Chapter 8

Multichannel Buffered Serial Port

This chapter describes the operation and hardware of the two multichannel buffered serial ports (McBSP). This chapter also includes register definitions, and timing diagrams for the McBSPs.

Topic **Page**

8.1 Features

The multichannel buffered serial port (McBSP) is based on the standard serial port interface found on the TMS320C2x, ’C2xx, ’C5x, and ’C54x devices. The McBSP provides:

- Full-duplex communication
- Double-buffered data registers, which allow a continuous data stream
- Independent framing and clocking for receive and transmit
- Direct interface to industry-standard codecs, analog interface chips (AICs), and other serially connected A/D and D/A devices
- External shift clock or an internal, programmable frequency shift clock for data transfer
- Autobuffering capability through the five channel DMA controller.

In addition, the McBSP has the following capabilities:

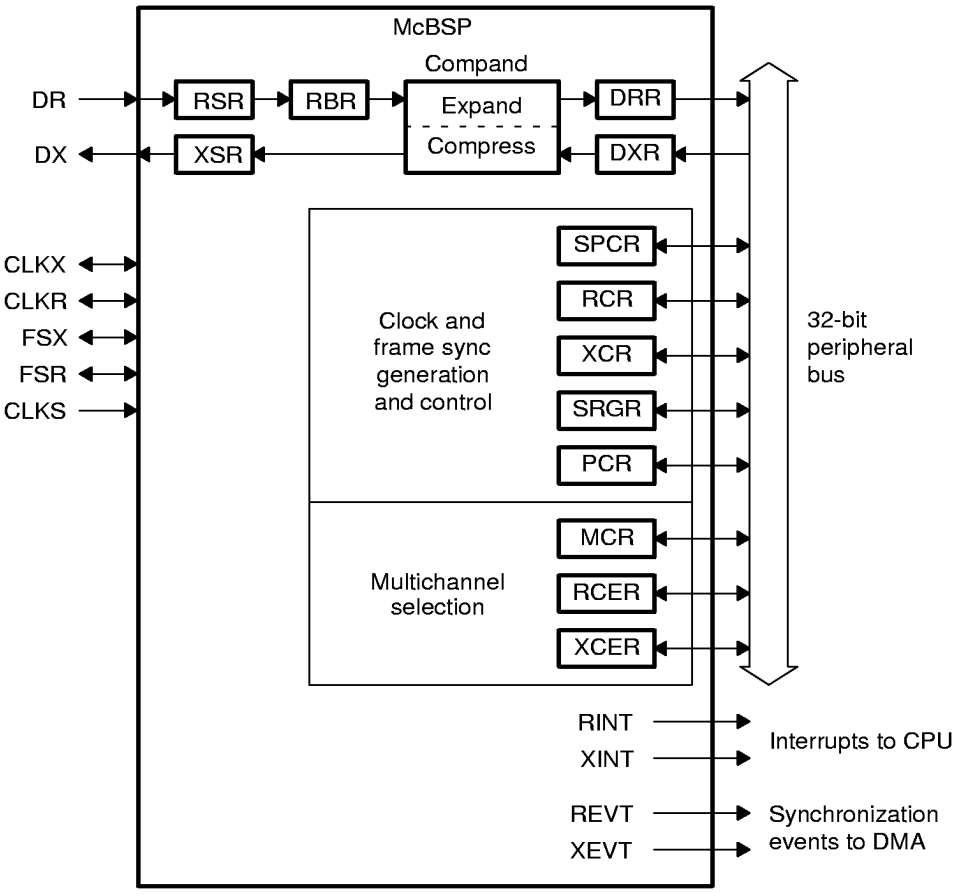
- Direct interface to:
  - T1/E1 framers
  - MVIP switching compatible and ST-BUS compliant devices including:
    - MVIP framers
    - H.100 framers
    - SCSA framers
  - IOM-2 compliant devices
  - AC97 compliant devices. The necessary multi-phase frame synchronization capability is provided
  - IIS compliant devices
  - SPI™ devices
- Multichannel transmit and receive of up to 128 channels
- A wide selection of data sizes including 8, 12, 16, 20, 24, and 32 bits
- μ-Law and A-Law companding
- 8-bit data transfers with option of LSB or MSB first
- Programmable polarity for both frame synchronization and data clocks
- Highly programmable internal clock and frame generation

8.2 McBSP Interface Signals and Registers

The multichannel buffered serial port (McBSP) consists of a data path and control path, which connect to external devices. Data is communicated to these external devices via separate pins for transmission and reception. Control information (clocking and frame synchronization) is communicated via four other pins. The device communicates to the McBSP via 32-bit wide control registers accessible via the internal peripheral bus.

The McBSP consists of a data path and control path as shown in Figure 8–1. Seven pins listed in Table 8–1 connect the control and data paths to external devices.

Figure 8–1. McBSP Block Diagram



Data is communicated to devices interfacing to the McBSP via the data transmit (DX) pin for transmission and the data receive (DR) pin for reception. Control information (clocking and frame synchronization) is communicated via CLKX, CLKR, FSX, and FSR. The 'C6x communicates to the McBSP via 32-bit-wide control registers accessible via the internal peripheral bus (see section 3.4 *Peripheral Bus*). Either the CPU or the DMA controller reads the received data from the data receive register (DRR) and writes the data to be transmitted to the data transmit register (DXR). Data written to the DXR is shifted out to DX via the transmit shift register (XSR). Similarly, receive data on the DR pin is shifted into the receive shift register (RSR) and copied into the receive buffer register (RBR). RBR is then copied to DRR, which can be read by the CPU or the DMA controller. This allows internal data movement and external data communications simultaneously.

The remaining registers accessible to the CPU configure the control mechanism of the McBSP. The McBSP registers are listed in Table 8–2. The control block consists of internal clock generation, frame-synchronization signal generation and control of these signals, and multichannel selection. This control block sends notification of important interrupts to the CPU and events to the DMA controller via the four signals shown in Table 8–3.

Table 8–1. McBSP Interface Signals

Pin	I/O/Z	Description
CLKR	I/O/Z	Receive clock
CLKX	I/O/Z	Transmit clock
CLKS	I	External clock
DR	I	Received serial data
DX	O/Z	Transmitted serial data
FSR	I/O/Z	Receive frame synchronization
FSX	I/O/Z	Transmit frame synchronization

Table 8–2. McBSP Registers

Hex Byte Address				
McBSP 0	**McBSP 1**	**Abbreviation**	**McBSP Register Name†**	**Section**
–	–	RBR	Receive buffer register	8.2
–	–	RSR	Receive shift register	8.2
–	–	XSR	Transmit shift register	8.2
018C 0000	0190 0000	DRR	Data receive register‡	8.2
018C 0004	0190 0004	DXR	Data transmit tegister	8.2
018C 0008	0190 0008	SPCR	Serial port control register	8.2.1
018C 000C	0190 000C	RCR	Receive control register	8.2.2
018C 0010	0190 0010	XCR	Transmit control register	8.2.2
018C 0014	0190 0014	SRGR	Sample rate generator register	8.5.1.1
018C 0018	0190 0018	MCR	Multichannel control register	8.6.1
018C 001C	0190 001C	RCER	Receive channel enable register	8.6.3.1
018C 0020	0190 0020	XCER	Transmit channel enable register	8.6.3.1
018C 0024	0190 0024	PCR	Pin control register	8.2.1

† The RBR, RSR, and XSR are not directly accessible via the CPU or the DMA controller.
‡ The CPU and DMA controller can only read this register.

Table 8–3. McBSP CPU Interrupts and DMA Synchronization Events

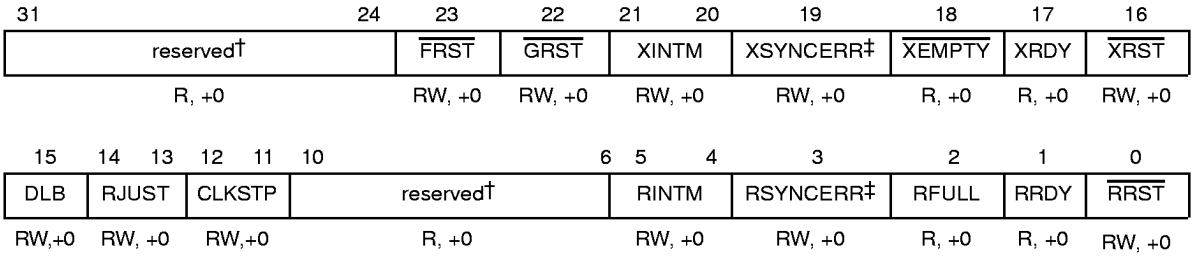
Interrupt Name	Description	Section
RINT	Receive Interrupt to CPU	8.3.3
XINT	Transmit Interrupt to CPU	8.3.3
REVT	Receive synchronization event to the DMA controller	8.3.2.1
XEVT	Transmit synchronization event to the DMA controller	8.3.2.2

8.2.1 Serial Port Configuration

The serial port is configured via the 32-bit serial port control register (SPCR) and the pin control register (PCR) shown in Figure 8–2 and Figure 8–3. The SPCR and PCR contain McBSP status control bits. Table 8–4 and Table 8–5 summarize the description of the SPCR and the PCR fields, respectively.

The PCR is also used to configure the serial port pins as general purpose inputs or outputs during receiver and/or transmitter reset. This is described in Section 8.8.

Figure 8–2. Serial Port Control Register (SPCR)



† Reserved-fields have *no* storage associated with them. However, they are always read as 0.
‡ Writing a 1 to this bit will set the error condition. Thus, it is used mainly for testing purposes or if this operation is desired.

Table 8–4. Serial Port Control Register (SPCR) Field Descriptions

Name	Function	Section
$\overline{\text{FRST}}$	Frame sync generator reset $\overline{\text{FRST}}$ = 0: The frame sync generation logic is reset. Frame sync signal is not generated by the sample rate generator. $\overline{\text{FRST}}$ = 1: Frame sync signal is generated after eight CLKG clocks. All frame counters are loaded with their programmed values.	8.5.3
$\overline{\text{GRST}}$	sample rate generator reset $\overline{\text{GRST}}$ = 0: Sample rate generator is reset. $\overline{\text{GRST}}$ = 1: Sample rate generator is pulled out of reset; CLKG is driven as per programmed values in sample rate generator register (SRGR).	8.5.1.2
RINTM	Receive interrupt mode RINTM = 00b: RINT driven by RRDY RINTM = 01b: RINT generated by end-of-subframe in multichannel operation RINTM = 10b: RINT generated by a new frame synchronization RINTM = 11b: RINT generated by RSYNCERR	8.3.3
XINTM	Transmit interrupt mode XINTM = 00b: XINT driven by XRDY XINTM = 01b: XINT generated by end-of-subframe or in multichannel operation XINTM = 10b: XINT generated by a new frame synchronization XINTM = 11b: XINT generated by XSYNCERR	8.3.3
RSYNCERR	Receive synchronization error RSYNCERR = 0: no frame synchronization error RSYNCERR = 1: frame synchronization error detected by McBSP.	8.3.7.2 8.3.7.5
XSYNCERR	Transmit synchronization error XSYNCERR = 0: no frame synchronization error XSYNCERR = 1: frame synchronization error detected by McBSP.	8.3.7.2 8.3.7.5
$\overline{\text{XEMPTY}}$	Transmit shift register (XSR) empty $\overline{\text{XEMPTY}}$ = 0: XSR is empty $\overline{\text{XEMPTY}}$ = 1: XSR is not empty	8.3.7.4
RFULL	Receive shift register (RSR) Full error condition RFULL = 0: Receiver is not in overrun condition. RFULL = 1: DRR is not read, RBR is full, and RSR is full with a new element.	8.3.7.1

Table 8–4. Serial Port Control Register (SPCR) Field Descriptions (Continued)

Name	Function	Section
RRDY	Receiver ready RRDY = 0: Receiver/transmitter is not ready. RRDY = 1: Receiver is ready with data to be read from DRR.	8.3.2
XRDY	Transmitter ready XRDY = 0: The transmitter is not ready. XRDY = 1: The transmitter is ready for data to be written to DXR.	8.3.2
$\overline{\text{RRST}}$	Receiver reset. This resets or enables the receiver. $\overline{\text{RRST}}$ = 0: The serial port receiver is disabled and is in reset state. $\overline{\text{RRST}}$ = 1: The serial port receiver is enabled.	8.3.1
$\overline{\text{XRST}}$	Transmitter reset. This resets or enables the transmitter. $\overline{\text{XRST}}$ = 0: The serial port transmitter is disabled and in reset state. $\overline{\text{XRST}}$ = 1: The serial port transmitter is enabled.	8.3.1
DLB	Digital loop back mode DLB = 0: digital loop back mode disabled DLB = 1: digital loop back mode enabled	8.5.2.5 8.5.2.6 8.5.3.2

Table 8–4. Serial Port Control Register (SPCR) Field Descriptions (Continued)

Name	Function	Section
RJUST	Receive data sign-extension and justification mode RJUST = 00b: right-justify and zero-fill MSBs in DRR RJUST = 01b: right-justify and sign-extend MSBs in DRR RJUST = 10b: left-justify and zero-fill LSBs in DRR RJUST = 11b: reserved	8.3.8
CLKSTP	Clock stop mode CLKSTP = 0Xb: Clock stop mode disabled. Normal clocking enabled for non-SPI mode. Clock stop mode enabled for various SPI™ modes when: CLKSTP = 10b and CLKXP = 0: Clock starts with rising edge without delay CLKSTP = 10b and CLKXP = 1: Clock starts with falling edge without delay CLKSTP = 11b and CLKXP = 0: Clock starts with rising edge with delay CLKSTP = 11b and CLKXP = 1: Clock starts with falling edge with delay	8.7

Figure 8–3. Pin Control Register (PCR)

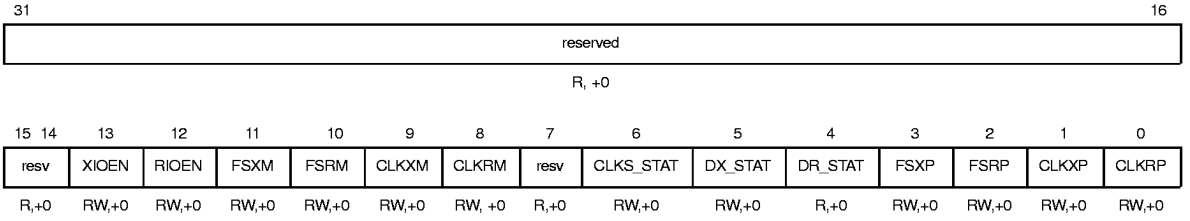


Table 8–5. Pin Control Register (PCR) Field Descriptions

Name	Function	Section
RIOEN	Receiver in general-purpose I/O mode ONLY when $\overline{\text{RRST}}$ = 0 in SPCR RIOEN = 0: DR and CLKS pins are not general-purpose inputs; FSR, CLKR are not general-purpose I/Os. RIOEN = 1: DR and CLKS pins are general-purpose inputs. FSR, CLKR are general-purpose I/Os. These serial port pins do not perform serial port operation.	8.8
XIOEN	Transmitter in general-purpose I/O mode ONLY when $\overline{\text{XRST}}$ = 0 in SPCR XIOEN = 0: CLKS pin is not a general-purpose input; DX pin is not a general purpose output; FSX, CLKX are not general-purpose I/Os. XIOEN = 1: CLKS pin is a general-purpose input; DX pin is a general-purpose output. FSX, CLKX are general-purpose I/Os. These serial port pins do not perform serial port operation.	8.8
FSXM	Transmit frame synchronization mode FSXM = 0: Frame synchronization signal is provided by an external source. FSX is an input pin. FSXM = 1: Frame synchronization generation is determined by the Sample Rate Generator frame synchronization mode bit FSGM in the SRGR.	8.5.3.3 and 8.8
FSRM	Receive frame synchronization mode FSRM = 0: Frame synchronization signals are generated by an external device. FSR is an input pin. FSRM = 1: Frame synchronization signals are generated internally by sample rate generator. FSR is an output pin except when GSYNC = 1 (subsection 8.5.1.1) in SRGR.	8.5.3.2 and 8.8
CLKRM	Receiver clock mode Case 1: Digital loop back mode not set (DLB = 0) in SPCR CLKRM = 0: Receive clock (CLKR) is an input driven by an external clock. CLKRM = 1: CLKR is an output pin and is driven by the sample rate generator. Case 2: Digital loop back mode set (DLB = 1) in SPCR CLKRM = 0: Receive clock (not the CLKR pin) is driven by the transmit clock (CLKX) which is based on the CLKXM bit in PCR. CLKR is in high-impedance. CLKRM = 1: CLKR is an output pin and is driven by the transmit clock. The transmit clock is derived based on CLKXM bit in the PCR.	8.5.2.6 and 8.8

Table 8–5. Pin Control Register (PCR) Field Descriptions (Continued)

Name	Function	Section
CLKXM	Transmitter clock mode CLKXM = 0: Transmitter clock is driven by an external clock with CLKX as an input pin. CLKXM = 1: CLKX is an output pin and is driven by the internal sample rate generator. During SPI mode (CLKSTP in SPCR is a nonzero value): CLKXM = 0: McBSP is a slave and (CLKX) is driven by the SPI master in the system. CLKR is internally driven by CLKX. CLKXM = 1: McBSP is a master and generates the transmitter clock (CLKX) to drive its receiver clock (CLKR) and the shift clock of the SPI-compliant slaves in the system.	8.5.2.7 and 8.8 8.7
CLKS_STAT	CLKS pin status. Reflects value on CLKS pin when selected as a general purpose input.	8.8
DX_STAT	DX pin status. Reflects value driven on to DX pin when selected as a general purpose output.	8.8
DR_STAT	DR pin status. Reflects value on DR pin when selected as a general purpose input.	8.8
FSRP	Receive frame synchronization polarity FSRP = 0: Frame synchronization pulse FSR is active high FSRP = 1: Frame synchronization pulse FSR is active low	8.3.4.1 and 8.8
FSXP	Transmit frame synchronization polarity FSXP = 0: Frame synchronization pulse FSX is active high FSXP = 1: Frame synchronization pulse FSX is active low	8.3.4.1 and 8.8
CLKXP	Transmit clock polarity CLKXP = 0: Transmit data driven on rising edge of CLKX CLKXP = 1: Transmit data driven on falling edge of CLKX	8.3.4.1 and 8.8
CLKRP	Receive clock polarity CLKRP = 0: Receive data sampled on falling edge of CLKR CLKRP = 1: Receive data sampled on rising edge of CLKR	8.3.4.1 and 8.8

8.2.2 Receive and Transmit Control Registers: RCR and XCR

The receive and transmit control registers (RCR and XCR), shown in Figure 8–4 and Figure 8–5, configure various parameters of the receive and transmit operations respectively.

Figure 8–4. Receive Control Register (RCR)

31	30 – 24	23 – 21	20 – 19	18	17 – 16
RPHASE	RFRLEN2	RWDLEN2	RCOMPAND	RFIG	RDATDLY
RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0

15	14 – 8	7 – 5	4 – 0
reserved	RFRLEN1	RWDLEN1	reserved
R, +0	RW, +0	RW, +0	R, +0

Figure 8–5. Transmit Control Register (XCR)

31	30 – 24	23 – 21	20 – 19	18	17 – 16
XPHASE	XFRLEN2	XWDLEN2	XCOMPAND	XFIG	XDATDLY
RW, +0	RW, +0	RW, +0	RW, +0	RW, +0	RW, +0

15	14 – 8	7 – 5	4 – 0
reserved	XFRLEN1	XWDLEN1	reserved
R, +0	RW, +0	RW, +0	R, +0

Table 8–6. Receive/Transmit Control Register (RCR/XCR) Field Description

Name	Function	Section
RPHASE	Receive phases RPHASE = 0: single phase frame RPHASE = 1: dual phase frame	8.3.4.2
XPHASE	Transmit phases XPHASE = 0: single phase frame XPHASE = 1: dual phase frame	8.3.4.2
RFRLEN(1/2)	Receive frame length in phase 1 and phase 2 RFRLEN(1/2) = 000 0000b: 1 word per phase RFRLEN(1/2) = 000 0001b: 2 words per phase • • • RFRLEN(1/2) = 111 1111b: 128 words per phase	8.3.4.3
XFRLEN(1/2)	Transmit frame length in phase 1 and phase 2 XFRLEN(1/2) = 000 0000b: 1 word per phase XFRLEN(1/2) = 000 0001b: 2 words per phase • • • XFRLEN(1/2) = 111 1111b: 128 words per phase	8.3.4.3
RWDLEN(1/2)	Receive element length in phase 1 and phase 2 RWDLEN(1/2) = 000b: 8 bits RWDLEN(1/2) = 001b: 12 bits RWDLEN(1/2) = 010b: 16 bits RWDLEN(1/2) = 011b: 20 bits RWDLEN(1/2) = 100b: 24 bits RWDLEN(1/2) = 101b: 32 bits RWDLEN(1/2) = 11Xb: reserved	8.3.4.4

Table 8–6. Receive/Transmit Control Register (RCR/XCR) Field Description (Continued)

Name	Function	Section
XWDLEN(1/2)	Transmit element length in phase 1 and phase 2 XWDLEN(1/2) = 000b: 8 bits XWDLEN(1/2) = 001b: 12 bits XWDLEN(1/2) = 010b: 16 bits XWDLEN(1/2) = 011b: 20 bits XWDLEN(1/2) = 100b: 24 bits XWDLEN(1/2) = 101b: 32 bits XWDLEN(1/2) = 11Xb: reserved	8.3.4.4
RCOMPAND	Receive companding mode. Modes other than 00b are only applicable when the appropriate RWDLEN is 000b, indicating 8-bit data. RCOMPAND = 00b: no companding, data transfer starts with MSB first. RCOMPAND = 01b: no companding, 8-bit data, transfer starts with LSB first. RCOMPAND = 10b: compand using μ-law for receive data. RCOMPAND = 11b: compand using A-law for receive data.	8.4
XCOMPAND	Transmit companding mode. Modes other than 00b are only applicable when the appropriate XWDLEN is 000b, indicating 8-bit data. XCOMPAND = 00b: no companding, data transfer starts with MSB first. XCOMPAND = 01b: no companding, 8-bit data, transfer starts with LSB first. XCOMPAND = 10b: compand using μ-law for transmit data. XCOMPAND = 11b: compand using A-law for transmit data.	8.4
RFIG	Receive frame ignore RFIG = 0: Unexpected receive frame synchronization pulses restart the transfer. RFIG = 1: Unexpected receive frame synchronization pulses are ignored.	8.3.6.1
XFIG	Transmit frame ignore XFIG = 0: Unexpected transmit frame synchronization pulses restart the transfer. XFIG = 1: Unexpected transmit frame synchronization pulses are ignored.	8.3.6.1

Table 8–6. Receive/Transmit Control Register (RCR/XCR) Field Description (Continued)

Name	Function	Section
RDATDLY	Receive data delay RDATDLY = 00b: 0-bit data delay RDATDLY = 01b: 1-bit data delay RDATDLY = 10b: 2-bit data delay RDATDLY = 11b: reserved	8.3.4.6
XDATDLY	Transmit data delay XDATDLY = 00b: 0-bit data delay XDATDLY = 01b: 1-bit data delay XDATDLY = 10b: 2-bit data delay XDATDLY = 11b: reserved	8.3.4.6

8.3 Data Transmission and Reception

As shown in Figure 8–1, on page 8-3 the receive operation is triple buffered and transmit operation is double buffered. Receive data arrives on DR and is shifted into the RSR. Once a full element (8, 12, 16, 20, 24, or 32 bits) is received, the RSR is copied to the receive buffer register (RBR) only if RBR is not full. RBR is then copied to DRR unless DRR has not been read by the CPU or the DMA controller.

