Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

I²C-bus/SPI to UART Bridge Controller w/ 64 bytes of TX/RX FIFOs

Features

- Single channel full-duplex UART
- Support I²C-bus or SPI interface
- 64 bytes FIFO (transmitter and receiver)
- Fully compatible with industrial standard 16C450 and equivalent
- Baud Rates up to 16Mbit/s in 4X sampling clock rate
- Programmable character formatting
  - 5-bit, 6-bit, 7-bit or 8-bit character
  - Even, odd, or no parity
  - 1, 1.5, or 2 stop bits
- Programmable Receive and Transmit FIFO trigger levels
- Special character detection
- Internal Loopback mode
- Line break generation and detection

Flow control

- Support hardware flow control using RTS/CTS
- Support software flow control with programmable Xon/Xoff characters
- Programmable single or double Xon/Xoff characters

Interface control

- Automatic RS-485 slave address detection
- RS-485 driver direction control via RTS signal
- RS-485 driver direction control inversion
- Built-in IrDA encoder and decoder interface
- Supports IrDA SIR with speeds up to 115.2 kbit/s (optional 1.152Mbps)
- Up to eight user programmable GPIO pins
- Software reset

Others

- Low standby current at 3.3 V
- Wide operation voltage (1.8V, 2.5V or 3.3V)
- Industrial and commercial temperature ranges
- Available in QFN24, TSSOP24 and TSSOP16 Packages

I²C interface

- Compliant with I²C-bus fast speed
- Support slave mode only
- Crystal oscillator (up to 24MHz) or external clock (up to 64MHz) input

SPI interface

- Support 20 Mbit/s maximum SPI clock speed
- Support SPI mode 0 (slave mode only)

Description

The PI7C9X760B is a I²C-bus/SPI to a single-channel high performance UART bridge controller. It offers data rates up to 20 Mbps and guarantees low operating and sleeping current. The PI7C9X760B also has up to 8 additional programmable general purpose I/O [GPIO] pins. The device comes in very small TSSOP24 packages, which makes it ideally suitable for cost efficient, handheld, battery operated applications. These UARTs provide a bridge for protocol conversion from I²C -bus or SPI to and RS-232/RS-485 and are fully bidirectional.

The PI7C9X760B supports SPI clock speeds up to 20 Mbps and IrDA SIR up to 1.152 Mbit/s.

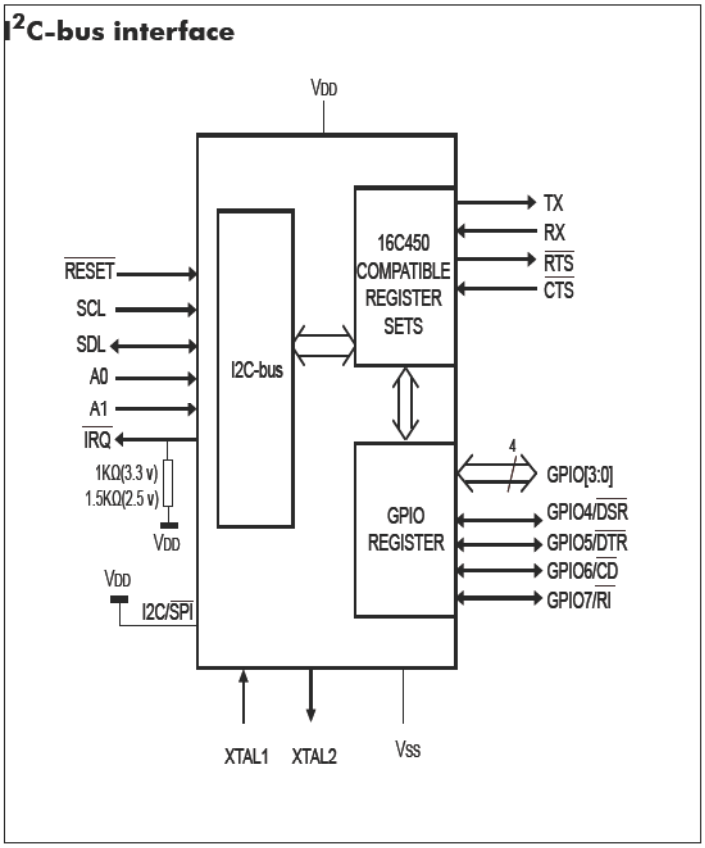
PI7C9X760B's internal register set is backward-compatible with the widely used and widely popular 16C450 UART. The PI7C9X760B also provides additional advanced features such as auto hardware and software flow control, automatic RS-485 support, support for fractional baud rates and software reset. This allows the software to reset the UART at any moment, independent of the hardware reset signalThis allows the software to reset the UART at any moment, independent of the hardware reset signal.

Application

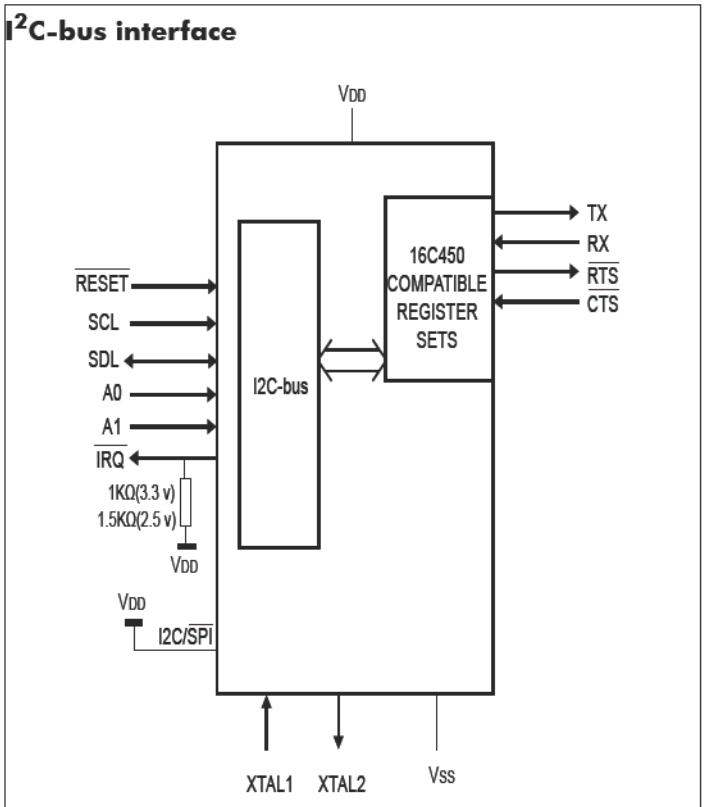
- Industrial computing
- Automation
- Factory process control
- Mobile computing
- Embedded applications
- Battery operated devices
- Networking

