register. This feature is useful in a multi-drop configuration of the UART where a single controller connected to multiple peripherals can communicate with a particular peripheral through its address or set of addresses along with a qualifier for an address byte. In 9-bit UART mode, the parity enable/mode bits are ignored and the UART word length (UARTx.LCRH.WLEN) must be set to 8 bit.

Receive Transaction:

In 9-bit mode, a peripheral checks for the address qualifier at the location of the parity bit. If set, the received byte is compared with the preprogrammed address in UARTx.ADDR register:

- If the address matches, a 9-bit mode address match interrupt (ADDR_MATCH) is generated, if enabled and further data get received.
- If the address does not match, the address byte and the subsequent data bytes get dropped. .

The address can be predefined in the UART.ADDR register to match with the received byte. The matching can be extended to a set of addresses using the address mask in the UART.AMASK register. By default, the UART.AMASK is 0xFF, meaning that only the specified address must match

Transmit Transaction:

All the send transactions in 9-bit UART mode are interpreted as:

- Address bytes, if the 9th bit is set
- Data bytes, if the 9th bit is cleared

In 9-bit mode, the 9th bit can be controlled by software. The EPS bit setting of the LCRH register reflects the 9th bit for transmit transactions. To indicate an address byte, the software must set the EPS bit before the byte transmission. For data byte transmissions, the EPS bit must be cleared before the byte transmission. For a complete transmit transaction, the address byte must be transmitted as a single byte transaction with EPS bit set, followed by a data byte burst with EPS bit cleared.

9^{th} bit handling:

Table 11-4. 9th Bit Handling

PEN	SPS	EPS	9^{th} Bit (Transmitted or Verified)
0	X	X	Not transmitted or verified
1	1	0	0 (= Data)

Table 11-4. 9th Bit Handling (continued)

PEN	SPS	EPS	9th Bit (Transmitted or Verified)
1	1	1	1 (= Address)

11.2.3.11 RS485 Support

RS485 is a standard used in serial communications systems. This standard can be used effectively over long distances and in electrically noisy environments. Multiple receivers can be connected to such a network. These characteristics make RS-485 useful in industrial control systems and similar applications.

With the RS485 direction signal an external RS485 PHY can be controlled. The RTS I/O is used in this mode for the direction signal. The signal is set automatically to high once a data transmit is started. It will stay set between bytes if they are sent back to back. If a data receive is ongoing a new transmit should be delayed till this data has been received and the direction signal has been set to transmit.

Data exchange sequence as show in Figure 11-11:

- Wait till an ongoing receive has been finished.
- Activate direction signal to transmit on RTS Pin
- Send data (one or more Bytes)
- Wait till an ongoing receive has been finished.
- Deactivate direction signal to transmit on RTS Pin

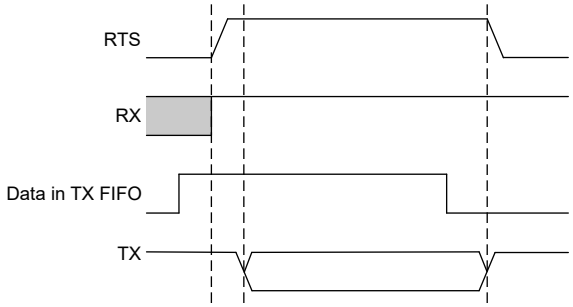


Figure 11-11. RS485 Data Exchange

Two bit fields to the LCRH register define the setup and hold time of the external driver direction control:

- EXTDIR_SETUP bits defines the number of UART clock ticks the signal to control the external driver for the RS485 will be set before the START bit is send. The generated setup time will be between EXDIR_SETUP value and EXDIR_SETUP + one baud rate cycle
- EXTDIR_HOLD bits defines the number of UART clock ticks the signal to control the external driver for the RS485 will be reset after the beginning of the stop bit. (If 2 STOP bits are enabled the beginning of the 2nd STOP bit.)

