Chapter 14
UART Modules

This chapter describes the use of the universal asynchronous/synchronous receiver/transmitters (UARTs) implemented on the MCF5307 and includes programming examples. All references to UART refer to one of these modules.

14.1 Overview

The MCF5307 contains two independent UARTs. Each UART can be clocked by BCLKO, eliminating the need for an external crystal. As Figure 14-1 shows, each UART module interfaces directly to the CPU and consists of the following:

- Serial communication channel
- Programmable transmitter and receiver clock generation
- Internal channel control logic
- Interrupt control logic

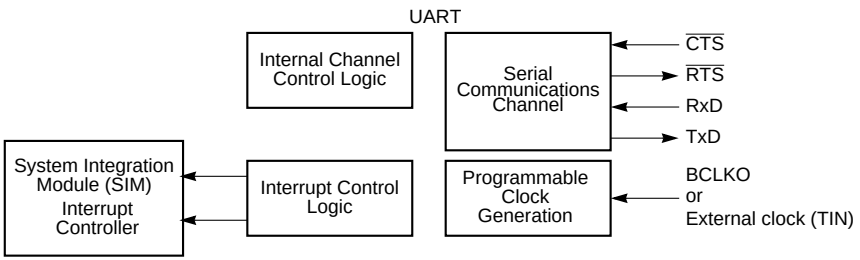


Figure 14-1. Simplified Block Diagram

The serial communication channel provides a full-duplex asynchronous/synchronous receiver and transmitter deriving an operating frequency from BCLKO or an external clock using the timer pin. The transmitter converts parallel data from the CPU to a serial bit stream, inserting appropriate start, stop, and parity bits. It outputs the resulting stream on the channel transmitter serial data output (TxD). See Section 14.5.2.1, “Transmitting.”

The receiver converts serial data from the channel receiver serial data input (RxD) to parallel format, checks for a start, stop, and parity bits, or break conditions, and transfers the assembled character onto the bus during read operations. The receiver may be polled- or interrupt-driven. See Section 14.5.2.2, “Receiver.”

14.2 Serial Module Overview

The MCF5307 contains two independent UART modules, whose features are as follows:

- Each can be clocked by BCLKO, eliminating a need for an external crystal
- Full-duplex asynchronous/synchronous receiver/transmitter channel
- Quadruple-buffered receiver
- Double-buffered transmitter
- Independently programmable receiver and transmitter clock sources
- Programmable data format:
  - 5–8 data bits plus parity
  - Odd, even, no parity, or force parity
  - One, one-and-a-half, or two stop bits
- Each channel programmable to normal (full-duplex), automatic echo, local loop-back, or remote loop-back mode
- Automatic wake-up mode for multidrop applications
- Four maskable interrupt conditions
- UART0 and UART1 have interrupt capability to DMA channels 2 and 3, respectively, when either the RxRDY or FFULL bit is set in the USR.
- Parity, framing, and overrun error detection
- False-start bit detection
- Line-break detection and generation
- Detection of breaks originating in the middle of a character
- Start/end break interrupt/status

14.3 Register Descriptions

This section contains a detailed description of each register and its specific function. Flowcharts in Section 14.5.6, “Programming,” describe basic UART module programming. The operation of the UART module is controlled by writing control bytes into the appropriate registers. Table 14-1 is a memory map for UART module registers.

Table 14-1. UART Module Programming Model

MBAR Offset		[31:24]	[23:16]	[15:8]	[7:0]
UART0	UART1				
0x1C0	0x200	UART mode registers[1]—(UMR1n) [p. 14-4], (UMR2n) [p. 14-6]	—		
0x1C4	0x204	(Read) UART status registers—(USRn) [p. 14-7]	—		
		(Write) UART clock-select register[1]—(UCSRn) [p. 14-8]	—		
0x1C8	0x208	(Read) Do not access[2]	—		
		(Write) UART command registers—(UCRn) [p. 14-9]	—		
0x1CC	0x20C	(UART/Read) UART receiver buffers—(URBn) [p. 14-11]	—		
		(UART/Write) UART transmitter buffers—(UTBn) [p. 14-11]	—		
0x1D0	0x210	(Read) UART input port change registers—(UIPCRn) [p. 14-12]	—		
		(Write) UART auxiliary control registers[1]—(UACRn) [p. 14-12]	—		
0x1D4	0x214	(Read) UART interrupt status registers—(UISRn) [p. 14-13]	—		
		(Write) UART interrupt mask registers—(UIMRn) [p. 14-13]	—		
0x1D8	0x218	UART divider upper registers—(UDUn) [p. 14-14]	—		
0x1DC	0x21C	UART divider lower registers—(UDLn) [p. 14-14]	—		

Table 14-1. UART Module Programming Model (Continued)

MBAR Offset		[31:24]	[23:16]	[15:8]	[7:0]
UART0	UART1				
0x1E0–0x1EC	0x220–0x22C	Do not access[2]	—		
0x1F0	0x230	UART interrupt vector register—(UIVRn) [p. 14-15]	—		
0x1F4	0x234	(Read) UART input port registers—(UIPn) [p. 14-15]	—		
		(Write) Do not access[2]	—		
0x1F8	0x238	(Read) Do not access[2]	—		
		(Write) UART output port bit set command registers—(UOP1n[3]) [p. 14-15]	—		
0x1FC	0x23C	(Read) Do not access[2]	—		
		(Write) UART output port bit reset command registers—(UOP0n[3]) [p. 14-15]	—		

1 UMR1n, UMR2n, and UCSRn should be changed only after the receiver/transmitter is issued a software reset command. That is, if channel operation is not disabled, undesirable results may occur.

2 This address is for factory testing. Reading this location results in undesired effects and possible incorrect transmission or reception of characters. Register contents may also be changed.

3 Address-triggered commands

NOTE:

UART registers are accessible only as bytes. Although external masters cannot access on-chip memories or MBAR, they can access any UART registers.

14.3.1 UART Mode Registers 1 (UMR1*n*)

The UART mode registers 1 (UMR1*n*) control configuration. UMR1*n* can be read or written when the mode register pointer points to it, at RESET or after a RESET MODE REGISTER POINTER command using UCR*n*[MISC]. After UMR1*n* is read or written, the pointer points to UMR2*n*.

	7	6	5	4	3	2	1	0
Field	RxRTS	RxIRQ/FFULL	ERR	PM		PT	B/C	
Reset	0000_0000							
R/W	R/W							
Address	MBAR + 0x1C0 (UART0), 0x200 (UART1). After UMR1n is read or written, the pointer points to UMR2n.							

Figure 14-2. UART Mode Registers 1 (UMR1*n*)

Table 14-2 describes UMR1*n* fields.

Table 14-2. UMR1*n* Field Descriptions

Bits	Name	Description
7	RxRTS	Receiver request-to-send. Allows the $\overline{\text{RTS}}$ output to control the $\overline{\text{CTS}}$ input of the transmitting device to prevent receiver overrun. If both the receiver and transmitter are incorrectly programmed for $\overline{\text{RTS}}$ control, $\overline{\text{RTS}}$ control is disabled for both. Transmitter RTS control is configured in UMR2n[TxRTS]. 0 The receiver has no effect on $\overline{\text{RTS}}$. 1 When a valid start bit is received, $\overline{\text{RTS}}$ is negated if the UART's FIFO is full. $\overline{\text{RTS}}$ is reasserted when the FIFO has an empty position available.
6	RxIRQ/ FFULL	Receiver interrupt select. 0 RxRDY is the source that generates IRQ. 1 FFULL is the source that generates IRQ.
5	ERR	Error mode. Configures the FIFO status bits, USRn[RB,FE,PE]. 0 Character mode. The USRn values reflect the status of the character at the top of the FIFO. ERR must be 0 for correct A/D flag information when in multidrop mode. 1 Block mode. The USRn values are the logical OR of the status for all characters reaching the top of the FIFO because the last RESET ERROR STATUS command for the channel was issued. See Section 14.3.5, “UART Command Registers (UCRn).”
4–3	PM	Parity mode. Selects the parity or multidrop mode for the channel. The parity bit is added to the transmitted character, and the receiver performs a parity check on incoming data. The value of PM affects PT, as shown below.
2	PT	Parity type. PM and PT together select parity type (PM = 0x) or determine whether a data or address character is transmitted (PM = 11).
1–0	B/C	Bits per character. Select the number of data bits per character to be sent. The values shown do not include start, parity, or stop bits. 00 5 bits 01 6 bits 10 7 bits 11 8 bits

PM	Parity Mode	Parity Type (PT= 0)	Parity Type (PT= 1)
00	With parity	Even parity	Odd parity
01	Force parity	Low parity	High parity
10	No parity	n/a	
11	Multidrop mode	Data character	Address character

14.3.2 UART Mode Register 2 (UMR2*n*)

UART mode registers 2 (UMR2*n*) control UART module configuration. UMR2*n* can be read or written when the mode register pointer points to it, which occurs after any access to UMR1*n*. UMR2*n* accesses do not update the pointer.