Block Diagram

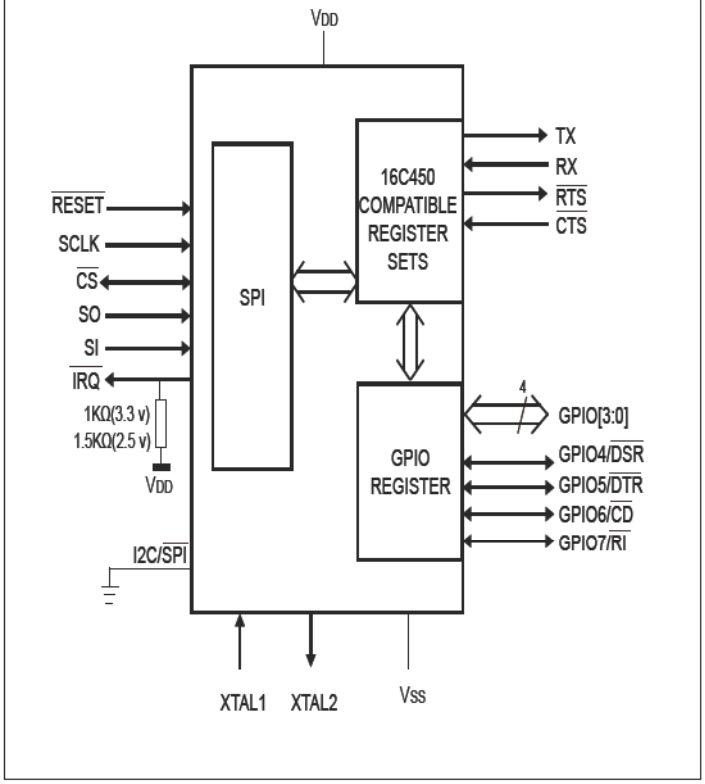
I²C-bus interface



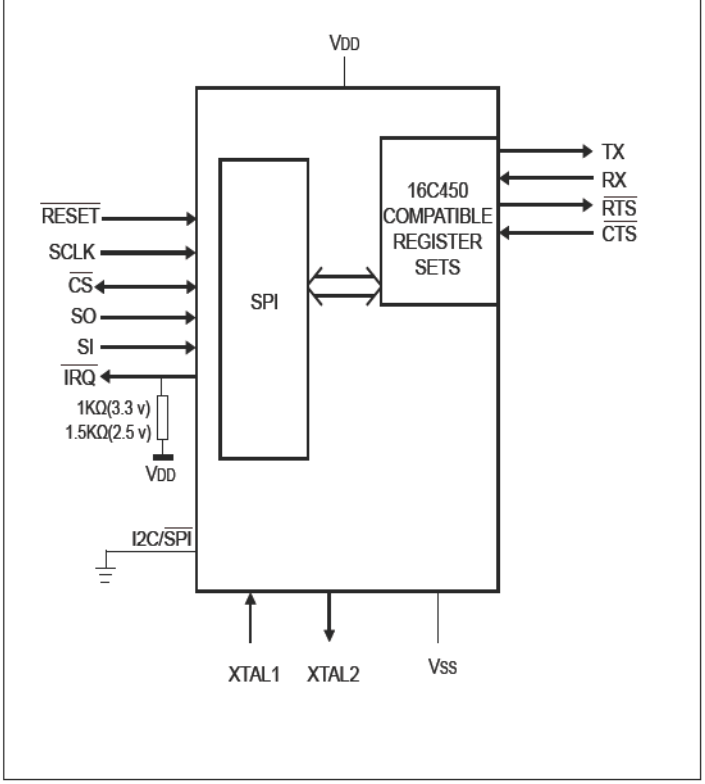
I²C-bus interface



SPI interface

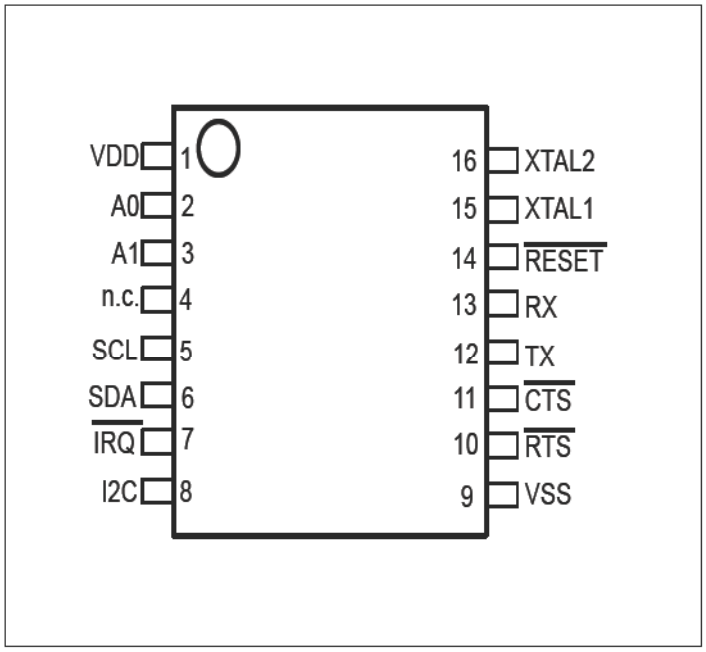


SPI interface



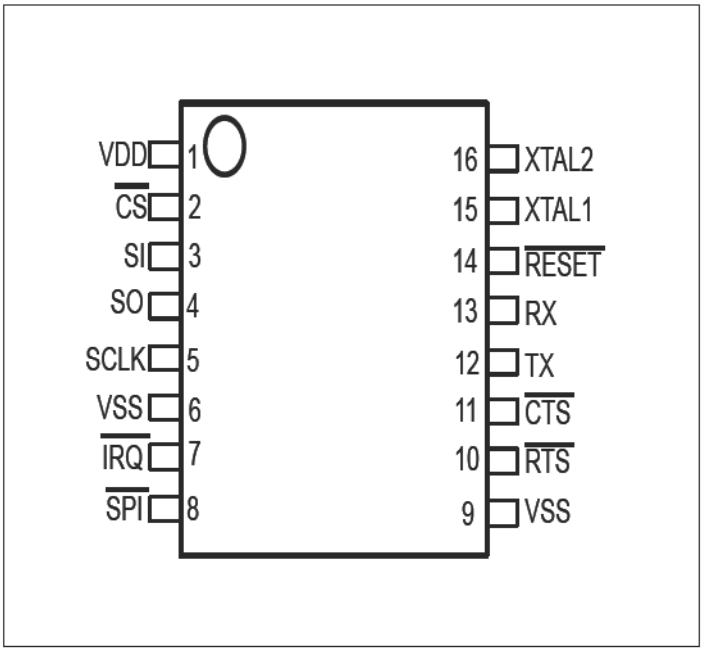
Pin Configuration (I²C-Bus Interface)

16-Pin TSSOP



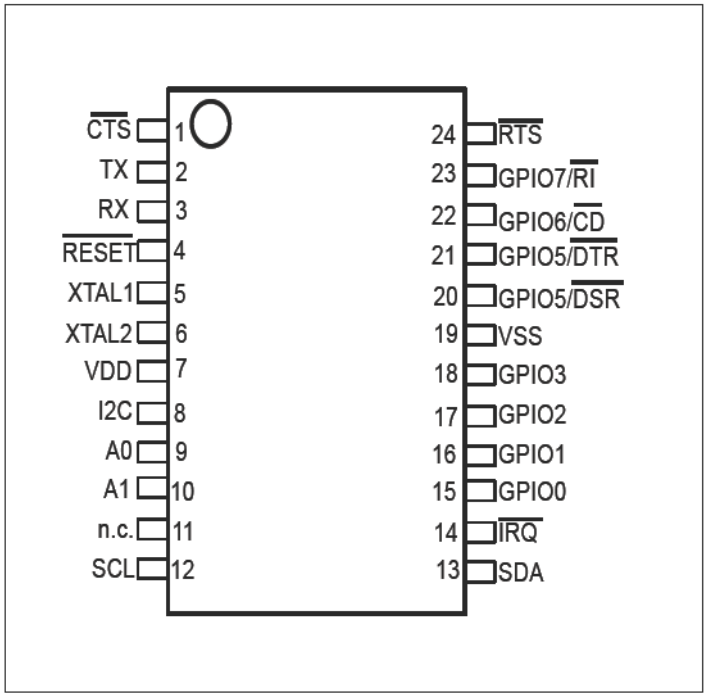
Pin Configuration (SPI Interface)

16-Pin TSSOP



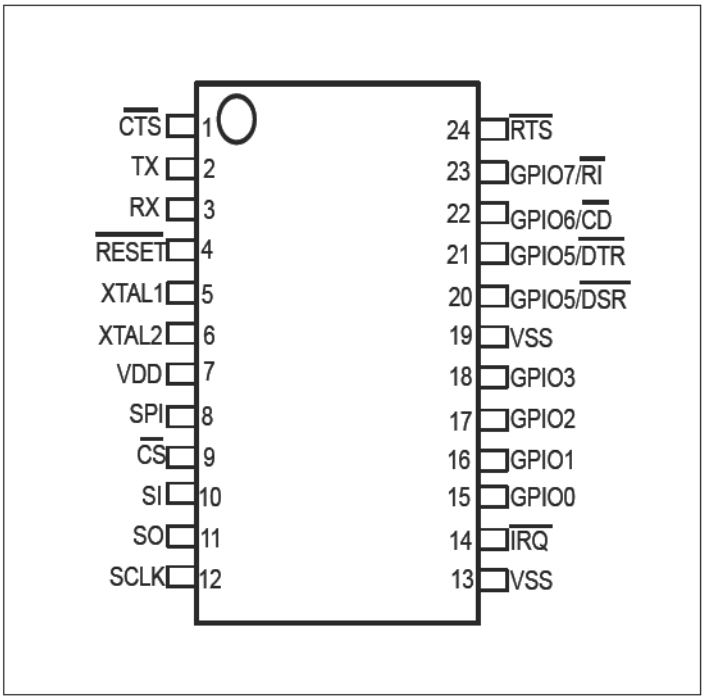
Pin Configuration (I²C-Bus Interface)

24-Pin TSSOP



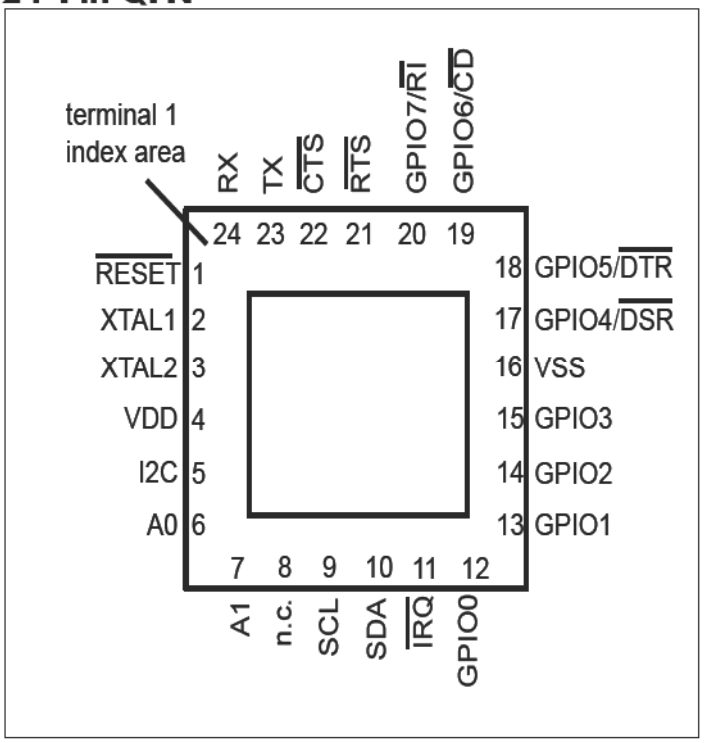
Pin Configuration (SPI Interface)

24-Pin TSSOP



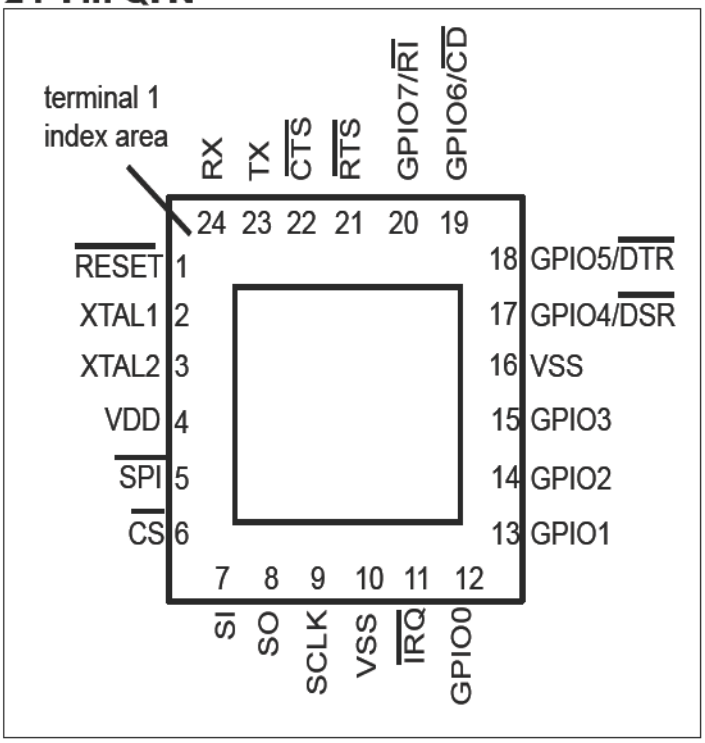
Pin Configuration (I²C-Bus Interface)

24-Pin QFN



Pin Configuration (SPI Interface)