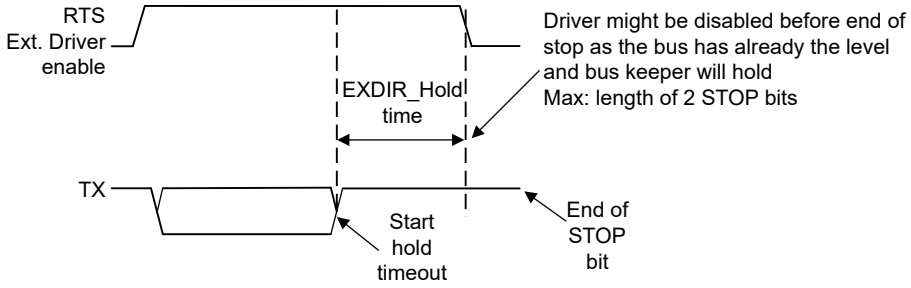


Figure 11-12. RS485 External Control

11.2.3.12 DALI Protocol

DALI stands for Digital Addressable Lighting Interface. It is an International Standard (IEC 62386) lighting control system, providing a single interface for all electronic control gear (light sources) and electronic control devices (lighting controllers).

The UART module supports the low level DALI Protocol sending and receiving the bit streams for forward and backward frames. The timing between any forward and backward frame sequence needs to be handled and checked by software.

Transmitting a forward or backward frame:

When transmitting a forward frame user needs to ensure to write second byte to buffer before first byte has been shifted out. The hardware will then send the two bytes without inserting the stop bits in-between. Otherwise the stop bits will be sent and the data will be handled like a backward frame.

Receiving data:

The UART module in DALI mode will check the 9th bit after the start bit to detect a Forward frame or backward frame. If this bit does have a change of the phase (= no stop bit) a forward frame is detected and:

- Address compare is done in software or hardware depending on MASK
- Address and data are always transferred into the RX Buffer

Otherwise a backward frame is detected and the data is transferred into the RX Buffer without setting the ADDR_MATCH interrupt.

The AMASK register can be used as group assignment used during multicast operation with the MSB to indicate if the device is part of a DALI group. To enable the UART in DALI mode to respond on all ADDRESS the AMASK register needs to be cleared.

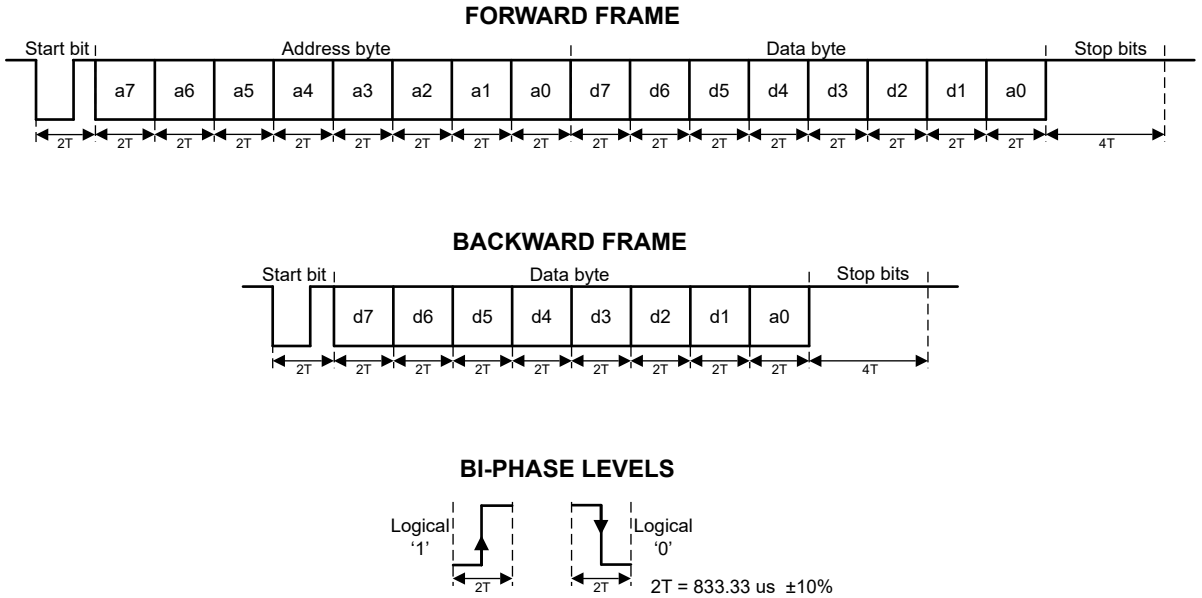


Figure 11-13. DALI Protocol

The limits for the times T shall be: 334 µs < T < 500 µs according to standard IEC 62386-102. DALI mode requires Manchester encoding to be enabled. When using DALI mode (CTL0.MODE set to DALI), the CTL0 and LCRH register must be set to:

- 8-bit word length (WLEN bit)
- No parity / 2 Stop Bit
- Manchester encoding enabled
- Baud-rate configuration set to match 2T = 833.33us
- DALI mode requires the FIFO to be enabled

11.2.3.13 Manchester Encoding and Decoding

UART provides option to receive and transmit Manchester encoded data. The function is enabled by the MENC bit in CTL0 register to generate the IEEE 802.3 compliant waveform. With the invert function in GPIO control module the output signals can be inverted to generate the G. E. Thomas compliant waveform.