	7	6	5	4	3	0
Field	CM		TxRTS	TxCTS	SB	
Reset	0000_0000					
R/W	R/W					
Address	MBAR + 0x1C0, 0x200. After UMR1n is read or written, the pointer points to UMR2n.					

Figure 14-3. UART Mode Register 2 (UMR2*n*)

Table 14-3 describes UMR2*n* fields.

Table 14-3. UMR2*n* Field Descriptions

Bits	Name	Description
7–6	CM	Channel mode. Selects a channel mode. Section 14.5.3, “Looping Modes,” describes individual modes. 00 Normal 01 Automatic echo 10 Local loop-back 11 Remote loop-back
5	TxRTS	Transmitter ready-to-send. Controls negation of $\overline{\text{RTS}}$ to automatically terminate a message transmission. Attempting to program a receiver and transmitter in the same channel for $\overline{\text{RTS}}$ control is not permitted and disables $\overline{\text{RTS}}$ control for both. 0 The transmitter has no effect on $\overline{\text{RTS}}$. 1 In applications where the transmitter is disabled after transmission completes, setting this bit automatically clears UOP[RTS] one bit time after any characters in the channel transmitter shift and holding registers are completely sent, including the programmed number of stop bits.
4	TxCTS	Transmitter clear-to-send. If both TxCTS and TxRTS are enabled, TxCTS controls the operation of the transmitter. 0 $\overline{\text{CTS}}$ has no effect on the transmitter. 1 Enables clear-to-send operation. The transmitter checks the state of $\overline{\text{CTS}}$ each time it is ready to send a character. If $\overline{\text{CTS}}$ is asserted, the character is sent; if it is negated, the channel TxD remains in the high state and transmission is delayed until $\overline{\text{CTS}}$ is asserted. Changes in $\overline{\text{CTS}}$ as a character is being sent do not affect its transmission.

Table 14-3. UMR2*n* Field Descriptions (Continued)

<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>3–0</td><td>SB</td><td>Stop-bit length control. Selects the length of the stop bit appended to the transmitted character. Stop-bit lengths of 9/16th to 2 bits are programmable for 6–8 bit characters. Lengths of 1 1/16th to 2 bits are programmable for 5-bit characters. In all cases, the receiver checks only for a high condition at the center of the first stop-bit position, that is, one bit time after the last data bit or after the parity bit, if parity is enabled. If an external 1x clock is used for the transmitter, clearing bit 3 selects one stop bit and setting bit 3 selects 2 stop bits for transmission.
<table>
<tr><th>SB</th><th>5 Bits</th><th>6–8 Bits</th><th>SB</th><th>5 Bits</th><th>6–8 Bits</th><th>SB</th><th>5–8 Bits</th><th>SB</th><th>5–8 Bits</th></tr>
<tr><td>0000</td><td>1.063</td><td>0.563</td><td>0100</td><td>1.313</td><td>0.813</td><td>1000</td><td>1.563</td><td>1100</td><td>1.813</td></tr>
<tr><td>0001</td><td>1.125</td><td>0.625</td><td>0101</td><td>1.375</td><td>0.875</td><td>1001</td><td>1.625</td><td>1101</td><td>1.875</td></tr>
<tr><td>0010</td><td>1.188</td><td>0.688</td><td>0110</td><td>1.438</td><td>0.938</td><td>1010</td><td>1.688</td><td>1110</td><td>1.938</td></tr>
<tr><td>0011</td><td>1.250</td><td>0.750</td><td>0111</td><td>1.500</td><td>1.000</td><td>1011</td><td>1.750</td><td>1111</td><td>2.000</td></tr>
</table></td></tr>
</table>

14.3.3 UART Status Registers (USR*n*)

The USR*n*, Figure 14-4, shows status of the transmitter, the receiver, and the FIFO.

	7	6	5	4	3	2	1	0
Field	RB	FE	PE	OE	TxEMP	TxRDY	FFULL	RxRDY
Reset	0000_0000							
R/W	Read only							
Address	MBAR + 0x1C4 (USR0), 0x204 (USR1)							

Figure 14-4. UART Status Register (USR*n*)

Table 14-4 describes USR*n* fields.

Table 14-4. USR*n* Field Descriptions

Bits	Name	Description
7	RB	Received break. The received break circuit detects breaks that originate in the middle of a received character. However, a break in the middle of a character must persist until the end of the next detected character time. 0 No break was received. 1 An all-zero character of the programmed length was received without a stop bit. RB is valid only when RxRDY = 1. Only a single FIFO position is occupied when a break is received. Further entries to the FIFO are inhibited until RxD returns to the high state for at least one-half bit time, which is equal to two successive edges of the UART clock.
6	FE	Framing error. 0 No framing error occurred. 1 No stop bit was detected when the corresponding data character in the FIFO was received. The stop-bit check occurs in the middle of the first stop-bit position. FE is valid only when RxRDY = 1.

Table 14-4. USR*n* Field Descriptions (Continued)

Bits	Name	Description
5	PE	Parity error. Valid only if RxRDY = 1. 0 No parity error occurred. 1 If UMR1n[PM] = 0x (with parity or force parity), the corresponding character in the FIFO was received with incorrect parity. If UMR1n[PM] = 11 (multidrop), PE stores the received A/D bit.
4	OE	Overrun error. Indicates whether an overrun occurs. 0 No overrun occurred. 1 One or more characters in the received data stream have been lost. OE is set upon receipt of a new character when the FIFO is full and a character is already in the shift register waiting for an empty FIFO position. When this occurs, the character in the receiver shift register and its break detect, framing error status, and parity error, if any, are lost. OE is cleared by the RESET ERROR STATUS command in UCRn.
3	TxEMP	Transmitter empty. 0 The transmitter buffer is not empty. Either a character is being shifted out, or the transmitter is disabled. The transmitter is enabled/disabled by programming UCRn[TC]. 1 The transmitter has underrun (both the transmitter holding register and transmitter shift registers are empty). This bit is set after transmission of the last stop bit of a character if there are no characters in the transmitter holding register awaiting transmission.
2	TxRDY	Transmitter ready. 0 The CPU loaded the transmitter holding register or the transmitter is disabled. 1 The transmitter holding register is empty and ready for a character. TxRDY is set when a character is sent to the transmitter shift register and when the transmitter is first enabled. If the transmitter is disabled, characters loaded into the transmitter holding register are not sent.
1	FFULL	FIFO full. 0 The FIFO is not full but may hold up to two unread characters. 1 A character was received and is waiting in the receiver buffer FIFO.
0	RxRDY	Receiver ready 0 The CPU has read the receiver buffer and no characters remain in the FIFO after this read. 1 One or more characters were received and are waiting in the receiver buffer FIFO.

14.3.4 UART Clock-Select Registers (UCSR*n*)

The UART clock-select registers (UCSR*n*) select an external clock on the TIN input (divided by 1 or 16) or a prescaled BCLKO as the clocking source for the transmitter and receiver. See Section 14.5.1, “Transmitter/Receiver Clock Source.” The transmitter and receiver can use different clock sources. To use BCLKO for both, set UCSR*n* to 0xDD.

	7 4	3 0
Field	RCS	TCS
Reset	0000_0000	
R/W	Write only	
Address	MBAR + 0x1C4 (UCSR0), 0x204 (UCSR1)	

Figure 14-5. UART Clock-Select Register (UCSR*n*)

Table 14-5 describes UCSR*n* fields.

Table 14-5. UCSR*n* Field Descriptions

Bits	Name	Description
7–4	RCS	Receiver clock select. Selects the clock source for the receiver channel. 1101 Prescaled BCLKO 1110 TIN divided by 16 1111 TIN
3–0	TCS	Transmitter clock select. Selects the clock source for the transmitter channel. 1101 Prescaled BCLKO 1110 TIN divided by 16 1111 TIN

14.3.5 UART Command Registers (UCR*n*)

The UART command registers (UCR*n*), Figure 14-6, supply commands to the UART. Only multiple commands that do not conflict can be specified in a single write to a UCR*n*. For example, RESET TRANSMITTER and ENABLE TRANSMITTER cannot be specified in one command.

	7	6–4	3–2	1–0
Field	—	MISC	TC	RC
Reset	0000_0000			
R/W	Write only			
Address	MBAR + 0x1C8, 0x208			

Figure 14-6. UART Command Register (UCR*n*)

Table 14-6 describes UCR*n* fields and commands. Examples in Section 14.5.2, “Transmitter and Receiver Operating Modes,” show how these commands are used.

Table 14-6. UCR*n* Field Descriptions

Bits	Value	Command	Description
7	—	—	Reserved, should be cleared.