Transmit data is written by the CPU or the DMA controller to the DXR. If there is no data in the XSR, the value in the DXR is copied to the XSR. Otherwise, the DXR is copied to the XSR when the last bit of data is shifted out on DX. After transmit frame synchronization, the XSR begins shifting out the transmit data on DX.

8.3.1 Resetting the Serial Port: $\overline{\text{(R/X)RST}}$, $\overline{\text{GRST}}$, and $\overline{\text{RESET}}$

The serial port can be reset in the following two ways:

- Device reset ($\overline{\text{RESET}}$ pin is low) places the receiver, the transmitter and the sample rate generator in reset. When the device reset is removed ($\overline{\text{RESET}} = 1$), $\overline{\text{FRST}} = \overline{\text{GRST}} = \overline{\text{RRST}} = \overline{\text{XRST}} = 0$, keeping the entire serial port in the reset state.
- The serial port transmitter and receiver can be independently reset by the $\overline{\text{XRST}}$ and $\overline{\text{RRST}}$ bits in the SPCR. The sample rate generator is reset by the $\overline{\text{GRST}}$ bit in the SPCR.

Table 8–7 shows the state of the McBSP pins when the serial port is reset by each of these methods.

Table 8–7. Reset State of McBSP Pins

McBSP Pins	Direction	Device Reset ($\overline{\text{RESET}}$ = 0)	McBSP Reset
			Receiver Reset ($\overline{\text{RRST}}$ = 0 and $\overline{\text{GRST}}$ = 1)
DR	I	Input	Input
CLKR	I/O/Z	Input	Known state if input; CLKR if output
FSR	I/O/Z	Input	Known state if input; FSRP(inactive state) if output
CLKS	I	Input	Input
			Transmitter Reset ($\overline{\text{XRST}}$ = 0 and $\overline{\text{GRST}}$ = 1)
DX	O	High impedance†	High impedance
CLKX	I/O/Z	Input	Known state if input; CLKX if output
FSX	I/O/Z	Input	Known state if input; FSXP(inactive state) if output
CLKS	I	Input	Input

- **Device reset or McBSP reset:** When the McBSP is reset in either of these two ways, the state machine is reset to its initial state. This initial state includes resetting all counters and status bits. The receive status bits include RFULL, RRDY, and RSYNCERR. The transmit status bits are $\overline{\text{XEMPTY}}$, XRDY, and XSYNCERR.
- **Device reset:** When McBSP is reset due to device reset, the entire serial port (including the transmitter, receiver, and the sample rate generator) is reset. All input-only pins and 3-state pins should be in a known state. The output-only pin, DX, is in the high impedance state. Since the sample rate generator is also reset ($\overline{\text{GRST}}$ = 0), the sample rate generator clock, CLKG, is driven by a divide-by-2 CPU clock, and the frame sync signal, FSG, is not generated. See subsection 8.5.1.2 for more information on sample rate generator reset. When the device is pulled out of reset, the serial port remains in the reset condition ($\overline{\text{(R/X)RST}}$ = $\overline{\text{FRST}}$ = 0), and in this condition the serial port pins can be used as general-purpose I/O as described in section 8.8.

- **McBSP reset:** When the receiver and transmitter reset bits, $\overline{\text{RRST}}$ and $\overline{\text{XRST}}$, are written with 0, the respective portions of the McBSP are reset, and activity in the corresponding section stops. All input-only pins, such as DR and CLKS, and all other pins that are configured as inputs are in a known state. FS(R/X) is driven to its inactive state (same as its polarity bit, FS(R/X)P) if it is an output. If CLK(R/X) are programmed as outputs, they are driven by CLKG, provided that $\overline{\text{GRST}}$ = 1. Lastly, the DX pin will be in the high-impedance state when the transmitter is reset. During normal operation, the sample rate generator can be reset by writing a 0 to $\overline{\text{GRST}}$. $\overline{\text{GRST}}$ should be low only when neither the transmitter nor the receiver is using the sample rate generator. In this case, the internal sample rate generator clock CLKG, and its frame sync signal (FSG) is driven inactive (low). When the sample rate generator is not in the reset state ($\overline{\text{GRST}}$ = 1), FSR and FSX are in an inactive state when $\overline{\text{RRST}}$ = 0 and $\overline{\text{XRST}}$ = 0, respectively, even if they are outputs driven by FSG. This ensures that when only one portion of the McBSP is in reset, the other portion can continue operation when $\overline{\text{FRST}}$ = 1 and frame sync is driven by FSG.

- **Sample rate generator reset:** As mentioned previously, the sample rate generator is reset when the device is reset or when its reset bit, $\overline{\text{GRST}}$, is written with 0. In the case of device reset, the CLKG signal is driven by a divide-by-2 CPU clock and FSG is driven inactive (low). If you want to reset the sample rate generator when neither the transmitter nor receiver is fed by the CLKG and FSG, $\overline{\text{GRST}}$ in the SRGR can be programmed to 0. Here, CLKG and FSG are driven inactive (low). When $\overline{\text{GRST}}$ = 1, CLKG comes up running as programmed in the SRGR. Later if $\overline{\text{FRST}}$ = 1, FSG is driven active (high) after eight cycles have elapsed.

The serial port initialization procedure is as follows:

1) Set $\overline{\text{XRST}}$ = $\overline{\text{RRST}}$ = $\overline{\text{FRST}}$ = 0 in SPCR. If the device is coming out of reset, this step is not required.
2) Program only the McBSP configuration registers (and not the data registers) listed in Table 8–2 as required when the serial port is in the reset state ($\overline{\text{XRST}}$ = $\overline{\text{RRST}}$ = $\overline{\text{FRST}}$ = 0).
3) Wait two bit clocks. This is to ensure proper synchronization internally.
4) Set up data acquisition as desired.
5) Set $\overline{\text{XRST}}$ = $\overline{\text{RRST}}$ = 1 to enable the serial port. The value written to the SPCR at this time should have only the reset bits changed to 1 and the remaining bit-fields should have the same value as in Step 2 above.
6) Set $\overline{\text{FRST}}$ = 1. Now the McBSP is ready to transmit and/or receive, if it is the frame master.

Alternatively, on either write (steps 1 and 5 above), the transmitter and receiver can be placed in or taken out of reset individually by modifying only the desired bit. The necessary duration of the active(low) period of $\overline{XRST}$ or $\overline{RRST}$ is at least two bit clocks (CLKR/CLKX). This procedure for reset initialization can be applied generally when the receiver or transmitter has to be reset during its normal operation and also when the sample rate generator is not used for either operation. The sample rate generator reset procedure is explained in section 8.5.1.2.

Notes:

1) The appropriate-fields in the serial port configuration registers SPCR, PCR, RCR, XCR, and SRGR should only be modified when the affected portion of the serial port is in reset.
2) The data transmit register, DXR, should be loaded by the CPU or DMA only when the transmitter is not in reset ($\overline{XRST}$ = 1). The exception to this rule is during non-digital loop back mode, which is described in section 8.4.1.
3) The multichannel selection registers MCR, XCER, and RCER can be modified at any time as long as they are not being used by the current block in the multichannel selection. See section 8.6.3.2 for further information .

8.3.2 Determining Ready Status

RRDY and XRDY indicate the ready state of the McBSP receiver and transmitter, respectively. Writes and reads of the serial port can be synchronized by polling RRDY and XRDY, or by using the events sent to the DMA controller (REVT and XEVT) or the interrupts to the CPU (RINT and XINT) that the events generate. Note that reading the DRR and writing to DXR affects RRDY and XRDY, respectively.

8.3.2.1 Receive Ready Status: REVT, RINT, and RRDY

RRDY = 1 indicates that the RBR contents have been copied to the DRR and that the data can be read by either the CPU or the DMA controller. Once that data has been read by either the CPU or the DMA controller, RRDY is cleared to 0. Also, at device reset or serial port receiver reset ($\overline{RRST}$ = 0), the RRDY is cleared to 0 to indicate no data has yet been received and loaded into DRR. RRDY directly drives the McBSP receive event to the DMA controller (via REVT). Also, the McBSP receive interrupt (RINT) to the CPU can be driven by RRDY if RINTM = 00b (default value) in the SPCR.

8.3.2.2 Transmit Ready Status: XEVT, XINT, and XRDY

XRDY = 1 indicates that the DXR contents have been copied to XSR and that DXR is ready to be loaded with a new data word. When the transmitter transitions from reset to nonreset ($\overline{\text{XRST}}$ transitions from 0 to 1), XRDY also transitions from 0 to 1 indicating that the DXR is ready for new data. Once new data is loaded by the CPU or the DMA controller, XRDY is cleared to 0. However, once this data is copied from the DXR to the XSR, XRDY transitions again from 0 to 1. The CPU or the DMA controller can write to DXR although XSR has not yet been shifted out on DX. XRDY directly drives the transmit synchronization event to the DMA controller (via XEVT). Also, the transmit interrupt (XINT) to the CPU can be driven by XRDY if XINTM = 00b (default value) in the SPCR.

8.3.3 CPU Interrupts: (R/X)INT

The receive interrupt (RINT) and transmit interrupt (XINT) signal the CPU of changes to the serial port status. Four options exist for configuring these interrupts. These options are set by the receive/transmit interrupt mode field, (R/X)INTM, in the SPCR. The possible values of the mode and the configurations they represent are:

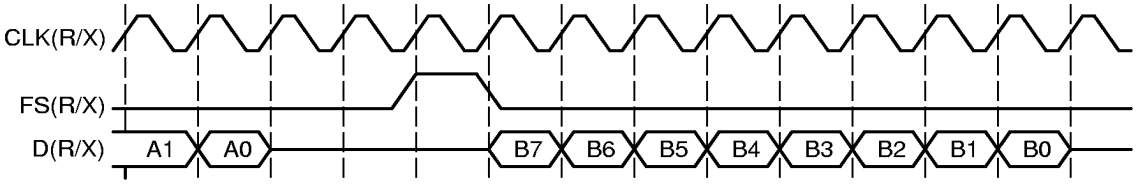
- (R/X)INTM = 00b. Interrupt on every serial element by tracking the (R/X)RDY bits in the SPCR.
- (R/X)INTM = 01b. Interrupt at the end of a subframe (16 elements or less) within a frame. See subsection 8.6.3.3 for details.
- (R/X)INTM = 10b. Interrupt on detection of frame synchronization pulses. This generates an interrupt even when the transmitter/receiver is in reset. This is done by synchronizing the incoming frame sync pulse to the CPU clock and sending it to the CPU via (R/X)INT. This is described in subsection 8.5.3.4.
- (R/X)INTM = 11b. Interrupt on frame synchronization error. Note that if any of the other interrupt modes are selected, (R/X)SYNCERR may be read when servicing the interrupts to detect this condition. See subsections 8.3.7.2 and 8.3.7.5 for more details on synchronization error.

8.3.4 Frame and Clock Configuration

Figure 8–6 shows typical operation of the McBSP clock and frame sync signals. Serial clocks CLKR and CLKX define the boundaries between bits for receive and transmit respectively. Similarly, frame sync signals FSR and FSX define the beginning of an element transfer. The McBSP allows configuration of various parameters for data frame synchronization. This can be done independently for receive and transmit and consists of the following items:

- Polarities of FSR, FSX, CLKX, and CLKR
- A choice of single- or dual-phase frames
- For each phase, the number of elements per frame
- For each phase, the number of bits per element
- Subsequent frame synchronization can restart the serial data stream or be ignored.
- The data delay from frame synchronization to first data bit can be 0-, 1-, or 2-bit delays.
- Right- or left-justification as well as sign-extension or zero-filling can be chosen for receive data.

Figure 8–6. Frame and Clock Operation



8.3.4.1 Frame and Clock Operation

Receive and transmit frame sync pulses can be generated either internally by the sample rate generator (see section 8.5.1) or driven by an external input. The source of frame sync is selected by programming the mode bit, FS(R/X)M, in the PCR. FSR is also affected by the GSYNC bit in the SRGR (see section 8.5.3.2 for details). Similarly, receive and transmit clocks can be selected to be inputs or outputs by programming the mode bit, CLK(R/X)M, in the PCR.

When FSR and FSX are inputs (FSXM = FSRM = 0), the McBSP detects them on the internal falling edge of clock, CLKR_int and CLKX_int, respectively (see Figure 8–36 on page 8-49). The receive data arriving at DR pin is also sampled

on the falling edge of CLKR_int. These internal clock signals are either derived from external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs driven by the sample rate generator, they are generated (transition to their active state) on the rising edge of the internal clock, CLK(R/X)_int. Similarly, data on DX is output on the rising edge of CLKX_int. See section 8.3.4.6 for further details.

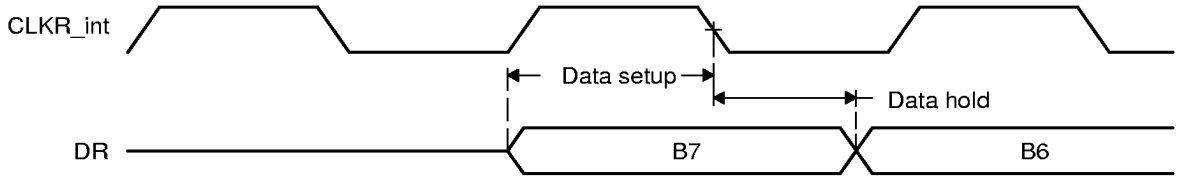
FSRP, FSXP, CLKRP, and CLKXP configure the polarities of FSR, FSX, CLKR, and CLKX as shown in Table 8–5. All frame sync signals (FSR_int and FSX_int) internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to the McBSP), and FSRP = FSXP = 1, the external active low frame sync signals are inverted before being sent to the receiver signal (FSR_int) and transmitter signal (FSX_int). Similarly, if internal synchronization is selected (FSR/FSX are outputs and GSYNC = 0), the internal active-high sync signals are inverted if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin. Figure 8–36 shows this inversion using XOR gates.

On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Data is always transmitted on the rising edge of CLKX_int. If CLKXP = 1 and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling-edge triggered input clock on CLKX is inverted to a rising-edge triggered clock before being sent to the transmitter. If CLKXP = 1, and internal clocking selected (CLKXM = 1 and CLKX is an output pin), the internal (rising-edge triggered) clock, CLKX_int, is inverted before being sent out on the CLKX pin.

Similarly, the receiver can reliably sample data that is clocked with a rising-edge clock (by the transmitter). The receive clock polarity bit, CLKRP, sets the edge used to sample received data. The receive data is always sampled on the falling edge of CLKR_int. Therefore, if CLKRP = 1 and external clocking is selected (CLKRM = 0 and CLKR is an input pin), the external rising-edge triggered input clock on CLKR is inverted to a falling-edge clock before being sent to the receiver. If CLKRP = 1 and internal clocking is selected (CLKRM = 1), the internal falling-edge triggered clock is inverted to a rising-edge before being sent out on the CLKR pin.

In a system where the same clock (internal or external) is used to clock the receiver and transmitter, CLKRP = CLKXP. The receiver uses the opposite edge as the transmitter to ensure valid setup and hold times of data around this edge. Figure 8–7 shows how data clocked by an external serial device using a rising-edge clock can be sampled by the McBSP receiver with the falling edge of the same clock.

Figure 8–7. Receive Data Clocking



8.3.4.2 Frame Synchronization Phases

Frame synchronization indicates the beginning of a transfer on the McBSP. The data stream following frame synchronization can have up to two phases, phase 1 and phase 2. The number of phases can be selected by the phase bit, (R/X)PHASE, in the RCR and XCR. The number of elements per frame and bits per element can be independently selected for each phase via (R/X)FRLEN(1/2) and (R/X)WDLEN(1/2), respectively. Figure 8–8 shows a frame in which the first phase consists of two elements of 12 bits each followed by a second phase of three elements of 8 bits each. The entire bit stream in the frame is contiguous; no gaps exist either between elements or phases. Table 8–8 shows the fields in the receive/transmit control registers (RCR/XCR) that control the frame length and element length for each phase for both the receiver and the transmitter. The maximum number of elements per frame is 128 for a single-phase frame and 256 for a dual phase frame. The number of bits per element can be 8, 12, 16, 20, 24, or 32.

Figure 8–8. Dual-Phase Frame Example

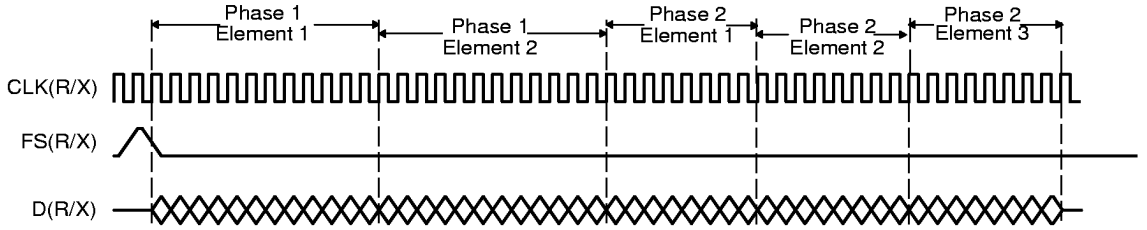


Table 8–8. RCR/XCR Fields Controlling Elements per Frame and Bits per Element

Serial Port McBSP0/1	Frame Phase	RCR/XCR field Control	
		Elements per Frame	Bits per Element
Receive	1	RFRLEN1	RWDLEN1
Receive	2	RFRLEN2	RWDLEN2
Transmit	1	XFRLEN1	XWDLEN1
Transmit	2	XFRLEN2	XWDLEN2

8.3.4.3 Frame Length: (R/X)FRLEN(1/2)

Frame length can be defined as the number of serial elements transferred per frame. The length corresponds to the number of elements or logical time slots or channels per frame synchronization signal. The 7-bit (R/X)FRLEN(1/2) field in the (R/X)CR supports up to 128 elements per frame as shown in Table 8–9. (R/X)PHASE = 0 represents a single-phase data frame and a (R/X)PHASE = 1 selects a dual-phase for the data stream. For a single-phase frame, the value of FRLEN2 does not matter. Program the frame length fields with (*w minus 1)*, where *w* represents the number of elements per frame. For Figure 8–8, (R/X)FRLEN1 = 1 or 0000001b and (R/X)FRLEN2 = 2 or 0000010b.

Table 8–9. McBSP Receive/Transmit Frame Length 1/2 Configuration

(R/X)PHASE	(R/X)FRLEN1	(R/X)FRLEN2	Frame Length
0	$0 \le n \le 127$	x	Single phase frame; (n+1) words per frame
1	$0 \le n \le 127$	$0 \le m \le 127$	Dual phase frame; (n+1) plus (m+1) words per frame

8.3.4.4 Element Length: (R/X)WDLEN(1/2)

The (R/X)WDLEN(1/2) fields in the receive/transmit control register determine the element length in bits per element for the receiver and the transmitter for each phase of the frame, as shown in Table 8–8. Table 8–10 shows how the value of these fields selects particular element lengths in bits. For the example in Figure 8–8, (R/X)WDLEN1 = 001b and (R/X)WDLEN2 = 000b. If (R/X)PHASE = 0, indicating a single-phase frame, (R/X)WDLEN2 is not used by the McBSP and its value does not matter.

