# TMS320DM357 DMSoC Audio Serial Port (ASP) Interface

# Reference Guide

Literature Number: SPRUG35
November 2008

# Contents

# List of Figures

# List of Tables

# Read This First

## About This Manual

Describes the operation of the audio serial port (ASP) audio interface in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The primary audio modes that are supported by the ASP are the AC97 and IIS modes. In addition to the primary audio modes, the ASP supports general serial port receive and transmit operation, but is not intended to be used as a high-speed interface.

## Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.
- Registers in this document are shown in figures and described in tables.
  - Each register figure shows a rectangle divided into fields that represent the fields of the register. Each field is labeled with its bit name, its beginning and ending bit numbers above, and its read/write properties below. A legend explains the notation used for the properties.
  - Reserved bits in a register figure designate a bit that is used for future device expansion.

## *Related Documentation From Texas Instruments*

The following documents describe the TMS320DM357 Digital Media System-on-Chip (DMSoC). Copies of these documents are available on the Internet at www.ti.com. *Tip:* Enter the literature number in the search box provided at www.ti.com.

**SPRUG06** — ***TMS320DM357 DMSoC Video Processing Back End (VPBE) User's Guide.*** Describes the video processing back end (VPBE) in the TMS320DM357 Digital Media System-on-Chip (DMSoC) video processing subsystem. Included in the VPBE is the video encoder, on-screen display, and digital LCD controller.

**SPRUG25** — ***TMS320DM357 DMSoC ARM Subsystem Reference Guide.*** Describes the ARM subsystem in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The ARM subsystem is designed to give the ARM926EJ-S (ARM9) master control of the device. In general, the ARM is responsible for configuration and control of the device; including the video processing subsystem, and a majority of the peripherals and external memories.

**SPRUG26** — ***TMS320DM357 DMSoC Universal Asynchronous Receiver/Transmitter (UART) User's Guide.*** This document describes the universal asynchronous receiver/transmitter (UART) peripheral in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The UART peripheral performs serial-to-parallel conversion on data received from a peripheral device, and parallel-to-serial conversion on data received from the CPU.

**SPRUG27** — ***TMS320DM357 DMSoC Inter-Integrated Circuit (I2C) Peripheral User's Guide.*** Describes the inter-integrated circuit (I2C) peripheral in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The I2C peripheral provides an interface between the DMSoC and other devices compliant with the I2C-bus specification and connected by way of an I2C-bus. External components attached to this 2-wire serial bus can transmit and receive up to 8-bit wide data to and from the DMSoC through the I2C peripheral. This document assumes the reader is familiar with the I2C-bus specification.

**SPRUG28** — ***TMS320DM357 DMSoC 64-Bit Timer User's Guide.*** Describes the operation of the software-programmable 64-bit timer in the TMS320DM357 Digital Media System-on-Chip (DMSoC). Timer 0 and Timer 1 are used as general-purpose (GP) timers and can be programmed in 64-bit mode, dual 32-bit unchained mode, or dual 32-bit chained mode; Timer 2 is used only as a watchdog timer. The GP timer modes can be used to generate periodic interrupts or enhanced direct memory access (EDMA) synchronization events. The watchdog timer mode is used to provide a recovery mechanism for the device in the event of a fault condition, such as a non-exiting code loop.

**SPRUG29** — ***TMS320DM357 DMSoC Serial Peripheral Interface (SPI) User's Guide.*** Describes the serial peripheral interface (SPI) in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The SPI is a high-speed synchronous serial input/output port that allows a serial bit stream of programmed length (1 to 16 bits) to be shifted into and out of the device at a programmed bit-transfer rate. The SPI is normally used for communication between the DMSoC and external peripherals. Typical applications include an interface to external I/O or peripheral expansion via devices such as shift registers, display drivers, SPI EPROMs and analog-to-digital converters.

**SPRUG30** — ***TMS320DM357 DMSoC Host Port Interface (HPI) Reference Guide.*** This document describes the host port interface in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The HPI provides a parallel port interface through which an external host processor can directly access the TMS320DM357 DMSoC processor's resources (configuration and program/data memories).

**SPRUG31** — ***TMS320DM357 DMSoC General-Purpose Input/Output (GPIO) User's Guide.*** Describes the general-purpose input/output (GPIO) peripheral in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The GPIO peripheral provides dedicated general-purpose pins that can be configured as either inputs or outputs. When configured as an input, you can detect the state of the input by reading the state of an internal register. When configured as an output, you can write to an internal register to control the state driven on the output pin.

**SPRUG32** — ***TMS320DM357 DMSoC Multimedia Card (MMC)/Secure Digital (SD) Card Controller User's Guide.*** Describes the multimedia card (MMC)/secure digital (SD) card controller in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The MMC/SD card is used in a number of applications to provide removable data storage. The MMC/SD controller provides an interface to external MMC and SD cards. The communication between the MMC/SD controller and MMC/SD card(s) is performed by the MMC/SD protocol.

**SPRUG33** — ***TMS320DM357 DMSoC Asynchronous External Memory Interface (EMIF) User's Guide.*** Describes the asynchronous external memory interface (EMIF) in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The EMIF supports a glueless interface to a variety of external devices.

**SPRUG34** — ***TMS320DM357 DMSoC Enhanced Direct Memory Access (EDMA) Controller User's Guide.*** Describes the operation of the enhanced direct memory access (EDMA3) controller in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The EDMA3 controller's primary purpose is to service user-programmed data transfers between two memory-mapped slave endpoints on the DMSoC.

**SPRUG35** — ***TMS320DM357 DMSoC Audio Serial Port (ASP) User's Guide.*** Describes the operation of the audio serial port (ASP) audio interface in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The primary audio modes that are supported by the ASP are the AC97 and IIS modes. In addition to the primary audio modes, the ASP supports general serial port receive and transmit operation, but is not intended to be used as a high-speed interface.

**SPRUG36** — ***TMS320DM357 DMSoC Ethernet Media Access Controller (EMAC)/Management Data Input/Output (MDIO) Module User's Guide.*** Discusses the ethernet media access controller (EMAC) and physical layer (PHY) device management data input/output (MDIO) module in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The EMAC controls the flow of packet data from the DMSoC to the PHY. The MDIO module controls PHY configuration and status monitoring.

**SPRUG37** — ***TMS320DM357 DMSoC Pulse-Width Modulator (PWM) Peripheral User's Guide.*** Describes the pulse-width modulator (PWM) peripheral in the TMS320DM357 Digital Media System-on-Chip (DMSoC).

**SPRUG38** — ***TMS320DM357 DMSoC DDR2 Memory Controller User's Guide.*** Describes the DDR2 memory controller in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The DDR2 memory controller is used to interface with JESD79D-2A standard compliant DDR2 SDRAM devices.

**SPRUG39** — ***TMS320DM357 DMSoC Video Processing Front End (VPFE) User's Guide.*** Describes the video processing front end (VPFE) in the TMS320DM357 Digital Media System-on-Chip (DMSoC) video processing subsystem. Included in the VPFE is the preview engine, CCD controller, resizer, histogram, and hardware 3A (H3A) statistic generator.

**SPRUGH2** — ***TMS320DM357 DMSoC Peripherals Overview Reference Guide.*** This document provides an overview of the peripherals in the TMS320DM357 Digital Media System-on-Chip (DMSoC).

**SPRUGH3** — ***TMS320DM357 DMSoC Universal Serial Bus Controller User's Guide.*** This document describes the universal serial bus (USB) controller in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The USB controller supports data throughput rates up to 480 Mbps. It provides a mechanism for data transfer between USB devices and also supports host negotiation.

## Trademarks

# Audio Serial Port (ASP) Interface

## 1 Introduction

This document describes the operation of the audio serial port (ASP) in the TMS320DM357 Digital Media System-on-Chip (DMSoC).

### 1.1 Purpose of the Peripheral

The primary use for the audio serial port (ASP) is for audio interface purposes. The ASP is a specialized version of the multichannel buffered serial port (McBSP) peripheral used on other TI digital signal processors (DSPs). The primary audio modes that are supported by the ASP are the AC97 and IIS modes. In addition to the primary audio modes, the ASP can be programmed to support other serial formats but is not intended to be used as a high-speed interface.

The ASP can be controlled by the ARM CPU.

### 1.2 Features

The ASP provides the following functions:

- 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 analog-to-digital (A/D) and digital-to-analog (D/A) devices
- External shift clock or an internal, programmable frequency shift clock for data transfer

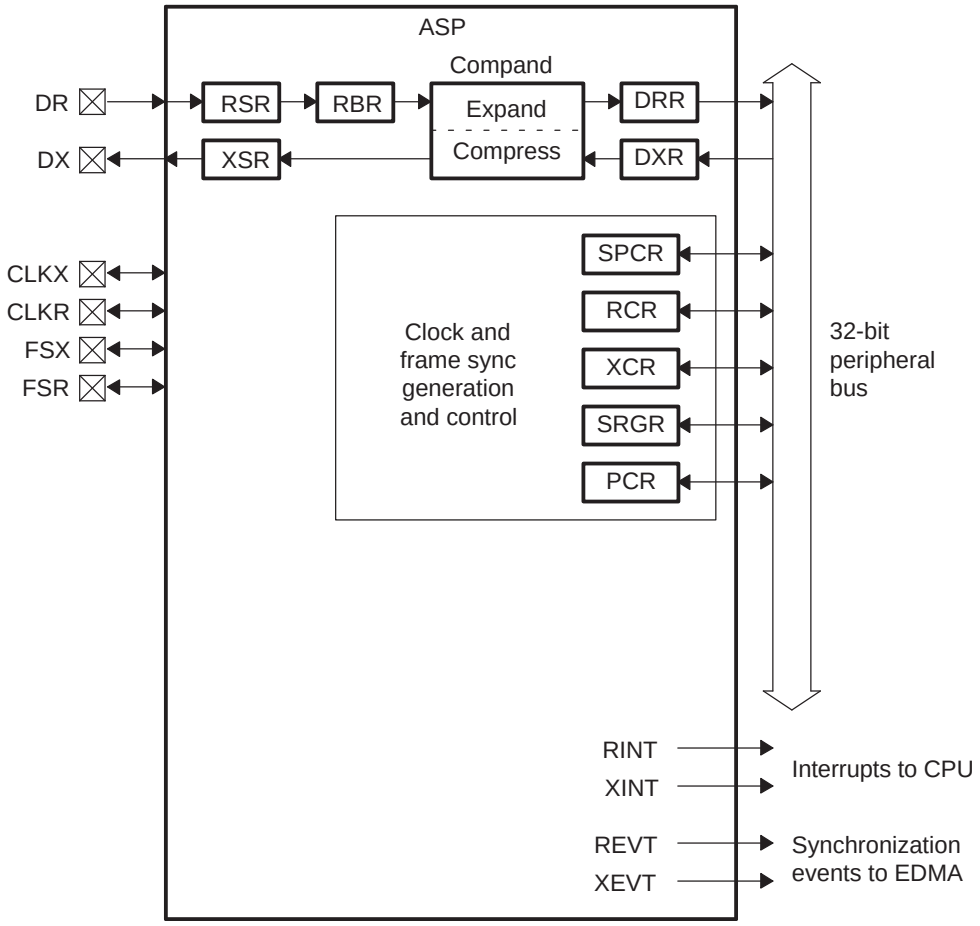
In addition, the ASP has the following capabilities:

- Direct interface to:
  - AC97 compliant devices (the necessary multiphase frame synchronization capability is provided)
  - IIS compliant devices
- 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 the option of LSB or MSB first
- Programmable polarity for both frame synchronization and data clocks
- Highly programmable internal clock and frame generation

## 1.3 Functional Block Diagram

The ASP consists of a data path and control path, as shown in Figure 1

**Figure 1. ASP Block Diagram**




## 1.4 Industry Standard Compliance Statement

The ASP supports the following industry standard interfaces:

**AC97—** The AC97 standard specifies a 5-wire digital serial link between an audio codec device and its digital controller.