Table 14-6. UCR*n* Field Descriptions (Continued)

Bits	Value	Command	Description
6–4	**MISC Field** (This field selects a single command.)		
	000	NO COMMAND	—
	001	RESET MODE REGISTER POINTER	Causes the mode register pointer to point to UMR1n.
	010	RESET RECEIVER	Immediately disables the receiver, clears USRn[FFULL,RxRDY], and reinitializes the receiver FIFO pointer. No other registers are altered. Because it places the receiver in a known state, use this command instead of RECEIVER DISABLE when reconfiguring the receiver.
	011	RESET TRANSMITTER	disables the transmitter and clears USRn[TxEMP,TxRDY]. No other registers are altered. Because it places the transmitter in a known state, use this command instead of TRANSMITTER DISABLE when reconfiguring the transmitter.
	100	RESET ERROR STATUS	lears USRn[RB,FE,PE,OE]. Also used in block mode to clear all error bits after a data block is received.
	101	RESET BREAK–CHANGE INTERRUPT	Clears the delta break bit, UISRn[DB].
	110	START BREAK	Forces TxD low. If the transmitter is empty, the break may be delayed up to one bit time. If the transmitter is active, the break starts when character transmission completes. The break is delayed until any character in the transmitter shift register is sent. Any character in the transmitter holding register is sent after the break. The transmitter must be enabled for the command to be accepted. This command ignores the state of $\overline{\text{CTS}}$.
	111	STOP BREAK	Causes TxD to go high (mark) within two bit times. Any characters in the transmitter buffer are sent.
3–2	**TC Field** (This field selects a single command)		
	00	NO ACTION TAKEN	Causes the transmitter to stay in its current mode: if the transmitter is enabled, it remains enabled; if the transmitter is disabled, it remains disabled.
	01	TRANSMITTER ENABLE	Enables operation of the channel's transmitter. USRn[TxEMP,TxRDY] are set. If the transmitter is already enabled, this command has no effect.
	10	TRANSMITTER DISABLE	Terminates transmitter operation and clears USRn[TxEMP,TxRDY]. If a character is being sent when the transmitter is disabled, transmission completes before the transmitter becomes inactive. If the transmitter is already disabled, the command has no effect.
	11	—	Reserved, do not use.

Table 14-6. UCR*n* Field Descriptions (Continued)

Bits	Value	Command	Description
1–0	**RC** (This field selects a single command)		
	00	NO ACTION TAKEN	Causes the receiver to stay in its current mode. If the receiver is enabled, it remains enabled; if disabled, it remains disabled.
	01	RECEIVER ENABLE	If the UART module is not in multidrop mode (UMR1n[PM] ≠ 11), RECEIVER ENABLE enables the channel's receiver and forces it into search-for-start-bit state. If the receiver is already enabled, this command has no effect.
	10	RECEIVER DISABLE	Disables the receiver immediately. Any character being received is lost. The command does not affect receiver status bits or other control registers. If the UART module is programmed for local loop-back or multidrop mode, the receiver operates even though this command is selected. If the receiver is already disabled, the command has no effect.
	11	—	Reserved, do not use.

14.3.6 UART Receiver Buffers (URB*n*)

The receiver buffers contain one serial shift register and three receiver holding registers, which act as a FIFO. RxD is connected to the serial shift register. The CPU reads from the top of the stack while the receiver shifts and updates from the bottom when the shift register is full (see Figure 14-20). RB contains the character in the receiver.

	7 … 0
Field	RB
Reset	0000_0000
R/W	Read only
Address	MBAR + 0x1CC,0x20C

Figure 14-7. UART Receiver Buffer (URB0)

14.3.7 UART Transmitter Buffers (UTB*n*)

The transmitter buffers consist of the transmitter holding register and the transmitter shift register. The holding register accepts characters from the bus master if channel's USR*n*[TxRDY] is set. A write to the transmitter buffer clears TxRDY, inhibiting any more characters until the shift register can accept more data. When the shift register is empty, it checks if the holding register has a valid character to be sent (TxRDY = 0). If there is a valid character, the shift register loads it and sets USR*n*[TxRDY] again. Writes to the transmitter buffer when the channel's TxRDY = 0 and when the transmitter is disabled have no effect on the transmitter buffer.

Figure 14-8 shows UTB0. TB contains the character in the transmitter buffer.

	7	0
Field	TB	
Reset	0000_0000	
R/W	Write only	
Address	MBAR + 0x1CC,0x20C	

Figure 14-8. UART Transmitter Buffer (UTB0)

14.3.8 UART Input Port Change Registers (UIPCR*n*)

The input port change registers (UIPCR*n*), Figure 14-9, hold the current state and the change-of-state for $\overline{\text{CTS}}$.

	7	5	4	3	1	0
Field	—		COS	111		CTS
Reset	0000			0	11	$\overline{\text{CTS}}$
R/W	Read only					
Address	MBAR + 0x1D0 (UIPCR0), 0x210 (UIPCR1)					

Figure 14-9. UART Input Port Change Register (UIPCR*n*)

Table 14-7 describes UIPCR*n* fields.

Table 14-7. UIPCR*n* Field Descriptions

Bits	Name	Description
7–5	—	Reserved, should be cleared.
4	COS	Change of state (high-to-low or low-to-high transition). 0 No change-of-state since the CPU last read UIPCRn. Reading UIPCRn clears UISRn[COS]. 1 A change-of-state longer than 25–50 μs occurred on the $\overline{\text{CTS}}$ input. UACRn can be programmed to generate an interrupt to the CPU when a change of state is detected.
3–1	—	Reserved, should be cleared.
0	CTS	Current state. Starting two serial clock periods after reset, CTS reflects the state of $\overline{\text{CTS}}$. If $\overline{\text{CTS}}$ is detected asserted at that time, COS is set, which initiates an interrupt if UACRn[IEC] is enabled. 0 The current state of the $\overline{\text{CTS}}$ input is asserted. 1 The current state of the $\overline{\text{CTS}}$ input is negated.

14.3.9 UART Auxiliary Control Register (UACR*n*)

The UART auxiliary control registers (UACR*n*), Figure 14-7, control the input enable.

	7 … 1	0
Field	—	IEC
Reset	0000_0000	
R/W	Write only	
Address	MBAR + 0x1D0 (UACR0), 0x210 (UACR1)	

Figure 14-10. UART Auxiliary Control Register (UACR*n*)

Table 14-8 describes UACR*n* fields.

Table 14-8. UACR*n* Field Descriptions

Bits	Name	Description
7–1	—	Reserved, should be cleared.
0	IEC	Input enable control. 0 Setting the corresponding UIPCRn bit has no effect on UISRn[COS]. 1 UISRn[COS] is set and an interrupt is generated when the UIPCRn[COS] is set by an external transition on the $\overline{\text{CTS}}$ input (if UIMRn[COS] = 1).

14.3.10 UART Interrupt Status/Mask Registers (UISR*n*/UIMR*n*)

The UART interrupt status registers (UISR*n*), Figure 14-11, provide status for all potential interrupt sources. UISR*n* contents are masked by UIMR*n*. If corresponding UISR*n* and UIMR*n* bits are set, the internal interrupt output is asserted. If a UIMR*n* bit is cleared, the state of the corresponding UISR*n* bit has no effect on the output.

NOTE:

True status is provided in the UISR*n* regardless of UIMR*n* settings. UISR*n* is cleared when the UART module is reset.

	7	6 … 3	2	1	0
Field	COS	—	DB	FFULL/RxRDY	TxRDY
Reset	0000_0000				
R/W	Read only for status, write only for mask.				
Address	MBAR + 0x1D4 (UISR0), 0x214 (UISR1); MBAR + 0x1D4 (UIMR0), 0x214 (UIMR1)				

Figure 14-11. UART Interrupt Status/Mask Registers (UISR*n*/UIMR*n*)

Table 14-9 describes UISR*n* and UIMR*n* fields.

Table 14-9. UISR*n*/UIMR*n* Field Descriptions

Bits	Name	Description
7	COS	Change-of-state. 0 UIPCRn[COS] is not selected. 1 Change-of-state occurred on $\overline{\text{CTS}}$ and was programmed in UACRn[IEC] to cause an interrupt.
6–3	—	Reserved, should be cleared.
2	DB	Delta break. 0 No new break-change condition to report. Section 14.3.5, “UART Command Registers (UCRn),” describes the RESET BREAK-CHANGE INTERRUPT command. 1 The receiver detected the beginning or end of a received break.
1	FFULL/ RxRDY	RxRDY (receiver ready) if UMR1n[FFULL/RxRDY] = 0; FIFO full (FFULL) if UMR1n[FFULL/RxRDY] = 1. Duplicate of USRn[FFULL/RxRDY]. If FFULL is enabled for UART0 or UART1, DMA channels 2 or 3 are respectively interrupted when the FIFO is full.
0	TxRDY	Transmitter ready. This bit is the duplication of USRn[TxRDY]. 0 The transmitter holding register was loaded by the CPU or the transmitter is disabled. Characters loaded into the transmitter holding register when TxRDY = 0 are not sent. 1 The transmitter holding register is empty and ready to be loaded with a character.