The output signal is generated by XORing the data with UART clock signal. The UART clock needs to be double the speed of the baud-rate. So for the data transmit there is an edge at the beginning and the middle of each data bit. For the receive signal the edge in the middle of the bit is detected to decode the RX data.

11.2.3.14 IrDA Encoding and Decoding

When IREN bit in UARTx.IRCTL register is set, the IrDA encoder and decoder are enabled and provide hardware bit shaping for IrDA communication. IrDA en/decoding should only be used with UART mode (UARTx.CTL0.MODE is 0)

IrDA Encoding

The encoder sends a pulse for every zero bit in the transmit bit stream coming from the UART (see Figure 11-14). The pulse duration is defined by IRTXPL bits specifying the number of one-half clock periods of the clock selected by IRTXCLK bit.

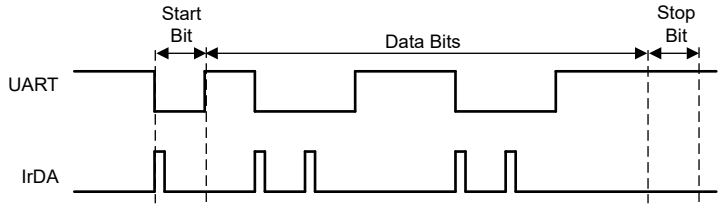


Figure 11-14. IrDA Protocol

(more information will be added in next revision)

IrDA Decoding

The decoder detects high pulses when IRRXPL = 0. Otherwise, it detects low pulses.

A programmable digital filter stage can be enabled by setting UARTx.GFCTL.DGFSEL > 0. When IRCTL.IREN is set, also the digital glitch filter should be set so that only pulses longer than the programmed filter length are passed and shorter pulses are discarded. (See the Glitch Suppression chapter on how to set the filter.)

11.2.3.15 ISO7816 Smart Card Support

The UART offers basic support to allow communication with an ISO7816 smart card. When configuring the MODE bits to smart card (0x4) of the UARTx.CTL0 register, the TXD signal is used as a bit clock, and the RXD signal is used as the half-duplex communication line connected to the smart card. Further smart card signals are not supported by the UART.

The clock rate of the UART clock in ISO7816 mode must be in the range of 1MHz to 5MHz when using ISO7816 mode, the UARTx.LCRH register must be set to:

- 8-bit word length (WLEN bits configured to 0x3)
- Even parity (PEN set and EPS set)
- No stick parity (SPS cleared)

In ISO7816 mode, the UART automatically uses 2 stop bits; therefore the STP2 bit of the LCRH register is ignored.

If a parity error is detected during a transmission, RXD is pulled low during the second stop bit. In this case, the UART aborts the transmission, it flushes the transmit FIFO and discards any data it contains. Additionally it raises a parity error interrupt, allowing the software to detect the problem and initiate retransmission of the affected data, as the UART does not support automatic retransmission in this case. The UART does not

support automatic retransmission on parity errors. If a parity error is detected on transmission, all further transmit operations are aborted and software must handle retransmission of the affected byte or message.

In Smartcard Mode, the receiver in case of parity error will drive the line low and a parity interrupt flag is asserted. The transmitter responds based on the value of this bit.

11.2.3.16 Address Detection

The UARTx.ADDR register is used to set the specific address that should be matched with receiving address byte. This register is used in conjunction with UARTx.AMASK register to form a match for address-byte received. Only bits where the AMASK is set to '1' are considered. So for full address the AMASK register is set to 0xFF. This feature is used in DALI, UART 9-Bit or Idle-Line mode.