24-Pin QFN



Pin Description

Pin Name	16-TSSOP Pin#	24-TSSOP Pin#	24-QFN Pin#	Type	Description
I²C (SPI) INTERFACE					
$\overline{CTS}$	11	1	22	I	UART clear to send (active LOW). A logic 0 (LOW) on the $\overline{CTS}$ pin indicates the modern or data set is ready to accept transmit data from the PI7C9X760B. Status can be tested by reading MSR[4]. This pin only affects the transmit and receive operations when auto $\overline{CTS}$ function is enabled via the Enhanced Feature Register EFR[7] for hardware flow control operation.
TX	12	2	23	O	UART transmitter output. During the local Loopback mode, the TX output pin is disabled and TX data is internally connected to the UART RX input.
RX	13	3	24	I	UART receiver input. During the local Loopback mode, the RX input pin is disabled and TX data is connected to the UART RX input internally.
$\overline{RESET}$	14	4	1	I	Device hardware reset (active LOW).
XTAL1	15	5	2	I	Crystal input or external clock input. Functions as a crystal input or as an external clocl input. A crystal can be connected between XTAL1 and XTAL2 to form an internal oscillator circuit (see Figure 16). Alternatively, an external clock can be connected to this pin.
XTAL2	16	6	3	O	Crystal output or clock output. (See also XTAL1.) XTAL2 is used as a crystal oscillator output.
V_{DD}	1	7	4		Power supply.
I2C/$\overline{SPI}$	8	8	5	I	I²C-bus or SPI interface select. I²C-bus interface is selected if this pin is HIGH. SPI interface is selected if this pin is LOW.
$\overline{CS}$/A0	2	9	6	I	SPI chip select or I2C-bus device address select A0. If SPI configuration is selected by I²C/SPI pin, this pin is the SPI chip select pin (Schmitt-trigger, active LOW).If I²C-bus configuration is selected by I²C/SPI pin, this pin along with the A1 pin allows user to change the device's base address.
SI/A1	3	10	7	I	SPI chip select or I²C-bus device address select A1. If SPI configuration is selected by I²C/SPI pin, this is the SPI data input pin. If I²C-bus configuration is selected by I²C/SPI pin, this pin along with the A0 pin allows user to change the device's base address. To select the device address, please refer to Table32.
SO	4	11	8	O	SPI data output pin. If SPI configuration is selected by I²C/SPI pin, this is a 3-stateable output pin. If I²C-bus configuration is selected by I²C/SPI pin, this pin function is undefined and must be left as N.C. (not connected).
SCL/SCLK	5	12	9	I	I²C-bus or SPI input clock.

Pin Description Cont...

Pin Name	16-TSSOP Pin#	24-TSSOP Pin#	24-QFN Pin#	Type	Description
SDA	6	13	10	I/O	I^2C-bus data input/output, open-drain if I^2C-bus configuration is selected by I^2C/SPI pin. If SPI configuration is selected, then this pin is undefined and must be connected to V_{SS}.
$\overline{IRQ}$	7	14	11	O	Interrupt (open-drain, active LOW). Interrupt is enabled when interrupt sources are enables in the Interrupt Enable Register(IER). Interrupt conditions include: change of state of the input pins, receiver errors, available receiver buffer data, available transmit buffer space, or when a modem status flag is detected. An external resistor (1 ohm for 3.3V, 1.5 ohm for 2.5V) must be connected between this pin and V_{DD}.
GPIO0		15	12	I/O	Programmable I/O pin.
GPIO1		16	13	I/O	Programmable I/O pin.
GPIO2		17	14	I/O	Programmable I/O pin.
GPIO3		18	15	I/O	Programmable I/O pin.
GPIO4/$\overline{DSR}$		20	17	I/O	Programmable I/O pin or modem's $\overline{DSR}$ pin.
GPIO5/$\overline{DTR}$		21	18	I/O	Programmable I/O pin or modem's $\overline{DSR}$ pin.
GPIO6/$\overline{CD}$		22	19	I/O	Programmable I/O pin or modem's $\overline{DSR}$ pin.
GPIO7/$\overline{RI}$		23	20	I/O	Programmable I/O pin or modem's $\overline{DSR}$ pin.
$\overline{RTS}$	10	24	21	O	UART request to send (active LOW). A logic 0 on the $\overline{RTS}$ pin indicates the transmitter has data ready and waiting to send. Writing a logic 1 in the modem control register MCR[1] will set this pin to a logic 0, indicating data is available. After a reset this pin set to a logic 1. This pin only affects the transmit and receive operations when auto $\overline{RTS}$ function is enabled via the Enhanced Feature Register (EFR[6]) for hardware flow control operation.
V_{SS}	9	19	16[4]		Ground.
V_{SS}			Certer Pad		The center pad on the back side of the QFN24 package is metallic and should be connected to grond on the printed-circuit board.

Functional Description

The UART will perform serial-to-I²C-bus conversion on data characters received from peripheral devices or modems, and I²C-bus-to-serial conversion on data characters transmitted by the host. The complete status of the UART can be read at any time during functional operation by the host.

The UART can be placed in an alternate mode (FIFO mode) relieving the host of excessive software overhead by buffering received/transmitted characters. Both the receiver and transmitter FIFOs can store up to 64 characters (including three additional bits of error status per character for the receiver FIFO) and have selectable or programmable trigger levels.

The UART has selectable hardware flow control and software flow control. Hardware flow control significantly reduces software overhead and increases system efficiency by automatically controlling serial data flow using the $\overline{RTS}$ output and $\overline{CTS}$ input signals. Software flow control automatically controls data flow by using programmable Xon/Xoff characters.

The UART includes a programmable baud rate generator that can divide the timing reference clock input by a divisor between 1 and $(2^{16} - 1)$.

1. Trigger levels

The UART provides independently selectable and programmable trigger levels for both receiver and transmitter interrupt generation. After reset, both transmitter and receiver FIFOs are disabled and so, in effect, the trigger level is the default value of one character. The selectable trigger levels are available via the FIFO Control Register (FCR). The programmable trigger levels are available via the Trigger Level Register (TLR). If TLR bits are cleared, then selectable trigger level in FCR is used. If TLR bits are not cleared, then programmable trigger level in TLR is used.

2. Hardware flow control

Hardware flow control is comprised of Auto-$\overline{CTS}$ and Auto-RTS (see Figure 1). Auto-$\overline{CTS}$ and Auto-$\overline{RTS}$ can be enabled/disabled independently by programming EFR[7:6].

With Auto-$\overline{CTS}$, $\overline{CTS}$ must be active before the UART can transmit data.

Auto-$\overline{RTS}$ only activates the $\overline{RTS}$ output when there is enough room in the FIFO to receive data and de-activates the $\overline{RTS}$ output when the RX FIFO is sufficiently full. The halt and resume trigger levels in the Transmission Control Register (TCR) determine the levels at which $\overline{RTS}$ is activated/deactivated. If TCR bits are cleared, then selectable trigger levels in FCR are used in place of TCR.

If both Auto-$\overline{CTS}$ and Auto-$\overline{RTS}$ are enabled, when $\overline{RTS}$ is connected to $\overline{CTS}$, data transmission does not occur unless the receiver FIFO has empty space. Thus, overrun errors are eliminated during hardware flow control. If not enabled, overrun errors occur if the transmit data rate exceeds the receive FIFO servicing latency.

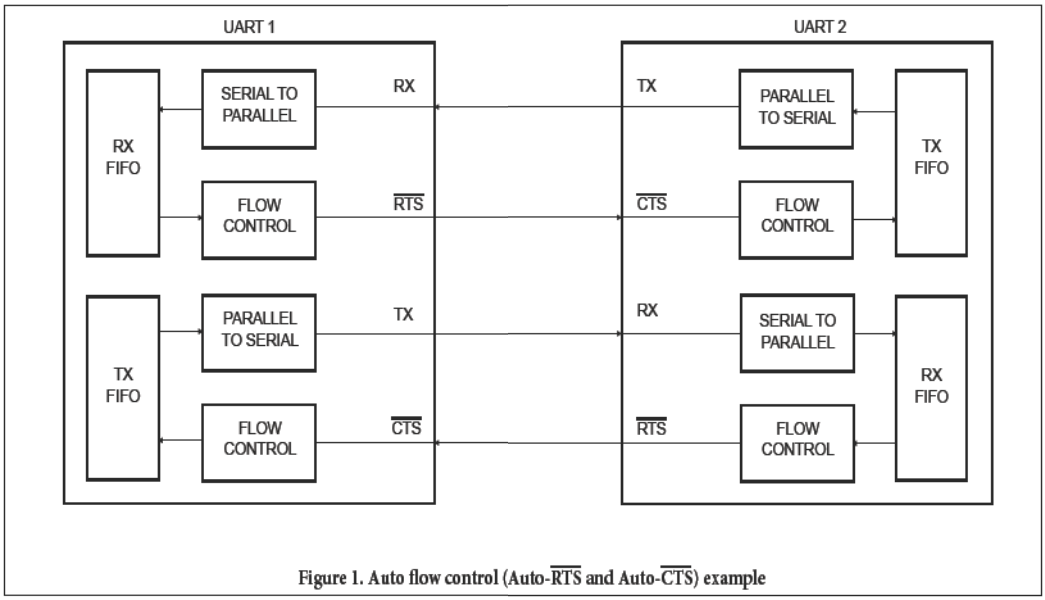
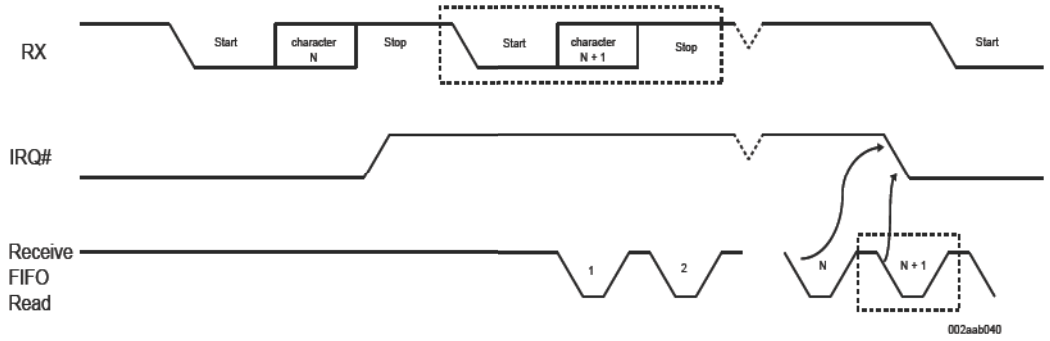


Figure 1. Auto flow control (Auto-$\overline{RTS}$ and Auto-$\overline{CTS}$) example

2.1 Auto-RTS

Figure 2 shows $\overline{RTS}$ functional timing. The receiver FIFO trigger levels used in Auto-$\overline{RTS}$ are stored in the TCR. RTS is active if the RX FIFO level is below the halt trigger level in TCR[3:0]. When the receiver FIFO halt trigger level is reached, RTS is de-asserted. The sending device (for example, another UART) may send an additional character after the trigger level is reached (assuming the sending UART has another character to send) because it may not recognize the de-assertion of $\overline{RTS}$ until it has begun sending the additional character. $\overline{RTS}$ is automatically reasserted once the receiver FIFO reaches the resume trigger level programmed via TCR[7:4]. This re-assertion allows the sending device to resume transmission.