Table 8–10. McBSP Receive/Transmit element Length Configuration

(R/X)WDLEN(1/2)	McBSP Element Length (Bits)
000	8
001	12
010	16
011	20
100	24
101	32
110	Reserved
111	Reserved

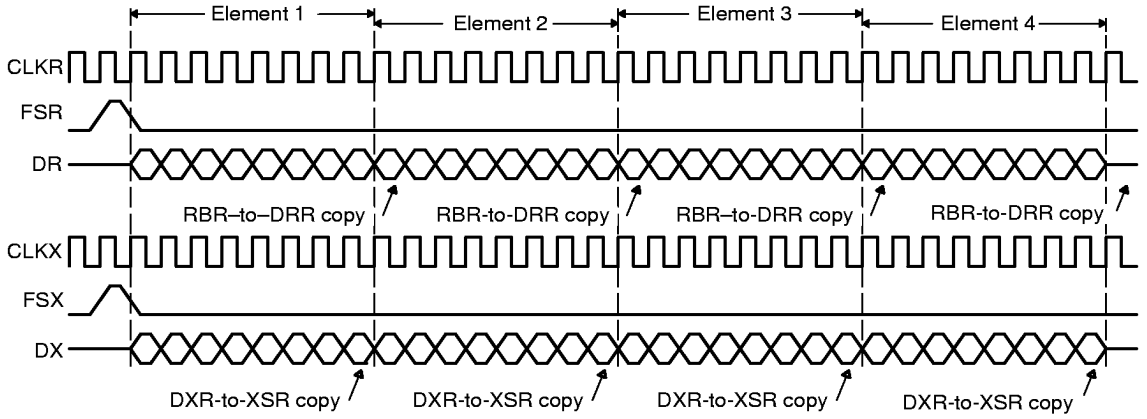
8.3.4.5 Data Packing using Frame Length and Element Length

The frame length and element length can be manipulated to effectively pack data. For example, consider a situation in which four 8-bit elements are transferred in a single-phase frame, as shown in Figure 8–9. In this case:

- (R/X)PHASE = 0, indicating a single-phase frame
- (R/X)FRLEN1 = 0000011b, indicating a 4-element frame
- (R/X)WDLEN1 = 000b, indicating 8-bit elements

In this case, four 8-bit data elements are transferred to and from the McBSP by the CPU or the DMA controller. Four reads of DRR and four writes of DXR are necessary for each frame.

Figure 8–9. Single-Phase Frame of Four 8-Bit Elements

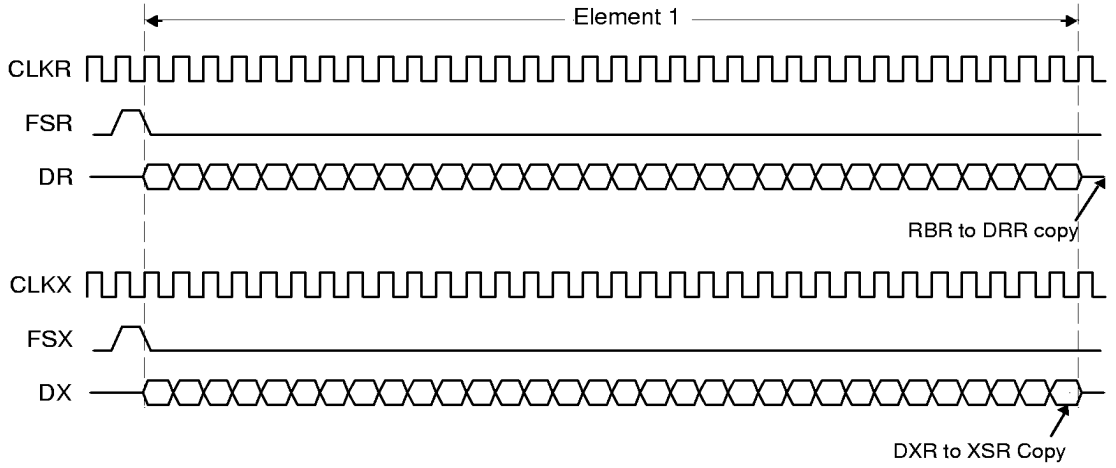


The example in Figure 8–9 can also be viewed as a data stream of a single-phase frame consisting of one 32-bit data element as shown in Figure 8–10. In this case:

- (R/X)PHASE = 0, indicating a single phase frame
- (R/X)FRLEN1 = 0b, indicating a 1-element frame
- (R/X)WDLEN1 = 101b, indicating 32-bit elements

In this case, one 32-bit data element is transferred to and from the McBSP by the CPU or the DMA controller. Thus, one read of DRR and one write of DXR is necessary for each frame. This results in only one-fourth the number of transfers compared to the previous case. This manipulation reduces the percentage of bus time required for serial port data movement.

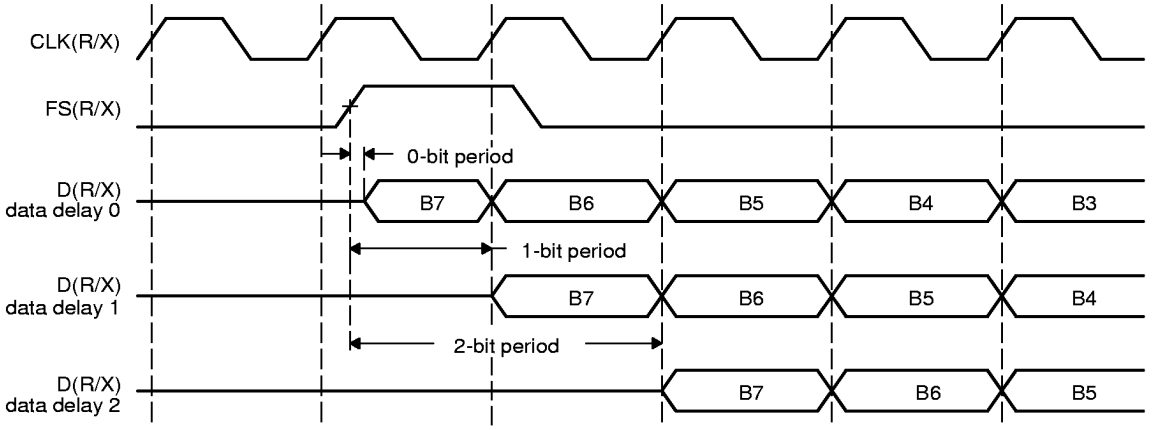
Figure 8–10. Single-Phase Frame of One 32-Bit Element



8.3.4.6 Data Delay: (R/X)DATDLY

The start of a frame is defined by the first clock cycle in which frame synchronization is found to be active. The beginning of actual data reception or transmission with respect to the start of the frame can be delayed if required. This delay is called data delay. RDATDLY and XDATDLY specify the data delay for reception and transmission, respectively. The range of programmable data delay is zero to two bit-clocks ((R/X)DATDLY = 00b to10b) as described in Table 8–6 and shown in Figure 8–11. Typically 1-bit delay is selected because data often follows a 1-cycle active frame sync pulse.

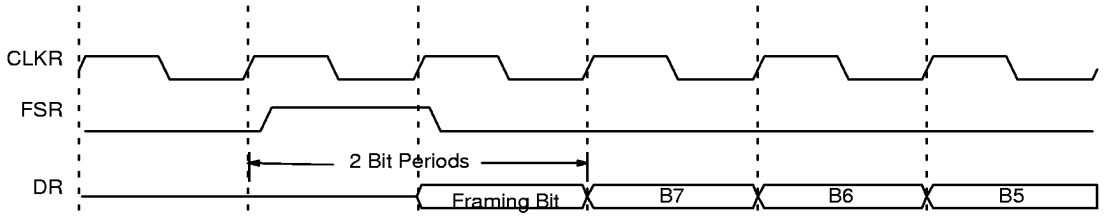
Figure 8–11. Data Delay



Normally a frame sync pulse is detected or sampled with respect to an edge of serial clock CLK(R/X). Thus, on a subsequent cycle (depending on data delay value), data can be received or transmitted. However, in the case of zero-bit data delay, the data must be ready for reception and/or transmission on the same serial clock cycle. For reception, this problem is solved by receive data being sampled on the first falling edge of CLKR when an active (high) FSR is detected. However, data transmission must begin on the rising edge of CLKX that generated the frame synchronization. Therefore, the first data bit is assumed to be present in the XSR and DX. The transmitter then asynchronously detects the frame synchronization, FSX goes active, and immediately starts driving the first bit to be transmitted on the DX pin.

Another common operation uses a data delay of 2. This configuration allows the serial port to interface to different types of T1 framing devices in which the data stream is preceded by a framing bit. During reception of such a stream with a data delay of two bits the framing bit appears after a 1-bit delay and data appears after a 2-bit delay), the serial port essentially discards the framing bit from the data stream, as shown in Figure 8–12. In transmission, by delaying the first transfer bit, the serial port essentially inserts a blank period (a high-impedance period) in place of the framing bit. Here, it is expected that the framing device inserts its own framing bit or that the framing bit is generated by another device. Alternatively, you may pull-up or pull-down DX to achieve the desired value.

Figure 8–12. 2-Bit Data Delay Used to Discard Framing Bit

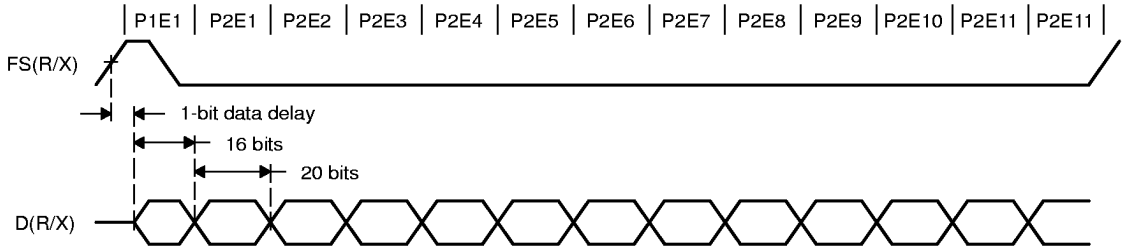


8.3.4.7 Multiphase Frame Example: AC97

Figure 8–13 shows an example of the Audio Codec '97 (AC97) standard, which uses the dual-phase frame feature. The first phase consists of a single 16-bit element. The second phase consists of 12 20-bit elements. The phases are configured as follows:

- (R/X)PHASE = 1b: specifying dual-phase frame
- (R/X)FRLEN1 = 0b: specifying 1 element per frame in phase 1
- (R/X)WDLEN1 = 010b: specifying 16-bits per element in phase 1
- (R/X)FRLEN2 = 0001011b: specifying 12 elements per frame in phase 2
- (R/X)WDLEN2 = 011b: specifying 20-bits per element in phase 2
- CLK(R/X)P = 0: specifying that the receive data sampled on falling edge of CLKR and transmit data is clocked on rising edge of CLKX
- FS(R/X)P = 0: indicating active frame sync signals are used
- (R/X)DATDLY = 01b: indicating a data delay of one bit-clock

Figure 8–13. AC97 Dual-Phase Frame Format†

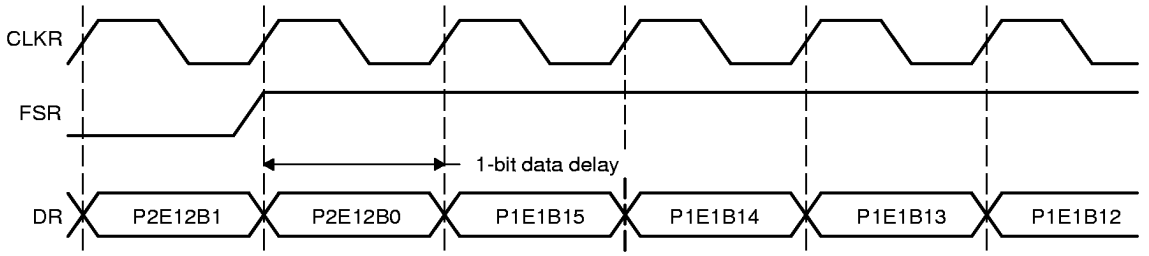


† PxEy denotes Phase x and Element y

Figure 8–13 shows the AC97 timing near frame synchronization. First the frame sync pulse itself overlaps the first element. In McBSP operation, the inactive-to-active transition of the frame synchronization signal actually indicates frame synchronization. For this reason, frame synchronization can be high an arbitrary number of bit clocks. Only after the frame synchronization is recognized as inactive and then active again is the next frame synchronization recognized.

In Figure 8–14, there is 1-bit data delay. Regardless of the data delay, transmission can occur without gaps. The last bit of the previous (last) element in phase 2 is immediately followed by the first data bit of the first element in phase 1 of the next data frame.

Figure 8–14. AC97 Bit Timing Near Frame Synchronization†



† PxEyBz denotes Phase x, Element y, Bit z

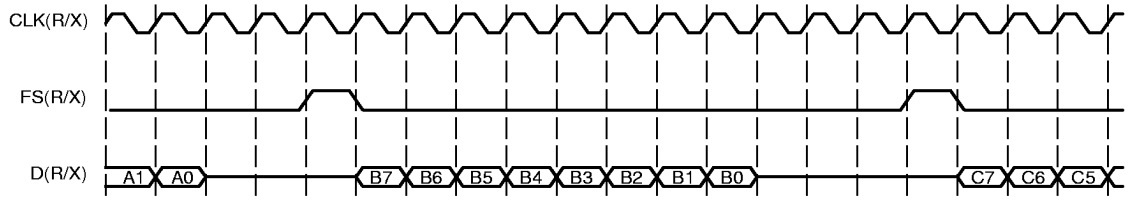
8.3.5 McBSP Standard Operation

During a serial transfer, there are typically periods of serial port inactivity between packets or transfers. The receive and transmit frame synchronization pulse occurs for every serial transfer. When the McBSP is not in the reset state and has been configured for the desired operation, a serial transfer can be initiated by programming (R/X)PHASE = 0 for a single-phase frame with the required number of elements programmed in (R/X)FRLEN1. The number of elements can range from 1 to 128 ((R/X)FRLEN1 = 0x0 to 0x7F). The required serial element length is set in the (R/X)WDLEN1 field in the (R/X)CR. If dual phase is required for the transfer, RPHASE = 1 and each (R/X)FRLEN(1/2) can be set to any value between 0x0 to 0x7F.

Figure 8–15 shows a single-phase data frame of one 8-bit element. Since the transfer is configured for 1-bit data delay, the data on the DX and DR pins are available one bit clock after FS(R/X) goes active. This figure as well as all others in this section uses the following assumptions:

- (R/X)PHASE = 0, specifying a single-phase frame
- (R/X)FRLEN1 = 0b, specifying 1 element per frame
- (R/X)WDLEN1 = 000b, specifying 8-bit element
- (R/X)FRLEN2 = (R/X)WDLEN2 = (value is ignored)
- CLK(R/X)P = 0, specifying that the receive data is clocked on the falling edge and transmit data is clocked on the rising edge
- FS(R/X)P = 0, indicating that active (high) frame sync signals are used
- (R/X)DATDLY = 01b, indicating a one-bit data delay

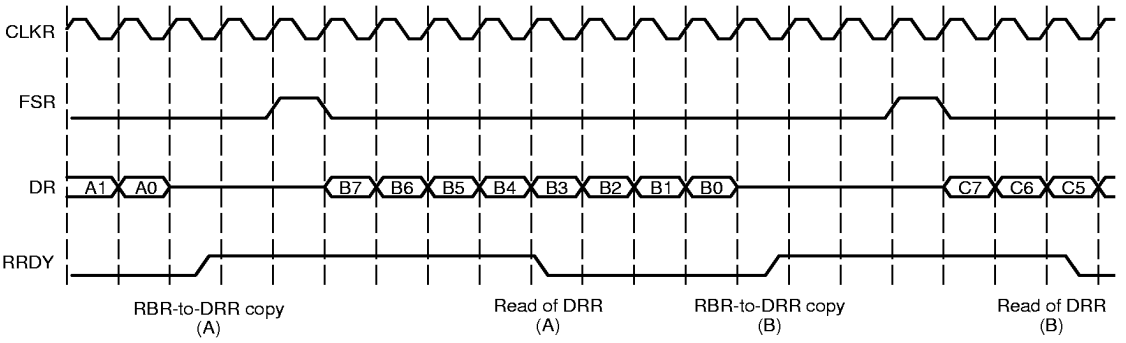
Figure 8–15. McBSP Standard Operation



8.3.5.1 Receive Operation

Figure 8–16 shows serial reception. Once the receive frame synchronization signal (FSR) transitions to its active state, it is detected on the first falling edge of the receiver's CLKR. The data on the DR pin is then shifted into the receive shift register (RSR) after the appropriate data delay as set by RDATDLY. The contents of RSR is copied to RBR at the end of every element on the rising edge of the clock, provided RBR is not full with the previous data. Then, an RBR-to-DRR copy activates the RRDY status bit to 1 on the following falling edge of CLKR. This indicates that the receive data register (DRR) is ready with the data to be read by the CPU or the DMA controller. RRDY is deactivated when the DRR is read by the CPU or the DMA controller.

Figure 8–16. Receive Operation

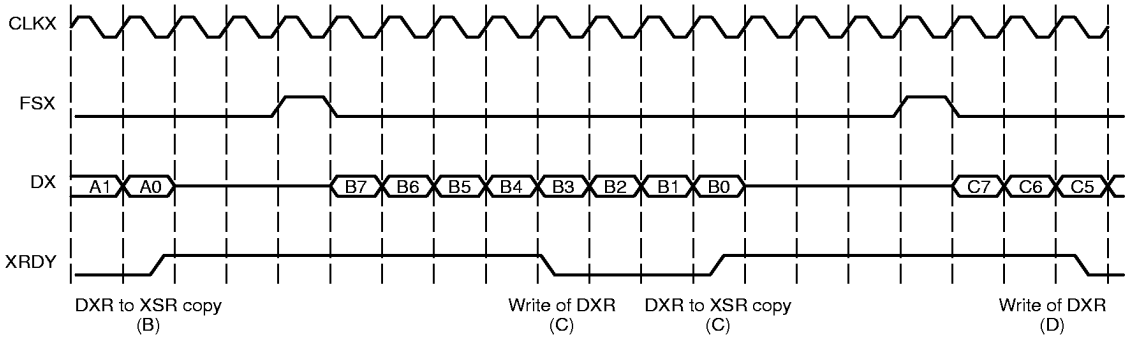


8.3.5.2 Transmit Operation

Once transmit frame synchronization occurs, the value in the transmit shift register, XSR, is shifted out and driven on the DX pin after the appropriate data delay as set by XDATDLY. XRDY is activated on every DXR-to-XSR copy on the following falling edge of CLKX, indicating that the data transmit register (DXR) is written with the next data to be transmitted. XRDY is deactivated when the DXR is written by the CPU or the DMA controller. Figure 8–17 illus-

trates serial transmission. See section 8.3.7.4 for transmit operation when the transmitter is pulled out of reset ($\overline{\text{XRST}}$ = 1).

Figure 8–17. Transmit Operation



8.3.5.3 Maximum Frame Frequency

The frame frequency is determined by the period between frame synchronization signals:

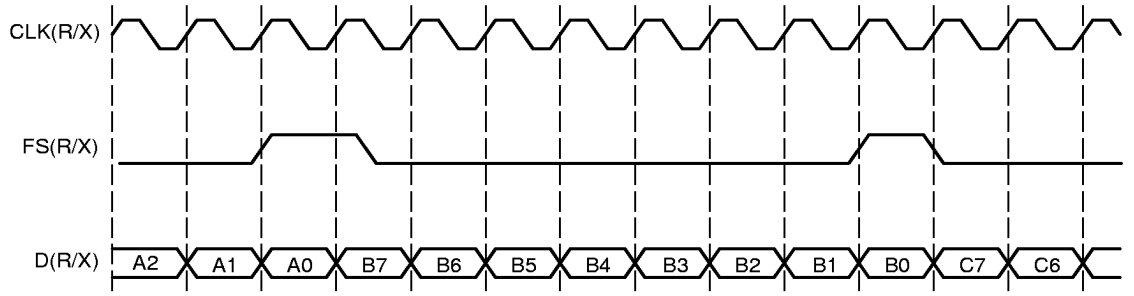
$$\text{Frame frequency} = \frac{\text{Bit-clock frequency}}{\text{Number of bit clocks between frame sync signals}}$$

The frame frequency may be increased by decreasing the time between frame synchronization signals in bit clocks (limited only by the number of bits per frame). As the frame transmit frequency is increased, the inactivity period between the data frames for adjacent transfers decreases to zero. The minimum time between frame synchronization pulses is the number of bits transferred per frame. This time also defines the maximum frame frequency:

$$\text{Maximum frame frequency} = \frac{\text{Bit-clock frequency}}{\text{Number of bits per frame}}$$

Figure 8–18 shows the McBSP operating at maximum frame frequency. The data bits in consecutive frames are transmitted continuously with no inactivity between bits. If there is a 1-bit data delay, as shown, the frame synchronization pulse overlaps the last bit transmitted in the previous frame.

Figure 8–18. Maximum Frame Frequency Transmit and Receive



Note:

For (R/X)DATDLY = 0, the first bit of data transmitted is asynchronous to CLKX.

8.3.6 Frame Synchronization Ignore

The McBSP can be configured to ignore transmit and receive frame synchronization pulses. The (R/X)FIG bit in the (R/X)CR can be set to 0 to recognize frame sync pulses or it can be set to 1 to ignore frame sync pulses. This way, you can use (R/X)FIG to either pack data, if operating at maximum frame frequency, or to ignore unexpected frame sync pulses.