Table 11-5. Address Detection

Condition	DALI Mode	Idle Line Mode	9-Bit Mode
Address match	Address and Data is moved to RXDATA	Address and Data is moved to RXDATA	Address and Data is moved to RXDATA
Address mismatch	Address and Data will be dropped	Address and Data will be dropped	Address and Data will be dropped

11.2.3.17 FIFO Operation

The UART has two FIFOs with a depth of 4 entries, one for transmit and one for receive. The FIFOs are accessed through the UART Data (TXDATA/RXDATA) registers. Read operations of the RXDATA register return a 12-bit value consisting of 8 data bits and 4 error flags. Write operations to TXDATA place 8-bit data in the transmit FIFO.

Out of reset, both FIFOs are disabled and act as 1-byte-deep holding registers. The FIFOs are enabled by setting the FEN bit in UARTx.CTL0. FIFO status can be monitored through the UARTx.STAT register and the interrupt events.

Hardware monitors empty, full and overrun conditions

The UARTx.STAT register contains empty and full flags (TXFE, TXFF, RXFE, and RXFF bits), and the CPU_INT.RIS register shows overrun status of the receive FIFO through the OVRERR bit. There is no indicator for a transmit FIFO overrun. A write is just lost, in case it overruns the transmit FIFO. If the FIFOs are disabled, the empty and full flags are set according to the status of the 1-byte-deep holding registers. When receiving more data than the FIFO can capture the oldest data will be overwritten with the received data.

The trigger point at which the FIFOs generate interrupts is controlled through the UARTx.IFLS register. Both FIFOs can be individually configured to trigger interrupts at different levels. Available configurations for transmit FIFO include 3/4, 1/2, 1/4 and empty, for receive FIFO 1/4, 1/2, 3/4 and full. For example, if the 3/4 option is selected for the receive FIFO, the UART generates a receive interrupt after 3 data bytes are received. Out of reset, both FIFOs are configured to trigger an interrupt at the 1/2 mark. The FIFO integrity is indeterminate under the following conditions:

- After a UART Send Break has been initiated (LCRH.BRK = 1)
- If the software disables the UART in the middle of a transmission with data in the FIFO, and then re-enables it.

11.2.3.18 Loopback Operation

The UART can be placed into an internal loopback mode for diagnostic or debug work by setting the LBE bit in the UART.CTL0 register. In loopback mode, the UART operates with the following behavior:

- Data transmitted on the TX output is received on the RX input
- Data transmitted on the TX output is not propagated to the TX IO pin
- Data received on the RX IO pin is ignored