**IIS—** IIS is a protocol for transmitting two channels of digital audio data over a single serial connection. The IIS bus is an industry standard three-wire interface for streaming stereo audio between devices, typically between a CPU and a DAC/ADC.

# 2 Peripheral Architecture

This section describes the architecture of the ASP.

## 2.1 Clock Control

The ASP can use an internal or external clock source. The internal clock source to the ASP is the SYSCLK5 chip-level clock. Either clock source can be divided down inside the ASP to generate the actual interface bit clock frequency. For detailed timing information, see the device-specific data manual. Detailed information about how the interface clock and frame synchronization signals are generated is provided in Section 2.4.

## 2.2 Signal Descriptions

The signals used on the ASP audio interface are listed in Table 1.

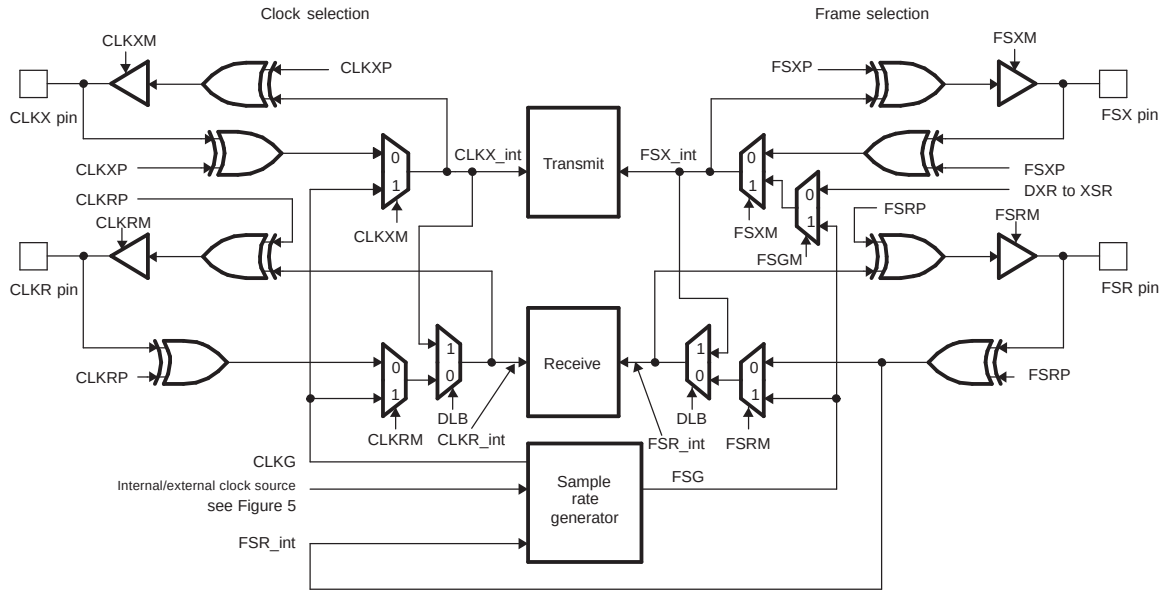
**Table 1. ASP Interface Signals**

| Pin | I/O/Z | Description |
|---|---|---|
| CLKR | I/O/Z | Receive clock - supplies or receives a reference clock for the receiver; or supplies a reference clock to the sample rate generator |
| CLKX | I/O/Z | Transmit clock - supplies or receives a reference clock for the transmitter; or supplies a reference clock to the sample rate generator |
| DR | I | Received serial data |
| DX | O/Z | Transmitted serial data |
| FSR | I/O/Z | Receive frame synchronization - control signal to synchronize the start of received data |
| FSX | I/O/Z | Transmit frame synchronization - control signal to synchronize the start of transmitted data |

## 2.3 Clock, Frames, and Data

The ASP has several ways 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 2 is a block diagram of the clock and frame selection circuitry.

**Figure 2. Clock and Frame Generation**



### 2.3.1 Frame and Clock Operation

Receive and transmit frame sync pulses (FSR/X), and clocks (CLKR/X), can either be generated internally by the sample rate generator (see section Section 2.3.2) or be driven by an external source. The source of frame sync and clock is selected by programming the mode bits, FS(R/X)M and CLK(R/X)M respectively, in the pin control register (PCR).

When FSR and FSX are inputs (FSXM = FSRM = 0), the ASP detects them on the internal falling edge of clock, CLKR_int and CLKX_int, respectively (see Figure 2). The receive data arriving at the DR pin is also sampled on the falling edge of CLKR_int. These internal clock signals are either derived from external source via the CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the ASP.

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.

FSRP, FSXP, CLKRP, and CLKXP configure the polarities of FSR, FSX, CLKR, and CLKX. 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 ASP) 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), 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 2 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 (see Figure 3). 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 is 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 (by the transmitter) with a rising-edge clock. 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 (see Figure 4). 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 4 shows how data clocked by an external serial device using a rising-edge clock can be sampled by the ASP receiver with the falling edge of the same clock.

**Figure 3. Transmit Data Clocking**

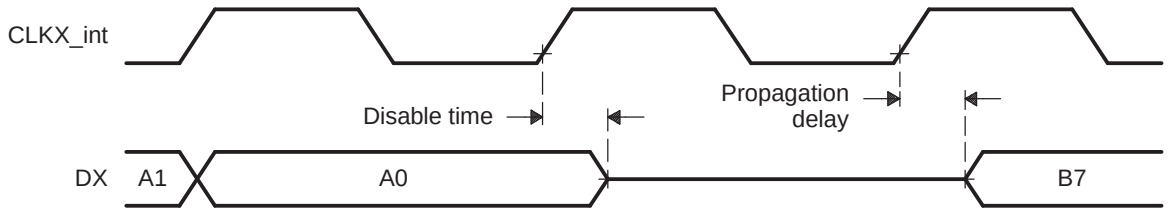
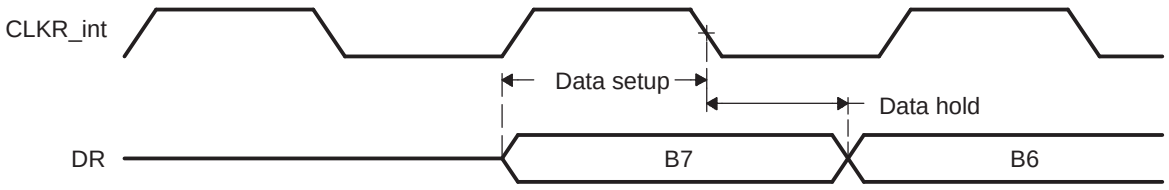



**Figure 4. Receive Data Clocking**



### 2.3.2 Sample Rate Generator Clocking and Framing

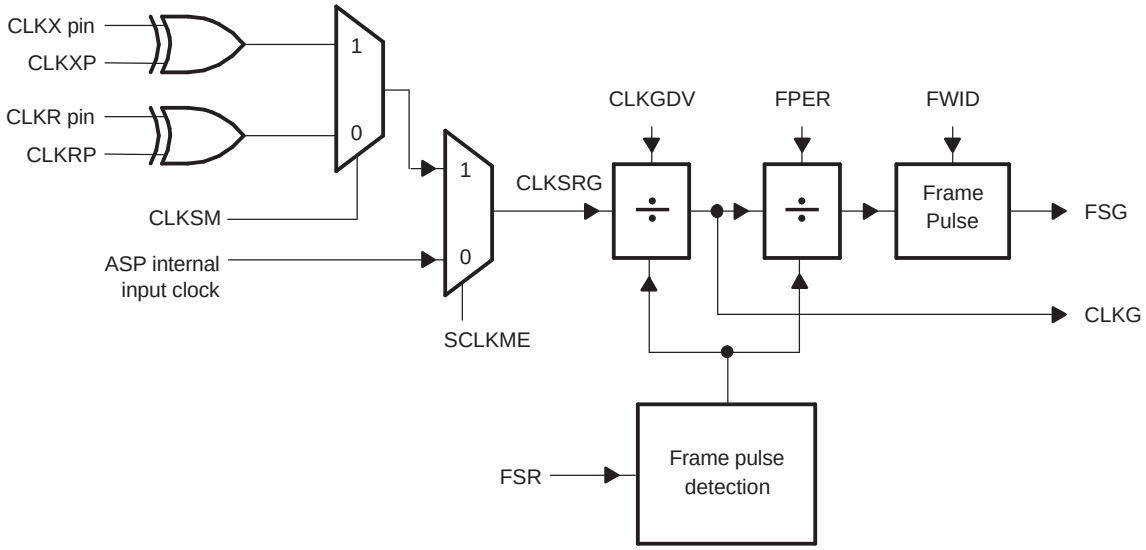
The sample rate generator is composed of a 3-stage clock divider that provides a programmable data clock (CLKG) and framing signal (FSG), as shown in Figure 5. CLKG and FSG are ASP internal signals that can be programmed to drive receive and/or transmit clocking, CLK(R/X), and framing, FS(R/X). The sample rate generator can be programmed to be driven by an internal clock source or an internal clock derived from an external clock source.

The sample rate generator is not used when CLKX, FSX, CLKR, and FSR are driven by an external source. Therefore, the GRST bit in SPCR does not need to be enabled (GRST = 1) for this setup. 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 2.6.

**Figure 5. Sample Rate Generator Block Diagram**



### 2.3.3 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 for the receiver and transmitter from a variety of data bit clocks including:

- The input clock to the sample rate generator, which can be either the internal clock source or a dedicated external clock source (CLKX/R). See Section 2.3.3.1 for details on the source of the ASP internal clock.
- The input clock source (internal clock source or external clock CLKX/R) 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 5) generates CLKG and FSG.

#### *2.3.3.1 Input Clock Source Mode: CLKSM and SCLKME*

The sample rate generator must be driven by an input clock signal from one of the three sources selectable with the SCLKME bit of PCR and the CLKSM bit of SRGR (see Table 2).

**Table 2. Choosing an Input Clock for the Sample Rate Generator With the SCLKME and CLKSM Bits**

| SCLKME | CLKSM | Input Clock For Sample Rate Generator |
|---|---|---|
| 0 | 0 | Reserved |
| 0 | 1 | ASP internal input clock |
| 1 | 0 | Signal on CLKR pin |
| 1 | 1 | Signal on CLKX pin |

#### *2.3.3.2 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 that is output on the sample rate generator output, CLKG, and that 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 to 256. The CLKGDV value chosen must result in a clock that meets the timing requirements/limitations specified in the device-specific data manual.

When CLKGDV is an odd value or equal to 0, the CLKG duty cycle is 50%. Note that an odd CLKGDV value means an even divide down of the source clock and an even CLKGDV value means an odd divide down of the source clock. When CLKGDV is an even value (2p), the high state duration is p + 1 cycles and the low state duration is p cycles. This is illustrated in Example 1, Example 2, and Example 3.

In the following examples:

$S_{IN}$ = sample generator input clock period

$f_{IN}$ = sample generator input clock frequency

$S_G$ = CLKG period

$f_G$ = CLKG frequency

The following equation is given above: $f_G = f_{IN}/(CLKGDV + 1)$; therefore, $S_G = (CLKGDV + 1) \times S_{IN}$.