8.3.6.1 Frame Sync Ignore and Unexpected Frame Sync Pulses

RFIG and XFIG are used to ignore unexpected frame sync pulses. Any frame sync pulse that occurs one or more bit clocks earlier than the programmed data delay from the end of the previous frame specified by ((R/X)DATDLY) is considered unexpected. Setting the frame ignore bits to 1 causes the serial port to ignore these unexpected frame sync signals.

In reception, if not ignored (RFIG = 0), unexpected FSR pulse discards the contents of RSR in favor of the incoming data. Therefore, if RFIG = 0, an unexpected frame synchronization pulse aborts the current data transfer, sets RSYNCERR in the SPCR to 1, and begins the reception of a new data element. When RFIG = 1, the unexpected frame sync pulses are ignored.

In transmission, if not ignored (XFIG = 0), an unexpected FSX pulse aborts the ongoing transmission, sets the XSYNCERR bit in the SPCR to 1, and re-initiates transmission of the current element that was aborted. When XFIG = 1, unexpected frame sync signals are ignored.

Figure 8–19 shows that element B is interrupted by an unexpected frame sync pulse when (R/X)FIG = 0. The reception of B is aborted (B is lost), and a new data element (C) is received after the appropriate data delay. This condition causes a receive synchronization error and thus sets the RSYNCERR bit. However, for transmission, the transmission of B is aborted, and the same data (B) is re-transmitted after the appropriate data delay. This condition is a transmit synchronization error and thus sets the XSYNCERR bit. Synchronization errors are discussed further in sections 8.3.7.2 and 8.3.7.5.

Figure 8–19. Unexpected Frame Synchronization with (R/X)FIG = 0

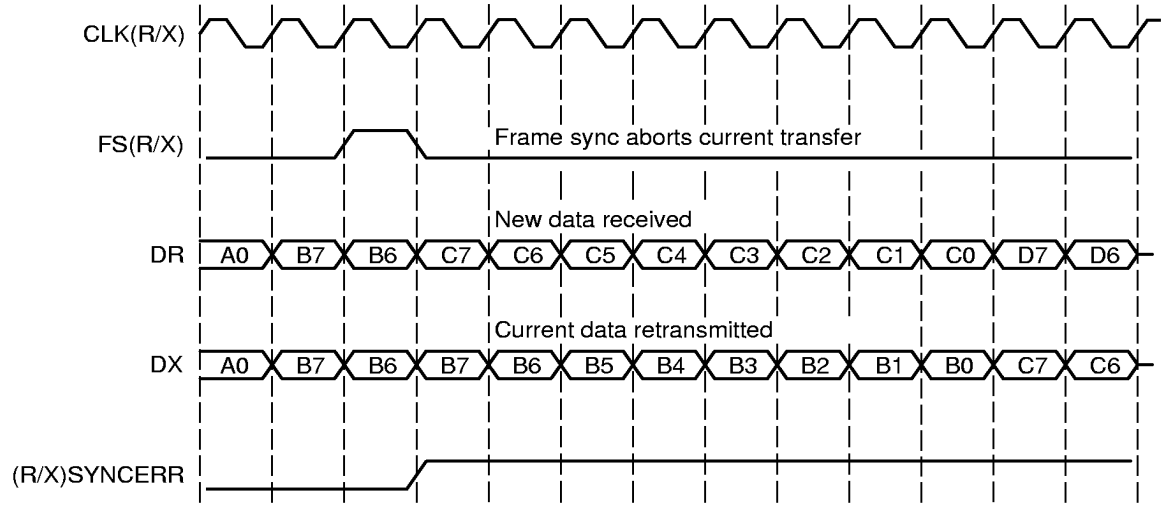
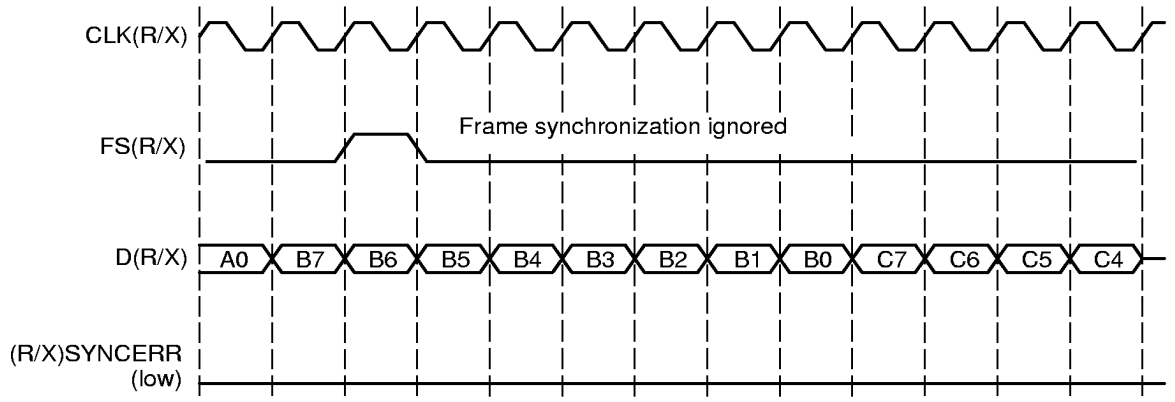


Figure 8–20 shows McBSP operation when unexpected frame synchronization signals are ignored by setting (R/X)FIG = 1. Here, the transfer of element B is not affected by an unexpected frame synchronization.

Figure 8–20. Unexpected Frame Synchronization With (R/X)FIG = 1



8.3.6.2 Data Packing using Frame Sync Ignore Bits

Section 8.3.4.5 describes one method of changing the element length and frame length to simulate 32-bit serial element transfers, thus requiring much less bus bandwidth than four 8-bit transfers require. This example works when there are multiple elements per frame. Now consider the case of the McBSP operating at maximum packet frequency as shown in Figure 8–21. Here, each frame has only a single 8-bit element. This stream takes one read transfer and one write transfer for each 8-bit element. Figure 8–22 shows the McBSP configured to treat this stream as a continuous stream of 32-bit elements. In this example, (R/X)FIG is set to 1 to ignore unexpected subsequent frames. Only one read transfer and one write transfer is needed every 32-bits. This configuration effectively reduces the required bus bandwidth to one-fourth of the bandwidth need to transfer four 8-bit blocks.

Figure 8–21. Maximum Frame Frequency Operation with 8-Bit Data

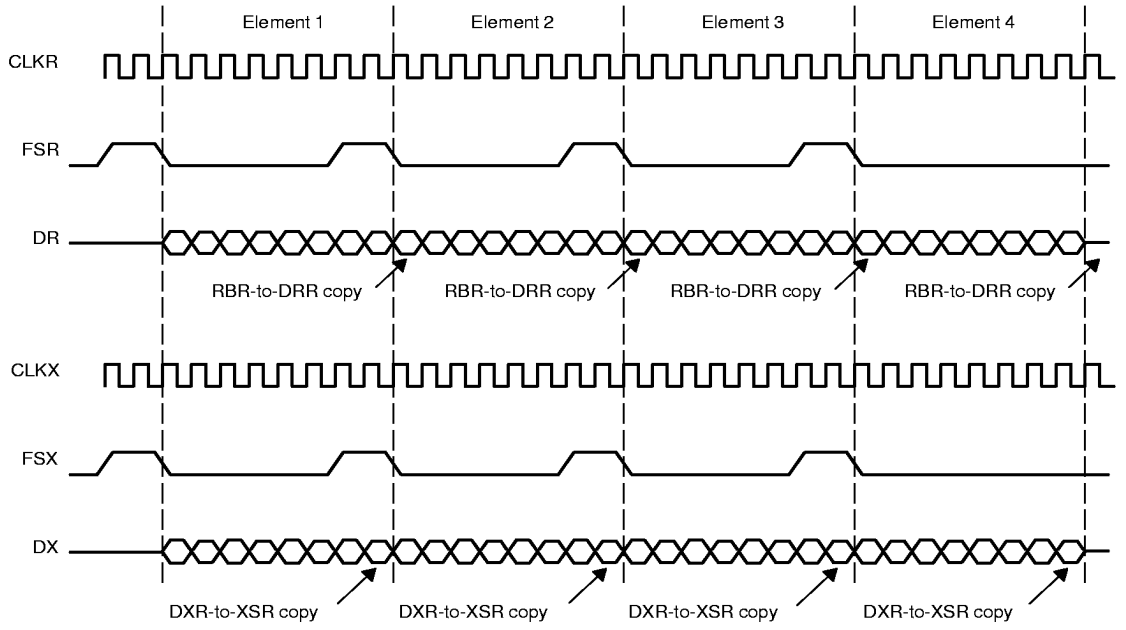
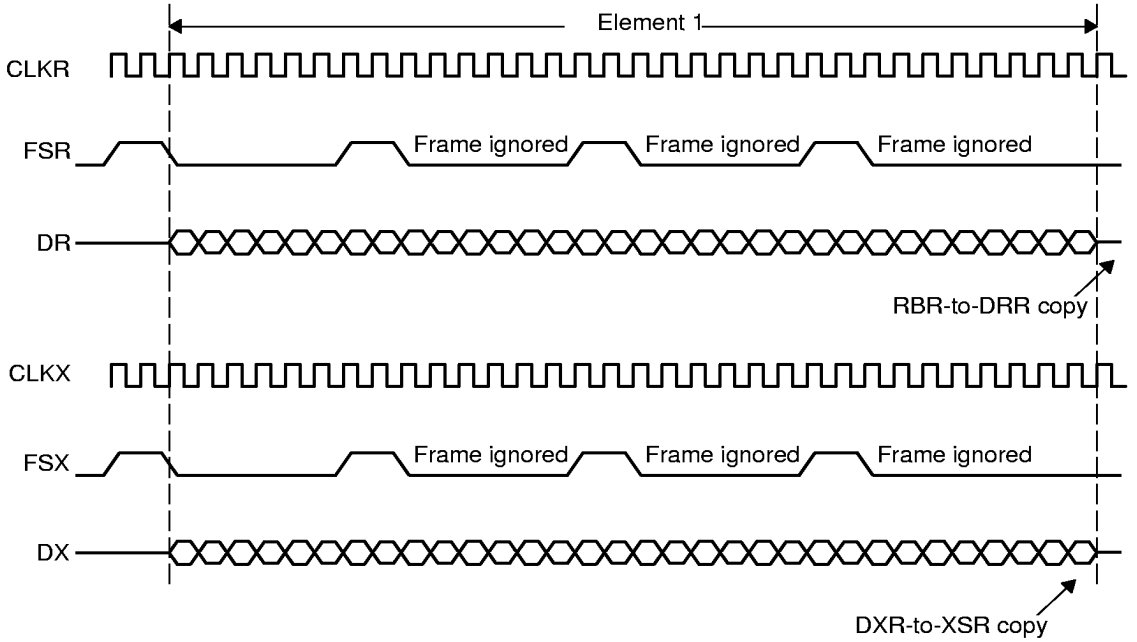


Figure 8–22. Data Packing at Maximum Frame Frequency With (R/X)FIG = 1



8.3.7 Serial Port Exception Conditions

There are five serial port events that can constitute a system error:

- Receive overrun (RFULL = 1)
- Unexpected receive frame synchronization (RSYNCERR = 1)
- Transmit data overwrite
- Transmit empty ($\overline{\text{XEMPTY}}$ = 0)
- Unexpected transmit frame synchronization (XSYNCERR = 1)

8.3.7.1 Reception With Overrun: RFULL

RFULL = 1 in the SPCR indicates that the receiver has experienced overrun and is in an error condition. RFULL is set when the following conditions are met:

- DRR has not been read since the last RBR-to-DRR transfer.
- RBR is full and an RBR-to-DRR copy has not occurred.
- RSR is full and an RSR-to-RBR transfer has not occurred.

The data arriving on DR is continuously shifted into RSR. Once a complete element is shifted into RSR, an RSR-to-RBR transfer can occur only if an RBR-to-DRR copy is complete. Therefore, if DRR has not been read by the CPU or the DMA controller since the last RBR-to-DRR transfer (RRDY = 1), an RBR-to-DRR copy does not take place until RRDY = 0. This prevents an RSR-to-RBR copy. New data arriving on DR pin is shifted into RSR and the previous contents of RSR is lost. After the receiver starts running from reset, a minimum of three elements must be received before RFULL can be set, because there was no last RBR-to-DRR transfer before the first element.

This data loss can be avoided if DRR is read no later than two and a half CLKR cycles before the end of the third element (data C) in RSR, as shown in Figure 8–23.

Either of the following events clears the RFULL bit to 0 and allows subsequent transfers to be read properly:

- Reading DRR
- Resetting the receiver ($\overline{\text{RRST}}$ = 0) or the device.

Another frame synchronization is required to restart the receiver.

Figure 8–23 shows the receive overrun condition. Because element A is not read before the reception of element B is complete, B is not transferred to DRR

yet. Another element, C, arrives and fills RSR. DRR is finally read, but not earlier than two and one half cycles before the end of element C. New data D overwrites the previous element C in RSR. If RFULL is still set after the DRR is read, the next element can overwrite D, if DRR is not read in time.

Figure 8–23. Serial Port Receive Overrun

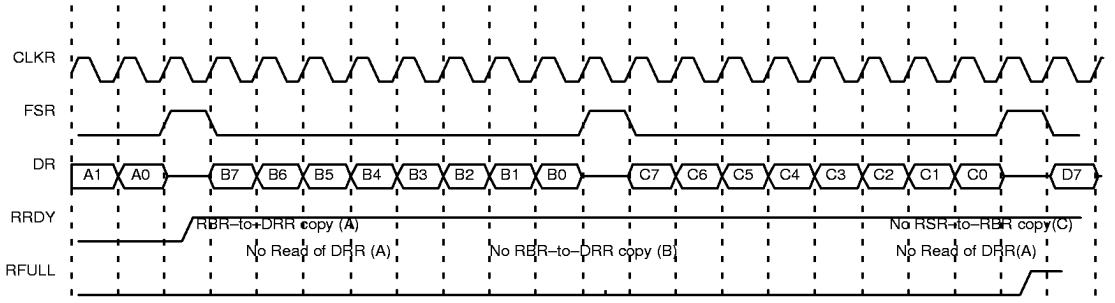
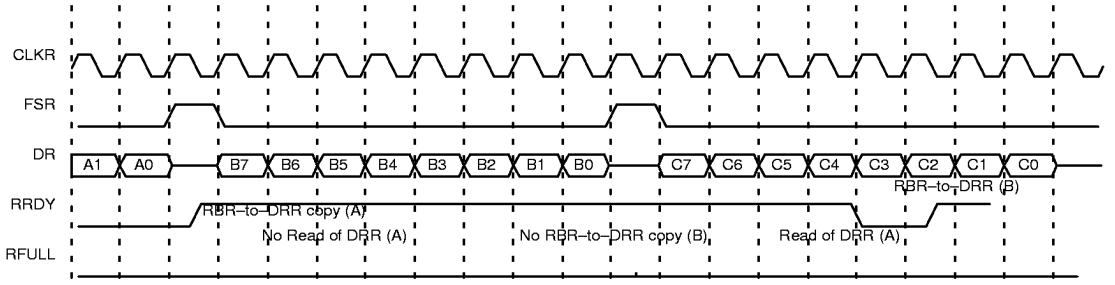


Figure 8–24 shows the case in which RFULL is set but the overrun condition is averted by reading the contents of DRR at least two and a half cycles before the next element, C, is completely shifted into RSR. This ensures that a RBR-to-DRR copy of data B occurs before the next element is transferred from RSR to RBR.

Figure 8–24. Serial Port Receive Overrun Avoided



8.3.7.2 Unexpected Receive Frame Synchronization: RSYNCERR

Figure 8–25 shows the decision tree that the receiver uses to handle all incoming frame synchronization pulses. The diagram assumes that the receiver has been activated ($\overline{\text{RRST}}$ = 1). Unexpected frame sync pulses can originate from an external source or from the internal sample rate generator. An unexpected frame sync pulse is defined as a sync pulse which occurs RDATDLY bit clocks earlier than the last transmitted bit of the previous frame. Any one of four cases can occur:

- Case 1 : Unexpected FSR pulses with RFIG = 1. This case is discussed in section 8.3.6.1 and shown in Figure 8–20. Here, receive frame sync pulses are ignored and the reception continues.
- Case 2: Normal serial port reception. There are three reasons for a receive *not* to be in progress:
  - This FSR is the first after $\overline{\text{RRST}}$ = 1.
  - This FSR is the first after DRR has been read clearing an RFULL condition.
  - The serial port is in the inter-packet intervals. The programmed data delay (RDATDLY) for reception may start during these inter-packet intervals for the first bit of the next element to be received. Thus, at maximum frame frequency, frame synchronization can still be received RDATDLY bit clocks before the first bit of the associated element.

For this case, reception continues normally, because these are not unexpected frame sync pulses.

- Case 3: Unexpected receive frame synchronization with RFIG = 0 (unexpected frame not ignored). This case was shown in Figure 8–19 for maximum packet frequency. Figure 8–26 shows this case during normal operation of the serial port with time intervals between packets. Unexpected frame sync pulses are detected when they occur the value in RDATDLY bit clocks before the last bit of the previous element is received on DR. In both cases, RSYNCERR in the SPCR is set. RSYNCERR can be cleared only by receiver reset or by writing a 0 to this bit in the SPCR. If RINTM = 11b in the SPCR, RSYNCERR drives the receive interrupt (RINT) to the CPU.

Note:

Note that the RSYNCERR bit in the SPCR is a read/write bit, so writing a 1 to it sets the error condition. Typically, writing a 0 is expected.

Figure 8–25. Decision Tree Response to Receive Frame Synchronization Pulse

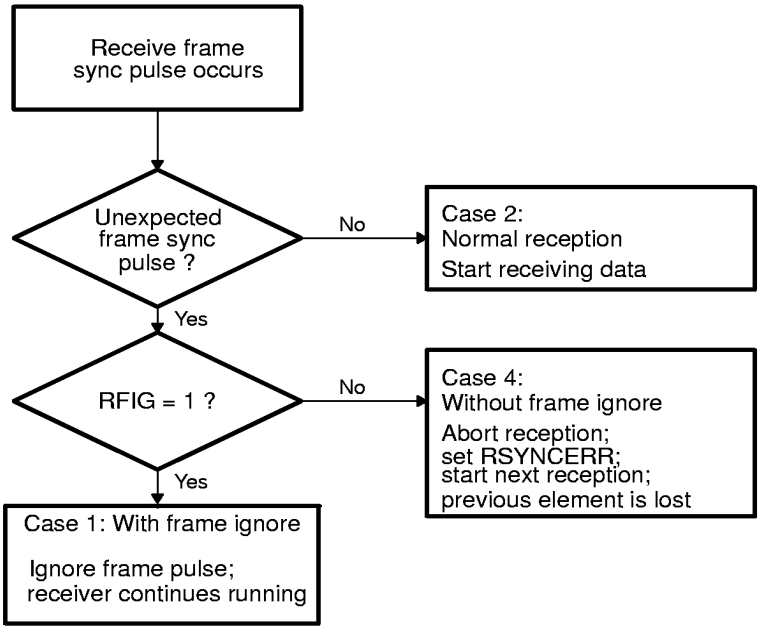
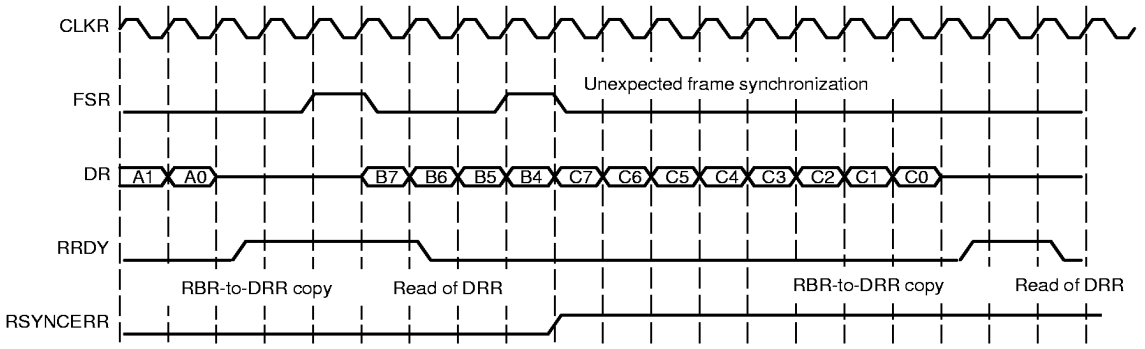


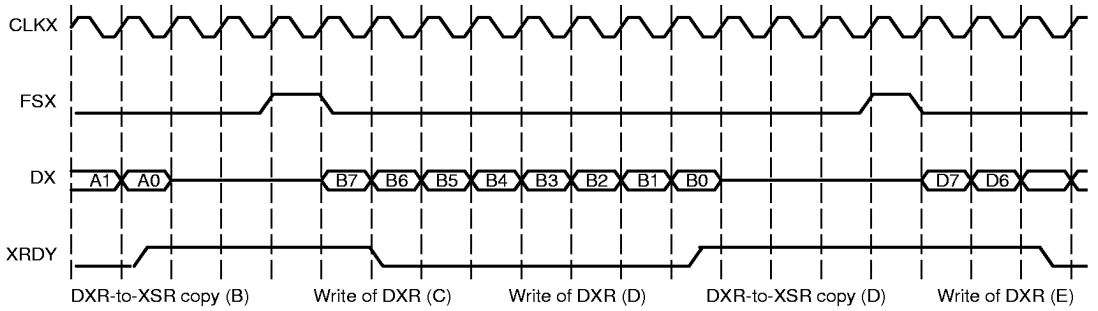
Figure 8–26. Unexpected Receive Synchronization Pulse



8.3.7.3 Transmit With Data Overwrite

Figure 8–27 depicts what happens if the data in DXR is overwritten before being transmitted. Suppose you load the DXR with data C. A subsequent write to the DXR overwrites C with D before C is copied to the XSR. Thus, C is never transmitted on DX. The CPU can avoid overwriting data by polling XRDY before writing to DXR or by waiting for a programmed XINT to be triggered by XRDY (XINTM = 00b). The DMA controller can avoid overwriting by synchronizing data writes with XEVT.