14.3.11 UART Divider Upper/Lower Registers (UDU*n*/UDL*n*)

The UDU*n* registers (formerly called UBG1*n*) holds the MSB, and the UDL*n* registers (formerly UBG2*n*) hold the LSB of the preload value. UDU*n* and UDL*n* concatenate to provide a divider to BCLKO for transmitter/receiver operation, as described in Section 14.5.1.2.1, “BCLKO Baud Rates.”

	7	0
Field	Divider MSB	
Reset	0000_0000	
R/W	R/W	
Address	MBAR + 0x1D8 (UDU0), 0x218 (UDU1)	

Figure 14-12. UART Divider Upper Register (UDU*n*)

	7	0
Field	Divider LSB	
Reset	0000_0000	
R/W	R/W	
Address	MBAR + 0x1DC (UDL0), 0x21C (UDL1)	

Figure 14-13. UART Divider Lower Register (UDL*n*)

NOTE:

The minimum value that can be loaded on the concatenation of UDU*n* with UDL*n* is 0x0002. Both UDU*n* and UDL*n* are write-only and cannot be read by the CPU.

14.3.12 UART Interrupt Vector Register (UIVR*n*)

The UIVR*n*, Figure 14-14, contain the 8-bit internal interrupt vector number (IVR).

	7	0
Field	IVR	
Reset	0000_1111	
R/W	R/W	
Address	MBAR + 0x1F0 (UIVR0), 0x230 (UIVR1)	

Figure 14-14. UART Interrupt Vector Register (UIVR*n*)

Table 14-10 describes UIVR*n* fields.

Table 14-10. UIVR*n* Field Descriptions

Bits	Name	Description
7–0	IVR	Interrupt vector. Indicates the vector number where the address of the exception handler for the specified interrupt is located. UIVRn is reset to 0x0F, indicating an uninitialized interrupt condition.

14.3.13 UART Input Port Register (UIP*n*)

The UART input port registers (UIP*n*), Figure 14-15, show the current state of the $\overline{\text{CTS}}$ input.

	7	1	0
Field	—		CTS
Reset	1111_1111		
R/W	Read only		
Address	MBAR + 0x1F4 (UIP0), 0x234 (UIP1)		

Figure 14-15. UART Input Port Register (UIP*n*)

Table 14-11 describes UIP*n* fields.

Table 14-11. UIP*n* Field Descriptions

Bits	Name	Description
7–1	—	Reserved, should be cleared.
0	CTS	Current state. The $\overline{\text{CTS}}$ value is latched and reflects the state of the input pin when UIPn is read. Note: This bit has the same function and value as UIPCRn[RTS]. 0 The current state of the $\overline{\text{CTS}}$ input is logic 0. 1 The current state of the $\overline{\text{CTS}}$ input is logic 1.

14.3.14 UART Output Port Command Registers (UOP1*n*/UOP0*n*)

In UART mode, the $\overline{\text{RTS}}$ output can be asserted by writing a 1 to UOP1*n*[RTS] and negated by writing a 1 to UOP0*n*[RTS]. See Figure 14-16.

	7 — 1	0
Field	—	RTS
Reset	0000_0000	
R/W	Write only	
Addr	UART0: MBAR + 0x1F8 (UOP1), 0x1FC (UOP0); UART1 0x238 (UOP1), 0x23C (UOP0)	

Figure 14-16. UART Output Port Command Register (UOP1/UOP0)

Table 14-12 describes UOP1 fields.

Table 14-12. UOP1/UOP0 Field Descriptions

Bits	Name	Description
7–1	—	Reserved, should be cleared.
0	RTS	Output port parallel output. Controls assertion (UOP1)/negation (UOP0) of $\overline{RTS}$ output. 0 Not affected. 1 Asserts $\overline{RTS}$ (UOP1). Negates $\overline{RTS}$ (UOP0).

14.4 UART Module Signal Definitions

Figure 14-17 shows both the external and internal signal groups.

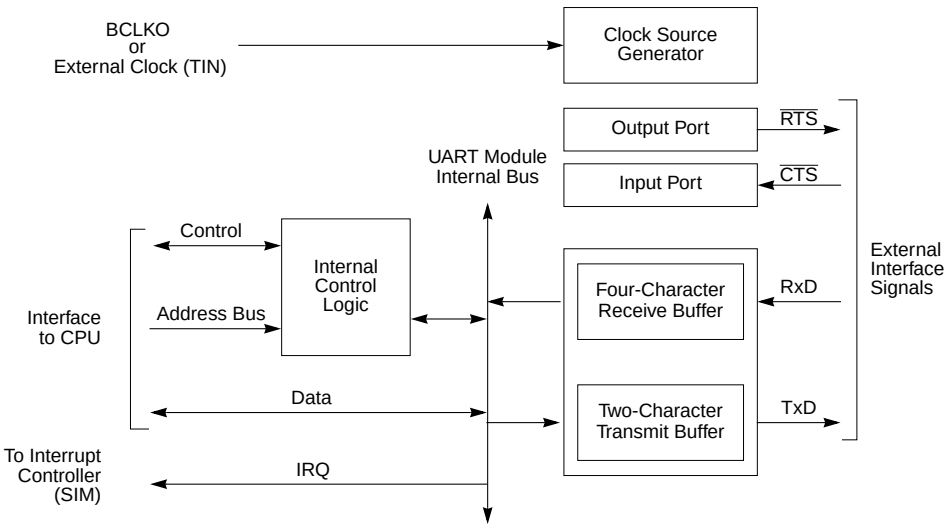


Figure 14-17. UART Block Diagram Showing External and Internal Interface Signals

An internal interrupt request signal ($\overline{IRQ}$) is provided to notify the interrupt controller of an interrupt condition. The output is the logical NOR of unmasked UISR*n* bits. The interrupt level of a UART module is programmed in the interrupt controller in the system integration module (SIM). The UART can use the autovector for the programmed interrupt level or supply the vector from the UIVR*n* when the UART interrupt is acknowledged.

The interrupt level, priority, and auto-vectoring capability is programmed in SIM register ICR4 for UART0 and ICR5 for UART1. See Section 9.2.1, “Interrupt Control Registers (ICR0–ICR9).”

Note that the UARTs can also automatically transfer data by using the DMA rather than interrupting the core. When UIMR[FFULL] is 1 and a receiver’s FIFO is full, it can send an interrupt to a DMA channel so the FIFO data can be transferred to memory. Note also that UART0 and UART1’s interrupt requests are connected to DMA channel 2 and channel 3, respectively.

Table 14-13 briefly describes the UART module signals.

NOTE:

The terms ‘assertion’ and ‘negation’ are used to avoid confusion between active-low and active-high signals. ‘Asserted’ indicates that a signal is active, independent of the voltage level; ‘negated’ indicates that a signal is inactive.

Table 14-13. UART Module Signals

Signal	Description
Transmitter Serial Data Output (TxD)	TxD is held high (mark condition) when the transmitter is disabled, idle, or operating in the local loop-back mode. Data is shifted out on TxD on the falling edge of the clock source, with the least significant bit (lsb) sent first.
Receiver Serial Data Input (RxD)	Data received on RxD is sampled on the rising edge of the clock source, with the lsb received first.
Clear-to-Send ($\overline{\text{CTS}}$)	This input can generate an interrupt on a change of state.
Request-to-Send ($\overline{\text{RTS}}$)	This output can be programmed to be negated or asserted automatically by either the receiver or the transmitter. When connected to a transmitter’s $\overline{\text{CTS}}$, $\overline{\text{RTS}}$ can control serial data flow.

Figure 14-18 shows a signal configuration for a UART/RS-232 interface.

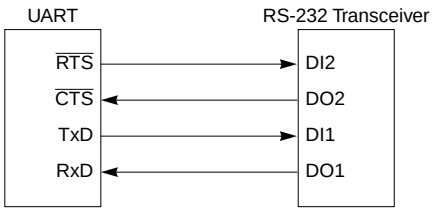


Figure 14-18. UART/RS-232 Interface

14.5 Operation

This section describes operation of the clock source generator, transmitter, and receiver.

14.5.1 Transmitter/Receiver Clock Source

BCLKO serves as the basic timing reference for the clock source generator logic, which consists of a clock generator and a programmable 16-bit divider dedicated to the UART. The clock generator cannot produce standard baud rates if BCLKO is used, so the 16-bit divider should be used.