***Example 1. CLKGDV = 0***

CLKGDV = 0

$S_G$ = (CLKGDV + 1) ɲ$S_{IN}$ = (0 + 1) ɲ$S_{IN}$ = $S_{IN}$

Pulse width high = $S_{IN}$ ɲ(CLKGDV + 1)/2 = $S_{IN}$ ɲ(0+ 1)/2 = 0.5 ɲ$S_{IN}$

Pulse width low = $S_{IN}$ ɲ(CLKGDV + 1)/2 = $S_{IN}$ ɲ(0+ 1)/2 = 0.5 ɲ$S_{IN}$

***Example 2. CLKGDV = 1***

CLKGDV = 1

$S_G$ = (CLKGDV + 1) ɲ$S_{IN}$ = (1 + 1) ɲ$S_{IN}$ = 2 ɲ$S_{IN}$

Pulse width high = $S_{IN}$ ɲ(CLKGDV + 1)/2 = $S_{IN}$ ɲ(1 +1)/2 = $S_{IN}$

Pulse width low = $S_{IN}$ ɲ(CLKGDV + 1)/2 = $S_{IN}$ ɲ(1+ 1)/2 = $S_{IN}$

***Example 3. CLKGDV = 2***

CLKGDV = 2

$S_G$ = (CLKGDV + 1) ɲ$S_{IN}$ = (2 + 1) ɲ$S_{IN}$ = 3 ɲ$S_{IN}$

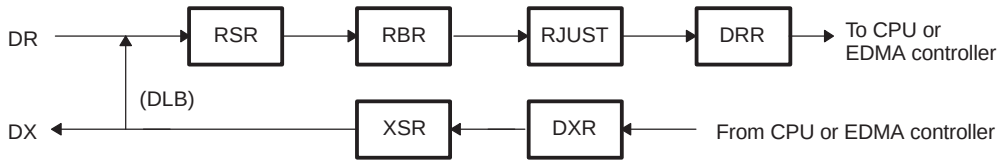
Pulse width high = $S_{IN}$ ɲ(CLKGDV/2 + 1) = $S_{IN}$ ɲ(2/2 + 1) = 2 ɲ$S_{IN}$

Pulse width low = $S_{IN}$ ɲCLKGDV/2 = $S_{IN}$ ɲ2/2 = 1 ɲ$S_{IN}$

### 2.3.3.3 Digital Loopback Mode: DLB

Setting DLB = 1 in SPCR enables digital loopback mode. In DLB mode, DR, FSR, and CLKR are internally connected through multiplexers to DX, FSX, and CLKX, respectively, as shown in Figure 2 and Figure 6. DLB mode allows testing of serial port code without using the external interface of the ASP. CLKX and FSX must be enabled as outputs (CLKXM = FSXM = 1) in DLB mode.

**Figure 6. Digital Loopback Mode**




### 2.3.3.4 Receive Clock Selection: DLB, CLKRM

Table 3 shows how the digital loopback bit (DLB) and the CLKRM bit in PCR select the receiver clock. In digital loopback mode (DLB = 1), the transmitter clock drives the receiver. CLKRM determines whether the CLKR pin is an input or an output.

**Table 3. Receive Clock Selection**

| DLB Bit in SPCR | CLKRM Bit in PCR | Source of Receive Clock | CLKR Function |
|---|---|---|---|
| 0 | 0 | CLKR acts as an input driven by the external clock and inverted as determined by CLKRP before being used. | Input. |
| 0 | 1 | The 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 is inverted. | High impedance. |
| 1 | 1 | CLKX_int drives CLKR_int as selected and is inverted. | Output. CLKR (same as CLKX) is inverted as determined by CLKRP before being driven out. |

### 2.3.3.5 Transmit Clock Selection: CLKXM

Table 4 shows how the CLKXM bit in PCR selects the transmit clock and whether the CLKX pin is an input or output.

**Table 4. Transmit Clock Selection**

| CLKXM Bit in PCR | Source of Transmit Clock | CLKX Function |
|---|---|---|
| 0 | The external clock drives the CLKX input pin. CLKX is inverted as determined by CLKXP before being used. | Input. |
| 1 | The sample rate generator clock (CLKG) drives the transmit clock. | Output. CLKG is inverted as determined by CLKXP before being driven out on CLKX. |

#### 2.3.3.6 Stopping Clocks

When the clocks are inputs to the ASP (CLKXM or CLKRM = 0), clocks can be stopped between data transfers. If the external device stops the serial clock between data transfers, the ASP interprets it as a slowed-down serial clock. Ensure that there are no glitches on the CLK(R/X) lines as the ASP may interpret them as clock-edge transitions. Restarting the serial clock is equivalent to a normal clock transition after a slow CLK(R/X) cycle. Note that just as in normal operations, transmit under flow (XEMPTY) may occur if the DXR is not properly serviced at least three CLKX cycles before the next frame sync. Therefore, if the serial clock is stopped before DXR is properly serviced, the external device needs to restart the clock at least three CLKX cycles before the next frame sync to allow the DXR write to be properly synchronized. See Figure 26 for a graphical explanation on when DXR needs to be written to avoid underflow.

### 2.3.4 Frame Sync Generation

Data frame synchronization is independently programmable for the receiver and the transmitter for all data delay values. When the FRST bit in the serial port control register (SPCR) is set to 1 the frame generation logic is activated to generate frame sync signals, provided that FSGM = 1 in the sample rate generator register (SRGR). The frame sync programming options are:

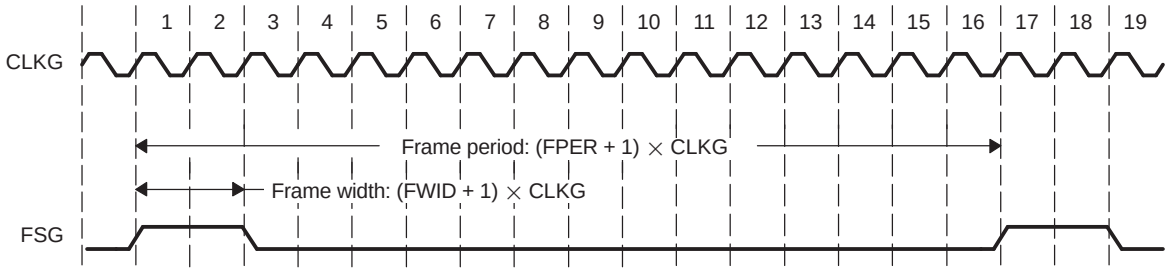
- A frame pulse with a programmable period between sync pulses and a programmable active width specified in SRGR.
- The transmitter can trigger its own frame sync signal that is generated by a DXR-to-XSR copy. This causes a frame sync to occur on every DXR-to-XSR copy. The data delays can be programmed 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.

#### 2.3.4.1 Frame Period (FPER) and Frame Width (FWID)

The FPER bits in the sample rate generator register (SRGR) are 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 bits are an 8-bit down-counter. FWID 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 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 an 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. Figure 7 shows a frame of 16 CLKG periods (FPER = 15 or 0000 1111b).

**Figure 7. Programmable Frame Period and Width**



#### 2.3.4.2 Receive Frame Synchronization Selection: DLB and FSRM

Table 5 shows how you can select various sources to provide the receive frame synchronization signal. Note that in digital loopback mode (DLB = 1), the transmit frame sync signal is used as the receive frame sync signal and that DR is internally connected to DX.

**Note:** FSR_int and FSX_int are shown in Figure 2.

**Table 5. Receive Frame Synchronization Selection**

| DLB Bit in SPCR | FSRM Bit in PCR | Source of Receive Frame Synchronization | FSR Pin Function |
|---|---|---|---|
| 0 | 0 | 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 | Sample rate generator frame sync signal (FSG) drives FSR_int, FRST = 1. | Output. FSG is inverted as determined by FSRP before being driven out on the FSR pin. |
| 1 | 0 | FSX_int drives FSR_int. FSX is selected as shown in Table 6. | High impedance. |
| 1 | 1 | FSX_int drives FSR_int. FSR_int is selected as shown in Table 6. | Output. Receive (same as transmit) frame synchronization is inverted as determined by FSRP before being driven out. |

#### 2.3.4.3 Transmit Frame Synchronization Selection: FSXM and FSGM

Table 6 shows how you can select the source of the transmit frame synchronization signal. 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

**Note:** FSR_int and FSX_int are shown in Figure 2.

**Table 6. Transmit Frame Synchronization Selection**

| FSXM Bit in PCR | FSGM Bit in SRGR | Source of Transmit Frame Synchronization | FSX Pin Function |
|---|---|---|---|
| 0 | X | External frame sync input on the 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. 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. |

#### 2.3.4.4 Frame Detection

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 serial port control register (SPCR). The associated portion (receiver/transmitter) of the ASP must be out of reset.

### 2.3.5 Data and Frames

#### 2.3.5.1 Frame Synchronization Phases

Frame synchronization indicates the beginning of a transfer on the ASP. 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 RCR and XCR. The number of elements per frame and bits per element can be independently selected for each phase via (R/X)FRLEN1/2 and (R/X)WDLEN1/2, respectively. Figure 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 7 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 elements in a dual-phase frame. The number of bits per element can be 8, 12, 16, 20, 24, or 32.

**Note:** For a dual-phase frame with internally generated frame synchronization, the maximum number of elements per phase depends on the word length. This is because the frame period, FPER, is only 12-bits wide and, therefore, provides 4096 bits per frame. Hence, the maximum number of 256 elements per dual- phase frame applies only when the WDLEN is 16 bits. However, any combination of element numbers and element size (defined by the FRLEN and WDLEN bits, respectively) is valid as long as their product is less than or equal to 4096 bits. This limitation does not apply for dual-phase with external frame sync.

**Figure 8. Dual-Phase Frame Example**

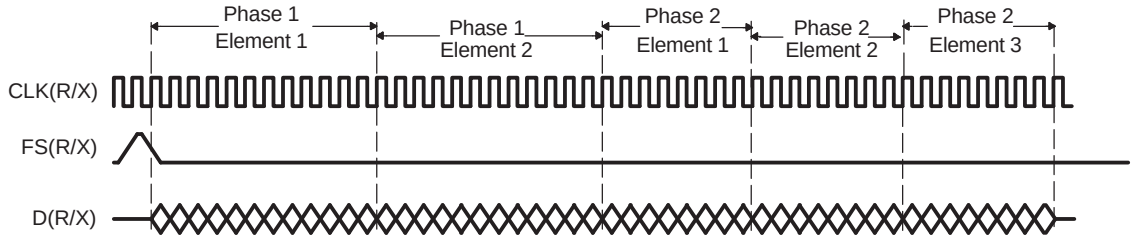



**Table 7. RCR/XCR Fields Controlling Elements per Frame and Bits per Element**

| Serial Port | 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 |

#### 2.3.5.2 Frame Length: RFRLEN1/2 and XFRLEN1/2

Frame length specifies the maximum number of serial elements or logical time slots or channels that are available for transfer per frame synchronization signal. The 7-bit (R/X)FRLEN1/2 bits in (R/X)CR support up to 128 elements per phase in a frame, as shown in Table 8. (R/X)PHASE = 0 selects a single-phase data frame, and (R/X)PHASE = 1 selects a dual-phase frame for the data stream. For a single-phase frame, the value of (R/X)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, (R/X)FRLEN1 = 1 or 000 0001b and (R/X)FRLEN2 = 2 or 000 0010b.

**Table 8. Receive/Transmit Frame Length Configuration**

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

#### 2.3.5.3 Element Length: RWDLEN1/2 and XWDLEN1/2

The (R/X)WDLEN1/2 fields in the receive/transmit control register (RCR and XCR) determine the element length in bits per element for the receiver and the transmitter for each phase of the frame, as indicated in Table 7. Table 9 shows how the value of these fields selects particular element lengths in bits. For the example in Figure 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 ASP and its value does not matter.