Figure 8–27. Transmit with Data Overwrite



8.3.7.4 Transmit Empty: $\overline{\text{XEMPTY}}$

$\overline{\text{XEMPTY}}$ indicates whether the transmitter has experienced under-flow. Either of the following conditions causes $\overline{\text{XEMPTY}}$ to become active ($\overline{\text{XEMPTY}}$ = 0):

- During transmission, DXR has not been loaded since the last DXR-to-XSR copy, and all bits of the data element in the XSR have been shifted out on DX.
- The transmitter is reset ($\overline{\text{XRST}}$ = 0 or device is reset) and then restarted.

During underflow condition, the transmitter continues to transmit the old data in DXR for every new frame sync signal that arrives on FSX until a new element is loaded into DXR by the CPU or the DMA controller. $\overline{\text{XEMPTY}}$ is deactivated ($\overline{\text{XEMPTY}}$ = 1) when this new element in DXR is transferred to XSR. In the case of internal frame sync generation, the transmitter regenerates a single FSX initiated by a DXR-to-XSR copy (FSXM = 1 in the PCR and FSGM = 0 in SRGR). Otherwise, the transmitter waits for the next frame synchronization.

When the transmitter is taken out of reset ($\overline{\text{XRST}}$ = 1), it is in a transmit ready (XRDY = 1) and transmit empty ($\overline{\text{XEMPTY}}$ = 0) condition. If DXR is loaded by the CPU or the DMA controller before FSX goes active, a valid DXR-to-XSR transfer occurs. This allows for the first element of the first frame to be valid even before the transmit frame sync pulse is generated or detected. Alternatively, if a transmit frame sync is detected before DXR is loaded, zeros will be output on DX.

Figure 8–28 depicts a transmit underflow condition. After B is transmitted, if you fail to reload the DXR before the subsequent frame synchronization, B will be retransmitted on DX. Figure 8–29 shows the case of writing to DXR just before a transmit underflow condition that would otherwise occur. After B is transmitted, C is written to DXR before the next transmit frame sync pulse occurs so that C is successfully transmitted on DX, averting a transmit empty condition.

Figure 8–28. Transmit Empty

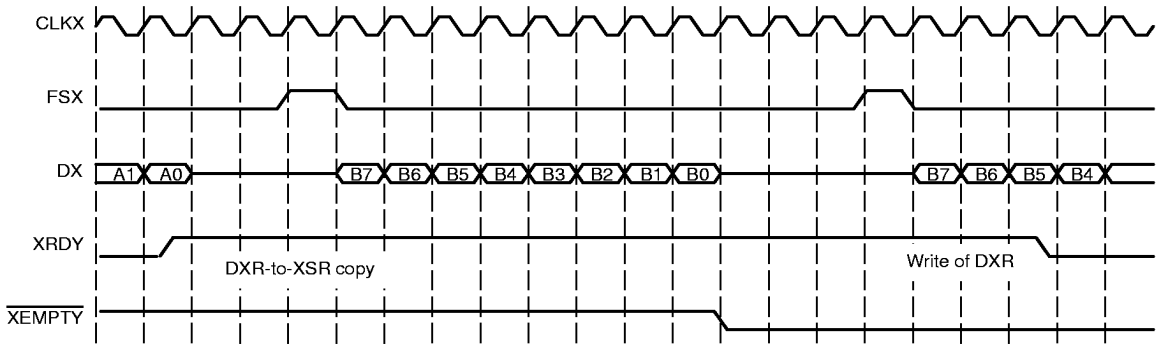
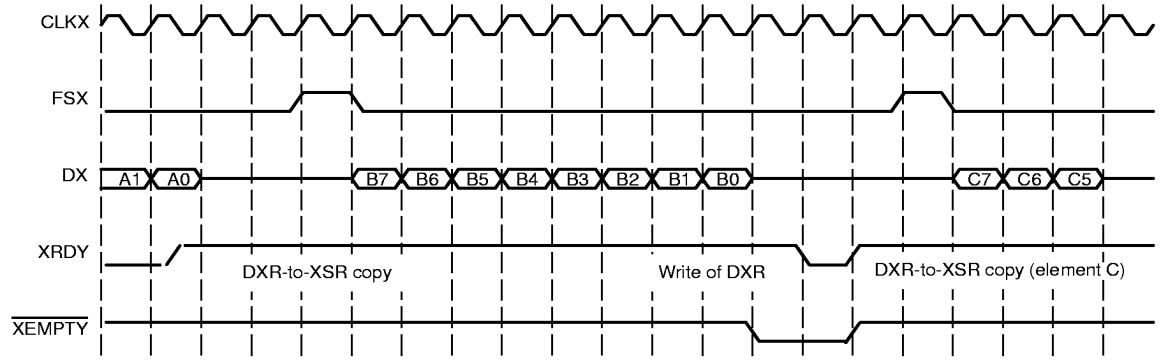


Figure 8–29. Transmit Empty Avoided

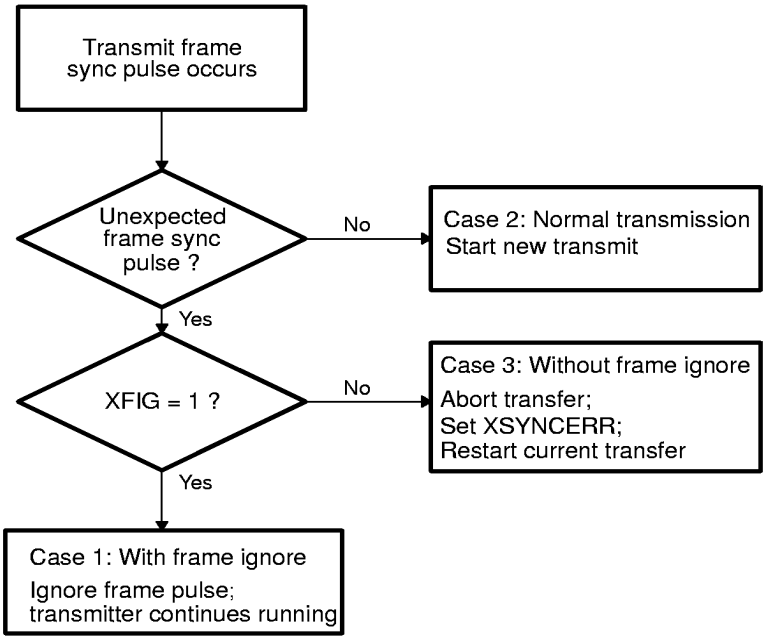


8.3.7.5 Unexpected Transmit Frame Synchronization: XSYNCERR

Figure 8–25 shows the decision tree that the transmitter uses to handle all incoming frame synchronization signals. The diagram assumes that the transmitter has been started ($\overline{XRST}$ = 1). An unexpected transmit frame sync pulse is defined as a sync pulse which occurs XDATDLY bit clocks earlier than the last transmitted bit of the previous frame. Any one of three cases can occur:

- Unexpected FSX pulses with XFIG = 1. This case is discussed in section 8.3.6.1 and shown in Figure 8–20.
- Normal serial port transmission. This situation is discussed in section 8.3.5.3. There are two possible reasons for a transmit *not* to be in progress:
  - This FSX pulse is the first after $\overline{XRST}$ = 1.
  - The serial port is in the inter-packet intervals. The programmed data delay (XDATDLY) may start during these inter-packet intervals before the first bit of the next element is transmitted. Thus, if operating at maximum packet frequency, frame synchronization can still be received XDATDLY bit clocks before the first bit of the associated element.

Figure 8–30. Response to Transmit Frame Synchronization

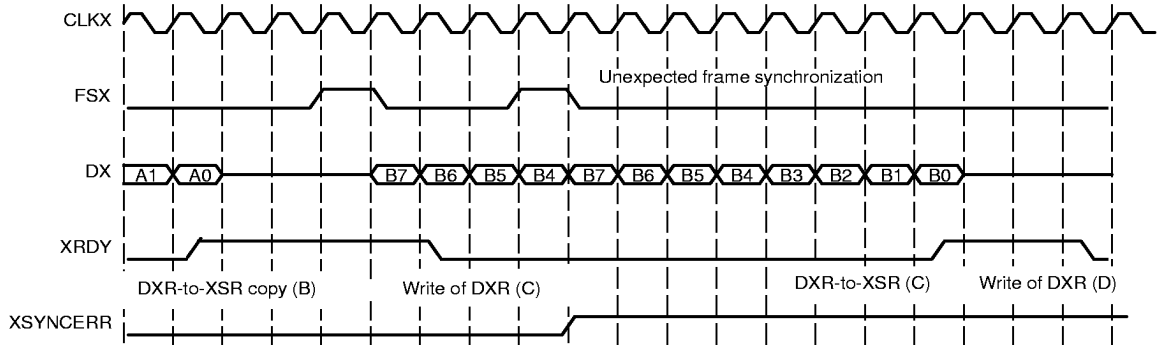


- Unexpected transmit frame synchronization with XFIG = 0. The case for frame synchronization with XFIG = 0 at maximum packet frequency is shown in Figure 8–19. Figure 8–31 shows the case for normal operation of the serial port with inter-packet intervals. In both cases, XSYNCERR in the SPCR is set. XSYNCERR can be cleared only by transmitter reset or by you writing a 0 to this bit in the SPCR. If XINTM = 11b in the SPCR, XSYNCERR drives the receive interrupt (XINT) to the CPU.

Note:

The XSYNCERR bit in the SPCR is a read/write bit, so writing a 1 to it sets the error condition. Typically, writing a 0 is expected.

Figure 8–31. Unexpected Transmit Frame Synchronization Pulse



8.3.8 Receive Data Justification and Sign-Extension: RJUST

RJUST in the SPCR selects whether data in the RBR is right or left justified (with respect to the MSB) in the DRR. If right-justification is selected, RJUST further selects whether the data is sign-extended or zero-filled. Table 8–11 shows the effect various values of RJUST have on an example 12-bit receive data value 0xABC.

Table 8–11. Use of RJUST Field With 12-Bit Example Data 0xABC

RJUST value	Justification	Extension	Value in DRR
00	Right	Zero-fill MSBs	0x00000ABC
01	Right	Sign-extend MSBs	0xFFFFFABC
10	Left	Zero-fill LSBs	0xABC00000
11	Reserved	Reserved	Reserved

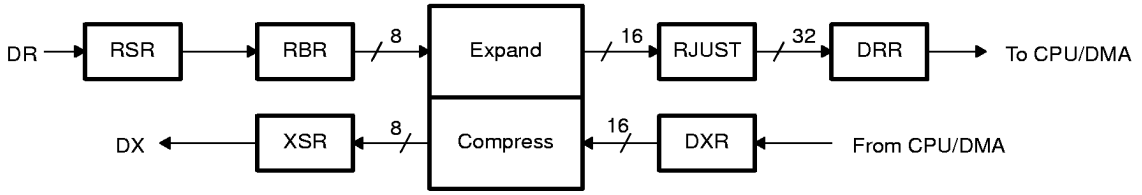
8.4 μ-LAW/A-LAW Companding Hardware Operation

Companding (*com*press and ex*pand*) hardware allows compression and expansion of data in either μ-law or A-law format. The specification for μ-law and A-law log PCM is part of the CCITT G.711 recommendation. The companding standard employed in the United States and Japan is μ-law and allows 14 bits of dynamic range. The European companding standard is A-law, and allows 13-bits of dynamic range. Any values outside these ranges will be set to the most positive or most negative value. Thus, for companding to work best here, the data transferred to and from the McBSP via the CPU or the DMA controller must be at least 16 bits wide.

The μ-law and A-law formats encode data into 8-bit code elements. Companded data is always 8 bits wide, so the appropriate (R/X)WDLEN(1/2) must be set to 0, indicating an 8-bit serial data stream. If companding is enabled, and either phase of the frame does not have an 8-bit element length, companding continues as if the element length is 8 bits.

When companding is used, transmit data is encoded according to the specified companding law, and receive data is decoded to 2s-complement format. Companding is enabled and the desired format selected by appropriately setting (R/X)COMPAND in the (R/X)CR as shown in Table 8–6. Compression occurs during the process of copying data from DXR to XSR and from RBR to DRR as shown in Figure 8–32.

Figure 8–32. Companding Flow



For transmit data to be compressed, it should be 16-bit, left-justified data, such as LAW16. The value can be either 13 or 14 bits depending on the companding law as shown in Figure 8–33. This 16-bit data is aligned in DXR as shown in Figure 8–34.

Figure 8–33. Companding Data Formats

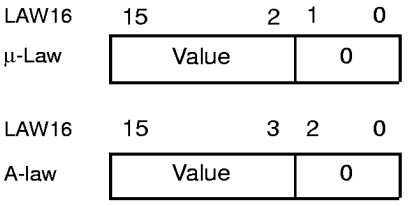
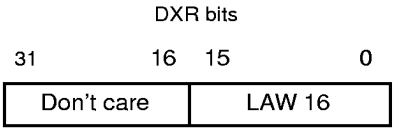


Figure 8–34. Transmit Data Companding Format in DXR



For reception, the 8-bit compressed data in RBR is expanded to a left-justified 16-bit data, LAW16. This can be further justified to a 32-bit data by programming the RJUST field in the SPCR as shown in Table 8–12.

Table 8–12. Justification of Expanded Data in DRR

	DRR bits			
RJUST	31	16	15	0
00	0		LAW16	
01	sign		LAW16	
10	LAW16		0	
11	reserved			

8.4.1 Companding Internal Data

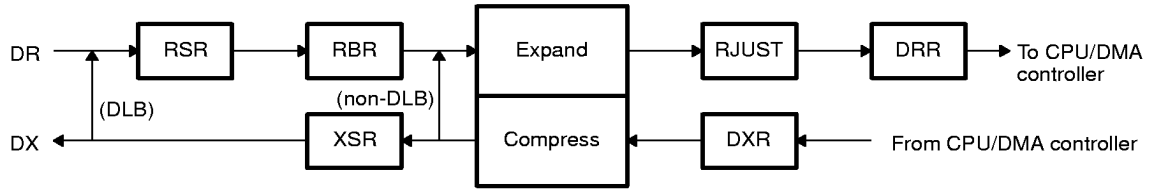
If the McBSP is otherwise unused, the companding hardware can compand internal data. This hardware can be used to:

- Convert linear data to the appropriate μ-law or A-law format.
- Convert μ-law or A-law data to the linear format.
- Observe the quantization effects in companding by transmitting linear data, and compressing and re-expanding this data. This is useful only if both XCOMPAND and RCOMPAND enable the same companding format.

Figure 8–35 shows two methods by which the McBSP can compand internal data. Data paths for these two methods are indicated by (DLB) and (non-DLB) arrows.

- **Non-DLB:** When both the transmit and receive sections of the serial port are reset, the DRR and DXR are internally connected through the companding logic. Values from the DXR are compressed as selected by XCOMPAND and then expanded as selected by RCOMPAND. RRDY and XRDY bits are not set. However, data is available in DRR four CPU clocks after being written to DXR. The advantage of this method is its speed. The disadvantage is that there is no synchronization available to the CPU and the DMA controller to control the flow of data.
- **DLB:** The McBSP is enabled in digital loop back (DLB) mode with companding appropriately enabled by RCOMPAND and XCOMPAND. Receive and transmit interrupts (RINT when RINTM = 0 and XINT when XINTM = 0) or synchronization events (REVT and XEVT) allow synchronization of the CPU or the DMA controller to these conversions, respectively. Here, the time for this companding depends on the serial bit rate selected.

Figure 8–35. Companding of Internal Data



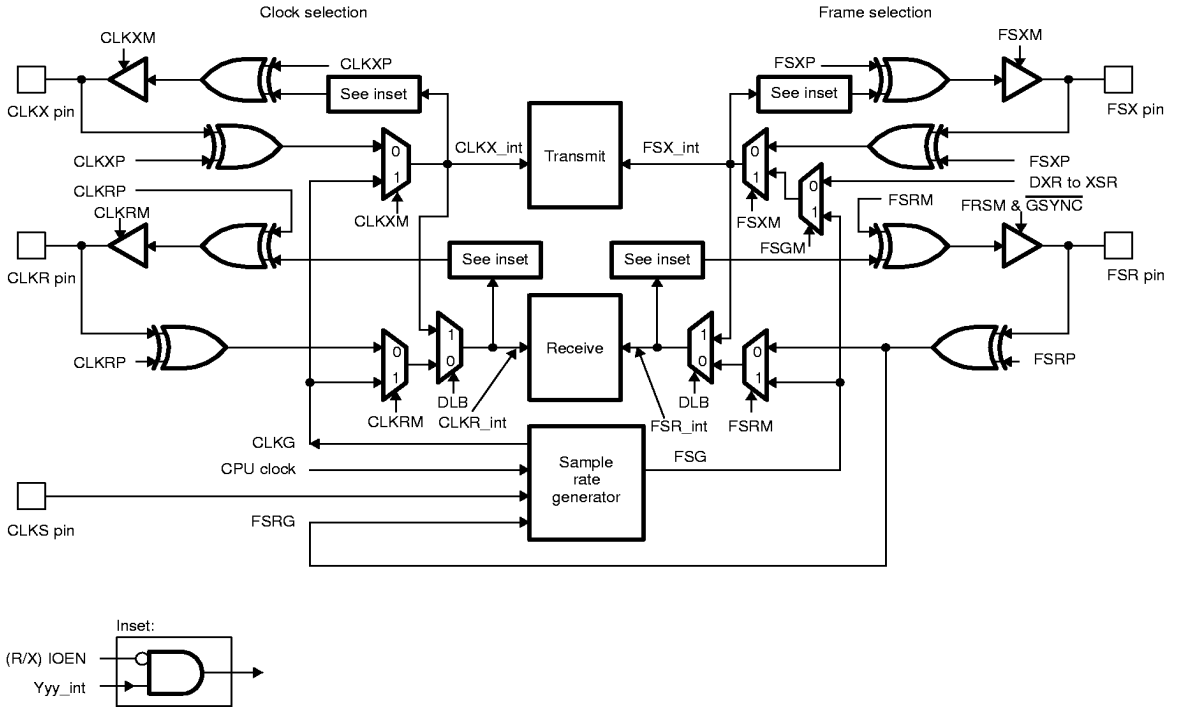
8.4.1.1 Bit Ordering

Normally, all transfers on the McBSP are sent and received with the MSB first. However, certain 8-bit data protocols (that do not use companded data) require the LSB to be transferred first. By setting the (R/X)COMPAND = 01b in the (R/X)CR, the bit ordering of 8-bit elements is reversed (LSB first) before being sent to the serial port. Similar to companding, this feature is enabled only if the appropriate (R/X)WDLEN(1/2) is set to 0, indicating 8-bit elements are to be transferred serially.

8.5 Programmable Clock and Framing

The McBSP has several means of selecting clocking and framing for both the receiver and transmitter. Clocking and framing can be sent to both portions by the sample rate generator. Each portion can select external clocking and/or framing independently. Figure 8–36 is a block diagram of the clock and frame selection circuitry.

Figure 8–36. Clock and Frame Generation



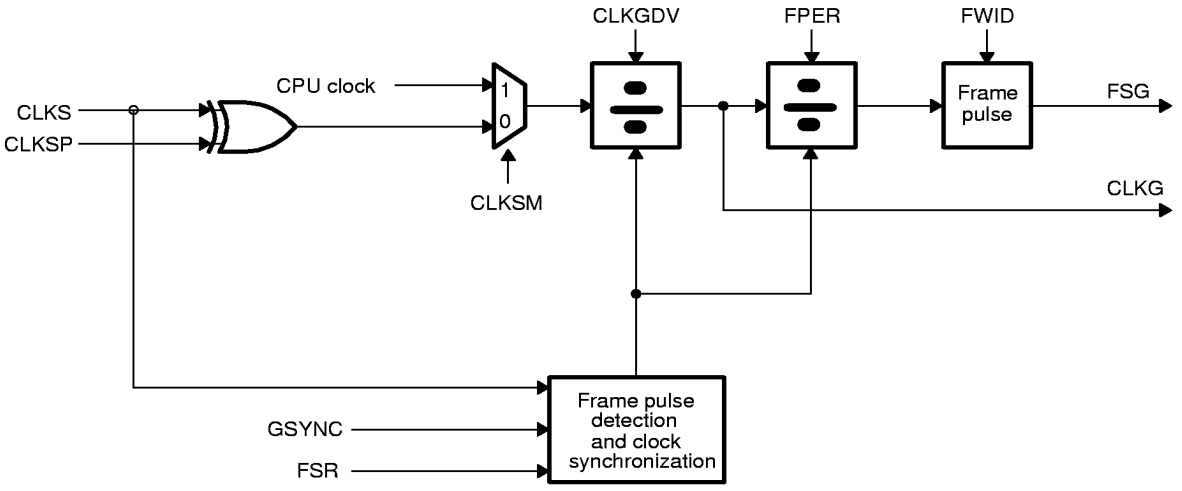
8.5.1 Sample Rate Generator Clocking and Framing

The sample rate generator is composed of a 3 stage clock divider that provides a programmable data clock (CLKG) and framing signal (FSG) as shown in Figure 8–37. CLKG and FSG are McBSP internal signals that can be programmed to drive receive and/or transmit clocking, CLK(R/X), and framing, FS(R/X). The sample rate generator can be programmed to be driven by an internal clock source or an internal clock derived from an external clock source. The three stages of the sample rate generator circuit compute:

- Clock divide-down (CLKGDV): The number of input clocks per data bit clock
- Frame period (FPER): The frame period in data bit clocks.
- Frame width (FWID): The width of an active frame pulse in data bit clocks.