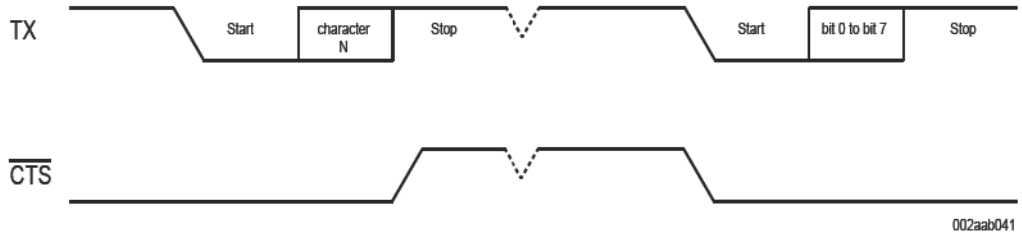


(1) N = receiver FIFO trigger level.
(2) The two blocks in dashed lines cover the case where an additional character is sent.

Figure 2. RTS functional timing

2.2 Auto-CTS

Figure 3 shows $\overline{CTS}$ functional timing. The transmitter circuitry checks $\overline{CTS}$ before sending the next data character. When $\overline{CTS}$ is active, the transmitter sends the next character. To stop the transmitter from sending the following character, $\overline{CTS}$ must be de-asserted before the middle of the last stop bit that is currently being sent. The Auto-$\overline{CTS}$ function reduces interrupts to the host system. When flow control is enabled, $\overline{CTS}$ level changes do not trigger host interrupts because the device automatically controls its own transmitter. Without Auto-$\overline{CTS}$, the transmitter sends any data present in the transmit FIFO and a receiver overrun error may result.



(1) When $\overline{CTS}$ is LOW, the transmitter keeps sending serial data out.
(2) When $\overline{CTS}$ goes HIGH before the middle of the last stop bit of the current character, the transmitter finishes sending the current
character, but it does not send the next character.
(3) When $\overline{CTS}$ goes from HIGH to LOW, the transmitter begins sending data again.

Figure 3. $\overline{CTS}$ functional timing

3 Software flow control

Software flow control is enabled through the Enhanced Features Register and the Modem Control Register. Different combinations of software flow control can be enabled by setting different combinations of EFR[3:0]. Table 1 shows software flow control options.

Table 1. Software flow control options (EFR[3:0])

EFR[3]	EFR[2]	EFR[1]	EFR[0]	TX, RX software flow control
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 and Xon2, Xoff1 and Xoff2
x	x	0	0	no receive flow control
x	x	1	0	receiver compares Xon1, Xoff1
x	x	0	1	receiver compares Xon2, Xoff2
1	0	1	1	transmit Xon1, Xoff1 receiver compares Xon1 or Xon2, Xoff1 or Xoff2
0	1	1	1	transmit Xon2, Xoff2 receiver compares Xon1 or Xon2, Xoff1 or Xoff2
1	1	1	1	transmit Xon1 and Xon2, Xoff1 and Xoff2 receiver compares Xon1 and Xon2, Xoff1 and Xoff2
0	0	1	1	no transmit flow control receiver compares Xon1 and Xon2, Xoff1 and Xoff2

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

• Xon Any function (MCR[5]): Receiving any character will resume operation after recognizing the Xoff character. It is possible that an Xon1 character is recognized as an Xon Any character, which could cause an Xon2 character to be written to the RX FIFO.

• Special character (EFR[5]): Incoming data is compared to Xoff2. Detection of the special character sets the Xoff interrupt (IIR[4]) but does not halt transmission. The Xoff interrupt is cleared by a read of the Interrupt Identification Register (IIR). The special character is transferred to the RX FIFO.

3.1 Receive flow control

When software flow control operation is enabled, UART will compare incoming data with Xoff1/Xoff2 programmed characters (in certain cases, Xoff1 and Xoff2 must be received sequentially). When the correct Xoff characters are received, transmission is halted after completing transmission of the current character. Xoff detection also sets IIR[4] (if enabled via IER[5]) and causes $\overline{\text{IRQ}}$ to go LOW.

To resume transmission, an Xon1/Xon2 character must be received (in certain cases Xon1 and Xon2 must be received sequentially). When the correct Xon characters are received, IIR[4] is cleared, and the Xoff interrupt disappears.

3.2 Transmit flow control

Xoff1/Xoff2 character is transmitted when the RX FIFO has passed the halt trigger level programmed in TCR[3:0], or the selectable trigger level in FCR[7:6].

Xon1/Xon2 character is transmitted when the RX FIFO reaches the resume trigger level programmed in TCR[7:4], or falls below the lower selectable trigger level in FCR[7:6].

The transmission of Xoff/Xon(s) follows the exact same protocol as transmission of an ordinary character from the FIFO. This means that even if the word length is set to be 5, 6, or 7 bits, then the 5, 6, or 7 least significant bits of Xoff1/Xoff2, Xon1/Xon2 will be transmitted. (Note that the transmission of 5, 6, or 7 bits of a character is seldom done, but this functionality is included to maintain compatibility with earlier designs.)

It is assumed that software flow control and hardware flow control will never be enabled simultaneously. Figure 4 shows an example of software flow control.

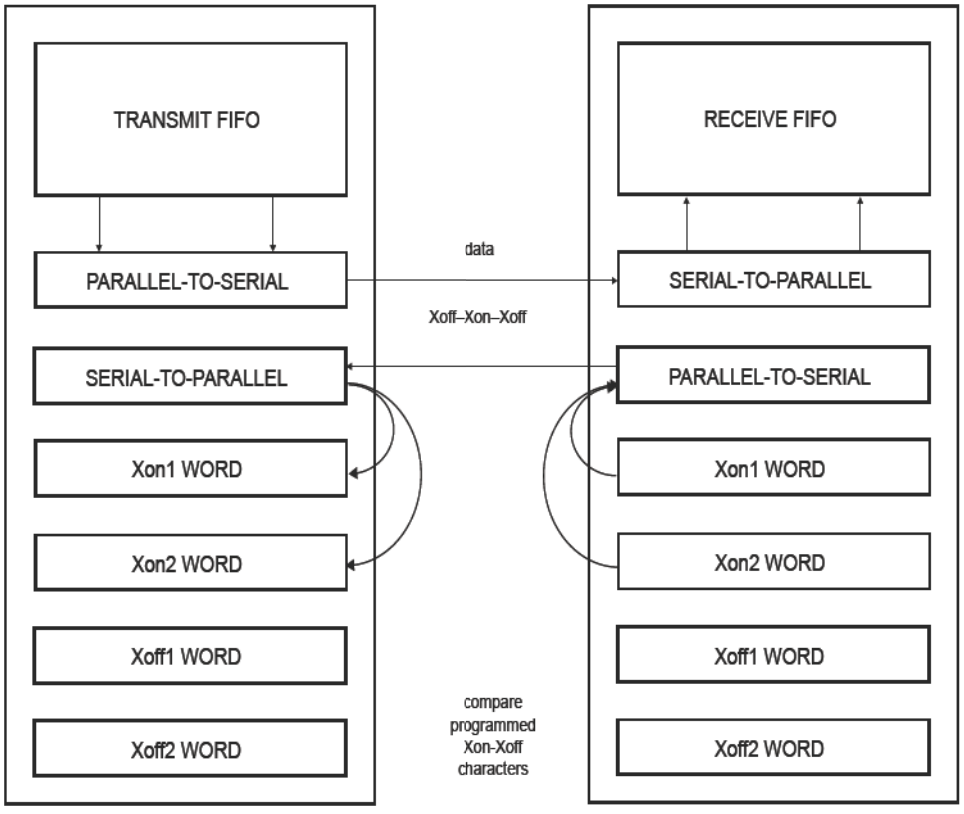


Figure 4. Example of software flow control

4. Hardware Reset, Power-On Reset (POR) and Software Reset

These three reset methods are identical and will reset the internal registers as indicated in Table 4.

Table 2 summarizes the state of register after reset.

Table 2. Register reset

Register	Reset state
Interrupt Enable Register	all bits cleared
Interrupt Identification Register	bit 0 is set; all other bits cleared
FIFO Control Register	all bits cleared
Line Control Register	reset to 0001 1101 (0x1D)
Modem Control Register	all bits cleared
Line Status Register	bit 5 and bit 6 set; all other bits cleared
Modem Status Register	bits 3:0 cleared; bits 7:4 input signals
Enhanced Features Register	all bits cleared
Receive Holding Register	pointer logic cleared
Transmit Holding Register	pointer logic cleared
Transmission Control Register	all bits cleared
Trigger Level Register	all bits cleared
Transmit FIFO level	reset to 0100 0000 (0x40)
Receive FIFO level	all bits cleared
I/O direction	all bits cleared
I/O interrupt enable	all bits cleared
I/O control	all bits cleared
Extra Features Control Register	all bits cleared

Remark: Registers DLL, DLH, SPR, XON1, XON2, XOFF1, XOFF2 are not reset by the top-level reset signal RESET, Software Reset, that is, they hold their initialization values during reset.

Table 3 summarizes the state of output signals after reset.

Table 3. Output signals after reset

Signal	Reset state
TX	HIGH
$\overline{\text{RTS}}$	HIGH
I/Os	inputs
$\overline{\text{IRQ}}$	HIGH by external pull-up

5 Interrupts

The UART has interrupt generation and prioritization (seven prioritized levels of interrupts) capability. The interrupt enable registers (IER and IOIntEna) enable each of the seven types of interrupts and the IRQ signal in response to an interrupt generation. When an interrupt is generated, the IIR indicates that an interrupt is pending and provides the type of interrupt through IIR[5:0]. Table 4 summarizes the interrupt control functions.