**Table 9. Receive/Transmit Element Length Configuration**

| (R/X)WDLEN1/2 | Element Length (Bits) |
|---|---|
| 000 | 8 |
| 001 | 12 |
| 010 | 16 |
| 011 | 20 |
| 100 | 24 |
| 101 | 32 |
| 110 | Reserved |
| 111 | Reserved |

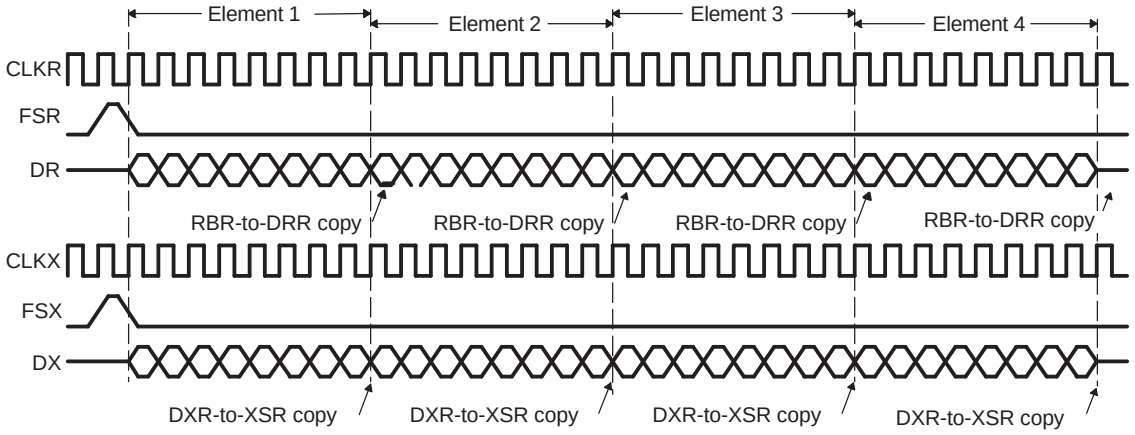
#### 2.3.5.4 Data Packing Using Frame Length and Element Length

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 9. In this case:

- (R/X)PHASE = 0, indicating a single-phase frame
- (R/X)FRLEN1 = 000 0011b, indicating a 4-element frame
- (R/X)WDLEN1 = 000b, indicating 8-bit elements

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

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

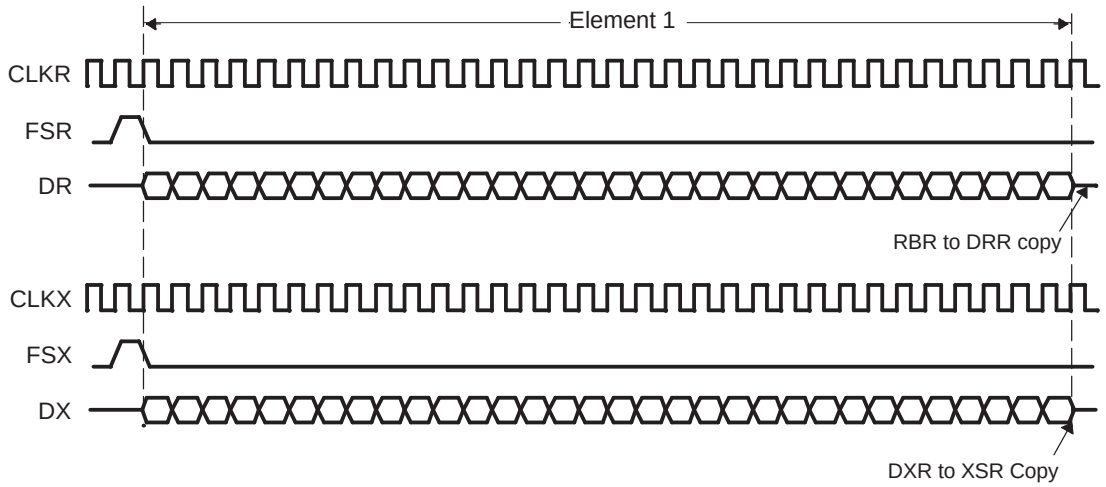



The example in Figure 9 can also be viewed as a data stream of a single-phase frame of one 32-bit data element, as shown in Figure 10. In this case:

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

In this example, one 32-bit data element is transferred to and from the ASP by the CPU or the EDMA controller. Thus, one read of DRR and one write of DXR is necessary for each frame. As a result, the number of transfers is one-fourth that of the previous example (Figure 9). This manipulation reduces the percentage of bus time required for serial port data movement.

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

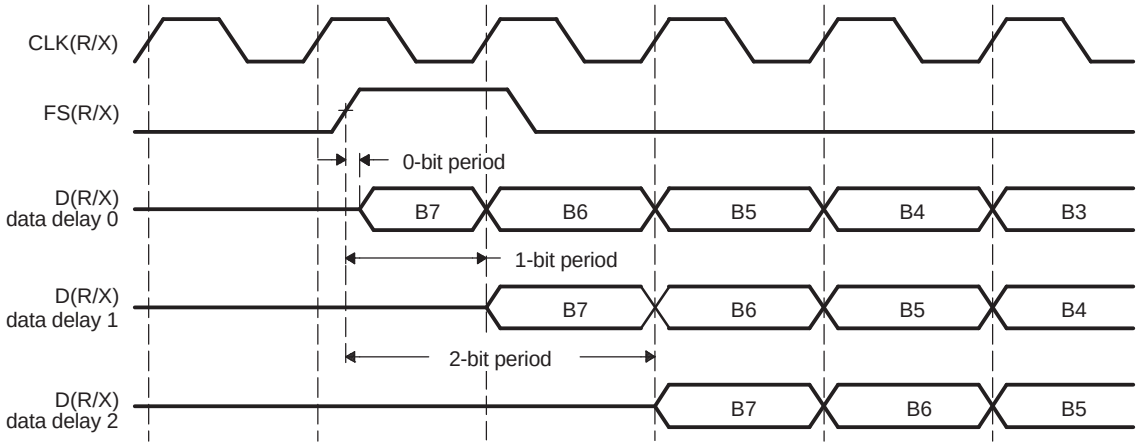


### 2.3.5.5 Data Delay: RDATDLY and XDATDLY

The start of a frame is defined by the first clock cycle in which frame synchronization is 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 (in RCR) and XDATDLY (in XCR) 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 shown in Figure 11. Typically, a 1-bit delay is selected because data often follows a 1-cycle active frame sync pulse.

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 a 0-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 in the XSR and DX. The transmitter then asynchronously detects the frame synchronization, FSX goes active, and it immediately starts driving the first bit to be transmitted on the DX pin.

**Figure 11. Data Delay**




### 2.3.5.6 Receive Data Justification and Sign Extension: RJUST

The RJUST bit in the serial port control register (SPCR) selects whether data in 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 10 and Table 11 summarize the effect that various values of RJUST have on example receive data.

**Table 10. Effect of RJUST Bit Values With 12-Bit Example Data ABCh**

| RJUST Bit in SPCR | Justification | Extension | Value in DRR |
|---|---|---|---|
| 00 | Right | Zero-fill MSBs | 0000 0ABCh |
| 01 | Right | Sign-extend MSBs | FFFF FABCh |
| 10 | Left | Zero-fill LSBs | ABC0 0000h |
| 11 | Reserved | Reserved | Reserved |

**Table 11. Effect of RJUST Bit Values With 20-Bit Example Data ABCDEh**

| RJUST Bit in SPCR | Justification | Extension | Value in DRR |
|---|---|---|---|
| 00 | Right | Zero-fill MSBs | 000A BCDEh |
| 01 | Right | Sign-extend MSBs | FFFA BCDEh |
| 10 | Left | Zero-fill LSBs | ABCD E000h |
| 11 | Reserved | Reserved | Reserved |

#### 2.3.5.7 32-Bit Bit Reversal: RWDREVRS, XWDREVRS

Normally all transfers are sent and received with the MSB first; however, you can reverse the receive/transmit bit ordering of a 32-bit element (LSB first) using the 32-bit reversal feature of the ASP by setting all of the following:

- (R/X)WDREVRS = 1 in the receive/transmit control register (RCR/XCR).
- (R/X)COMPAND = 01b in RCR/XCR.
- (R/X)WDLEN(1/2) = 101b in RCR/XCR to indicate 32-bit elements.

When you set the register fields as above, the bit ordering of the 32-bit element is reversed before being received by or sent from the serial port. If the (R/W)WDREVRS and (R/X)COMPAND fields are set as above, but the element size is not set to 32-bit, operation is undefined.

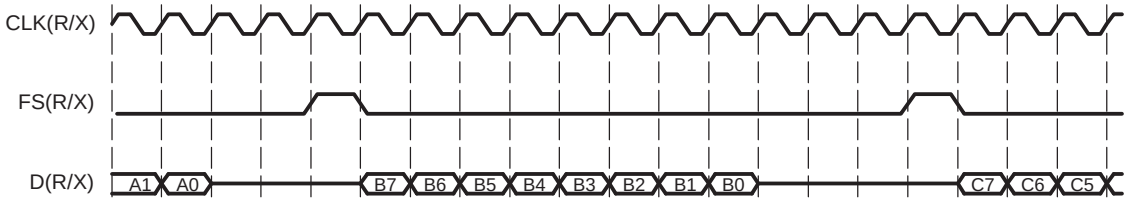
## 2.4 ASP Standard Operation

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 ASP 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 = 00h to 7Fh). The required serial element length is set in the (R/X)WDLEN1 field in the (R/X)CR. If a dual-phase frame is required for the transfer, RPHASE = 1 and each (R/X)FRLEN1/2 can be set to any value between 0h and 7Fh.

Figure 12 shows a single-phase data frame of one 8-bit element. Since the transfer is configured for a 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, use the following assumptions:

- (R/X)PHASE = 0, specifying a single-phase frame
- (R/X)FRLEN1 = 0b, specifying one element per frame
- (R/X)WDLEN1 = 000b, specifying eight bits per 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 that transmit data is clocked on the rising edge
- FS(R/X)P = 0, specifying that active (high) frame sync signals are used
- (R/X)DATDLY = 01b, specifying a 1-bit data delay

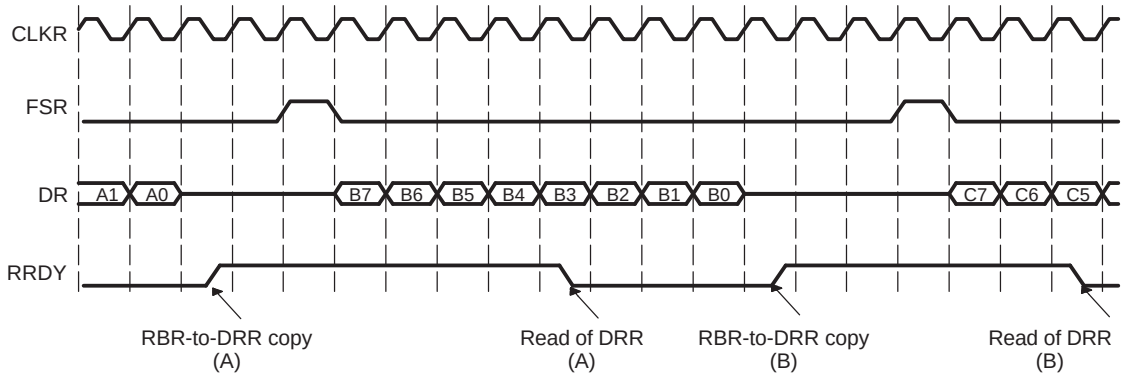
**Figure 12. ASP Standard Operation**



### 2.4.1 Receive Operation

Figure 13 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 receivers 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 EDMA controller. RRDY is deactivated when the DRR is read by the CPU or the EDMA controller. See also Section 2.8.1.2 and Section 2.9.1.

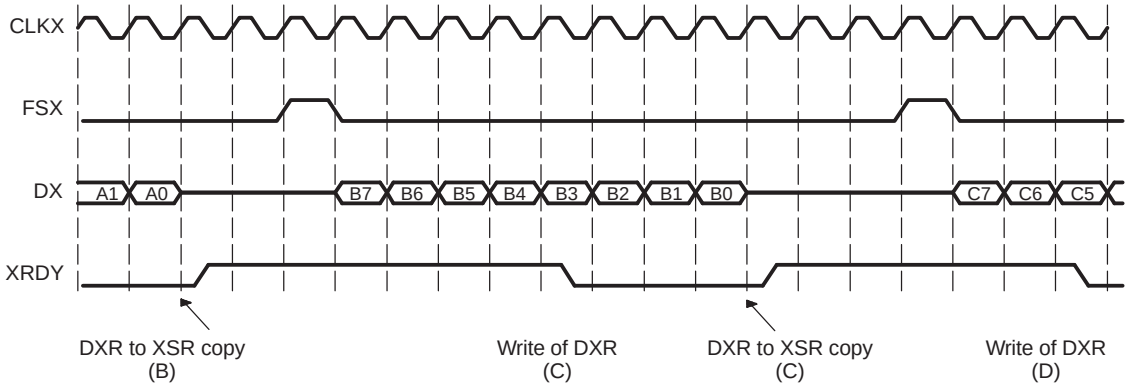
**Figure 13. Receive Operation**




### 2.4.2 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 after every DXR-to-XSR copy on the following falling edge of CLKX, indicating that the data transmit register (DXR) can be written with the next data to be transmitted. XRDY is deactivated when the DXR is written by the CPU or the EDMA controller. Figure 14 illustrates serial transmission. See Section 2.4.5.4 for information on transmit operation when the transmitter is pulled out of reset (XRST = 1). See also Section 2.8.1.3 and Section 2.9.2.