In addition, a frame pulse detection and clock synchronization module allows synchronization of the clock divide-down with an incoming frame pulse. The operation of the sample rate generator during device reset is described in section 8.3.1.

Figure 8–37. Sample Rate Generator



8.5.1.1 Sample Rate Generator Register (SRGR)

The sample rate generator register (SRGR) shown in Figure 8–38 and described in Table 8–13, controls the operation of various features of the sample rate generator. This section describes the fields within the SRGR.

Figure 8–38. Sample Rate Generator Register (SRGR)

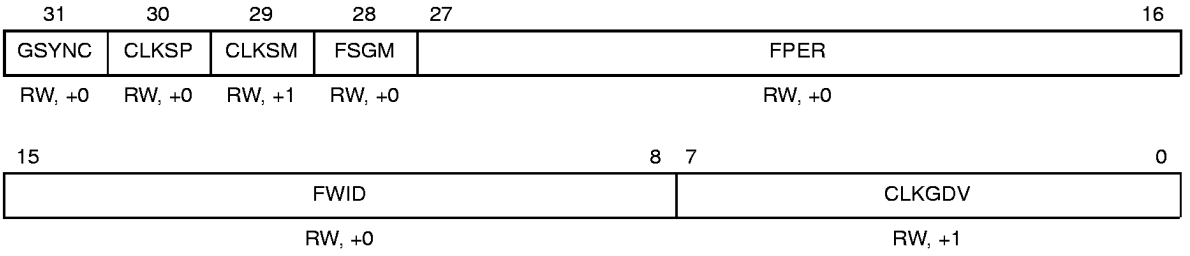


Table 8–13. Sample Rate Generator Register (SRGR) Field Summary

Name	Function	Section
GSYNC	Sample rate generator clock synchronization. Used only when the external clock (CLKS) drives the sample rate generator clock (CLKSM = 0). GSYNC = 0: the sample rate generator clock (CLKG) is free running. GSYNC = 1: (CLKG) is running, but is resynchronized and the frame sync signal (FSG) is generated only after the receive frame synchronization signal (FSR) is detected. Also, the frame period (FPER) is a don't care because the period is dictated by the external frame sync pulse.	8.5.2.4
CLKSP	CLKS polarity clock edge select. Used only when the external clock CLKS drives the sample rate generator clock (CLKSM = 0). CLKSP = 0: the rising edge of CLKS generates CLKG and FSG. CLKSP = 1: the falling edge of CLKS generates CLKG and FSG.	8.5.2.3
CLKSM	McBSP sample rate generator clock mode CLKSM = 0: Sample rate generator clock derived from CLKS. CLKSM = 1: (Default value) Sample rate generator clock derived from CPU clock	8.5.2.2
FSGM	Sample rate generator transmit frame synchronization mode. Used when FSXM = 1 in PCR. FSGM = 0: Transmit frame sync signal (FSX) is generated on every DXR-to-XSR copy. FSGM = 1: Transmit frame sync signal is driven by the sample rate generator frame sync signal, FSG.	8.5.3.3

Table 8–13. Sample Rate Generator Register (SRGR) Field Summary (Continued)

Name	Function	Section
FPER	Frame period. This field's value plus 1 determines when the next frame sync signal should become active. Valid value: 0 to 4095	8.5.3.1
FWID	Frame width. This field's value plus 1 is the width of the frame sync pulse, FSG, during its active period. Valid value: 0 to 255	8.5.3.1
CLKGDV	Sample rate generator clock divider. This value is used as the divide-down number to generate the required sample rate generator clock frequency. The default value is 1. Valid value: 0 to 127	8.5.2.2

8.5.1.2 Sample Rate Generator Reset Procedure

The sample rate generator reset and initialization procedure is as follows:

1) During device reset, $\overline{\text{GRST}}$ = 0. Otherwise, during normal operation, reset the sample rate generator with $\overline{\text{GRST}}$ = 0 in SPCR, provided CLKG and/or FSG ($\overline{\text{FRST}}$ = 1) is not used by any portion of the McBSP. If $\overline{\text{GRST}}$ is low due to device reset, CLKG is driven by a divide-by-2 CPU clock and FSG is driven inactive. CLKG and FSG are inactive when $\overline{\text{GRST}}$ = 0. If necessary, set $\overline{\text{(R/X)RST}}$ = 0.

2) Program SRGR as required. If necessary, other control registers can be written with desired values provided the respective portion (R/X) is in reset.

3) Wait two CLKSRG clocks. This is to ensure proper synchronization internally.

4) Set $\overline{\text{GRST}}$ = 1 to enable the sample rate generator.

5) Wait two CLKG bit clocks.

6) Pull the receiver and/or transmitter out of reset ($\overline{\text{(R/X)RST}}$ = 1) if required.

7) On the next rising edge of CLKSRG, CLKG transitions to 1 and starts clocking with a frequency equal to (CPU clock / (1+CLKGDV)) if CLKSM =1, or CLKS clock/(1 + CLKGDV) if CLKSM = 0.

8) After the required data acquisition setup is done such as writing to DXR, $\overline{\text{FRST}}$ can be written with 1 in the SPCR, if an internally generated frame pulse is required. FSG is generated on an active edge after eight CLKG clocks have elapsed.

8.5.2 Data Clock Generation

When the receive/transmit clock mode is set to 1 (CLK(R/X)M = 1), the data clocks (CLK(R/X)) are driven by the internal sample rate generator output clock, CLKG. You can select from a variety of data bit clocks for the receiver and transmitter independently. These options include:

- The input clock to the sample rate generator can be either the CPU clock or a dedicated external clock source (CLKS).
- The input clock (CPU clock or external clock CLKS) source to the sample rate generator can be divided down by a programmable value (CLKGDV) to drive CLKG.

Regardless of the source to the sample rate generator, the rising edge of CLKSRG (see Figure 8–37) generates CLKG and FSG (also, see section 8.5.2.3).

8.5.2.1 Input Clock Source Mode: CLKSM

The CLKSM bit in the SRGR selects either the CPU clock (CLKSM = 1) or the external clock input (CLKSM = 0), CLKS, as the source for the sample rate generator input clock. Any divide periods are divide-downs calculated by the sample rate generator and are timed by this input clock selection. When CLKSM = 1, the minimum value of CLKGDV should be 1.

8.5.2.2 Sample Rate Generator Data Bit Clock Rate: CLKGDV

The first divider stage generates the serial data bit clock from the input clock. This divider stage uses a counter that is preloaded by CLKGDV and that contains the divide ratio value. The output of this stage is the data bit clock which is output on the sample rate generator output, CLKG, and serves as the input for the second and third divider stages.

CLKG has a frequency equal to 1/(CLKGDV+1) of the sample rate generator input clock. Thus, the sample rate generator input clock frequency is divided by a value between 1 and 256. When CLKGDV is odd or equal to 0, the CLKG duty cycle is 50%. When CLKGDV is an even value, 2p, the high state duration is p + 1 cycles and the low state duration is p cycles.

8.5.2.3 Bit Clock Polarity: CLKSP

External clock (CLKS) is selected to drive the sample rate generator clock divider by selecting CLKSM = 0. In this case, the CLKSP bit in the SRGR selects the edge of CLKS on which sample rate generator data bit clock (CLKG) and frame sync signal (FSG) are generated. Since the rising edge of CLKSRG generates CLKG and FSG, the rising edge of CLKS when (CLKSP = 0) or the falling edge of CLKS when CLKSP = 1 causes the transition on CLKG and FSG.

8.5.2.4 Bit Clock and Frame Synchronization

When CLKS is selected to drive the sample rate generator (CLKSM = 0), GSYNC can be used to configure the timing of CLKG relative to CLKS. GSYNC = 1 ensures that the McBSP and the external device it is communicating to are dividing down the CLKS with the same phase relationship. If GSYNC = 0, this feature is disabled and CLKG runs freely and is not resynchronized. If GSYNC = 1, an inactive-to-active transition on FSR triggers a resynchronization of CLKG and the generation of FSG. CLKG always begins at a high state after synchronization. Also, FSR is always detected at the same edge of CLKS that generates CLKG, no matter how long the FSR pulse is. Although an external FSR is provided, FSG can still drive internal receive frame synchronization when GSYNC = 1. When GSYNC = 1, FPER is a don't care, because the frame period is determined by the arrival of the external frame sync pulse.

Figure 8–39 and Figure 8–40 shows this operation with various polarities of CLKS and FSR. These figures assume that FWID is 0, for a FSG = 1 CLKG wide.

Figure 8–39. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 1

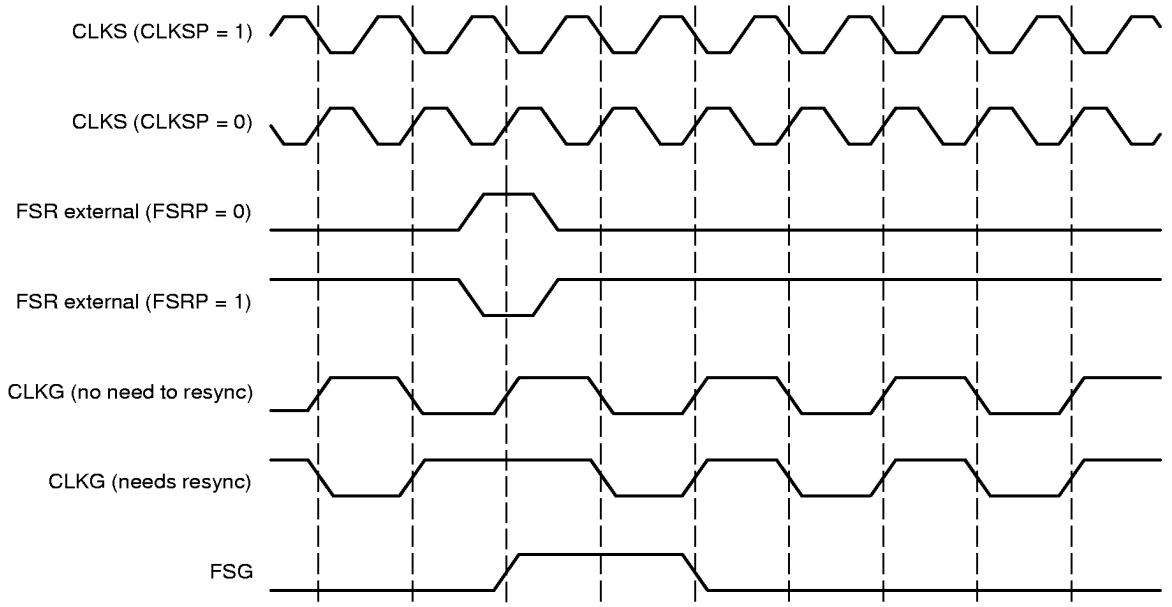
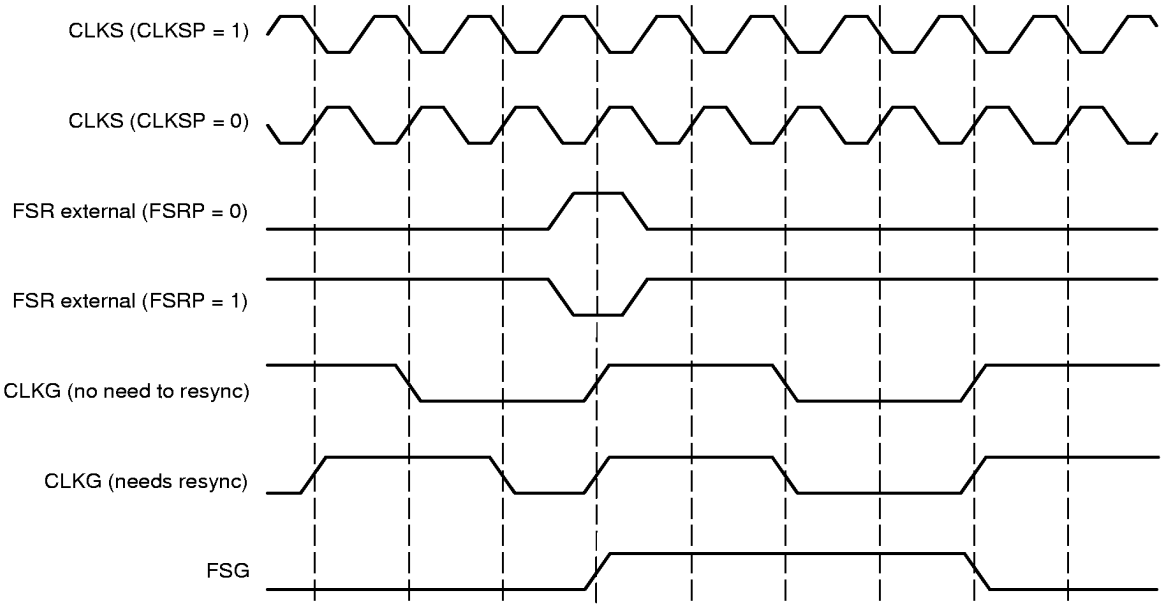


Figure 8–40. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 3



These figures show what happens to CLKG when it is initially in sync and GSYNC = 1, as well as when it is not in sync with the frame synchronization and GSYNC = 1.

When GSYNC = 1, the transmitter can operate synchronously with the receiver provided:

- FSX is programmed to be driven by the sample rate generator frame sync FSG (FSGM = 1 in the SRGR and FSXM = 1 in the PCR). If the input FSR has appropriate timing so that it can be sampled by the falling edge of CLKG, it can be used instead by setting FSXM = 0 in the PCR and connecting FSR to FSX externally.
- The sample rate generator clock should drive the transmit and receive bit clock (CLK(R/X)M = 1 in the SPCR). Therefore, the CLK(R/X) pin should not be driven by any other source.

8.5.2.5 Digital Loop Back Mode: DLB

Setting DLB = 1 in the SPCR enables digital loop back mode. In DLB mode, DR, FSR, and CLKR are internally connected through multiplexers to DX, FSX, CLKX respectively, as shown in Figure 8–36. DLB mode allows testing of serial port code with a single DSP device.

8.5.2.6 Receive Clock Selection: DLB, CLKRM

Table 8–14 shows how the digital loop back bit (DLB) and the CLKRM bit in the PCR select the receiver clock. In digital loop-back mode (DLB = 1), the transmitter clock drives the receiver. CLKRM determines whether the CLKR pin is an input or an output.

Table 8–14. Receive Clock Selection

DLB in SPCR	CLKRM in PCR	Source of Receive Clock	CLKR Function
0	0	CLKR acts as an input driven by external clock and inverted as determined by CLKRP before being used.	Input
0	1	Sample rate generator clock (CLKG) drives CLKR.	Output. CLKG inverted as determined by CLKRP before being driven out on CLKR.
1	0	CLKX_int drives the receive clock CLKR_int as selected and inverted as shown in Table 8–15.	High impedance
1	1	CLKX_int drives CLKR_int as selected and inverted as shown in Table 8–15.	Output. CLKR (same as CLKX) inverted as determined by CLKRP before being driven out.

8.5.2.7 Transmit Clock Selection: CLKXM

Table 8–15 shows how the CLKXM bit in the PCR selects the transmit clock, and whether the CLKX pin is an input or output.

Table 8–15. Transmit Clock Selection

CLKXM in PCR	Source of Transmit Clock	CLKX Function
0	External clock drives the CLKX input pin. CLKX is inverted as determined by CLKXP before being used.	Input
1	Sample rate generator clock, CLKG, drives the transmit clock	Output. CLKG inverted as determined by CLKXP before being driven out on CLKX.

8.5.3 Frame Sync Signal Generation

Similar to data bit clocking, data frame synchronization is also independently programmable for the receiver and the transmitter for all data delay values. The $\overline{\text{FRST}}$ bit in the SPCR when set to 1 activates the frame generation logic to generate frame sync signals provided FSGM = 1 in SRGR. Frame sync programming options are:

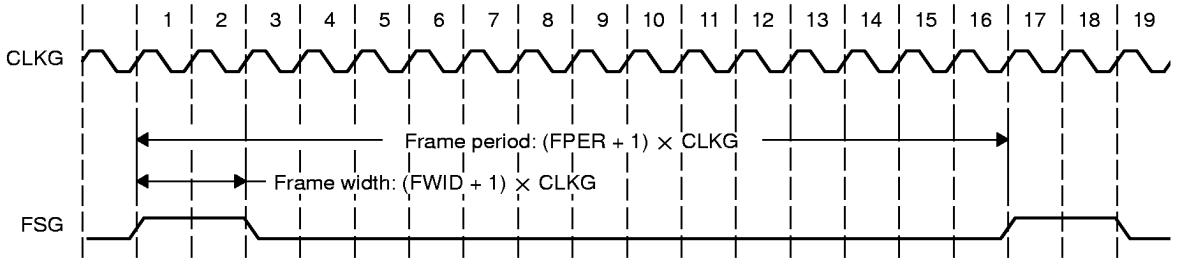
- A frame pulse with programmable period between sync pulses and programmable active width specified in the sample rate generator register (SRGR).
- The transmitter can trigger its own frame sync signal that is generated by a DXR-to-XSR copy. This causes a frame sync to occur on every DXR-to-XSR copy. The data delays can be programmed as required. However, maximum packet frequency cannot be achieved in this method for data delays of 1 and 2.
- Both the receiver and transmitter can independently select an external frame synchronization on the FSR and FSX pins, respectively.

8.5.3.1 Frame Period and Frame Width: FPER and FWID

The FPER block is a 12-bit down-counter that can count down the generated data clocks from 4095 to 0. FPER controls the period of active frame sync pulses. The FWID block in the sample rate generator is an 8-bit down counter. The FWID field controls the active width of the frame sync pulse.

When the sample rate generator comes out of reset, FSG is in an inactive-low state. After this, when $\overline{\text{FRST}}$ = 1 and FSGM = 1 frame sync signals are generated. The frame width value (FWID + 1) is counted down on every CLKG cycle until it reaches 0 when FSG goes low. Thus, the value of FWID+1 determines a active frame pulse width ranging from 1 to 256 data bit clocks. At the same time, the frame period value (FPER + 1) is also counting down, and when this value reaches 0, FSG goes high again, indicating a new frame is beginning. Thus, the value of FPER + 1 determines a frame length from 1 to 4096 data bits. When GSYNC = 1, the value of FPER does not matter. Figure 8–41 shows a frame of 16 CLKG periods (FPER = 15 or 00001111b).

Figure 8–41. Programmable Frame Period and Width



8.5.3.2 Receive Frame Sync Selection: DLB, FSRM, GSYNC

Table 8–16 shows how you may select various sources to provide the receive frame synchronization signal. Note that in digital loop back mode (DLB = 1) the transmit frame sync signal is used as the receive frame sync signal and that DR is connected to DX internally.

Table 8–16. Receive Frame Synchronization Selection

DLB in SPCR	FSR in PCR	GSYNC in SRGR	Source of Receive Frame Synchronization	FSR Pin Function
0	0	X	External frame sync signal drives the FSR input pin, whose signal is then inverted as determined by FSRP before being used as FSR_int.	Input
0	1	0	Sample rate generator frame sync signal (FSG) drives FSR_int, $\overline{\text{FRST}}$ = 1	Output. FSG inverted as determined by FSRP before being driven out on FSR pin.
0	1	1	Sample rate generator frame sync signal (FSG) drives FSR_int, $\overline{\text{FRST}}$ = 1	Input. The external frame sync input on FSR is used to synchronize CLKG and generate FSG.
1	0	0	FSX_int drives FSR_int. FSX is selected as shown in Table 8–17.	High impedance
1	X	1	FSX_int drives FSR_int and is selected as shown in Table 8–17.	Input. External FSR is not used for frame synchronization but is used to synchronize CLKG and generate FSG since GSYNC = 1.
1	1	0	FSX_int drives FSR_int and is selected as shown in Table 8–17.	Output. Receive (same as transmit) frame synchronization inverted as determined by FSRP before being driven out.

8.5.3.3 Transmit Frame Sync Signal Selection: FSXM, FSGM

Table 8–17 shows how you can select the source of transmit frame synchronization pulses. The three choices are:

- External frame sync input
- The sample rate generator frame sync signal, FSG
- A signal that indicates a DXR-to-XSR copy has been made

Table 8–17. Transmit Frame Synchronization Selection

FSXM in PCR	FSGM in SRGR	Source of Transmit Frame Synchronization	FSX Pin Function
0	X	External frame sync input on FSX pin. This is inverted by FSXP before being used as FSX_int.	Input
1	1	Sample rate generator frame sync signal (FSG) drives FSX_int. $\overline{\text{FRST}}$ = 1	Output. FSG is inverted by FSXP before being driven out on FSX.
1	0	A DXR-to-XSR copy activates transmit frame sync signal.	Output. 1-bit-clock-wide signal inverted as determined by FSXP before being driven out on FSX.