Table 4. Interrupt Source and Priority Level

IIR[5:0]	Priority level	Interrupt type	Interrupt source
00 0001	none	none	None
00 0110	1	receiver line status	Overrun Error (OE), Framing Error (FE), Parity Error (PE), or Break Interrupt (BI) errors occur in characters in the RX FIFO
00 1100	2	RX time-out	Stale data in RX FIFO
00 0100	2	RHR interrupt	Receive data ready (FIFO disable) or RX FIFO above trigger level (FIFO enable)
00 0010	3	THR interrupt	Transmit FIFO empty (FIFO disable) or TX FIFO passes above trigger level (FIFO enable)
00 0000	4	modem status	Change of state of modem input pins
11 0000	5	I/O pins	Input pins change of state
01 0000	6	Xoff interrupt	Receive Xoff character(s)/special character
10 0000	7	$\overline{\text{CTS}}$, $\overline{\text{RTS}}$	$\overline{\text{RTS}}$ pin or $\overline{\text{CTS}}$ pin change state from active (LOW) to inactive (HIGH)

It is important to note that for the framing error, parity error, and break conditions, Line Status Register bit 7 (LSR[7]) generates the interrupt. LSR[7] is set when there is an error anywhere in the RX FIFO, and is cleared only when there are no more errors remaining in the FIFO. LSR[4:2] always represent the error status for the received character at the top of the RX FIFO. Reading the RX FIFO updates LSR[4:2] to the appropriate status for the new character at the top of the FIFO. If the RX FIFO is empty, then LSR[4:2] are all zeros.

For the Xoff interrupt, if an Xoff flow character detection caused the interrupt, the interrupt is cleared by an Xon flow character detection. If a special character detection caused the interrupt, the interrupt is cleared by a read of the IIR.

5.1 Interrupt mode operation

In Interrupt mode (if any bit of IER[3:0] is 1) the host is informed of the status of the receiver and transmitter by an interrupt signal, $\overline{IRQ}$. Therefore, it is not necessary to continuously poll the Line Status Register (LSR) to see if any interrupt needs to be serviced. Figure 5 shows Interrupt mode operation.

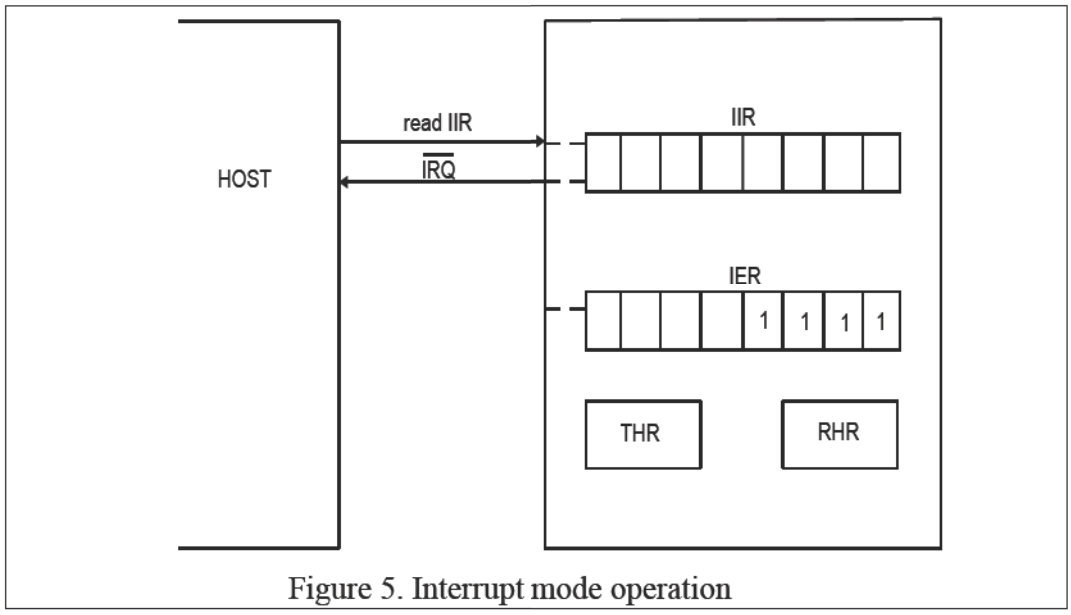


Figure 5. Interrupt mode operation

5.2 Polled mode operation

In Polled mode (IER[3:0] = 0000) the status of the receiver and transmitter can be checked by polling the Line Status Register (LSR). This mode is an alternative to the FIFO Interrupt mode of operation where the status of the receiver and transmitter is automatically known by means of interrupts sent to the CPU. Figure 6 shows FIFO Polled mode operation.

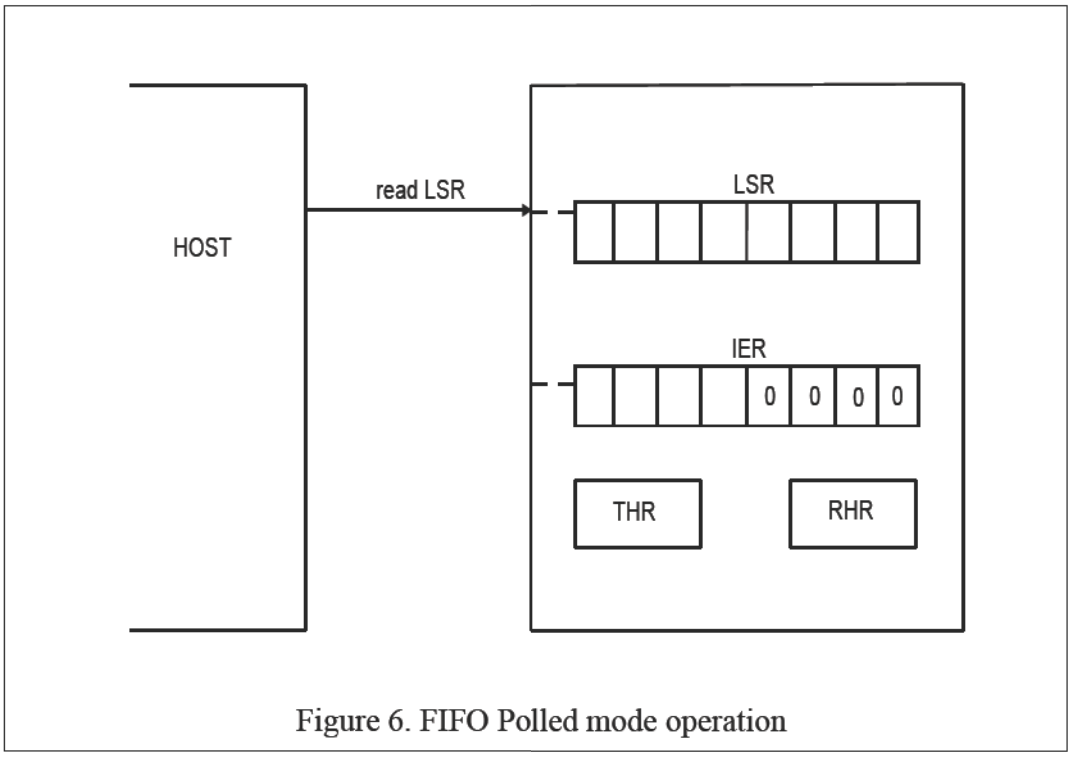


Figure 6. FIFO Polled mode operation

6 Sleep mode

Sleep mode is an enhanced feature of the UART. It is enabled when EFR[4], the enhanced functions bit, is set and when IER[4] is set. Sleep mode is entered when:

- The serial data input line, RX, is idle (see Section 7 "Break and time-out conditions").
- The TX FIFO and TX shift register are empty.
- There are no interrupts pending except THR.

Remark: Sleep mode will not be entered if there is data in the RX FIFO.

In Sleep mode, the clock to the UART is stopped. Since most registers are clocked using these clocks, the power consumption is greatly reduced. The UART will wake up when any change is detected on the RX line, when there is any change in the state of the modem input pins, or if data is written to the TX FIFO.

Remark: Writing to the divisor latches DLL and DLH to set the baud clock must not be done during Sleep mode. Therefore, it is advisable to disable Sleep mode using IER[4] before writing to DLL or DLH.

7 Break and time-out conditions

When the UART receives a number of characters and these data are not enough to set off the receive interrupt (because they do not reach the receive trigger level), the UART will generate a time-out interrupt instead, 4 character times after the last character is received. The time-out counter will be reset at the center of each stop bit received or each time the receive FIFO is read.

A break condition is detected when the RX pin is pulled LOW for a duration longer than the time it takes to send a complete character plus start, stop and parity bits. A break condition can be sent by setting LCR[6], when this happens the TX pin will be pulled LOW until LSR[6] is cleared by the software.

8 Programmable baud rate generator

The UART contains a programmable baud rate generator that takes any clock input and divides it by a divisor in the range between 1 and (2^{16} - 1). An additional divide-by-4 prescaler is also available and can be selected by MCR[7], as shown in Figure 7. The formula for the baud rate is:

$$\text{Baud rate} = \frac{\left(\frac{\text{XTAL1 crystal input frequency}}{\text{prescaler}}\right)}{\text{divisor x sample rate}}$$

where:

prescaler = 1, when MCR[7] is set to logic 0 after reset (divide-by-1 clock selected)

prescaler = 4, when MCR[7] is set to logic 1 after reset (divide-by-4 clock selected).

Divisor = {DLH, DLL}

Sample rate = 16 - SCR + CPRN

Remark: The default value of prescaler after reset is divide-by-1.

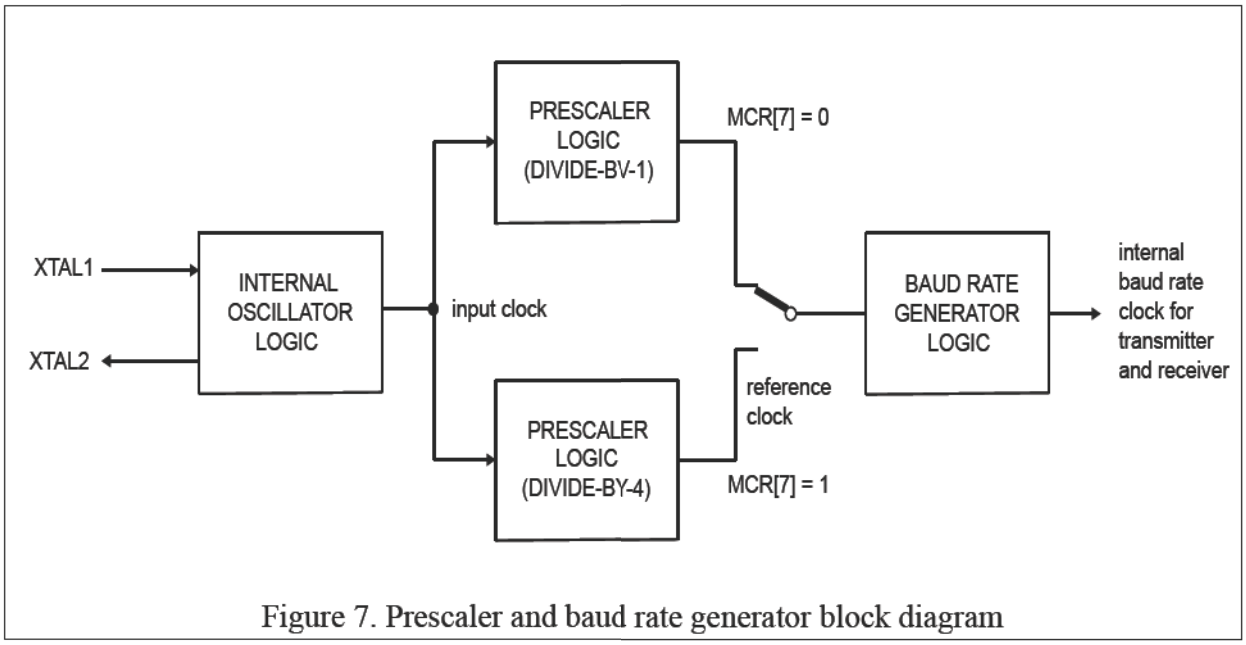


Figure 7. Prescaler and baud rate generator block diagram

DLL and DLH must be written to in order to program the baud rate. DLL and DLH are the least significant and most significant byte of the baud rate divisor. If DLL and DLH are both zero, the UART is effectively disabled, as no baud clock will be generated.