**Figure 14. Transmit Operation**



### 2.4.3 Maximum Frame Frequency

The frame frequency is determined by the following equation, which calculates the period between frame synchronization signals:

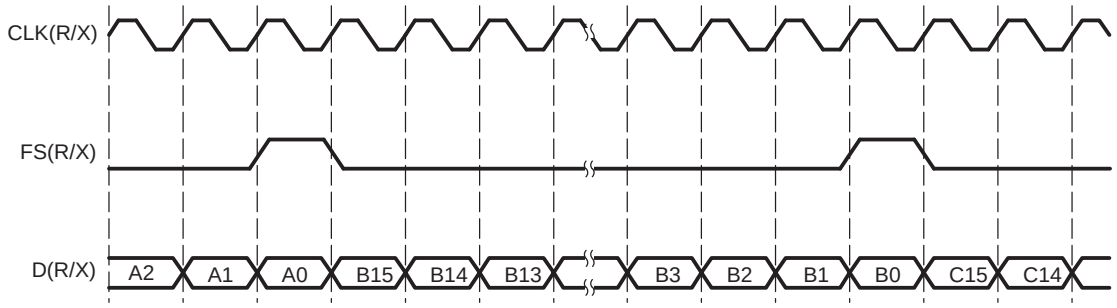
$$\text{Frame frequency} = \frac{\text{Bit-clock frequency}}{\text{Number of bit clocks between frame sync signals}}$$

The frame frequency may be increased by decreasing the time between frame synchronization signals in bit clocks (which is 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 0. The minimum time between frame synchronization pulses is the number of bits transferred per frame. This time also defines the maximum frame frequency, which is calculated by the following equation:

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

Figure 15 shows the ASP 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 15. Maximum Frame Frequency for Transmit and Receive**




**Note:** For (R/X)DATDLY = 0, the first bit of data transmitted is asynchronous to CLKX, as shown in Figure 11.

Maximum frame frequency may not be possible when the word length is 8-bits, depending on the the clock divide value CLKGDV. The CPU or EDMA may not be able to service serial port requests that occur as frequently as every 8 bit clocks. This situation can be resolved by allowing additional space between words or choosing a slower bit clock (larger CLKGDV value).

### 2.4.4 Frame Synchronization Ignore

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

#### *2.4.4.1 Frame Sync Ignore and Unexpected Frame Sync Pulses*

RFIG and XFIG are used to ignore unexpected internal or external frame sync pulses. Any frame sync pulse is considered unexpected if it occurs one or more bit clocks earlier than the programmed data delay from the end of the previous frame specified by ((R/X)DATDLY). 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), an 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 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 SPCR to 1, and reinitiates transmission of the current element that was aborted. When XFIG = 1, unexpected frame sync signals are ignored.

Figure 16 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 retransmitted after the appropriate data delay. This condition is a transmit synchronization error and thus sets the XSYNCERR bit. Synchronization errors are discussed in Section 2.4.5.2 and Section 2.4.5.5.

**Figure 16. Unexpected Frame Synchronization With (R/X)FIG = 0**

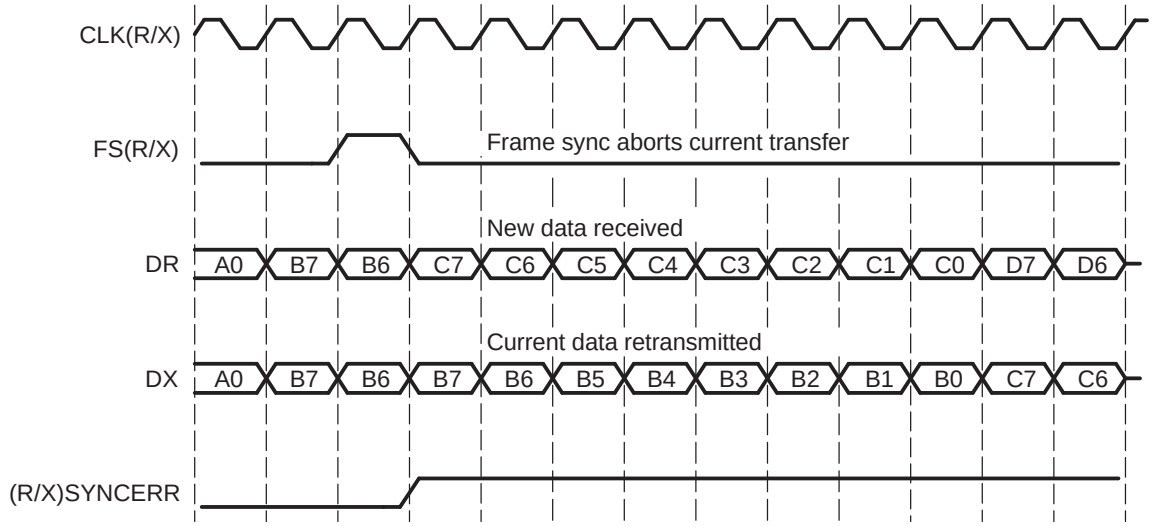
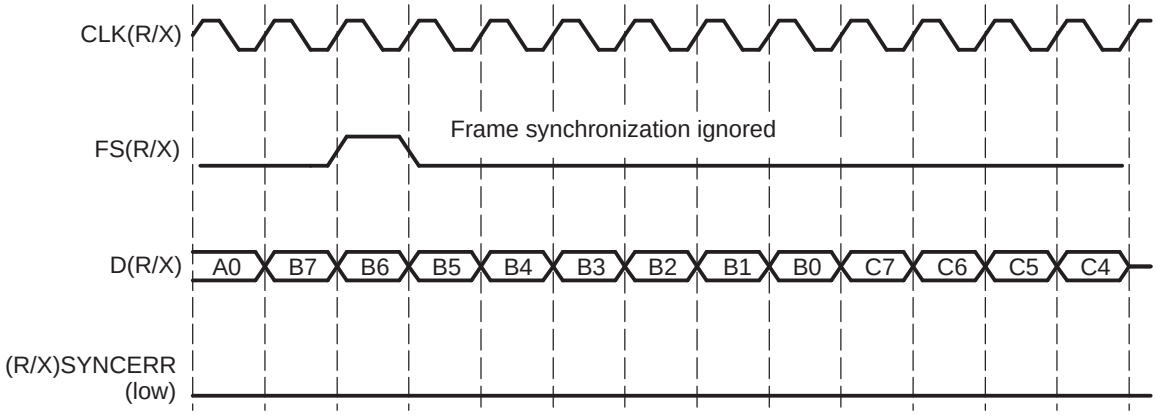



Figure 17 shows ASP operation when unexpected internal or external frame synchronization signals are ignored by setting (R/X)FIG = 1. Here, the transfer of element B is not affected by an unexpected frame synchronization.

**Figure 17. Unexpected Frame Synchronization With (R/X)FIG = 1**




### 2.4.4.2 Data Packing using Frame Sync Ignore Bits

Section 2.3.5.4 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 ASP operating at maximum packet frequency, as shown in Figure 18. 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 19 shows the ASP 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 needed to transfer four 8-bit blocks.

**Figure 18. Maximum Frame Frequency Operation With 8-Bit Data**

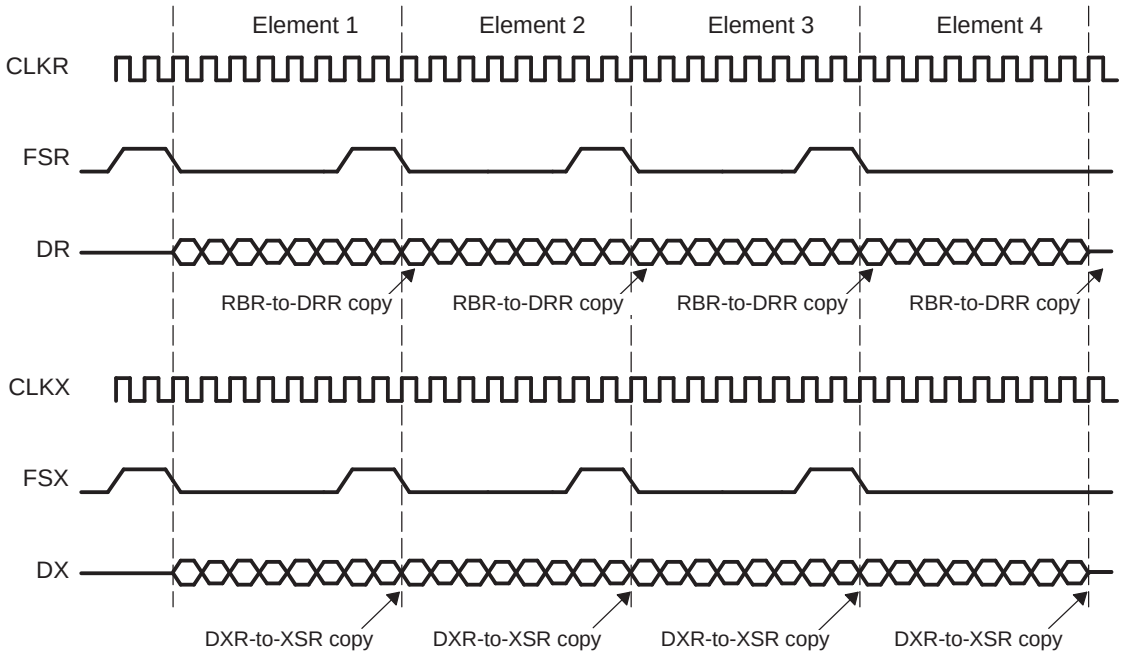
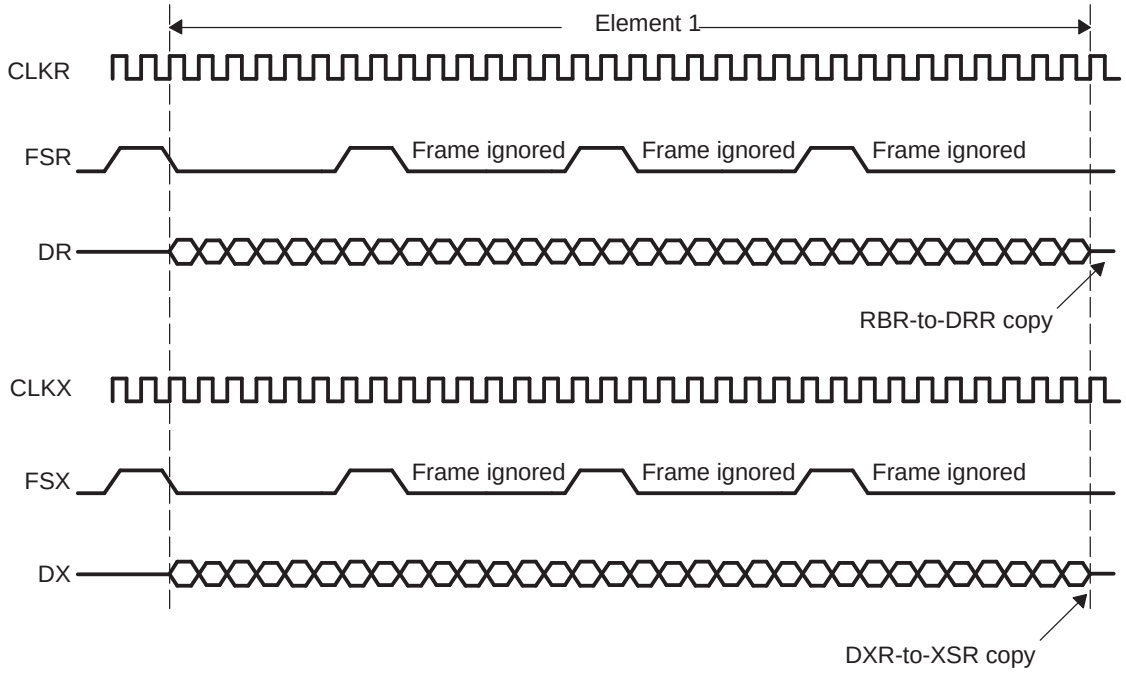


**Figure 19. Data Packing at Maximum Frame Frequency With (R/X)FIG = 1**




### 2.4.5 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 (XEMPTY = 0)
- Unexpected transmit frame synchronization (XSYNCERR = 1)

#### *2.4.5.1 Receive Overrun: RFULL*

RFULL = 1 in the serial port control register (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 (Figure 20). 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 EDMA 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 the DR pin is shifted into RSR, and the previous contents of RSR are 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 21.