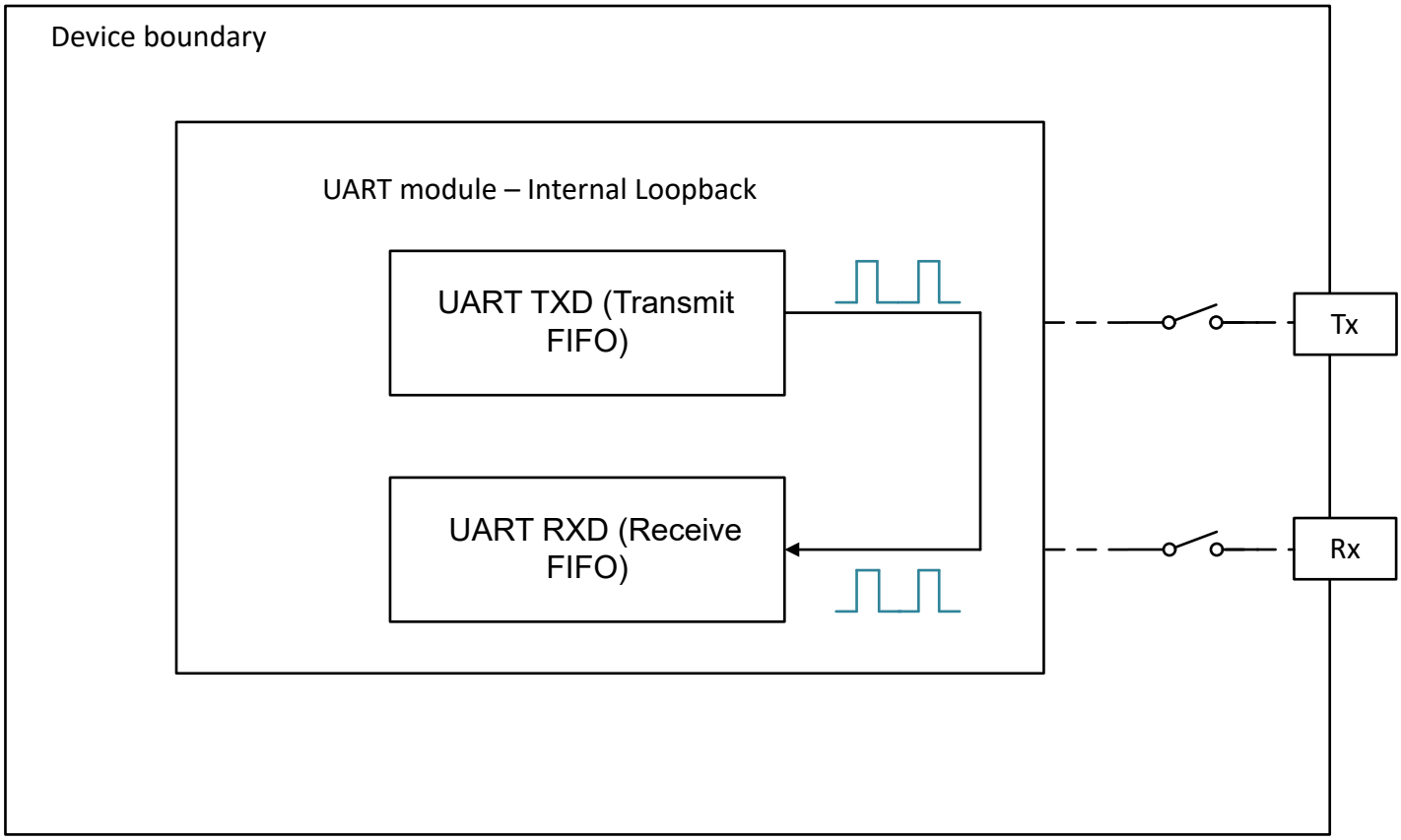


Figure 11-15. UART Loopback mode

11.2.3.19 Glitch Suppression

Digital filter

Digital filter is based on the UART functional clock. The DGFSEL bits in the UARTx.GFCTL register can be programmed to provide glitch suppression on the RX line and assure proper signal values. The glitch suppression value is in terms of functional clocks. All signals are delayed internally when glitch suppression is nonzero. For example, if DGFSEL is set to 0x5, 5 clocks should be added onto the calculation for the expected transaction time. The DGFSEL need to be configured for the glitch suppression pulse width to be shorter than 1/3 of a normal data pulse, to avoid a normal pulse is filtered unexpectedly.

Analog filter

The analog glitch suppression on the RX line is based on the analog glitch filter and it can be selected with the AGFSEL bits in the UARTx.GFCTL register. See data sheet for the select-able glitch filter values. The analog glitch filter is enabled with the AGFEN, if not set the input signals will be passed through to the UART module without filtering.

11.2.4 Low Power Operation

(More information will be added in next revision)

11.2.5 Reset Considerations

Software Reset Considerations

A software reset can be executed with setting the RESETASSERT together with the KEY in the RSTCTL register. An ongoing transfer will be terminated immediately and can leave the software in an undefined state. Therefore, before requesting a reset an ongoing transfer should be terminated.

Hardware Reset Considerations

A hardware reset also initializes the IO configuration. This sets the IOs to a high impedance state and the data lines can float. If this is critical for the application or connected devices on the UART interface external pull up or down resistors might be required.

11.2.6 Initialization

Before the UART is setup or configuration changes, the ENABLE bit should be cleared to avoid unpredictable behavior during the updates or for the first data receive or transmitted afterward.

To enable and initialize the UART, use the following steps:

1. Configure RX and TX pin functions by using the IOMUX registers.
2. Reset the peripheral using UARTx.RSTCTL register
3. Enable the power to UART peripheral using the UARTx.PWREN register
4. Select the UART function clock source and divide options using UART.CLKSEL and UART.CLKDIV registers.
5. Disable the UART by clearing the UART.CTL0.ENABLE bit.
6. Use the baud-rate equation in Section 11.2.3.4 to calculate the UARTx.IBRD and UARTx.FBRD registers.
7. Write the integer portion of the BRD to the UART.IBRD register.
8. Write the fractional portion of the BRD to the UART.FBRD register.
9. Write the desired oversampling and FIFO configuration to the UART.CTL0 register
10. Write the desired serial parameters to the UART.LCRH register.
11. Enable the UART by setting the UART.CTL0.ENABLE bit.