14.5.1.1 Programmable Divider

As Figure 14-19 shows, the UART transmitter and receiver can use the following clock sources:

- An external clock signal on the TIN pin that can be divided by 16. When not divided, TIN provides a synchronous clock mode; when divided by 16, it is asynchronous**.**
- BCLKO supplies an asynchronous clock source that is divided by 32 and then divided by the 16-bit value programmed in UDU*n* and UDL*n*. See Section 14.3.11, “UART Divider Upper/Lower Registers (UDUn/UDLn).”

The choice of TIN or BCLKO is programmed in the UCSR.

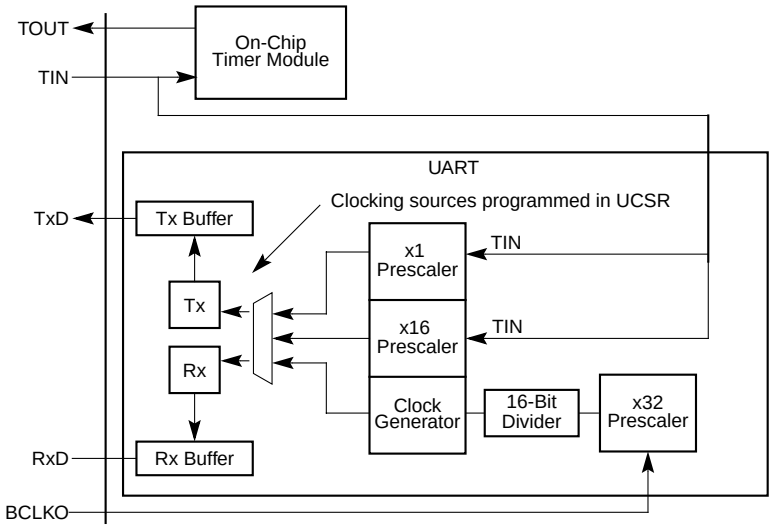


Figure 14-19. Clocking Source Diagram

NOTE:

If TIN is a clocking source for either the timer or UART, the timer module cannot use TIN for timer capture.

14.5.1.2 Calculating Baud Rates

The following sections describe how to calculate baud rates.

14.5.1.2.1 BCLKO Baud Rates

When BCLKO is the UART clocking source, it goes through a divide-by-32 prescaler and then passes through the 16-bit divider of the concatenated UDU*n* and UDL*n* registers. Using a 45-MHz BCLKO, the baud-rate calculation is as follows:

$$\text{Baudrate} = \frac{45\text{MHz}}{[32 \times \text{divider}]}$$

let baud rate = 9600, then

$$\text{Divider} = \frac{45\text{MHz}}{[32 \times 9600]} = 146(\text{decimal}) = 0092(\text{hexadecimal})$$

therefore UDU = 0x00 and UDL = 0x92.

14.5.1.2.2 External Clock

An external source clock (TIN) can be used as is or divided by 16.

$$\text{Baudrate} = \frac{\text{Externalclockfrequency}}{16\text{or}1}$$

14.5.2 Transmitter and Receiver Operating Modes

Figure 14-20 is a functional block diagram of the transmitter and receiver showing the command and operating registers, which are described generally in the following sections and described in detail in Section 14.3, “Register Descriptions.”

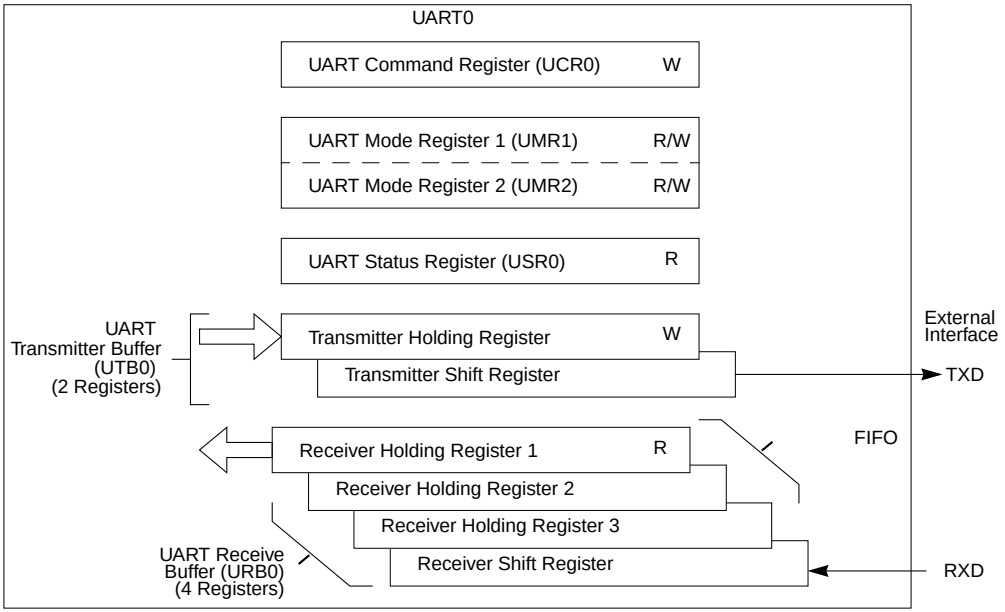


Figure 14-20. Transmitter and Receiver Functional Diagram

14.5.2.1 Transmitting

The transmitter is enabled through the UART command register (UCR*n*). When it is ready to accept a character, the UART sets USR*n*[TxRDY]. The transmitter converts parallel data from the CPU to a serial bit stream on TxD. It automatically sends a start bit followed by the programmed number of data bits, an optional parity bit, and the programmed number of stop bits. The lsb is sent first. Data is shifted from the transmitter output on the falling edge of the clock source.

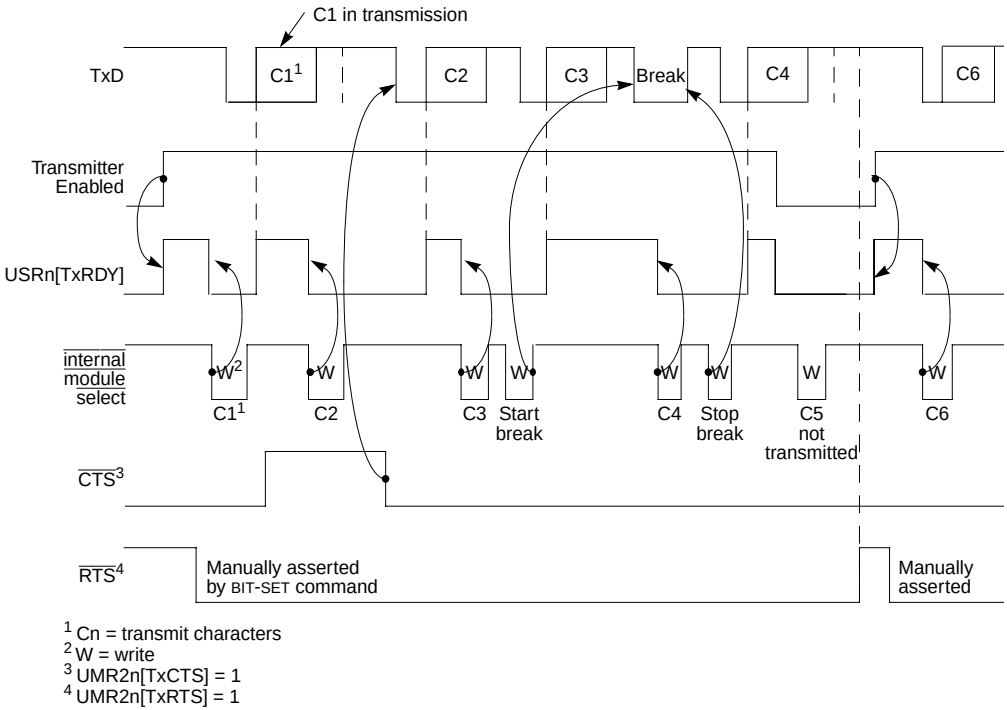
After the stop bits are sent, if no new character is in the transmitter holding register, the TxD output remains high (mark condition) and the transmitter empty bit, USR*n*[TxEMP], is set. Transmission resumes and TxEMP is cleared when the CPU loads a new character into the UART transmitter buffer (UTB*n*). If the transmitter receives a disable command, it continues until any character in the transmitter shift register is completely sent.

If the transmitter is reset through a software command, operation stops immediately (see Section 14.3.5, “UART Command Registers (UCRn)”). The transmitter is reenabled through the UCR*n* to resume operation after a disable or software reset.

If the clear-to-send operation is enabled, $\overline{\text{CTS}}$ must be asserted for the character to be transmitted. If $\overline{\text{CTS}}$ is negated in the middle of a transmission, the character in the shift register is sent and TxD remains in mark state until $\overline{\text{CTS}}$ is reasserted. If the transmitter is forced to send a continuous low condition by issuing a SEND BREAK command, the transmitter ignores the state of $\overline{\text{CTS}}$.