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

- Reading DRR
- Resetting the receiver (RRST = 0) or the device

Another frame synchronization is required to restart the receiver.

Figure 20 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 21 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 20. Serial Port Receive Overrun**

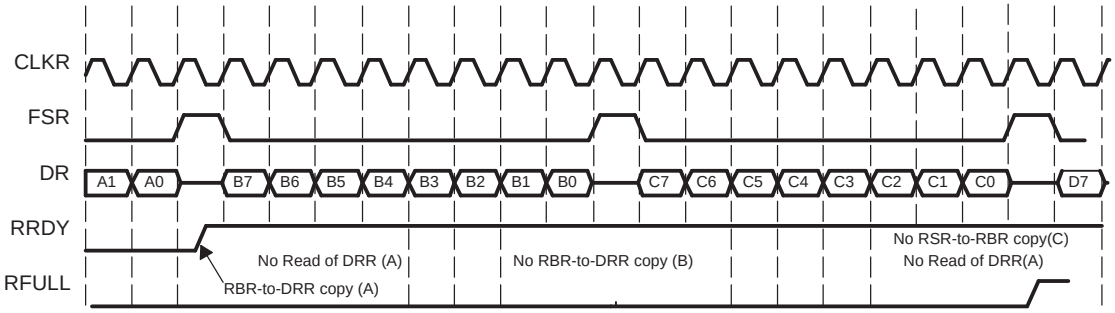
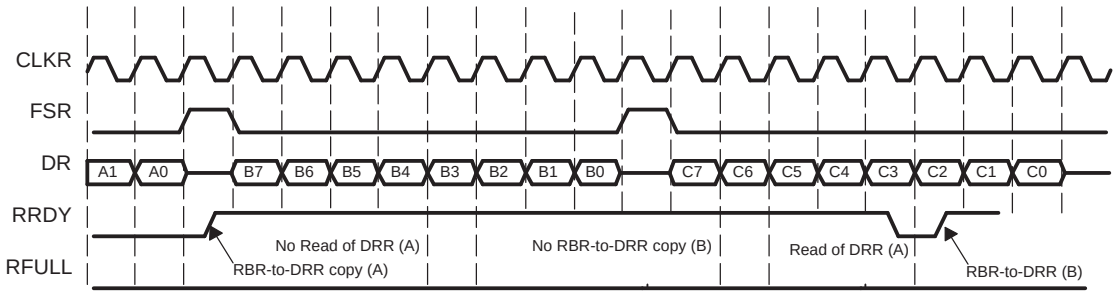



**Figure 21. Serial Port Receive Overrun Avoided**



### 2.4.5.2 Unexpected Receive Frame Synchronization: RSYNCERR

Figure 22 shows the decision tree that the receiver uses to handle all incoming frame synchronization pulses. The diagram assumes that the receiver has been activated (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 three cases can occur:

- Case 1: Unexpected FSR pulses with RFIG = 1. This case is discussed in Section 2.4.4.1 and shown in Figure 17. 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 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 16 for maximum packet frequency. Figure 23 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 SPCR is set. RSYNCERR can be cleared only by receiver reset or by writing a 0 to this bit in SPCR. If RINTM = 11b in SPCR, RSYNCERR drives the receive interrupt (RINT) to the CPU.

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

**Figure 22. Decision Tree Response to Receive Frame Synchronization Pulse**

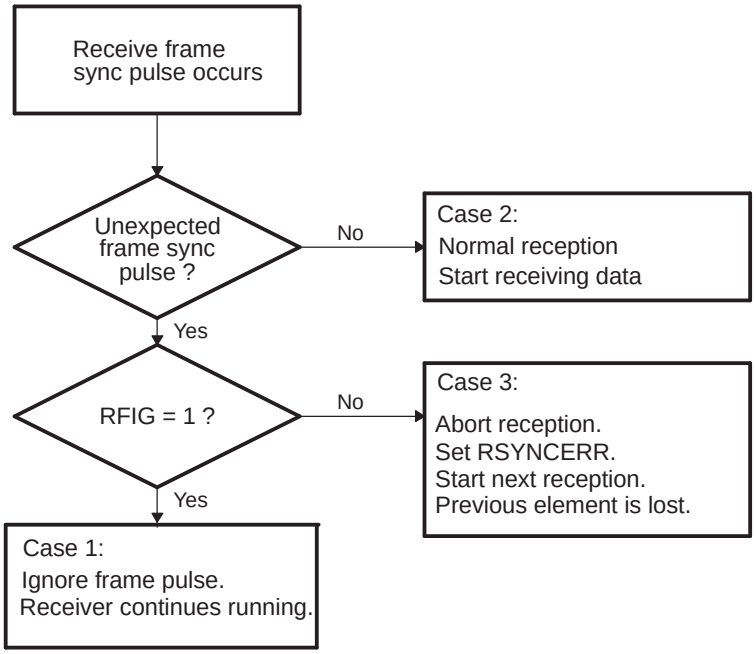
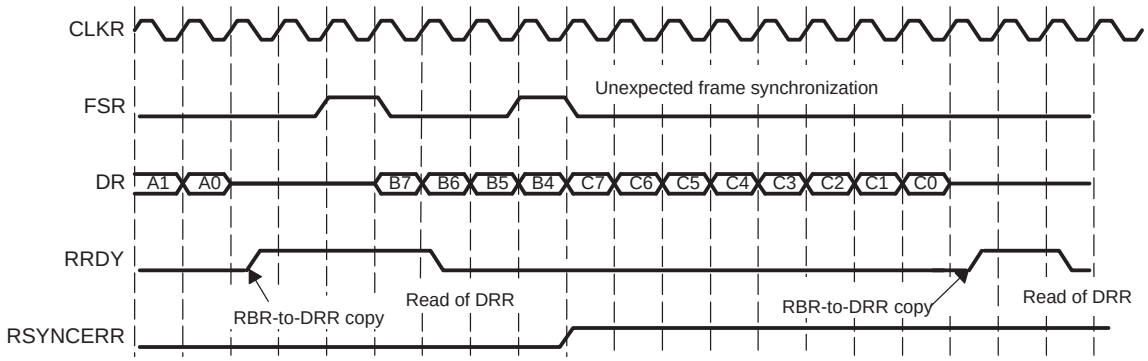


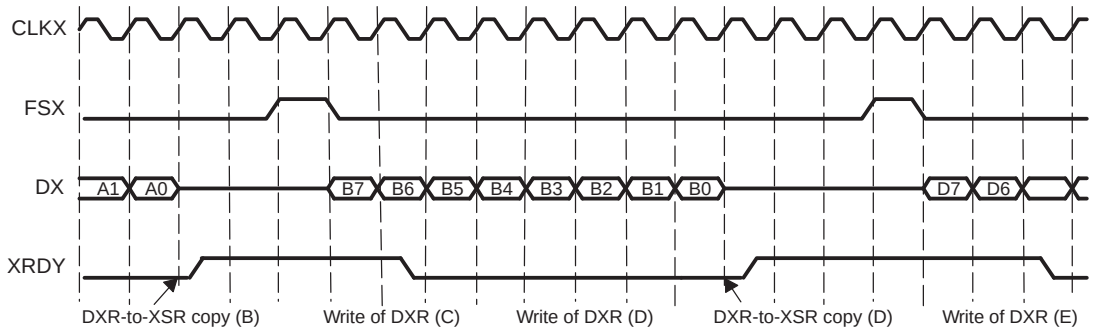
**Figure 23. Unexpected Receive Frame Synchronization Pulse**




### 2.4.5.3 Transmit With Data Overwrite

Figure 24 shows what happens if the data in DXR is overwritten before it is 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 EDMA controller can avoid overwriting by synchronizing data writes with XEVT. See also Section 2.8.1.3.

**Figure 24. Transmit With Data Overwrite**



### 2.4.5.4 Transmit Empty: XEMPTY

XEMPTY indicates whether the transmitter has experienced underflow. Either of the following conditions causes XEMPTY to become active (XEMPTY = 0):

- During transmission, DXR has not been loaded since the last DXR-to-XSR copy, and all bits of the data element in XSR have been shifted out on DX.
- The transmitter is reset (XRST = 0 or the 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 FSX (generated by an external device, or by the internal sample rate generator) until a new element is loaded into DXR by the CPU or the EDMA controller. XEMPTY is deactivated (XEMPTY = 1) when this new element in DXR is transferred to XSR. In the case when the FSX is generated by a DXR-to-XSR copy (FSXM = 1 in PCR and FSGM = 0 in SRGR), the ASP does not generate any new frame sync until new data is written to the DXR and a DXR-to-XSR copy occurs.

When the transmitter is taken out of reset (XRST = 1), it is in a transmit ready (XRDY = 1) and transmit empty (XEMPTY = 0) condition. If DXR is loaded by the CPU or the EDMA 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, 0s are output on DX.

Figure 25 shows a transmit underflow condition. After B is transmitted, B is retransmitted on DX if you fail to reload the DXR before the subsequent frame synchronization. Figure 26 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.

**Figure 25. Transmit Empty**

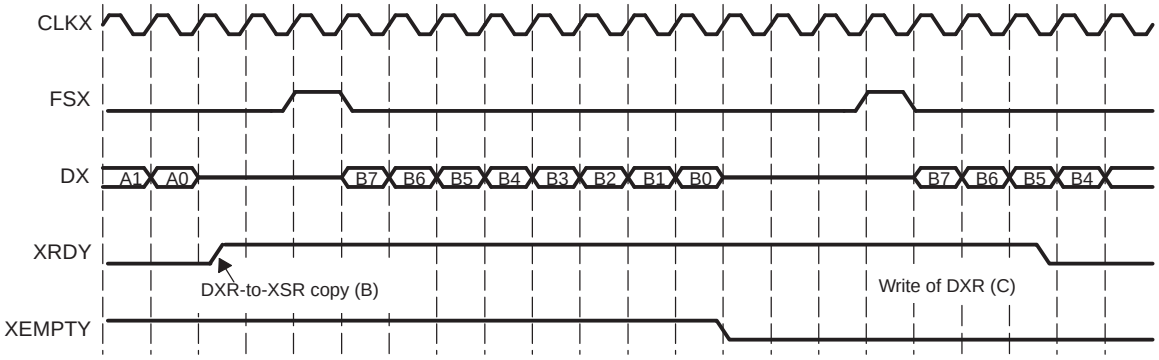
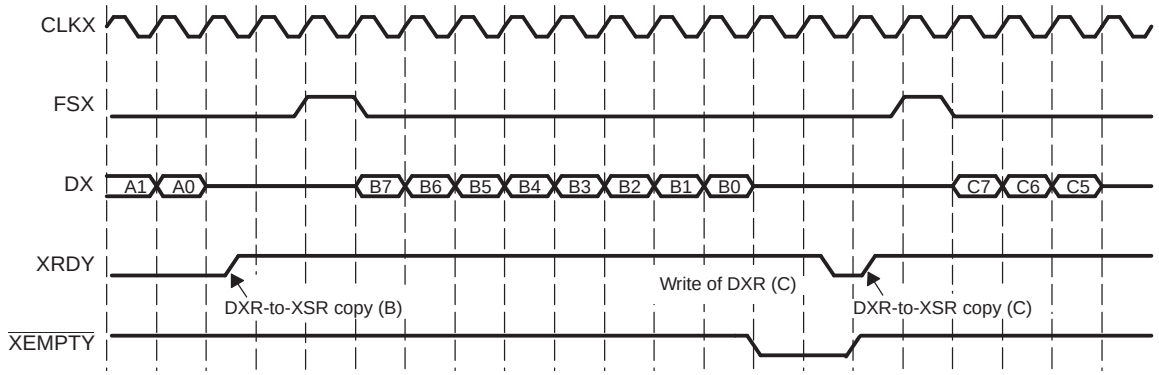



**Figure 26. Transmit Empty Avoided**



### 2.4.5.5 Unexpected Transmit Frame Synchronization: XSYNCERR

A transmit frame sync error (XSYNCERR) may occur the first time the transmitter is enabled (XRST = 1) after a device reset. To avoid this, after enabling the transmitter for the first time, the following procedure must be followed:

1. Wait for two CLKG cycles. The unexpected frame sync error (XSYNCERR), if any, occurs within this time period.
2. Disable the transmitter (XRST = 0). This clears any XSYNCERR.
3. Re-enable the transmitter (XRST = 1).