11.2.7 Interrupt and Events Support

The UART module contains three event publishers and no event subscribers. One event publisher (CPU_INT) manages UART interrupt requests (IRQs) to the CPU subsystem through a static event route. The second and third event publishers (DMA_TRIG_RX, DMA_TRIG_TX) are used to setup the trigger signaling for the DMA through DMA event route.

The UART events are summarized in Table 11-6.

Table 11-6. UART Events

Event	Type	Source	Destination	Route	Configuration	Functionality
CPU interrupt	Publisher	UART	CPU Subsystem	Static route	CPU_INT registers	Fixed interrupt route from UART to CPU
DMA trigger	Publisher	UART	DMA	DMA event route	DMA_TRIG_RX registers	Fixed interrupt route from UART to DMA
DMA trigger	Publisher	UART	DMA	DMA event route	DMA_TRIG_TX registers	Fixed interrupt route from UART to DMA

11.2.7.1 CPU Interrupt Event Publisher (CPU_INT)

The UART module provides 18 interrupt sources which can be configured to source a CPU interrupt event. In order of decreasing interrupt priority, the CPU interrupt events from the UART are:

Table 11-7. UART CPU Interrupt Event Conditions (CPU_INT)

IIDX STAT	Name	Description
0x01	RTOUT	UART receive timeout interrupt, This interrupt is asserted when the receive FIFO is not empty, and no further data is received specified time in the UARTx.IFLS.RXTOSEL bits. More information provided below.
0x02	FRMERR	UART framing error interrupt, see Section 11.2.3.6 for more information.
0x03	PARERR	UART parity error interrupt, see Section 11.2.3.6 for more information.

Table 11-7. UART CPU Interrupt Event Conditions (CPU_INT) (continued)

IIDX STAT	Name	Description
0x04	BRKERR	UART break error interrupt, see Section 11.2.3.6 for more information.
0x05	OVRERR	UART receive overrun error interrupt, see Section 11.2.3.6 for more information.
0x06	RXNE	Falling edge on RX interrupt, this interrupt triggers when there is a falling edge on RX line.
0x07	RXPE	Rising edge on RX interrupt, this interrupt triggers when there is a rising edge on RX line.
0x08	LINC0	LIN capture 0 match interrupt, this interrupt triggers when the defined capture 0 value is reached in LIN counter.
0x09	LINC1	LIN capture 1 match interrupt, this interrupt triggers when the defined capture 1 value is reached in LIN counter.
0x0A	LINOVF	LIN counter overflow interrupt, this interrupt triggers when the 16bit LIN counter overflows.
0x0B	RXINT	UART receive interrupt. More information provided below.
0x0C	TXINT	UART transmit interrupt. More information provided below.
0x0D	EOT	UART end of transmission interrupt, it indicates that the last bit of all transmitted data and status has left the serializer and without any further data in the TX FIFO.
0x0E	ADDR_MATCH	Address match interrupt, used in protocols with address to indicate address match happened.
0x0F	CTS	UART clear to send interrupt, indicate the CTS signal status.
0x10	DMA_DONE_RX	This interrupt is set if the RX DMA channel sends the DONE signal.
0x11	DMA_DONE_TX	This interrupt is set if the TX DMA channel sends the DONE signal.

The CPU interrupt event configuration is managed with the CPU_INT event management registers. See Section 6.2.5 for guidance on configuring the Event registers for CPU interrupts.

The receive timeout interrupt is asserted when the receive FIFO is not empty, and no further data is received specified time in the IFLS.RXTOSEL bits. The receive timeout interrupt is cleared either when the FIFO becomes empty through reading all the data (or by reading the holding register), by reading the interrupt index from IIDX or when a 1 is written to the corresponding bit in the ICLR register.

The receive interrupt (RXINT, 0x0B) changes state when one of the following events occurs:

- If the FIFOs are enabled and the receive FIFO reaches the programmed trigger level, the RXINT bit is set. The receive interrupt is cleared by reading data from the receive FIFO until it becomes less than the trigger level, by reading the interrupt index from IIDX or by writing a 1 to the RXINT bit in ICLR.
- If the FIFOs are disabled (have a depth of one location) and data is received thereby filling the location, the RXINT bit is set. The receive interrupt is cleared by performing a single read of the receive FIFO, by reading the interrupt index from IIDX or by writing a 1 to the RXINT bit in ICLR.