If the transmitter is programmed to automatically negate $\overline{\text{RTS}}$ when a message transmission completes, $\overline{\text{RTS}}$ must be asserted manually before a message is sent. In applications in which the transmitter is disabled after transmission is complete and $\overline{\text{RTS}}$ is appropriately programmed, $\overline{\text{RTS}}$ is negated one bit time after the character in the shift register is completely transmitted. The transmitter must be manually reenabled by reasserting $\overline{\text{RTS}}$ before the next message is to be sent.

Figure 14-21 shows the functional timing information for the transmitter.



[1] Cn = transmit characters
[2] W = write
[3] UMR2n[TxCTS] = 1
[4] UMR2n[TxRTS] = 1

Figure 14-21. Transmitter Timing Diagram

14.5.2.2 Receiver

The receiver is enabled through its UCR*n*, as described in Section 14.3.5, "UART Command Registers (UCRn)." Figure 14-22 shows receiver functional timing.

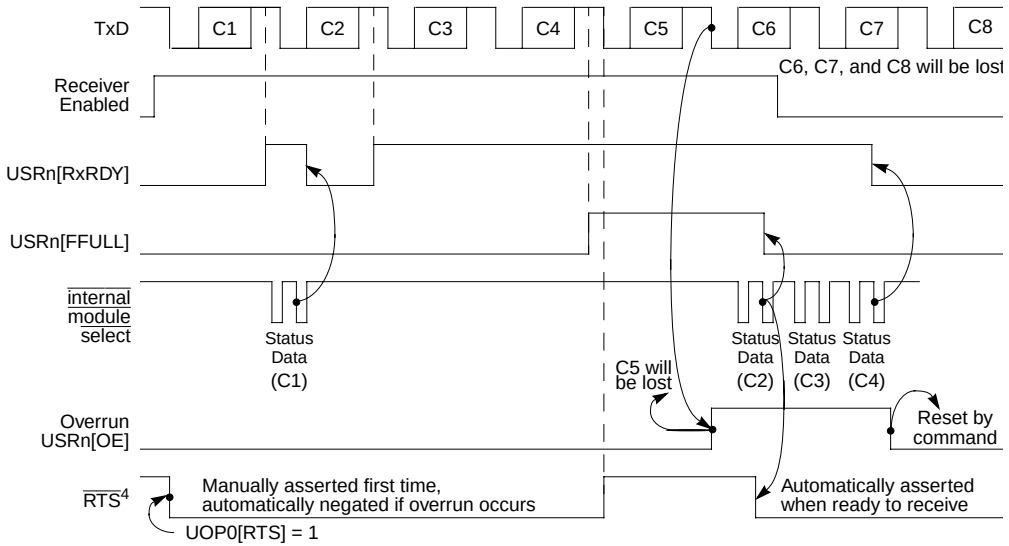


Figure 14-22. Receiver Timing

When the receiver detects a high-to-low (mark-to-space) transition of the start bit on RxD, the state of RxD is sampled each 16× clock for eight clocks, starting one-half clock after the transition (asynchronous operation) or at the next rising edge of the bit time clock (synchronous operation). If RxD is sampled high, the start bit is invalid and the search for the valid start bit begins again.

If RxD is still low, a valid start bit is assumed and the receiver continues sampling the input at one-bit time intervals, at the theoretical center of the bit, until the proper number of data bits and parity, if any, is assembled and one stop bit is detected. Data on the RxD input is sampled on the rising edge of the programmed clock source. The lsb is received first. The data is then transferred to a receiver holding register and USR*n*[RxRDY] is set. If the character is less than eight bits, the most significant unused bits in the receiver holding register are cleared.

After the stop bit is detected, the receiver immediately looks for the next start bit. However, if a non-zero character is received without a stop bit (framing error) and RxD remains low for one-half of the bit period after the stop bit is sampled, the receiver operates as if a new start bit were detected. Parity error, framing error, overrun error, and received break conditions set the respective PE, FE, OE, RB error and break flags in the USR*n* at the received character boundary and are valid only if USR*n*[RxRDY] is set.

If a break condition is detected (RxD is low for the entire character including the stop bit), a character of all zeros is loaded into the receiver holding register (RHR) and USR*n*[RB,RxRDY] are set. RxD must return to a high condition for at least one-half bit time before a search for the next start bit begins.

The receiver detects the beginning of a break in the middle of a character if the break persists through the next character time. If the break begins in the middle of a character, the receiver places the damaged character in the Rx FIFO stack and sets the corresponding USR*n* error bits and USR*n*[RxRDY]. Then, if the break lasts until the next character time, the receiver places an all-zero character into the Rx FIFO and sets USR*n*[RB,RxRDY].

14.5.2.3 FIFO Stack

The FIFO stack is used in the UART's receiver buffer logic. The stack consists of three receiver holding registers. The receive buffer consists of the FIFO and a receiver shift register connected to the RxD (see Figure 14-20). Data is assembled in the receiver shift register and loaded into the top empty receiver holding register position of the FIFO. Thus, data flowing from the receiver to the CPU is quadruple-buffered.

In addition to the data byte, three status bits, parity error (PE), framing error (FE), and received break (RB), are appended to each data character in the FIFO; OE (overrun error) is not appended. By programming the ERR bit in the channel's mode register (UMR1*n*), status is provided in character or block modes.

USR*n*[RxRDY] is set when at least one character is available to be read by the CPU. A read of the receiver buffer produces an output of data from the top of the FIFO stack. After the read cycle, the data at the top of the FIFO stack and its associated status bits are popped and the receiver shift register can add new data at the bottom of the stack. The FIFO-full status bit (FFULL) is set if all three stack positions are filled with data. Either the RxRDY or FFULL bit can be selected to cause an interrupt.

The two error modes are selected by UMR1*n*[ERR] as follows:

- In character mode (UMR1*n*[ERR] = 0, status is given in the USR*n* for the character at the top of the FIFO.
- In block mode, the USR*n* shows a logical OR of all characters reaching the top of the FIFO stack since the last RESET ERROR STATUS command. Status is updated as characters reach the top of the FIFO stack. Block mode offers a data-reception speed advantage where the software overhead of error-checking each character cannot be tolerated. However, errors are not detected until the check is performed at the end of an entire message—the faulting character is not identified.

In either mode, reading the USR*n* does not affect the FIFO. The FIFO is popped only when the receive buffer is read. The USR*n* should be read before reading the receive buffer. If all three receiver holding registers are full, a new character is held in the receiver shift register until space is available. However, if a second new character is received, the contents of the the character in the receiver shift register is lost, the FIFOs are unaffected, and USR*n*[OE] is set when the receiver detects the start bit of the new overrunning character.

To support flow control, the receiver can be programmed to automatically negate and assert $\overline{\text{RTS}}$, in which case the receiver automatically negates $\overline{\text{RTS}}$ when a valid start bit is detected and the FIFO stack is full. The receiver asserts $\overline{\text{RTS}}$ when a FIFO position becomes

available; therefore, overrun errors can be prevented by connecting $\overline{\text{RTS}}$ to the $\overline{\text{CTS}}$ input of the transmitting device.

NOTE:

The receiver can still read characters in the FIFO stack if the receiver is disabled. If the receiver is reset, the FIFO stack, $\overline{\text{RTS}}$ control, all receiver status bits, and interrupt requests are reset. No more characters are received until the receiver is reenabled.

14.5.3 Looping Modes

The UART can be configured to operate in various looping modes as shown in Figure 14-22 on page 14-23. These modes are useful for local and remote system diagnostic functions. The modes are described in the following paragraphs and in Section 14.3, “Register Descriptions.”

The UART’s transmitter and receiver should be disabled when switching between modes. The selected mode is activated immediately upon mode selection, regardless of whether a character is being received or transmitted.

14.5.3.1 Automatic Echo Mode

In automatic echo mode, shown in Figure 14-23, the UART automatically resends received data bit by bit. The local CPU-to-receiver communication continues normally, but the CPU-to-transmitter link is disabled. In this mode, received data is clocked on the receiver clock and resent on TxD. The receiver must be enabled, but the transmitter need not be.

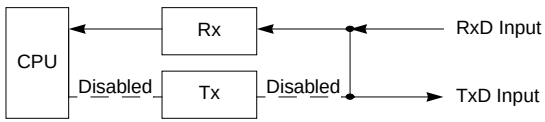


Figure 14-23. Automatic Echo

Because the transmitter is inactive, USR*n*[TxEMP,TxRDY] are inactive and data is sent as it is received. Received parity is checked but is not recalculated for transmission. Character framing is also checked, but stop bits are sent as they are received. A received break is echoed as received until the next valid start bit is detected.

14.5.3.2 Local Loop-Back Mode

Figure 14-24 shows how TxD and RxD are internally connected in local loop-back mode. This mode is for testing the operation of a local UART module channel by sending data to the transmitter and checking data assembled by the receiver to ensure proper operations.