See also Section 2.7 for details on initialization procedure.

Figure 27 shows the decision tree that the transmitter uses to handle all incoming frame synchronization signals. The diagram assumes that the transmitter has been started (XRST = 1). An unexpected transmit frame sync pulse is defined as a sync pulse that occurs XDATDLY bit clocks earlier than the last transmitted bit of the previous frame. Any one of three cases can occur:

- Case 1: Unexpected FSX pulses with XFIG = 1. This case is discussed in Section 2.4.4.1 and shown in Figure 17. In this case, unexpected FSX pulses are ignored, and the transmission continues.
- Case 2: FSX pulses with normal serial port transmission. This situation is discussed in Section 2.4.2. There are two possible reasons for a transmit not to be in progress:
  - This FSX pulse is the first one to occur after XRST = 1.
  - The serial port is in the interpacket intervals. The programmed data delay (XDATDLY) may start during these interpacket 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.
- Case 3: Unexpected transmit frame synchronization with XFIG = 0. The case was shown in Figure 16 for frame synchronization with XFIG = 0 at maximum packet frequency. Figure 28 shows the case for normal operation of the serial port with interpacket intervals. In both cases, XSYNCERR in SPCR is set. XSYNCERR can be cleared only by transmitter reset or by writing a 0 to this bit in SPCR. If XINTM = 11b in SPCR, XSYNCERR drives the receive interrupt (XINT) to the CPU.

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

**Figure 27. Decision Tree Response to Transmit Frame Synchronization Pulse**

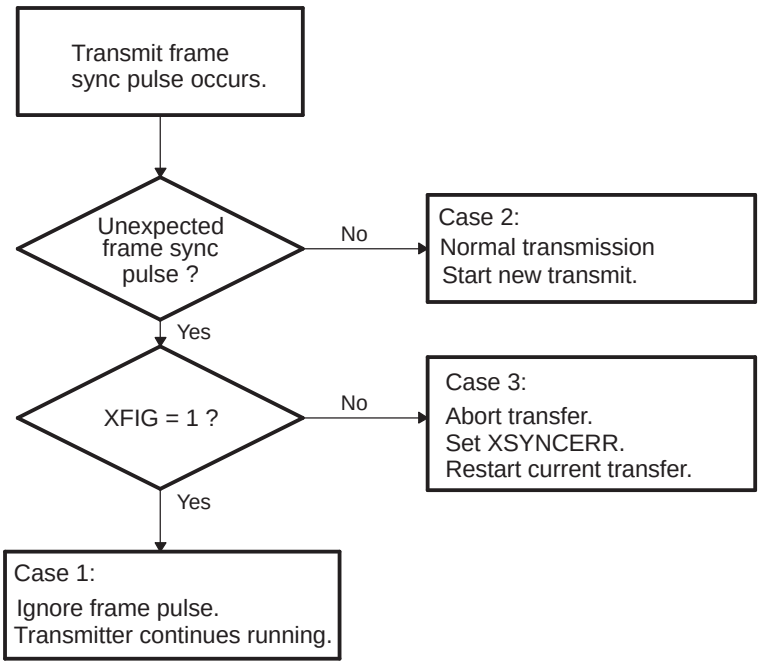
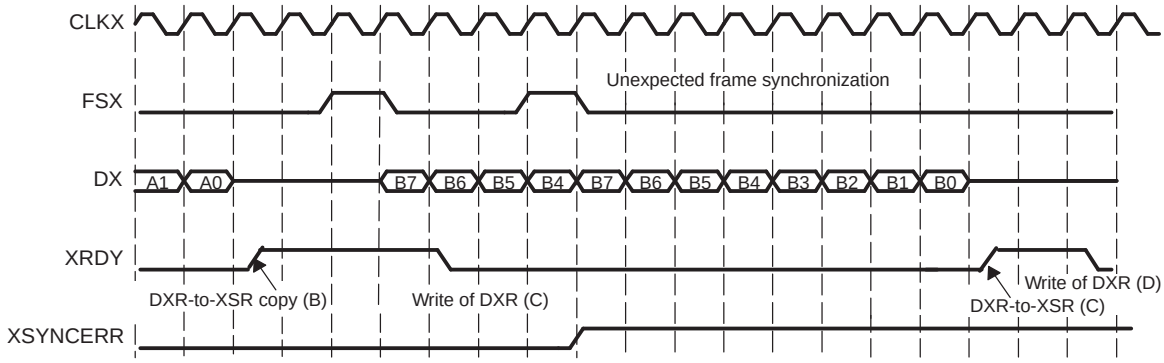


**Figure 28. Unexpected Transmit Frame Synchronization Pulse**



## 2.5 μ-Law/A-Law Companding Hardware Operation

Companding (compressing and expanding) 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 are set to the most positive or most negative value. Thus, for companding to work best here, the data transferred to and from the ASP via the CPU or the EDMA 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)WDLEN1/2 must be cleared 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 eight 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 is selected by appropriately setting (R/X)COMPAND in the (R/X)CR. Compression occurs during the process of copying data from DXR to XSR and expansion occurs from RBR to DRR, as shown in Figure 29.

For transmit data to be compressed, it should be 16-bit, left-justified data, such as LAW16, as shown in Figure 30. The value can be either 13 or 14 bits wide, depending on the companding law. This 16-bit data is aligned in DXR, as shown in Figure 31.

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 32-bit data by programming the RJUST bits in the serial port control register (SPCR), as shown in Table 12.

**Figure 29. Companding Flow**

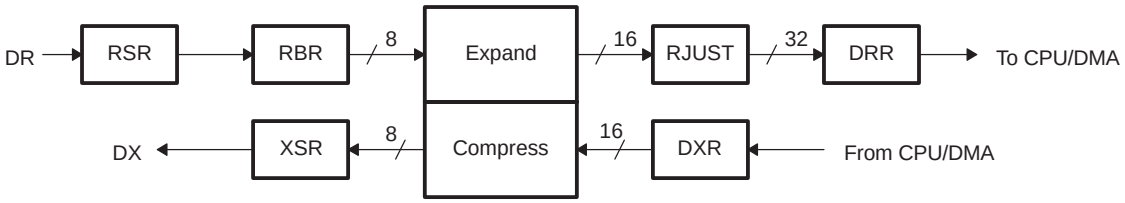



**Figure 30. Companding Data Formats**

| LAW16 | 15 | 2 | 1 | 0 |
|---|---|---|---|---|
| μ-Law | Value | | 0 | 0 |

| LAW16 | 15 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|
| A-Law | Value | | 0 | 0 | 0 |

**Figure 31. Transmit Data Companding Format in DXR**

| 31 | 16 | 15 | 0 |
|---|---|---|---|
| Don't care | | LAW16 | |

**Table 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 | | | |

### 2.5.1 Companding Internal Data

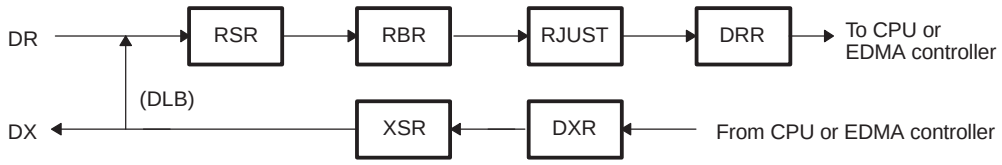
If the ASP is unused, the companding hardware can compand internal data. This hardware can be used to:

- 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 32 shows two methods by which the ASP 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, DRR and DXR are internally connected through the companding logic. Values from DXR are compressed as determined by the XCOMPAND bits and then expanded as determined by the RCOMPAND bits. RRDY and XRDY bits are not set. However, data is available in DRR four internal ASP 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 EDMA controller to control the flow of data.
- **DLB:** The ASP is enabled in digital loopback (DLB) mode with companding appropriately enabled by the RCOMPAND and XCOMPAND bits. 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 EDMA controller to these conversions, respectively. Here, the time for this companding depends on the serial bit rate selected.

**Figure 32. Companding of Internal Data**




### 2.5.2 Bit Ordering

Normally, all transfers on the ASP 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 (R/X)CR, the bit ordering of 8-bit elements is reversed (LSB first) before being sent to the serial port. Like the companding feature, this feature is enabled only if the appropriate (R/X)WDLEN1/2 bit is cleared to 0, indicating that 8-bit elements are to be serially transferred. A 32-bit bit reversal feature is also available, see Section 2.3.5.7

## 2.6 Resetting the Serial Port: RRST, XRST, GRST, and RESET

**Device reset or ASP reset:** When the ASP is reset by device reset or ASP reset, the state machine is reset to its initial state. All counters and status bits are reset. This includes the receive status bits RFULL, RRDY, and RSYNCERR, and the transmit status bits XEMPTY, XRDY, and XSYNCERR.

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

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

Table 13 shows the state of the ASP pins when the serial port is reset by these methods.

**Table 13. Reset State of ASP Pins**

| Pin | Direction | Device Reset ($\overline{\text{RESET}}$ = 0) | ASP Reset |
|---|---|---|---|
| | | | **Receiver Reset (RRST = 0 and 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 |
| | | | **Transmitter Reset (XRST = 0 and GRST = 1)** |
| DX | O/Z | 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 |

### 2.6.1 Software Reset Considerations

**ASP reset:** When the receiver and transmitter reset bits, RRST and XRST, are written with 0, the respective portions of the ASP are reset and activity in the corresponding section stops. All input-only pins, such as DR, 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 GRST = 1. The DX pin is 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 GRST. The sample rate generator should only be reset when not being used by the transmitter ro the receiver. 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 (GRST = 1), FSR and FSX are in an inactive state when RRST = 0 and XRST = 0, respectively, even if they are outputs driven by FSG. This ensures that when only one portion of the ASP is in reset, the other portion can continue operation when 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, GRST, is written with 0.

**Emulator software reset:** In the event of an emulator software reset initiated from the ARM side, the ASP register values are reset to their default values. In the event of an emulator software reset initiated from the CPU side, the ASP register values are unaffected.

### 2.6.2 Hardware Reset Considerations

**Device reset:** When the ASP 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. When the device is pulled out of reset, the serial port remains in the reset condition (RRST = XRST = FRST = GRST = 0).

## 2.7 ASP Initialization Procedure

The ASP initialization procedure varies depending on the specific system setup. Section 2.7.1 provides a general initialization sequence.

The transmitter and the receiver of the ASP can operate independently from each other. Therefore, they can be placed in or taken out of reset individually by modifying only the desired bit in the registers without disrupting the other portion. The steps in the following sections discuss the initialization procedure for taking both the transmitter and the receiver out of reset. To initialize only one portion, configure only the portion desired.

The ASP internal sample rate generator and internal frame sync generator are shared between the transmitter and the receiver. Table 14 and Table 15 describe their usage base upon the clock and frame sync configurations of the receiver and transmitter, respectively.