8.5.3.4 Frame Detection for Initialization

To facilitate detection of frame synchronization, the receive and transmit CPU interrupts (RINT and XINT) can be programmed to detect frame synchronization by setting RINTM = XINTM = 10b in the SPCR. Unlike other types of serial port interrupts, this one can operate while the associated portion of the serial port is in reset (such as activating RINT when the receiver is in reset). In that case, the FS(R/X)M and FS(R/X)P still select the appropriate source and polarity of frame synchronization. Thus, even when the serial port is in reset, these signals are synchronized to the CPU clock and then sent to the CPU in the form of RINT and XINT at the point at which they feed the receive and transmit portions of the serial port. A new frame synchronization pulse can be detected, after which the CPU can safely take the serial port out of reset.

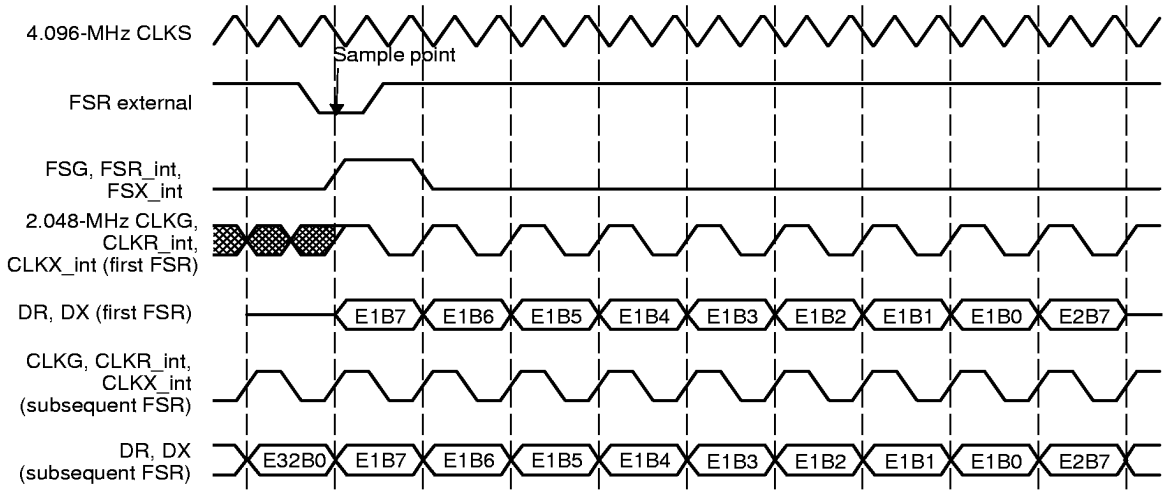
8.5.4 Clocking Examples

8.5.4.1 Double-Rate ST-BUS Clock

Figure 8–42 shows the McBSP timing to be compatible with the Mitel ST-Bus. The operation is running at maximum frame frequency.

- CLK(R/X)M = 1: CLK(R/X)_int generated internally by sample rate generator
- GSYNC = 1: CLKG is synchronized with the external frame sync signal input on FSR. CLKG is not synchronized (runs freely) until the frame sync signal is active. Also, FSR is regenerated internally to form a minimum pulse width.
- CLKSM = 0: external clock (CLKS) drives the sample rate generator
- CLKSP = 1: falling edge of CLKS generates CLKG and thus CLK(R/X)_int
- CLKGDV = 1: receive clock (shown as CLKR) is half of CLKS frequency
- FS(R/X)P = 1: active-low frame sync pulse
- (R/X)FRLEN1 = 11111b: 32 elements per frame
- (R/X)WDLEN1 = 0: 8-bit element
- (R/X)PHASE = 0: single phase frame and thus (R/X)FRLEN2 = (R/X)WDLEN2 = X
- (R/X)DATDLY = 0: no data delay

Figure 8–42. ST-BUS and MVIP Example

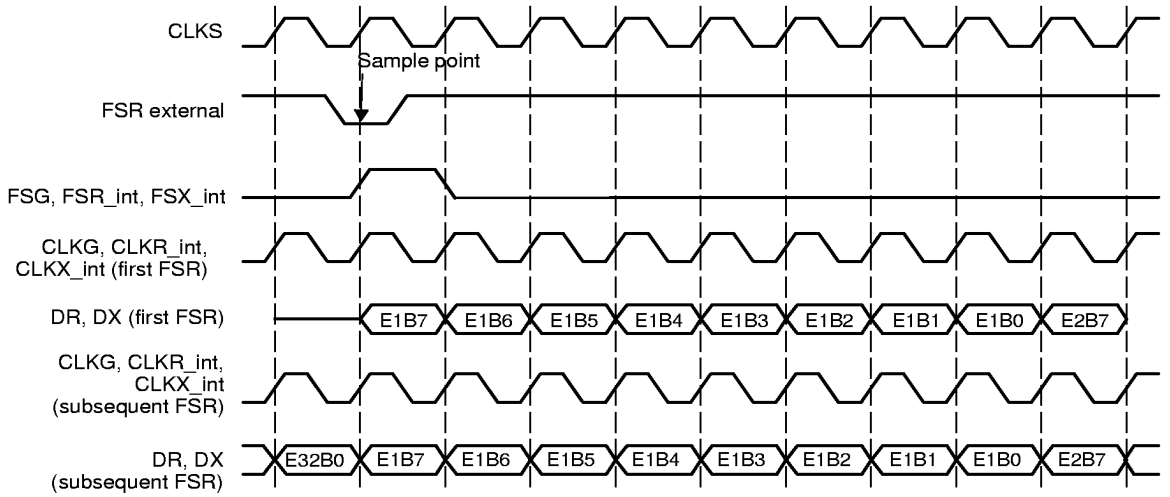


8.5.4.2 Single-Rate ST-BUS Clock

This example is the same as the ST-BUS example except for the following items:

- CLKGDV = 0: CLKS drives CLK(R/X)_int without any divide-down (single rate clock).
- CLKSP = 0: The rising edge of CLKS generates internal clocks CLKG and CLK(R/X)_int.

Figure 8–43. Single-Rate Clock Example



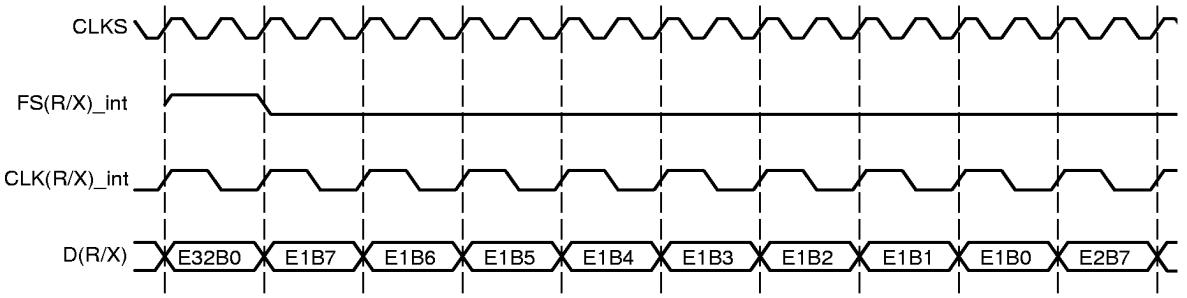
The rising edge of CLKS detects the external FSR. This external frame sync pulse resynchronizes the internal McBSP clocks and generates the frame sync for internal use. The internal frame sync is generated so that it is wide enough to be detected on the falling edge of the internal clocks.

8.5.4.3 Double-Rate Clock

This example is the same as the ST-BUS example except for the following:

- CLKSP = 0: The rising edge of CLKS generates CLKG and CLK(R/X).
- CLKGDV = 1: CLKG, CLKR_int, and CLKX_int frequencies are half of the CLKS frequency.
- GSYNC = 0: CLKS drives CLKG. CLKG runs freely and is not resynchronized by FSR.
- FS(R/X)M = 0: Frame synchronization is externally generated. The framing pulse is wide enough to be detected.
- FS(R/X)P = 0: active-high input frame sync signal
- (R/X)DATDLY = 1: specifies a data delay of 1-bit

Figure 8–44. Double Rate Clock Example



8.6 Multichannel Selection Operation

Multiple channels can be independently selected for the transmitter and receiver by configuring the McBSP with a single phase frame. Each frame represents a time-division multiplexed data stream. The number of elements per frame represented by (R/X)FRLEN1, denotes the number of channels available for selection. Thus, to save memory and bus bandwidth, multichannel selection allows independent enabling of particular elements for transmission and reception. Up to 32 elements in an up to 128 element bit stream can be enabled.

If a receive element is not enabled:

- RRDY is not set to 1 upon reception of the last bit of the element.
- RBR is not copied to DRR upon reception of the last bit of the element. Thus, RRDY is not set active. This feature also implies that no interrupts or synchronization events are generated for this element.

If a transmit element is not enabled:

- DX is in the high impedance state.
- A DXR-to-XSR transfer is not automatically triggered at the end of serial transmission of the related element.
- $\overline{\text{XEMPTY}}$ and XRDY are not affected by the end of transmission of the related serial element.

An enabled transmit element can have its data masked or transmitted. When data is masked, the DX pin is forced to the high-impedance state, even though the transmit channel is enabled.

8.6.1 Multichannel Operation Control Registers

The following control registers are used in multichannel operation:

- The multichannel control register (MCR)
- The transmit channel enable register (XCER)
- The receive channel enable register (RCER)

Figure 8–45. Multichannel Control Register

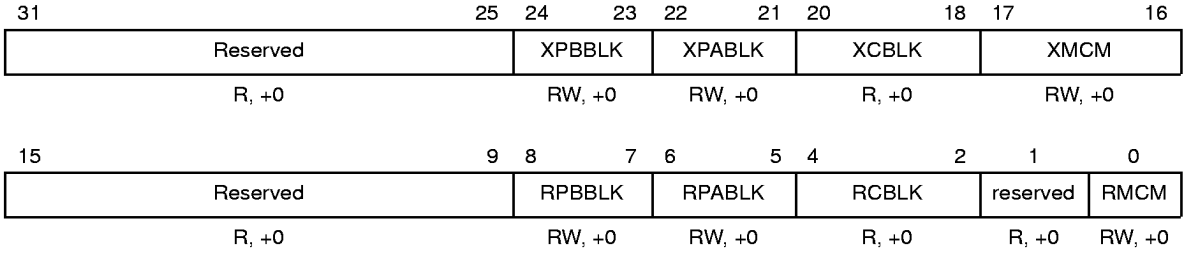


Table 8–18. Multichannel Control Register Field Descriptions

Name	Function	Section
RMCM	Receive multichannel selection enable RMCM = 0: All channels are enabled. RMCM = 1: All elements disabled (default value). Required channels are selected by enabling RP(A/B)BLK and RCER appropriately.	8.6.2
XMCM	Transmit multichannel selection enable XMCM = 00b: All elements are enabled without masking (DX is always driven during transmission of data). DX is masked or driven to hi–Z during inter–packet intervals, when a channel is masked regardless of whether it is enabled, or when a element is disabled. XMCM = 01b: All elements are disabled and therefore masked by default. Required elements are selected by enabling XP(A/B)BLK and XCER appropriately. Also, these selected elements are not masked, DX is always driven. XMCM = 10b: All elements are enabled, but masked. Selected elements that are enabled via XP(A/B)BLK and XCER are unmasked. XMCM = 11b: All elements are disabled and therefore masked by default. Required elements are selected by enabling RP(A/B)BLK and RCER appropriately. Selected elements can be unmasked by RP(A/B)BLK and XCER. This mode is used for symmetric transmit and receive operation. (See section 8.6.3 for more details on symmetric operation)	8.6.3

Table 8–18. Multichannel Control Register Field Descriptions (Continued)

Name	Function	Section
RCBLK	Receive current subframe	8.6.3.2
	RCBLK = 000b: Subframe 0. Element 0 to element 15	
	RCBLK = 001b: Subframe 1. Element 16 to element 31	
	RCBLK = 010b: Subframe 2. Element 32 to element 47	
	RCBLK = 011b: Subframe 3. Element 48 to element 63	
	RCBLK = 100b: Subframe 4. Element 64 to element 79	
	RCBLK = 101b: Subframe 5. Element 80 to element 95	
	RCBLK = 110b: Subframe 6. Element 96 to element 111	
	RCBLK = 111b: Subframe 7. Element 112 to element 127	
XCBLK	Transmit current subframe	8.6.3.2
	XCBLK = 000b: Subframe 0. Element 0 to channel 15	
	XCBLK = 001b: Subframe 1. Element 16 to element 31	
	XCBLK = 010b: Subframe 2. Element 32 to element 47	
	XCBLK = 011b: Subframe 3. Element 48 to element 63	
	XCBLK = 100b: Subframe 4. Element 64 to element 79	
	XCBLK = 101b: Subframe 5. Element 80 to element 95	
	XCBLK = 110b: Subframe 6. Element 96 to element 111	
	XCBLK = 111b: Subframe 7. Element 112 to element 127	
RPBBLK	Receive partition B subframe	8.6.3
	RPBBLK = 00b: Subframe 1. Element 16 to element 31	
	RPBBLK = 01b: Subframe 3. Element 48 to element 63	
	RPBBLK = 10b: Subframe 5. Element 80 to element 95	
	RPBBLK = 11b: Subframe 7. Element 112 to element 127	
XPBBLK	Transmit partition B subframe	8.6.3
	XPBBLK = 00b: Subframe 1. Element 16 to element 31	
	XPBBLK = 01b: Subframe 3. Element 48 to element 63	
	XPBBLK = 10b: Subframe 5. Element 80 to element 95	
	XPBBLK = 11b: Subframe 7. Element 112 to element 127	

Table 8–18. Multichannel Control Register Field Descriptions (Continued)

Name	Function	Section
RPABLK	Receive partition A subframe	8.6.3
	RPABLK = 00b: Subframe 0. Element 0 to element 15	
	RPABLK = 01b: Subframe 2. Element 32 to element 47	
	RPABLK = 10b: Subframe 4. Element 64 to element 79	
	RPABLK = 11b: Subframe 6. Element 96 to element 111	
XPABLK	Transmit partition A subframe	8.6.3
	XPABLK = 00b: Subframe 0. Element 0 to element 15	
	XPABLK = 01b: Subframe 2. Element 32 to element 47	
	XPABLK = 10b: Subframe 4. Element 64 to element 79	
	XPABLK = 11b: Subframe 6. Element 96 to element 111	

8.6.2 Enabling Multichannel Selection

Multichannel mode can be enabled independently for reception and transmission by setting RMCM = 1 and XMCM to a nonzero value in the MCR, respectively.

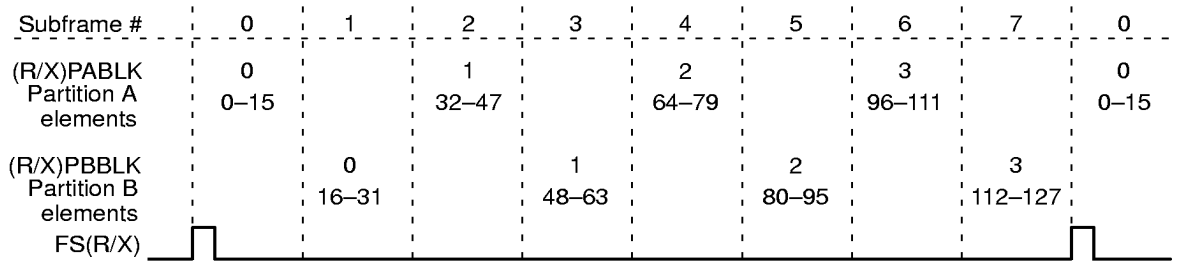
8.6.3 Enabling and Masking of Channels

A total of 32 of the available 128 elements can be enabled at any given time. The 128 elements comprise eight subframes (0 through 7) and each subframe has 16 contiguous elements. Further, even-numbered subframes 0, 2, 4, and 6 belong to partition A, and odd-numbered subframes 1, 3, 5, and 7 belong to partition B.

The number of elements enabled can be updated during the course of a frame to allow any arbitrary group of elements to be enabled. This update is accomplished using an alternating ping-pong scheme for controlling two subframes (one odd-numbered and other even-numbered) of 16 contiguous elements within a frame at any time. One subframe belongs to partition A and the other to partition B.

Any one 16-element block from partition A and from partition B can be selected, yielding a total of 32 elements that can be enabled. The subframes are allocated on 16-element boundaries within the frame, as shown in Figure 8–46. The (R/X)PABLK and (R/X)PBBLK fields in the MCR select the subframes in partition A and B respectively. This enabling is performed independently for transmit and receive.

Figure 8–46. Element Enabling by Subframes in Partitions A and B



Transmit data masking allows an element enabled for transmit to have its DX pin set to high impedance state during its transmit period. In systems where symmetric transmit and receive provides software benefits, this feature allows transmit elements to be disabled on a shared serial bus. A similar feature is not needed for receive, because multiple receptions cannot cause serial bus contention.

Note:

DX is masked or driven to the high-impedance state during (a) inter-packet intervals, (b) when an element is masked regardless of whether it is enabled, or (c) when an element is disabled.

Various XMCM values result in different operation, as described here:

- XMCM = 00b: The serial port transmits data over the DX pin for the number of elements programmed in XFRLEN1. Thus, DX is driven during transmit.
- XMCM = 01b: Only those elements that need to be transmitted are selected via XP(A/B)BLK and XCER. Only these selected elements are written to DXR and ultimately transmitted. In other words, if XINTM = 00b, which implies that an XINT is generated for every DXR-to-XSR copy, the number of XINT generated is equal to the number of elements selected via XCER (and *not* equal to XFRLEN1).
- XMCM = 10b: For this case, all elements are enabled which means all the elements in a data frame (XFRLEN1) are written to DXR and DXR-to-XSR copies occur at their respective times. However, DX is driven only for those elements that are selected via XP(A/B)BLK and XCER and is placed in the high-impedance state otherwise. In this case, if XINTM = 00b, the number of interrupts generated due to every DXR-to-XSR copy would equal the number of elements in that frame (XFRLEN1).

- XMCM = 11b: In this mode, symmetric transmit and receive operation is forced. Symmetric operation happens when a device transmits and receives on the same set of subframes. These subframes are determined by setting RP(A/B)BLK. The elements in each of these subframes can then be enabled/selected using the RCER register for receive. The transmit side uses the same blocks as the receive side (thus the value of X(P/A)BLK does not matter). In this mode, all elements are disabled, so DR and DX are in the high-impedance state. For receiving, a RBR-to-DRR copy occurs only for those elements that are selected via RP(A/B)BLK and RCER. If RINT were to be generated for every RBR-to-DRR copy, it would occur as many times as the number of elements selected in RCER (and *not* the number of elements programmed in RFRLEN1). For transmitting, the same subframe that is used for reception is used to maintain symmetry, so the value XP(A/B)BLK does not matter. DXR is loaded and DXR-to-XSR copy occurs for all the elements that are enabled via RP(A/B)BLK . However, DX is driven only for those elements that are selected via XCER. The elements enabled in XCER can be only a subset of or the same as those selected in RCER. Therefore, if XINTM = 00b, transmit interrupts to the CPU would be generated the same number of times as the number of elements selected in RCER (not XCER).

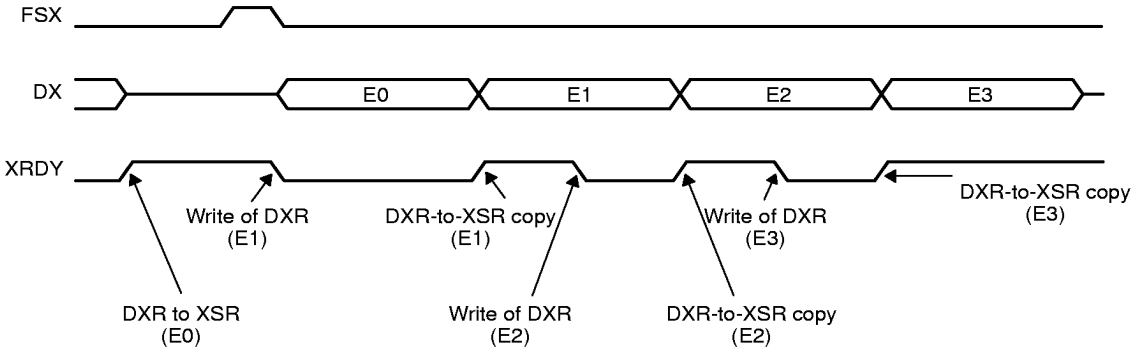
Figure 8–47 shows the activity on the McBSP pins for all the above XMCM modes with the following conditions:

- (R/X)PHASE = 0: Single-phase frame for multichannel selection enabled
- FRLEN1 = 011b: 4-element frame
- WDLEN1 = Any valid serial element length

In the following illustrations, the arrows indicating occurence of the various events are only a sample indication.