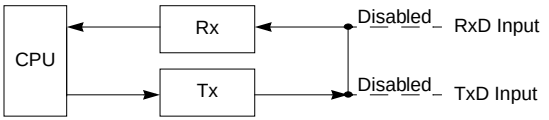


Figure 14-24. Local Loop-Back

Features of this local loop-back mode are as follows:

- Transmitter and CPU-to-receiver communications continue normally in this mode.
- RxD input data is ignored
- TxD is held marking
- The receiver is clocked by the transmitter clock. The transmitter must be enabled, but the receiver need not be.

14.5.3.3 Remote Loop-Back Mode

In remote loop-back mode, shown in Figure 14-25, the channel automatically transmits received data bit by bit on the TxD output. The local CPU-to-transmitter link is disabled. This mode is useful in testing receiver and transmitter operation of a remote channel. For this mode, the transmitter uses the receiver clock.

Because the receiver is not active, received data cannot be read by the CPU and error status conditions are inactive. Received parity is not checked and is not recalculated for transmission. Stop bits are sent as they are received. A received break is echoed as received until the next valid start bit is detected.

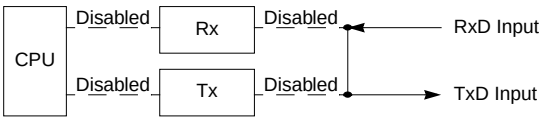


Figure 14-25. Remote Loop-Back

14.5.4 Multidrop Mode

Setting UMR1*n*[PM] programs the UART to operate in a wake-up mode for multidrop or multiprocessor applications. In this mode, a master can transmit an address character followed by a block of data characters targeted for one of up to 256 slave stations.

Although slave stations have their channel receivers disabled, they continuously monitor the master's data stream. When the master sends an address character, the slave receiver channel notifies its respective CPU by setting USR*n*[RxRDY] and generating an interrupt (if programmed to do so). Each slave station CPU then compares the received address to its station address and enables its receiver if it wishes to receive the subsequent data characters or block of data from the master station. Slave stations not addressed continue monitoring the data stream. Data fields in the data stream are separated by an address character. After

a slave receives a block of data, its CPU disables the receiver and repeats the process.Functional timing information for multidrop mode is shown in Figure 14-26.

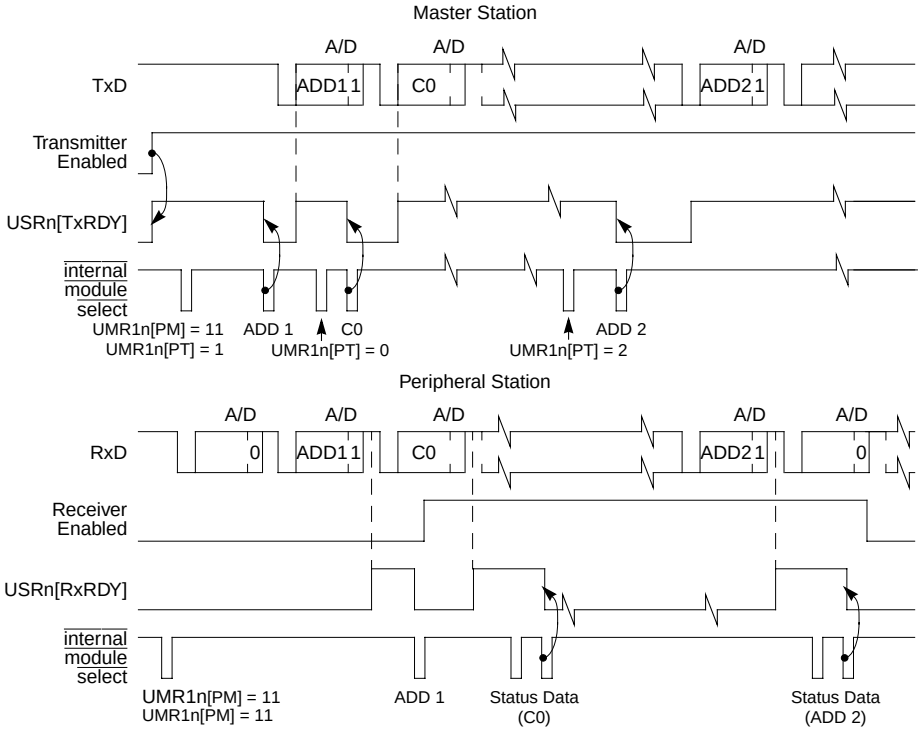


Figure 14-26. Multidrop Mode Timing Diagram

A character sent from the master station consists of a start bit, a programmed number of data bits, an address/data (A/D) bit flag, and a programmed number of stop bits. A/D = 1 indicates an address character; A/D = 0 indicates a data character. The polarity of A/D is selected through UMR1*n*[PT]. UMR1*n* should be programmed before enabling the transmitter and loading the corresponding data bits into the transmit buffer.

In multidrop mode, the receiver continuously monitors the received data stream, regardless of whether it is enabled or disabled. If the receiver is disabled, it sets the RxRDY bit and loads the character into the receiver holding register FIFO stack provided the received A/D bit is a one (address tag). The character is discarded if the received A/D bit is zero (data tag). If the receiver is enabled, all received characters are transferred to the CPU through the receiver holding register stack during read operations.

In either case, the data bits are loaded into the data portion of the stack while the A/D bit is loaded into the status portion of the stack normally used for a parity error (USR*n*[PE]).

Framing error, overrun error, and break detection operate normally. The A/D bit takes the place of the parity bit; therefore, parity is neither calculated nor checked. Messages in this

mode may still contain error detection and correction information. One way to provide error detection, if 8-bit characters are not required, is to use software to calculate parity and append it to the 5-, 6-, or 7-bit character.

14.5.5 Bus Operation

This section describes bus operation during read, write, and interrupt acknowledge cycles to the UART module.

14.5.5.1 Read Cycles

The UART module responds to reads with byte data. Reserved registers return zeros.

14.5.5.2 Write Cycles

The UART module accepts write data as bytes. Write cycles to read-only or reserved registers complete normally without exception processing, but data is ignored.

NOTE:

The UART module is accessed by the CPU with zero wait states, as BCLKO is used for the UART module.

14.5.5.3 Interrupt Acknowledge Cycles

The UART module supplies the interrupt vector in response to a UART IACK cycle. If UIVR*n* is not initialized to provide a vector number, a spurious exception is taken if an interrupt is generated. This works in conjunction with the interrupt controller, which allows a programmable priority level.

14.5.6 Programming

The software flowchart, Figure 14-27, consists of the following:

- UART module initialization—These routines consist of SINIT and CHCHK (sheets 1 and 2). Before SINIT is called at system initialization, the calling routine allocates 2 words on the system stack. On return to the calling routine, SINIT passes UART status data on the stack. If SINIT finds no errors, the transmitter and receiver are enabled. SINIT calls CHCHK to perform the checks. When called, SINIT places the UART in local loop-back mode and checks for the following errors:
  - — Transmitter never ready
  - — Receiver never ready
  - — Parity error
  - — Incorrect character received
- I/O driver routine—This routine (sheets 4 and 5) consists of INCH, the terminal input character routine which gets a character from the receiver, and OUTCH, which sends a character to the transmitter.

- Interrupt handling—Consists of SIRQ (sheet 4), which is executed after the UART module generates an interrupt caused by a change-in-break (beginning of a break). SIRQ then clears the interrupt source, waits for the next change-in-break interrupt (end of break), clears the interrupt source again, then returns from exception processing to the system monitor.

14.5.6.1 UART Module Initialization Sequence

NOTE:

UART module registers can be accessed by word or byte operations, but only data byte D[7:0] is valid.

Table 14-14 shows the UART module initialization sequence.