Remark: The programmable baud rate generator is provided to select both the transmit and receive clock rates.

Table 5 to 8 show the baud rate and divisor correlation for crystal with frequency 1.8432 MHz, 3.072 MHz, 14.74926 MHz, and 24MHz respectively.

Figure 8 shows the crystal clock circuit reference.

Table 5. Baud rates using a 1.8432 MHz crystal

Desired baud rate (bit/s)	Divisor used to generate 16x clock	Sample rate	Percent error difference between desired and actual
50	2304	16	0
75	1536	16	0
110	1047	16	0.026
134.5	857	16	0.058
150	768	16	0
300	384	16	0
600	192	16	0
1200	96	16	0
1800	64	16	0
2000	46	20	0.617
2400	48	16	0
3600	32	16	0
4800	24	16	0
7200	16	16	0
9600	12	16	0
19200	6	16	0
38400	3	16	0
56000	2	16	2.86

Table 6. Baud rates using a 3.072 MHz crystal

Desired baud rate (bit/s)	Divisor used to generate 16x clock	Sample rate	Percent error difference between desired and actual
50	2304	16	0
75	2560	16	0
110	1745	16	0.026
134.5	1428	16	0.034
150	1280	16	0
300	640	16	0
600	320	16	0
1200	160	16	0
1800	90	19	0.195
2000	96	16	0
2400	80	16	0
3600	45	19	0.195
4800	40	16	0
7200	25	17	0.392
9600	20	16	0
19200	10	16	0
38400	5	16	0

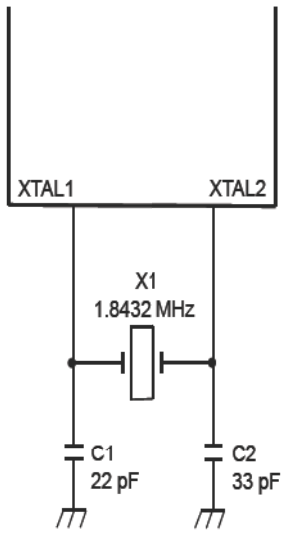


Figure 8. Crystal oscillator circuit reference

Table 7. Baud rates using a 14.74926 MHz crystal

Desired baud rate (bit/s)	Divisor used to generate 16x clock	Sample rate	Percent error difference between desired and actual
38400	24	16	0.025
56000	11	24	0.235
57600	16	16	0.025
115200	8	16	0.025
153600	6	16	0.025
921600	1	16	0.025

Table 8. Baud rates using a 24 MHz crystal

Desired baud rate (bit/s)	Divisor used to generate 16x clock	Sample rate	Percent error difference between desired and actual
4800	250	20	0
7200	159	21	0.17
25000	48	20	0
38400	25	25	0
57600	22	19	0.32
115200	8	26	0.16
225000	6	18	1.2
400000	3	20	0
921600	1	26	0.16
1000000	1	24	0

9. RS-485 features

9.1 Auto RS-485 RTS control

Normally the $\overline{RTS}$ pin is controlled by MCR bit 1, or if hardware flow control is enabled, the logic state of the $\overline{RTS}$ pin is controlled by the hardware flow control circuitry. EFCR register bit 4 will take the precedence over the other two modes; once this bit is set, the transmitter will control the state of the $\overline{RTS}$ pin. The transmitter automatically asserts the $\overline{RTS}$ pin (logic 0) once the host writes data to the transmit FIFO, and de-asserts $\overline{RTS}$ pin (logic 1) once the last bit of the data has been transmitted.

To use the auto RS-485 $\overline{RTS}$ mode the software would have to disable the hardware flow control function.

9.2 RS-485 RTS output inversion

EFCR bit 5 reverses the polarity of the $\overline{RTS}$ pin if the UART is in auto RS-485 $\overline{RTS}$ mode. When the transmitter has data to be sent it de-asserts the $\overline{RTS}$ pin (logic 1), and when the last bit of the data has been sent out the transmitter asserts the $\overline{RTS}$ pin (logic 0).

9.3 Auto RS-485

EFCR bit 0 is used to enable the RS-485 mode (multidrop or 9-bit mode). In this mode of operation, a 'master' station transmits an address character followed by data characters for the addressed 'slave' stations. The slave stations examine the received data and interrupt the controller if the received character is an address character (parity bit = 1).

To use the auto RS-485 RTS mode the software would have to disable the hardware flow control function.

9.3.1 Normal multidrop mode

The 9-bit mode in EFCR (bit 0) is enabled, but not Special Character Detect (EFR bit 5). The receiver is set to Force Parity 0 (LCR[5:3] = 111) in order to detect address bytes.

With the receiver initially disabled, it ignores all the data bytes (parity bit = 0) until an address byte is received (parity bit = 1). This address byte will cause the UART to set the parity error. The UART will generate a line status interrupt (IER bit 2 must be set to '1' at this time), and at the same time puts this address byte in the RX FIFO. After the controller examines the byte it must make a decision whether or not to enable the receiver; it should enable the receiver if the address byte addresses its ID address, and must not enable the receiver if the address byte does not address its ID address.

If the controller enables the receiver, the receiver will receive the subsequent data until being disabled by the controller after the controller has received a complete message from the 'master' station. If the controller does not disable the receiver after receiving a message from the 'master' station, the receiver will generate a parity error upon receiving another address byte. The controller then determines if the address byte addresses its ID address, if it is not, the controller then can disable the receiver. If the address byte addresses the 'slave' ID address, the controller take no further action; the receiver will receive the subsequent data.

9.3.2 Auto address detection

If Special Character Detect is enabled (EFR[5] is set and XOFF2 contains the address byte) the receiver will try to detect an address byte that matches the programmed character in XOFF2. If the received byte is a data byte or an address byte that does not match the programmed character in XOFF2, the receiver will discard these data. Upon receiving an address byte that matches the XOFF2 character, the receiver will be automatically enabled if not already enabled, and the address character is pushed into the RX FIFO along with the parity bit (in place of the parity error bit). The receiver also generates a line status interrupt (IER bit 2 must be set to 1 at this time). The receiver will then receive the subsequent data from the 'master' station until being disabled by the controller after having received a message from the 'master' station.

If another address byte is received and this address byte does not match XOFF2 character, the receiver will be automatically disabled and the address byte is ignored. If the address byte matches XOFF2 character, the receiver will put this byte in the RX FIFO along with the parity bit in the parity error bit (LSR[2]).

10. I²C-bus Interface

The I²C-bus interface is compliant with the Standard-mode and Fast-mode I²C-bus specifications. The I²C-bus interface consists of two lines: serial data (SDA) and serial clock (SCL). In the Standard-mode, the serial clock and serial data can go up to 100 kbps and in the Fast-mode, the serial clock and serial data can go up to 400 kbps. The first byte sent by an I²C-bus master contains a start bit (SDA transition from HIGH to LOW when SCL is HIGH), 7-bit slave address and whether it is a read or write transaction. The next byte is the sub-address that contains the address of the register to access. The UART responds to each write with an acknowledge (SDA driven LOW by UART for one clock cycle when SCL is HIGH). If the TX FIFO is full, the UART will respond with a negative acknowledge (SDA driven HIGH by UART for one clock cycle when SCL is HIGH) when the CPU tries to write to the TX FIFO. The last byte sent by an I²C-bus master is a stop bit (SDA transition from LOW to HIGH when SCL is HIGH). See Figures 8 - 10 below. For complete details, see the I²C-bus specifications.

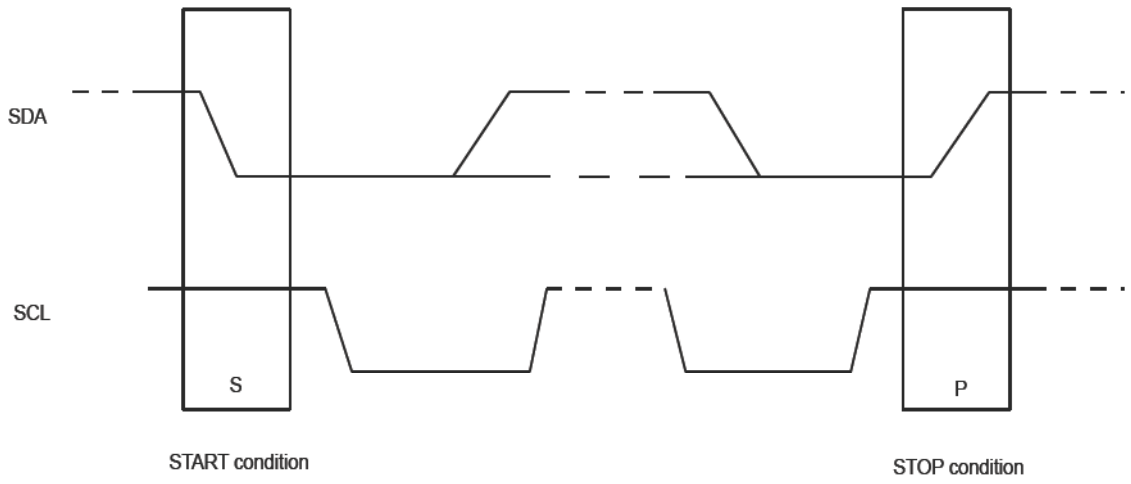
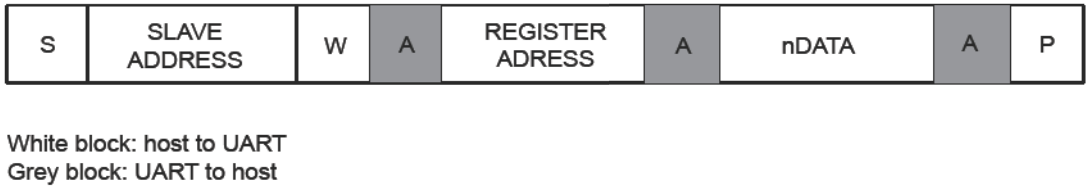