Figure 8–47. XMCM Operation

(a) XMCM = 00b



(b) XMCM = 01b, XPABLK = 00b, XCER = 1010b

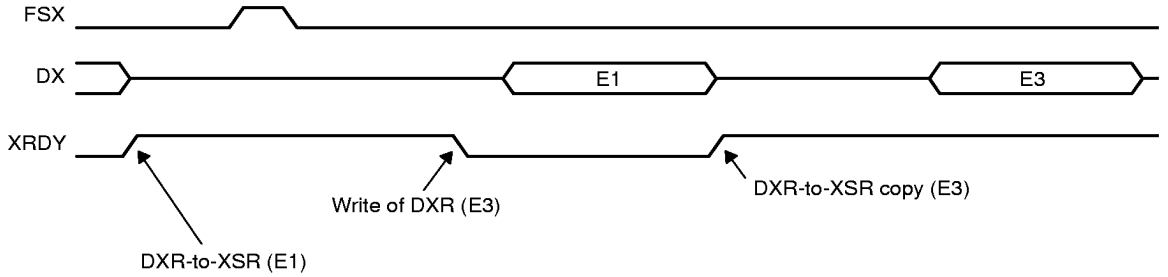
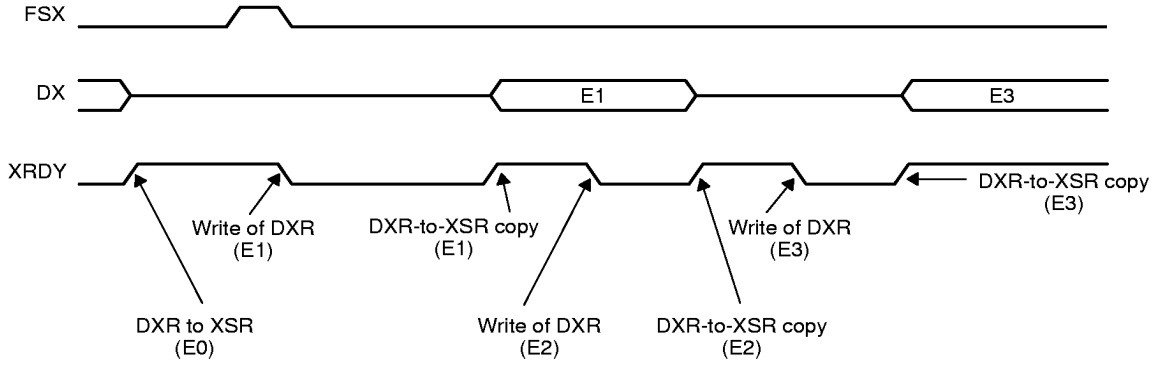
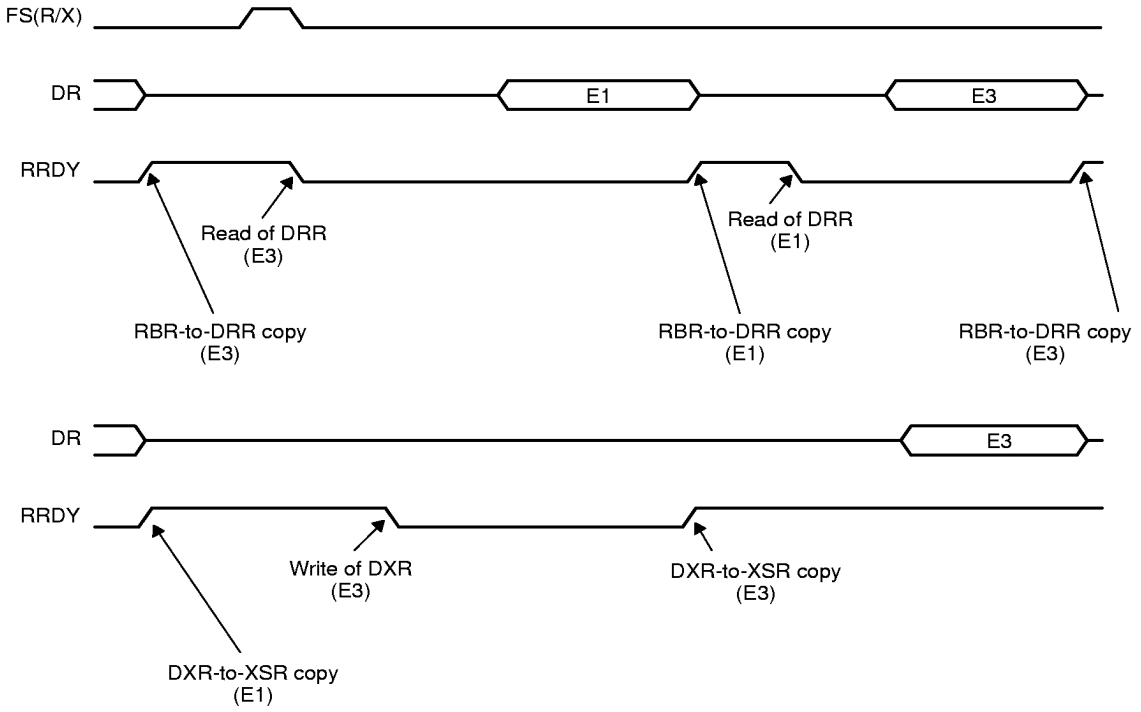


Figure 8–47. XMCM Operation (Continued)

(c) XMCM = 10b, XPABLK = 00b, XCER = 1010b



(d) XMCM = 11b, RPABLK = 00b, XPABLK = X, RCER = 1010b, XCER = 1000b



8.6.3.1 Channel Enable Registers: (R/X)CER

The receive channel enable register (RCER) and transmit channel enable register (XCER) are used to enable any of the 32 elements for receive and transmit, respectively. Of the 32 elements, 16 elements belong to a subframe in partition A and the other 16 belong to a subframe in partition B. They are shown in Figure 8–48 and Figure 8–49. (R/X)CEA and (R/X)CEB register fields shown in Table 8–19 enable elements within the 16-channel elements in partitions A and B, respectively. The (R/X)PABLK and (R/X)PBBLK fields in the MCR select which 16-element subframes are selected.

Figure 8–48. Receive Channel Enable Register (RCER)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RCEB 15	RCEB 14	RCEB 13	RCEB 12	RCEB 11	RCEB 10	RCEB 9	RCEB 8	RCEB 7	RCEB 6	RCEB 5	RCEB 4	RCEB 3	RCEB 2	RCEB 1	RCEB 0
RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCEA 15	RCEA 14	RCEA 13	RCEA 12	RCEA 11	RCEA 10	RCEA 9	RCEA 8	RCEA 7	RCEA 6	RCEA 5	RCEA 4	RCEA 3	RCEA 2	RCEA 1	RCEA 0
RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0

Figure 8–49. Transmit Channel Enable Register (XCER) Diagram

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
XCEB 15	XCEB 14	XCEB 13	XCEB 12	XCEB 11	XCEB 10	XCEB 9	XCEB 8	XCEB 7	XCEB 6	XCEB 5	XCEB 4	XCEB 3	XCEB 2	XCEB 1	XCEB 0
RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XCEA 15	XCEA 14	XCEA 13	XCEA 12	XCEA 11	XCEA 10	XCEA 9	XCEA 8	XCEA 7	XCEA 6	XCEA 5	XCEA 4	XCEA 3	XCEA 2	XCEA 1	XCEA 0
RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0

Table 8–19. Receive/Transmit Channel Enable Register Field Description

Name	Function
RCEA*n* 0 ≤ n ≤ 15	Receive channel enable RCEA *n* = 0: Disables reception of *n*th element in an even-numbered subframe in partition A RCEA *n* = 1: Enables reception of *n*th element in an even-numbered subframe in partition A
XCEA*n* 0 ≤ n ≤ 15	Transmit channel enable XCEA *n* = 0: Disables transmission of *n*th element in an even-numbered subframe in partition A XCEA *n* = 1: Enables transmission of *n*th element in an even-numbered subframe in partition A
RCEB*n* 0 ≤ n ≤ 15	Receive channel enable (R/X)CEB *n* = 0: Disables reception of *n*th element in odd-numbered subframe in partition B (R/X)CEB *n* = 1: Enables reception of *n*th element in odd-numbered subframe in partition B
XCEB*n* 0 ≤ n ≤ 15	Transmit channel enable XCEB *n* = 0: Disables transmission of *n*th element in odd-numbered subframe in partition B XCEB *n* = 1: Enables transmission of *n*th element in odd-numbered subframe in partition B

8.6.3.2 Changing Element Selection

Using the multichannel selection feature, a static group of 32 elements can be enabled and remains enabled with no CPU intervention until this allocation is modified. An arbitrary number, group, or all of the elements within a frame can be accessed by updating the block allocation registers during the course of the frame in response to the end-of-subframe interrupts (see section 8.6.3.3 for these interrupts).

You must be careful not to affect the currently selected subframe whenchanging the selection.

Note:

You must be careful not to affect the currently selected subframe when changing the selection.

The currently selected subframe is readable through the RCBLK and XCBLK fields in the MCR for receive and transmit, respectively. The associated channel enable register cannot be modified if it is selected by the appropriate (R/X)P(A/B)BLK register to point toward the current subframe. Similarly the (R/X)PABLK and (R/X)PBBLK fields in the MCR cannot be modified while pointing to or being changed to point to the currently selected subframe. If the total number of elements is 16 or fewer, the current partition is always pointed to. In this case, only a reset of the serial port can change the element enabling.

8.6.3.3 End-of-Subframe Interrupt

At the end of every subframe (16 elements or less) boundary during multichannel operation, the receive interrupt (RINT) or transmit interrupt (XINT) to the CPU is generated if RINTM = 01b or XINTM = 01b in the SPCR, respectively. This interrupt indicates a new partition has been crossed. You can then check the current partition and change the selection of subframes in the A and/or B partitions if they do not point to the current subframe. These interrupts are 2 CPU-clock high pulses. If RINTM = XINTM = 01b when (R/X)MCM = 0 (non-multichannel operation), interrupts are not generated.

8.7 SPI Protocol: CLKSTP

A system conforming to this protocol is a master-slave configuration. The SPI™ protocol is a 4-wire interface composed of serial data in (master in slave out or MISO), serial data out (master out slave in or MOSI), shift clock (SCK), and an active-low slave enable ($\overline{SS}$) signal. Communication between the master and the slave is determined by the presence or absence of the master clock. In the absence of a dedicated frame synchronization signal, the data transfer is initiated by the detection of the master clock and is terminated on absence of the master clock. The slave has to be enabled during this period of transfer. When the McBSP is the master, the slave enable is derived from the master transmit frame sync pulse, FSX. An example block diagram of the McBSP as a master and as a slave is shown in Figure 8–50 and Figure 8–51.

Figure 8–50. SPI Configuration: McBSP as the Master

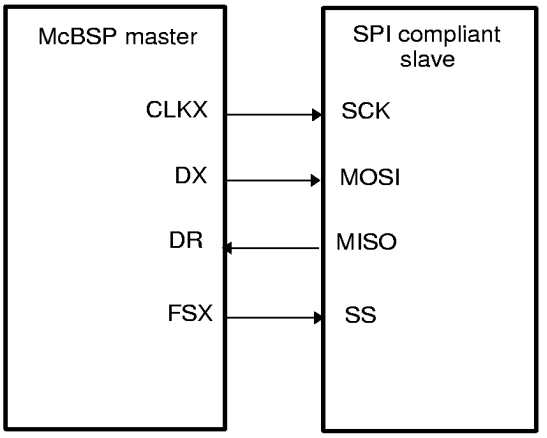
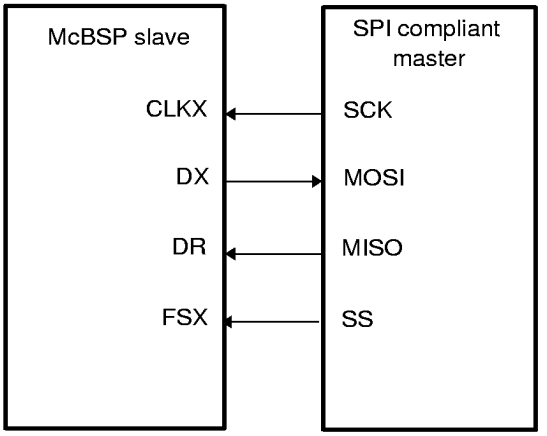


Figure 8–51. SPI Configuration: McBSP as the Slave



The clock stop mode (CLKSTP) in the McBSP provides compatibility with the SPI protocol. Clock stop mode works only with single-phase frames ((R/X)PHASE = 0) and one element per frame. The clock stop mode field (CLKSTP) in the SPCR in conjunction with the CLKXP bit in the PCR allows serial clocks to be stopped between transfers using one of four possible timing variations, as shown in Table 8–20.

Table 8–20. SPI-Mode Clock Stop Scheme

CLKSTP	CLKXP	Clock Scheme
0X	X	Clock stop mode disabled. Clock enabled for non-SPI mode.
10	0	Low inactive state without delay. The McBSP transmits data on the rising edge of CLKX and receives data on the falling edge of CLKR.
11	0	Low inactive state with delay. The McBSP transmits data one-half cycle ahead of the rising edge of CLKX and receives data on the rising edge of CLKR.
10	1	High inactive state without delay. The McBSP transmits data on the falling edge of CLKX and receives data on the rising edge of CLKR.
11	1	High inactive state with delay. The McBSP transmits data one-half cycle ahead of the falling edge of CLKX and receives data on the falling edge of CLKR.

When the McBSP is configured to operate in SPI mode, both the transmitter and the receiver operate together as a master or a slave. The McBSP is a master when it generates clocks. The master's transmit clock drives its own receive clock and the clocks to the slave device. In conjunction with CLKSTP enabled, CLKXM = 1 (in PCR) indicates that the McBSP is a master, and CLKXM = 0 indicates that the McBSP is an SPI slave.

The slave enable signal enables the serial data input and output driver on the slave device (the device not providing the output clock). Some devices do not have the slave enable pin. To interface to devices with or without the slave enable, an alternative framing scheme is used.

When the McBSP is the SPI master, CLKX should be configured as an output and FSX should be configured as an output (generated with a load of DXR) that can be connected to the slave enable ($\overline{SS}$) input on the slave device. The FSGM bit in SRGR is 0, so the DXR-to-XSR transfer generates the FSX, internal FSR, and slave enable signals. The slave should be enabled before beginning the transfer, which means that XDATDLY must be programmed to 1. The CLKGDV (clock divide ratio) in SRGR should be programmed to generate the required SPI data rate. The McBSP generates a continuous clock (CLKX) internally and gates the clock off (stops the clock) to the external interface when transfers are finished. The McBSP's receive clock is provided from the internal, continuously running clock, so the receiver and transmitter both work internally as if clocks do not stop.

When the McBSP is a slave device, the internal serial port logic performs transfers using only the exact number of input clock pulses per data bit. The CLK(R/X) pins are configured as inputs. Only CLKX needs to be driven by the SPI master clock in the system. If the master device provides a slave enable signal, it is connected to FS(R/X) and is used in its asynchronous form. Thus, transmit and receive frame sync mode bits, FSXM and FSRM, should be set to 0. FSX, then, controls only the initial drive of data to the DX pin. When the McBSP is a slave, the (R/X)DATDLY should be zero so that data is driven out or shifted in on the same edge where the frame sync occurs. If the master device does not provide a slave enable, the FSR/FSX pins are connected to an active level (as defined by the FS(R/X) polarity control bits), and data is driven to the DX pin as soon as DXR is loaded. The input clock and frame sync from the master are synchronized to the CPU clock to allow reset. The CLKGDV must be a value such that the CLKG is at least eight times the SPI data rate. This value is acheived by programming CLKGDV = 1 for all cases of SPI transfer. The first data to be transmitted is available on the DX pin but is enabled only after detection of the SPI clock.

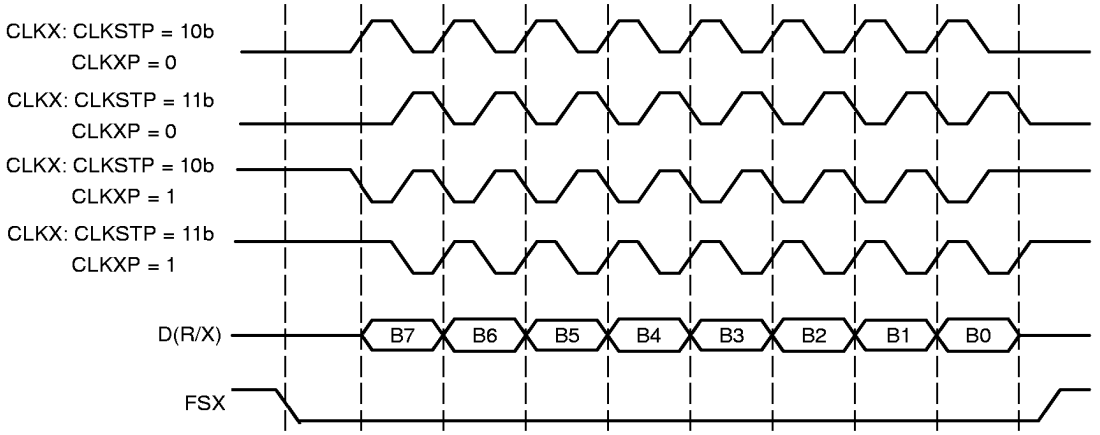
In clock stop mode for the transmitter and the receiver, stopping of clocks is handled differently depending on whether the clocks are externally or internally generated. In the case where clocks are externally generated, the external device that is generating clocks holds clocks appropriately between transmitted elements. When the serial port clock is internally generated, this clock runs continuously and is simply gated off (the clock stops) to the external pin when the transfer is complete. In this case, transfers are performed in the same fashion as with external free running clocks.

The CLKSTP and CLKXP bits select the appropriate clock scheme for a particular SPI interface as shown in Table 8–20 and Figure 8–52. The CLKSTP and CLKXP fields in the SPCR determine the following conditions:

- Whether clock stop mode is enabled or not
- In clock stop mode, whether the clock is high or low when stopped
- In clock stop mode, whether the first clock edge occurs at the start of the first data bit or at the middle of the first data bit

The CLKXP bit selects the edge on which data is transmitted (driven) and received (sampled) as shown in Table 8–20.

Figure 8–52. Clock Stop Mode Options



8.7.1 McBSP Initialization for SPI Mode

The operation of the serial port during device reset, transmitter reset, and receiver reset is described in subsection 8.3.1. For McBSP operation as a master or a slave in SPI mode, you must follow these steps for proper initialization:

1) Set $\overline{\text{XRST}}$ = $\overline{\text{RRST}}$ = 0 in SPCR.

2) Program the necessary McBSP configuration registers (and not the data registers) listed in Table 8–2 as required when the serial port is in reset state ($\overline{\text{XRST}}$ = $\overline{\text{RRST}}$ = 0) *except* for CLKSTP which should be disabled. Program CLKSTP to 0Xb, if CLKSTP is not disabled.

3) Wait two bit clocks for the McBSP to reinitialize.

4) Write the desired value into the CLKSTP field in the SPCR.

5) Set $\overline{\text{XRST}}$ = $\overline{\text{RRST}}$ = 1 to enable the serial port. The value written to the SPCR at this time should have only the reset bits changed to 1; the remaining fields should have the same value as in steps 2 and 4.

6) Wait two bit clocks for the receiver and transmitter to become active.

8.8 McBSP Pins as General-Purpose I/O

Two conditions allow the serial port pins (CLKX, FSX, DX, CLKR, FSR, DR, and CLKS) to be used as general-purpose I/O rather than serial port pins:

- The related portion (transmitter or receiver) of the serial port is in reset: $(\overline{R/X})\overline{RST} = 0$ in the SPCR
- General-purpose I/O is enabled for the related portion of the serial port: (R/X)IOEN = 1 in the PCR

Figure 8–3 shows the PCR bits that configure each of the McBSP pins as general purpose inputs or outputs. Table 8–21 shows how this is achieved. In the case of FS(R/X), FS(R/X)M = 0 configures the pin as an input and FS(R/X)M = 1 configures that pin as an output. When configured as an output, the value driven on FS(R/X) is the value stored in FS(R/X)P. If configured as an input, the FS(R/X)P becomes a read-only bit that reflects the status of that signal. CLK(R/X)M and CLK(R/X)P work similarly for CLK(R/X). When the transmitter is selected as general-purpose I/O, the value of the DX_STAT bit in the PCR is driven onto DX. DR is always an input and its value is held in the DR_STAT bit in the PCR. To configure CLKS as a general-purpose input, both the transmitter and receiver have to be in reset state and (R/X)IOEN = 1, because it is always an input to the McBSP and affects both transmit and receive operations.

Table 8–21. Configuration of Pins as General Purpose I/O

Pin	General–Purpose I/O Enabled When...	Selected as Output When...	Output Value Driven From	Selected as Input When ...	Input Value Readable on
CLKX	$\overline{XRST}$ = 0 XIOEN = 1	CLKXM = 1	CLKXP	CLKXM = 0	CLKXP
FSX	$\overline{XRST}$ = 0 XIOEN = 1	FSXM = 1	FSXP	FSXM = 0	FSXP
DX	$\overline{XRST}$ = 0 XIOEN = 1	Always	DX_STAT	Never	Does not apply
CLKR	$\overline{RRST}$ = 0 RIOEN = 1	CLKRM = 1	CLKRP	CLKRM = 0	CLKRP
FSR	$\overline{RRST}$ = 0 RIOEN = 1	FSRM = 1	FSRP	FSRM = 0	FSRP
DR	$\overline{RRST}$ = 0 RIOEN = 1	Never	Does not apply	Always	DR_STAT
CLKS	$\overline{RRST}$ = $\overline{XRST}$ = 0 RIOEN = XIOEN = 1	Never	Does not apply	Always	CLKS_STAT