The transmit interrupt (TXINT, 0x0C) changes state when one of the following events occurs:

- If the FIFOs are enabled and the transmit FIFO progresses through the programmed trigger level, the TXINT bit is set. The transmit interrupt is based on a transition through level, therefore the FIFO must be written past the programmed trigger level otherwise no further transmit interrupts will be generated. The transmit interrupt is cleared by writing data to the transmit FIFO until it becomes greater than the trigger level, by reading the interrupt index from IIDX or by writing a 1 to the TXINT bit in ICLR.
- If the FIFOs are disabled (have a depth of one location) and there is no data present in the transmitters single location, the TXINT bit is set. It is cleared by performing a single write to the transmit FIFO, by reading the interrupt index from IIDX or by writing a 1 to the TXINT bit in ICLR.

11.2.7.2 DMA Trigger Publisher (DMA_TRIG_RX, DMA_TRIG_TX)

DMA_TRIG_RX and DMA_TRIG_TX registers are used to setup the trigger signaling for the DMA. This can be setup in a flexible way to trigger the DMA for receive or transmit events with the trigger conditions in Table 11-8and Table 11-9.

DMA_TRIG_RX is used for triggering the DMA to do a receive data transfer and DMA_TRIG_TX is used for triggering the DMA to do a transmit data transfer.

Table 11-8. UART DMA Trigger Condition (DMA_TRIG_RX)

IIDX STAT	Name	Description
0x01	RTOUT	UART receive timeout interrupt, This interrupt is asserted when the receive FIFO is not empty, and no further data is received specified time in the UARTx.IFLS.RXTOSEL bits. More information provided below.
0x0B	RXINT	UART receive interrupt. More information provided below.

The receive timeout interrupt is asserted when the receive FIFO is not empty, and no further data is received specified time in the IFLS.RXTOSEL bits. The receive timeout interrupt is cleared either when the FIFO becomes empty through reading all the data (or by reading the holding register), by reading the interrupt index from IIDX or when a 1 is written to the corresponding bit in the ICLR register.

The receive interrupt (RXINT, 0x0B) changes state when one of the following events occurs:

- If the FIFOs are enabled and the receive FIFO reaches the programmed trigger level, the RXINT bit is set. The receive interrupt is cleared by reading data from the receive FIFO until it becomes less than the trigger level, by reading the interrupt index from IIDX or by writing a 1 to the RXINT bit in ICLR.
- If the FIFOs are disabled (have a depth of one location) and data is received thereby filling the location, the RXINT bit is set. The receive interrupt is cleared by performing a single read of the receive FIFO, by reading the interrupt index from IIDX or by writing a 1 to the RXINT bit in ICLR.

Table 11-9. UART DMA Trigger Condition (DMA_TRIG_TX)

IIDX STAT	Name	Description
0x0C	TXINT	UART transmit interrupt. More information provided below.

The transmit interrupt (TXINT, 0x0C) changes state when one of the following events occurs:

- If the FIFOs are enabled and the transmit FIFO progresses through the programmed trigger level, the TXINT bit is set. The transmit interrupt is based on a transition through level, therefore the FIFO must be written past the programmed trigger level otherwise no further transmit interrupts will be generated. The transmit interrupt is cleared by writing data to the transmit FIFO until it becomes greater than the trigger level, by reading the interrupt index from IIDX or by writing a 1 to the TXINT bit in ICLR.
- If the FIFOs are disabled (have a depth of one location) and there is no data present in the transmitters single location, the TXINT bit is set. It is cleared by performing a single write to the transmit FIFO, by reading the interrupt index from IIDX or by writing a 1 to the TXINT bit in ICLR.

The DMA trigger event configuration is managed with the DMA_TRIG_RX and DMA_TRIG_TX event management registers. See Section 6.2.5 for guidance on configuring the Event registers and Section 6.1.3.2 for on how DMA trigger event works.

11.2.8 Emulation Modes

The module behavior while the device is in debug mode is controlled by the FREE and SOFT bits in PDBGCTL register.

When the device is in debug mode and set into halt mode below behavior can be configured.

Table 11-10. Debug Mode Peripheral Behavior

PDBGCTL.FREE	PDBGCTL.SOFT	Function
1	**x**	Modules continues operation
0	**0**	Module stops immediately
0	**1**	Module stops after the next transfer has been finished