Table 14-14. UART Module Initialization Sequence

Register	Setting
UCRn	Reset the receiver and transmitter. Reset the mode pointer (MISC[2–0] = 0b001).
UIVRn	Program the vector number for a UART module interrupt.
UIMRn	Enable the preferred interrupt sources.
UACRn	Initialize the input enable control (IEC bit).
UCSRn	Select the receiver and transmitter clock. Use timer as source if required.
UMR1n	If preferred, program operation of receiver ready-to-send (RxRTS bit). Select receiver-ready or FIFO-full notification (RxRDY/FFULL bit). Select character or block error mode (ERR bit). Select parity mode and type (PM and PT bits). Select number of bits per character (B/Cx bits).
UMR2n	Select the mode of operation (CMx bits). If preferred, program operation of transmitter ready-to-send (TxRTS). If preferred, program operation of clear-to-send (TxCTS bit). Select stop-bit length (SBx bits).
UCR	

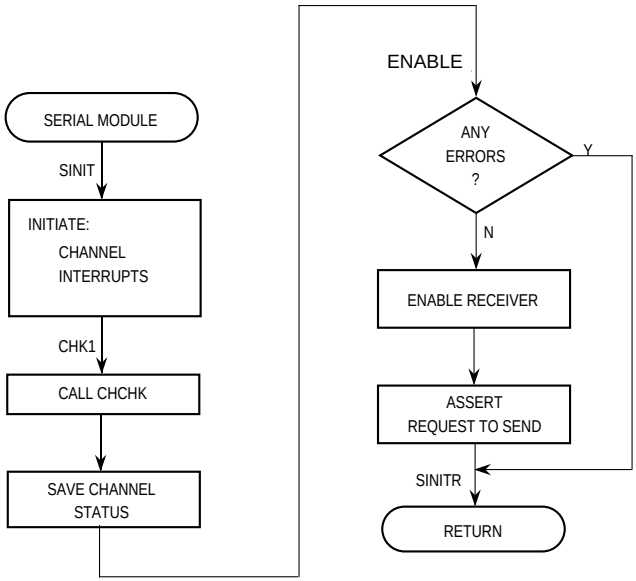


Figure 14-27. UART Mode Programming Flowchart (Sheet 1 of 5)

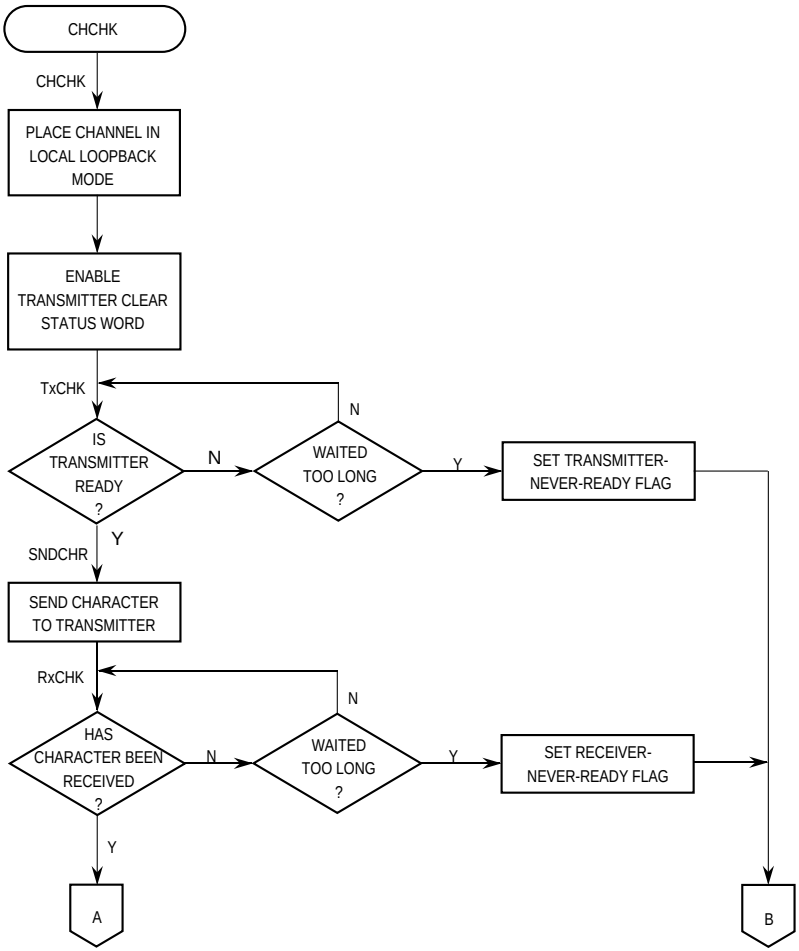


Figure 14-27. UART Mode Programming Flowchart (Sheet 2 of 5)

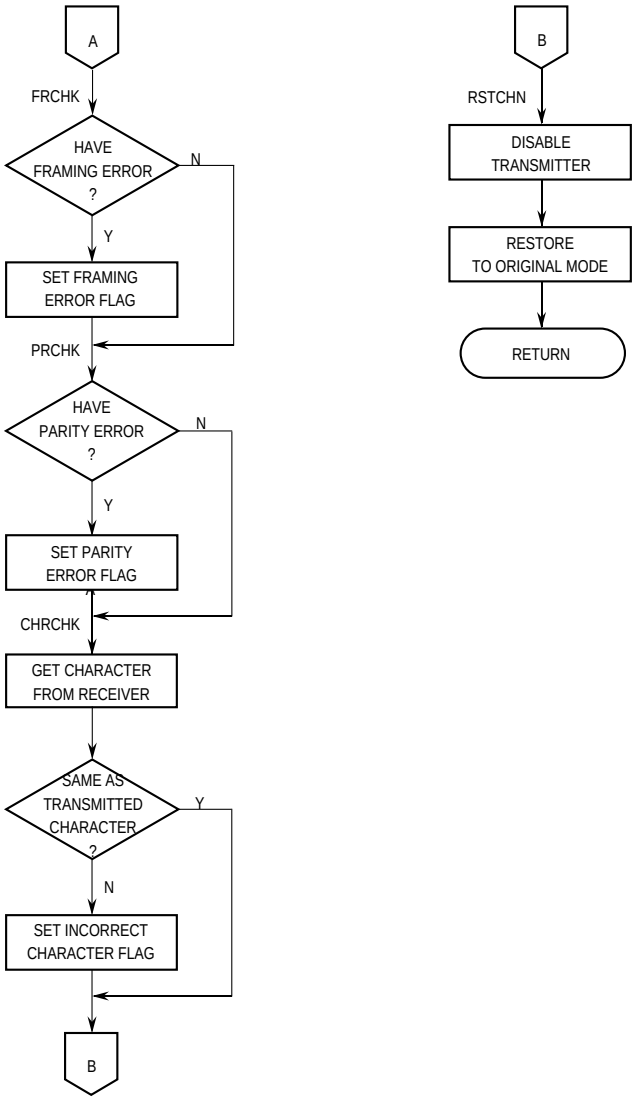


Figure 14-27. UART Mode Programming Flowchart (Sheet 3 of 5)

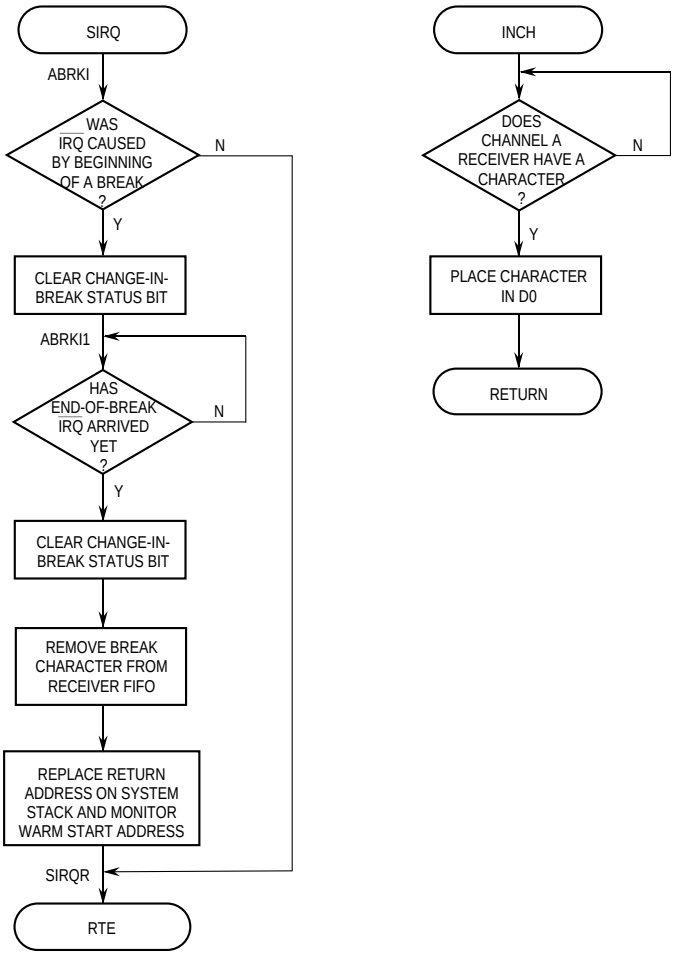


Figure 14-27. UART Mode Programming Flowchart (Sheet 4 of 5)

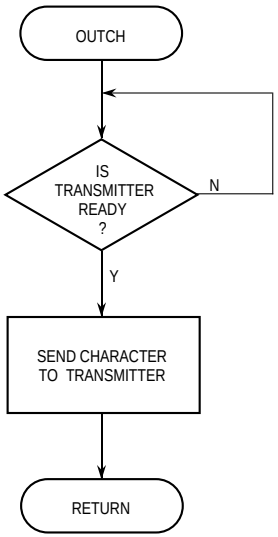


Figure 14-27. UART Mode Programming Flowchart (Sheet 5 of 5)