Figure 9. I²C Start and Stop Conditions



White block: host to UART
Grey block: UART to host

Figure 10. Master writes to slave (UART)

White block: host to UART
Grey block: UART to host

Figure 11. Master reads from slave (UART)

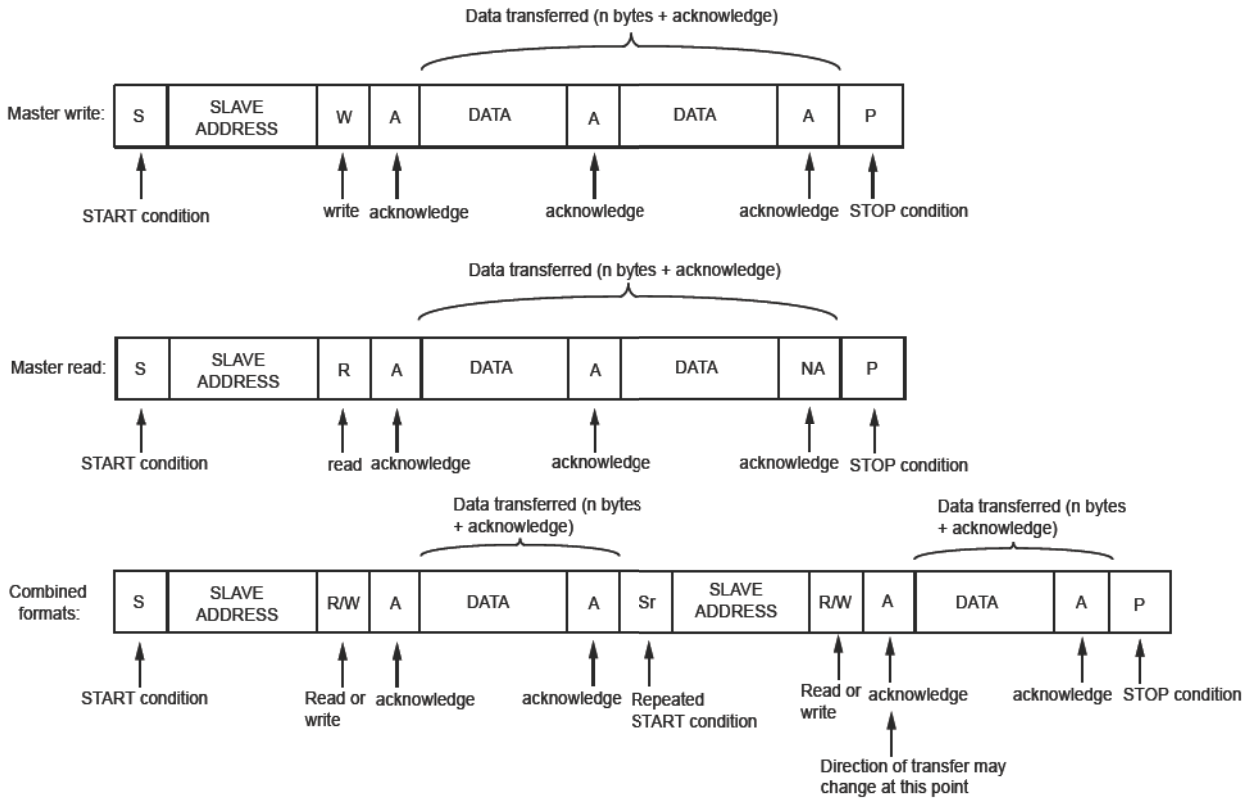


Figure 12. I²C data formats

10.1 I²C-bus Addressing

There could be many devices on the I²C-bus. To distinguish itself from the other devices on the I²C-bus, there are eight possible slave addresses that can be selected for the UART using the A1 and A0 address lines. Table 9 below shows the different addresses that can be selected. Note that there are two different ways to select each I2C address.

Table 9: I²C Address Map

A1	A0	I²C ADDRESS
V_{DD}	V_{DD}	0x90 (0110 000X)
V_{DD}	V_{SS}	0x92 (0110 001X)
V_{DD}	SCL	0x94 (0110 010X)
V_{DD}	SDA	0x96 (0110 011X)
V_{SS}	V_{DD}	0x98 (0110 100X)
V_{SS}	V_{SS}	0x9A (0110 101X)
V_{SS}	SCL	0x9C (0110 110X)
V_{SS}	SDA	0x9E (0110 111X)
SCL	V_{DD}	0xA0 (0110 000X)
SCL	V_{SS}	0xA2 (0110 001X)
SCL	SCL	0xA4 (0110 010X)
SCL	SDA	0xA6 (0110 011X)
SDA	V_{DD}	0xA8 (0110 100X)
SDA	V_{SS}	0xAA (0110 101X)
SDA	SCL	0xAC (0110 110X)
SDA	SDA	0xAE (0110 111X)

An I²C sub-address is sent by the I²C master following the slave address. The sub-address contains the UART register address being accessed. A read or write transaction is determined by bit-0 of the slave address (HIGH = Read, LOW = Write). Table 10 below lists the functions of the bits in the I²C sub-address.

Table 10: I²C Sub-Address (Register Address)

Bit	Function
7	Reserved
6:3	UART Internal Register Address A3:A0
2:1	UART Channel Select '00' – UART Channel A other values are reserved
0	Reserved

After the last read or write transaction, the I²C-bus master will set the SCL signal back to its idle state (HIGH).

11. SPI Bus Interface

The SPI interface consists of four lines: serial clock (SCL), chip select (CS#), slave output (SO) and slave input (SI). The serial clock, slave output and slave input can be as fast as 18 MHz at 3.3V. To access the device in the SPI mode, the CS# signal for the UART is asserted by the SPI master, then the SPI master starts toggling the SCL signal with the appropriate transaction information. The first bit sent by the SPI master includes whether it is a read or write transaction and the UART register being accessed. See Table 11 below.

Table 11: SPI First Byte Format

Bit	Function
7	Read/Write# Logic 1 = Read Logic 0 = Write
6:3	UART Internal Register Address A3:A0
2:1	UART Channel Select '00' = UART Channel A Other values are reserved
0	Reserved

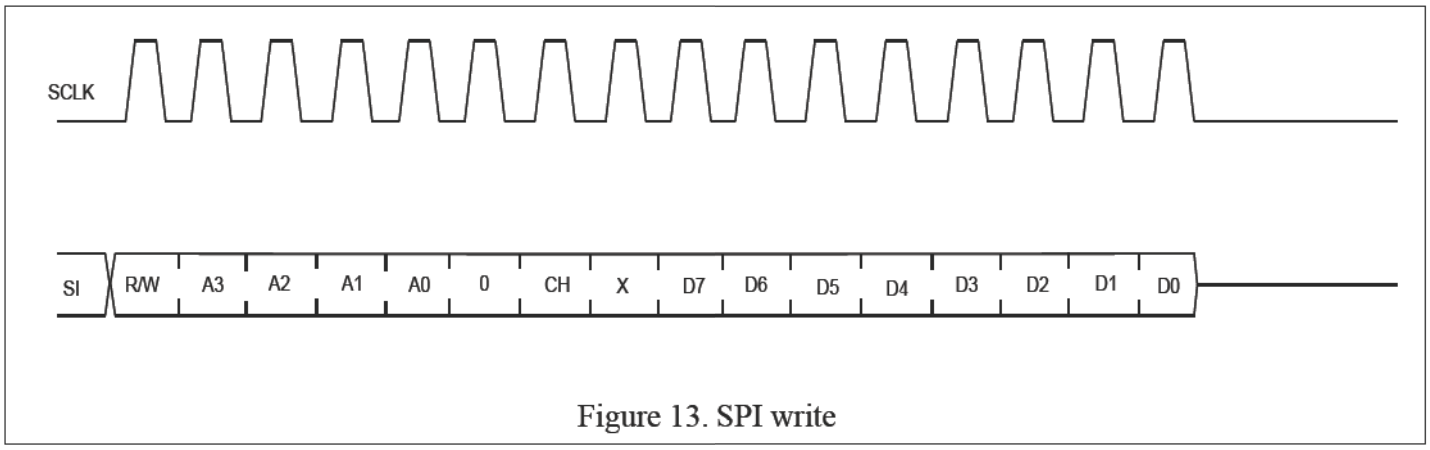


Figure 13. SPI write

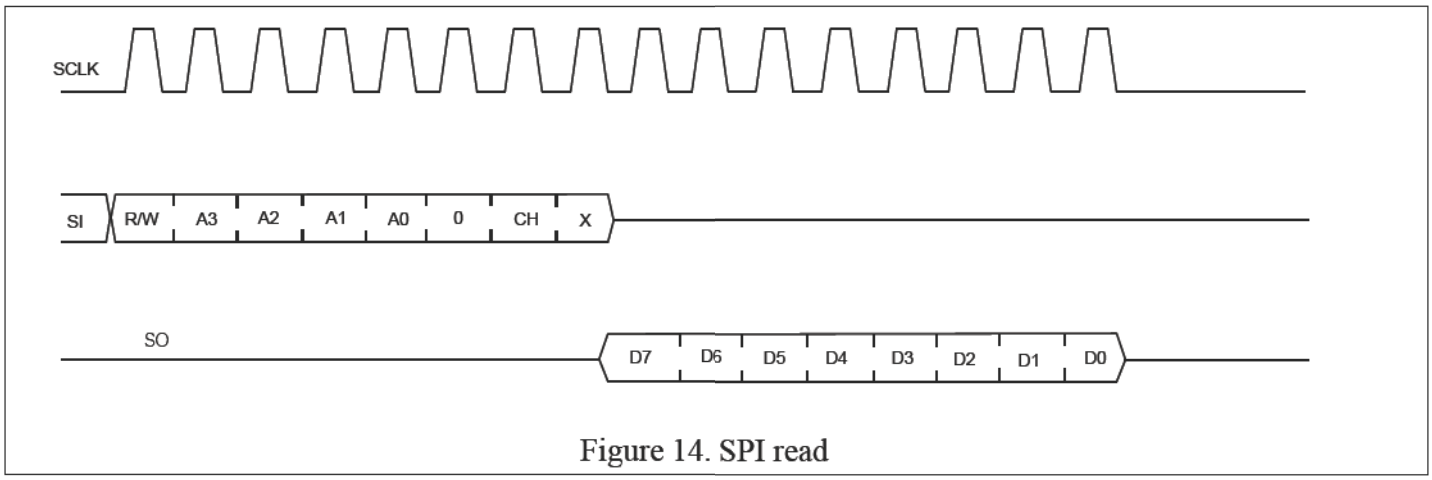


Figure 14. SPI read

The 64 byte TX FIFO can be loaded with data or 64 byte RX FIFO data can be unloaded in one SPI write or read sequence.

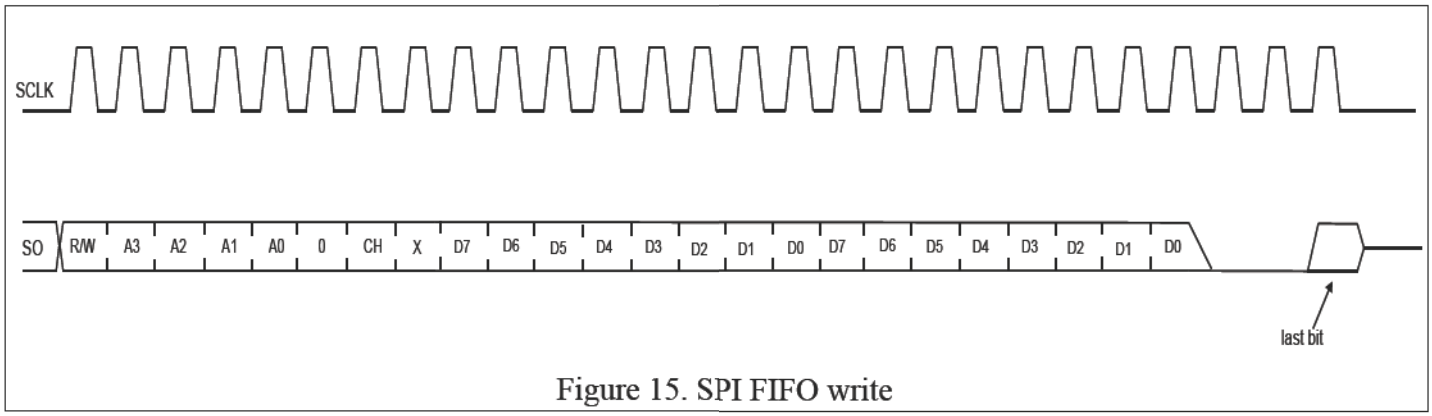


Figure 15. SPI FIFO write

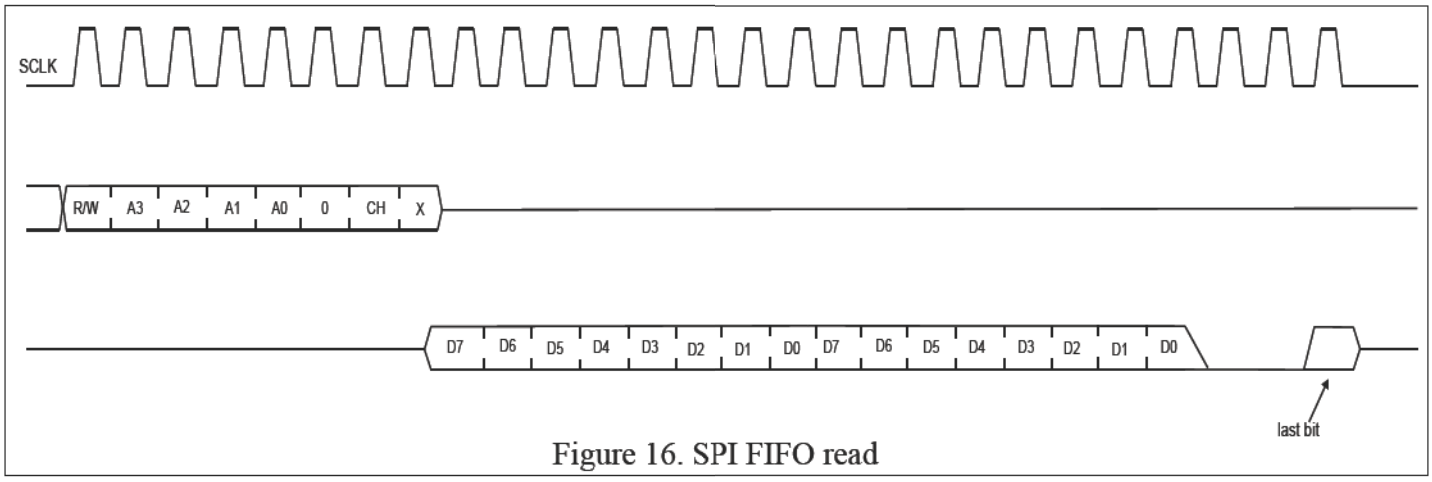


Figure 16. SPI FIFO read

After the last read or write transaction, the SPI master will set the SCL signal back to its idle state (LOW).

12 Infrared Mode

The UART includes the infrared encoder and decoder compatible to the IrDA (Infrared Data Association) version 1.0 and 1.1. The IrDA 1.0 standard that stipulates the infrared encoder sends out a 3/16 of a bit wide HIGH-pulse for each “0” bit in the transmit data stream with a data rate up to 115.2 Kbps. For the IrDA 1.1 standard, the infrared encoder sends out a 1/4 of a bit time wide HIGH-pulse for each "0" bit in the transmit data stream with a data rate up to 1.152 Mbps. This signal encoding reduces the on-time of the infrared LED, hence reduces the power consumption. See Figure 16 below.

The infrared encoder and decoder are enabled by setting MCR register bit-6 to a ‘1’. With this bit enabled, the infrared encoder and decoder is compatible to the IrDA 1.0 standard. For the infrared encoder and decoder to be compatible to the IrDA 1.1 standard, EFCR bit-7 will also need to be set to a ’1’. When the infrared feature is enabled, the transmit data output, TX, idles LOW. Likewise, the RX input also idles LOW, see Figure 16.

The wireless infrared decoder receives the input pulse from the infrared sensing diode on the RX pin. Each time it senses a light pulse, it returns a logic 1 to the data bit stream.

The UART can be in the infrared mode upon power-up if the ENIR# pin is LOW. After power-up, the infrared mode can be controlled via MCR bit-6.

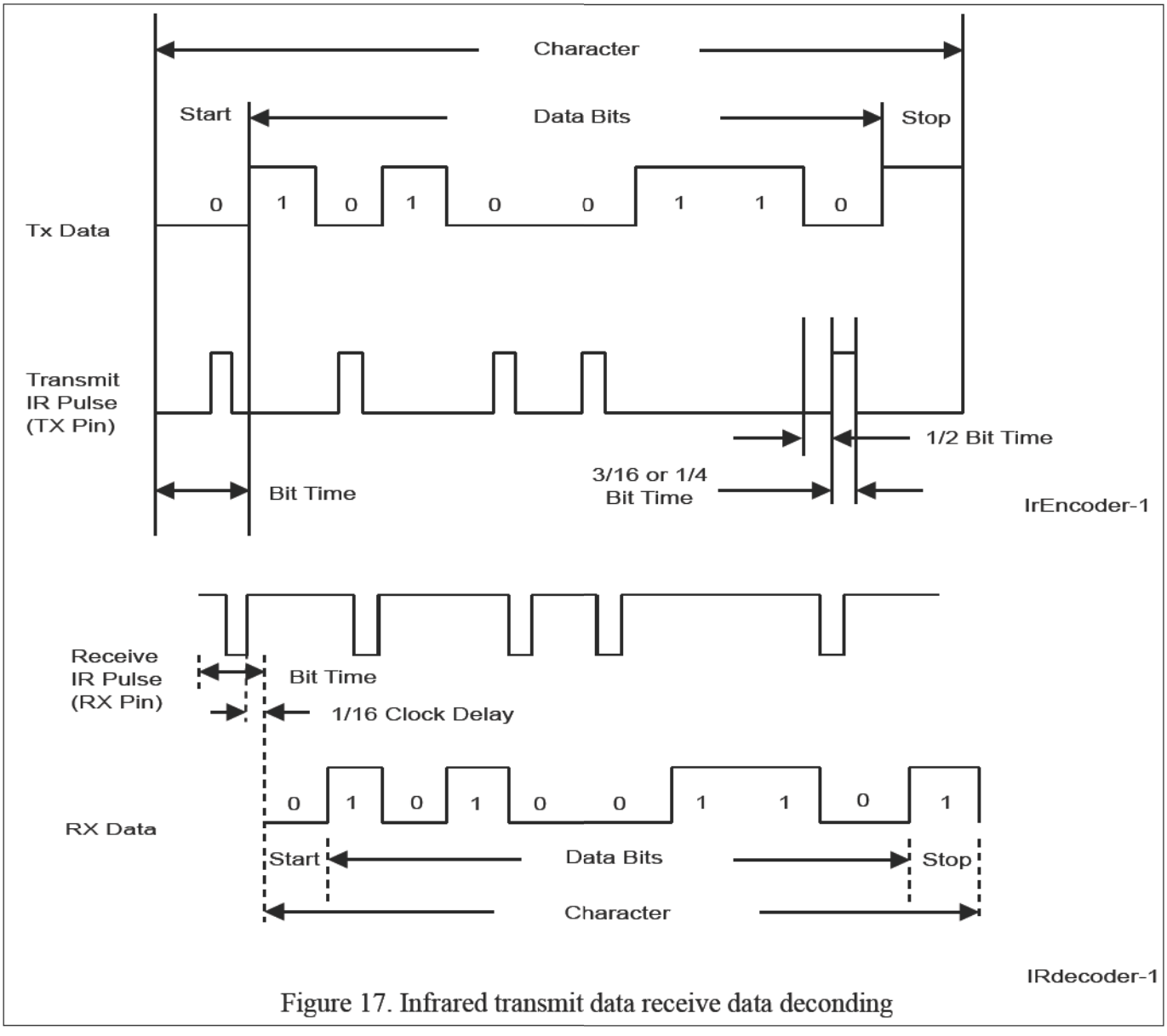


Figure 17. Infrared transmit data receive data deconding