**Table 14. Receiver Clock and Frame Configurations**

| CLKR Source | FSR Source | Comment on Configuration |
|---|---|---|
| Internal | Internal | The ASP internal sample rate generator and internal frame sync generator are used by the receiver. |
| External | Internal | The ASP internal sample rate generator and internal frame sync generator are used by the receiver. |
| Internal | External | The ASP internal sample rate generator is used but the internal frame sync generator is not used by the receiver. |
| External | External | The ASP internal sample rate generator and internal frame sync generator are not used by the receiver. |

**Table 15. Transmitter Clock and Frame Configurations**

| CLKX Source | FSX Source | Comment on Configuration |
|---|---|---|
| Internal | Internal | The ASP internal sample rate generator is used by the transmitter. The transmitter can generate frame sync FSX in one of two ways. First, it can generate FSX by using the internal frame sync generator (FSGM = 1). Alternatively, it can generate FSX upon each DXR-to-XSR copy (FSGM = 0). In this case, the internal frame sync generator can be kept in reset (FRST = 0) if it is not used by the receiver. You can follow the general initialization sequence in Section 2.7.1. |
| External | Internal | The ASP internal sample rate generator and internal frame sync generator are not used by the transmitter. This configuration is only valid with FSGM = 0 where the ASP transmitter generates FSX upon each DXR-to-XSR copy. You can follow the general initialization sequence in Section 2.7.1. |
| Internal | External | The ASP internal sample rate generator is used by the transmitter but the internal frame sync generator is not. |
| External | External | The ASP internal sample rate generator and internal frame sync generator are not used by the transmitter. |

### 2.7.1 General Initialization Procedure

This section provides the general initialization procedure.

1. With the ASP still in reset (Power and Sleep Controller (PSC) in the default state), perform the necessary device pin multiplexing setup (see the device-specific data manual).
2. Ensure that no portion of the ASP is using the internal sample rate generator signal CLKG and the internal frame sync generator signal FSG (GRST = FRST = 0). The respective portion of the ASP needs to be in reset (XRST = 0 and/or RRST = 0).
3. Program the control registers as required. Ensure the internal sample rate generator and the internal frame sync generator are still in reset (GRST = FRST = 0). Also ensure the respective portion of the ASP is still in reset in this step (XRST = 0 and/or RRST = 0).
4. Wait for proper internal synchronization. If the external device provides the bit clock, wait for two CLKR or CLKX cycles. If the ASP generates the bit clock as a clock master, wait for two CLKSRG cycles. In this case, the clock source to the sample rate generator (CLKSRG) is selected by the CLKSM bit in SRGR and the SCLKME bit in PCR.
5. Skip this step if the bit clock is provided by the external device. This step only applies if the ASP is the bit clock master and the internal sample rate generator is used.
   a. Start the sample rate generator by setting the GRST bit to 1. Wait two CLKG bit clocks for synchronization. CLKG is the output of the sample rate generator.
   b. On the next rising edge of CLKSRG, CLKG transitions to 1 and starts clocking with a frequency equal to 1/(CLKGDV + 1) of the sample rate generator source clock CLKSRG.
6. Skip this step if the transmitter is not used. If the transmitter is used, a transmit sync error (XSYNCERR) may occur when it is enabled for the first time after device reset. The purpose of this step is to clear any potential XSYNCERR that occurs on the transmitter at this time:
   a. Set the XRST bit to 1 to enable the transmitter.
   b. Wait for any unexpected frame sync error to occur. If the external device provides the bit clock, wait for two CLKR or CLKX cycles. If the ASP generates the bit clock as a clock master, wait for two CLKG cycles. The unexpected frame sync error (XSYNCERR), if any, occurs within this time period.
   c. Disable the transmitter (XRST = 0). This clears any outstanding XSYNCERR.
7. Setup data acquisition as required:
   a. If the EDMA is used to service the ASP, setup data acquisition as desired and start the EDMA in this step, before the ASP is taken out of reset.
   b. If CPU interrupt is used to service the ASP, enable the transmit and/or receive interrupt as required.
   c. If CPU polling is used to service the ASP, no action is required in this step.
8. Set the XRST bit and/or the RRST bit to 1 to enable the corresponding section of the ASP. The ASP is now ready to transmit and/or receive.
   a. If the EDMA is used to service the ASP, it services the ASP automatically upon receiving the XEVT and/or REVT.
   b. If CPU interrupt is used to service the ASP, the interrupt service routine is automatically entered upon receiving the XINT and/or RINT.
   c. If CPU polling is used to service the ASP, it can do so now by polling the XRDY and/or RRDY bit.
9. If the internal frame sync generator is used (FSGM = 1), proceed to the additional steps to turn on the internal frame sync generator. Initialization is complete if any one of the following is true:
   a. The external device generates frame sync FSX and/or FSR. The ASP is now ready to transmit and/or receive upon receiving external frame sync.
   b. The ASP generates transmit frame sync FSX upon each DXR-to-XSR copy. The internal frame sync generator is not used (FSGM = 0).

   The following steps to turn on the internal frame sync generator apply only if FSGM = 1:
10. Skip this step if the transmitter is not used. If the transmitter is used, ensure that DXR is serviced before you start the internal frame sync generator. You can do so by checking XEMPTY = 1 (XSR is not empty) in SPCR.
11. Set the FRST bit to 1 to start the internal frame sync generator. The internal frame sync signal FSG is generated on a CLKG active edge after 7 to 8 CLKG clocks have elapsed.

### 2.7.2 Special Case: External Device is the Transmit Frame Master

Care must be taken if the transmitter expects a frame sync from an external device. After the transmitter comes out of reset (XRST = 1), it waits for a frame sync from the external device. If the first frame sync arrives very shortly after the transmitter is enabled, the CPU or EDMA controller may not have a chance to service the data transmit register (DXR). In this case, the transmitter shifts out the default data in the transmit shift register (XSR) instead of the desired value, which has not yet arrived in DXR. This causes problems in some applications, as the first data element in the frame is invalid. The data stream appears element-shifted (the first data word may appear in the second channel instead of the first).

To ensure proper operation when the external device is the frame master, you must assure that DXR is already serviced with the first word when a frame sync occurs. To do so, you can keep the transmitter in reset until the first frame sync is detected. Software is setup such that it will only take the transmitter out of reset (XRST = 1) promptly after detecting the first frame sync. This assures that the transmitter does not begin data transfers at the data pin during the first frame sync period. This also provides almost an entire frame period for the ARM to service DXR with the first word before the second frame sync occurs. The transmitter only begins data transfers upon receiving the second frame sync. At this point, DXR is already serviced with the first word.

#### *2.7.2.1 How to Detect First Frame Sync*

Although the ASP is capable of generating an interrupt to the CPU upon the detection of frame synchronization (XINTM = 2h and/or RINTM = 2h in the serial port control register (SPCR)), the ASP requires the associated portion (receiver/transmitter) of the ASP to be out of reset in order for the interrupt to be generated. Therefore, instead of directly using the ASP interrupt to detect the first frame sync, you can use the GPIO peripheral. This can be achieved by connecting the frame sync signal to a GPIO pin. Software can either poll the GPIO pin to detect the first frame sync or program the GPIO peripheral to generate an interrupt to the CPU upon detecting the first frame sync edge. For more information on the GPIO peripheral, see the *TMS320DM357 DMSoC General-Purpose Input/Output (GPIO) User's Guide* (SPRUG31).

The following are some recommended GPIO pin(s) on the DMSoC that you can use to detect the first ASP external frame sync:

- **GPIO pin located near the ASP pins.** Connect the external frame sync to both the ASP FSX/FSR pin(s) and the dedicated GPIO pin.
- **GPIO pin multiplexed with the ASP FSX signal.** Note that on the DM357 DMSoC, the GPIO pins (of the GPIO peripheral) are multiplexed with the ASP pins. Software can program the DMSoC pin multiplexing register (PINMUX) to default these pins to the GPIO function, and only switch them to the ASP function upon detecting the first frame sync. This method is only recommended if the external device is both the frame sync and clock master; that is, the external device drives both the FSX and CLKX signals. This method is not recommended if the ASP is the clock master (driving CLKX and/or CLKR), as the “on-the-fly” pin multiplexed switching can cause a glitch on the CLKX/CLKR pin. For more details on pin multiplexing, see the device-specific data manual.

#### *2.7.2.2 Initialization Procedure When External Device is Frame Sync Master*

The initialization procedure assumes the following:

- Using a GPIO pin multiplexed with the ASP FSX signal. If a dedicated GPIO pin is used instead, skip step 1 and step 8b.
- Software polls the GPIO pin to detect the first frame sync. If the GPIO interrupt is used instead to detect the first frame sync, step 8 can be performed within an interrupt service routine (ISR).

1. On the DM357 DMSoC, the GPIO and ASP signals are multiplexed together. Start by programming the pin multiplexing register (PINMUX) to select the GPIO function on the GPIO/ASP multliplexed pins. Program the GPIO peripheral so that these pins function as GPIO inputs.
2. Ensure that no portion of the ASP is using the internal sample rate generator signal CLKG and the internal frame sync generator signal FSG (GRST = FRST = 0 in SPCR). The respective portion of the ASP needs to be in reset (XRST = 0 and/or RRST = 0 in SPCR).

3. Program the sample rate generator register (SRGR) and other control registers as required. Ensure the internal sample rate generator and the internal frame sync generator are still in reset (GRST = FRST = 0 in SPCR). Also ensure the respective portion of the ASP is still in reset in this step (XRST = 0 and/or RRST = 0 in SPCR).
4. Wait for proper ASP internal synchronization:
   a. If the external device provides the bit clock, wait for two CLKR or CLKX cycles. Skip step 5.
   b. If the ASP generates the bit clock as a clock master, wait for two CLKSRG cycles. In this case, the clock source to the sample rate generator (CLKSRG) is selected by the CLKSM bit in SRGR.
5. Skip this step if the bit clock is provided by the external device. This step only applies if the ASP is the bit clock master and the internal sample rate generator is used.
   a. Start the sample rate generator by setting the GRST bit in SPCR to 1. Wait two CLKG bit clocks for synchronization. CLKG is the output of the sample rate generator.
   b. On the next rising edge of CLKSRG, CLKG transitions to 1 and starts clocking with a frequency equal to 1/(CLKGDV + 1) of the sample rate generator source clock CLKSRG.
6. A transmit sync error (XSYNCERR) may occur when it is enabled for the first time after device reset. The purpose of this step is to clear any potential XSYNCERR that occurs on the transmitter at this time:
   a. Set the XRST bit in SPCR to 1 to enable the transmitter.
   b. Wait for any unexpected frame sync error to occur. If the external device provides the bit clock, wait for two CLKR or CLKX cycles. If the ASP generates the bit clock as a clock master, wait for two CLKG cycles. The unexpected frame sync error (XSYNCERR), if any, occurs within this time period.
   c. Disable the transmitter (XRST = 0). This clears any outstanding XSYNCERR.
7. Setup data acquisition as required:
   a. If the EDMA controller is used to service the ASP, setup data acquisition as desired and start the EDMA controller in this step, before the ASP is taken out of reset.
   b. If the CPU interrupt is used to service the ASP, no action is required in this step.
   c. If CPU polling is used to service the ASP, no action is required in this step.
8. Poll the GPIO pin (through reading the appropriate registers in the GPIO peripheral) to detect the first transmit frame sync from the external device. Upon detection of the first frame sync, perform the following in this order:
   a. Set the XRST bit and/or the RRST bit to 1 to enable the respective portion of the ASP. The ASP is now ready to transmit and/or receive.
   b. Program PINMUX to switch the GPIO/ASP multiplexed pins to the ASP function.
9. Service the ASP:
   a. If CPU polling is used to service the ASP in normal operations, it can do so upon exit from the ISR.
   b. If the CPU interrupt is used to service the ASP in normal operations, upon XRDY interrupt service routine is entered. The ISR should be setup to verify that XRDY = 1 and service the ASP accordingly.
   c. If the EDMA controller is used to service the ASP in normal operations, it services the ASP automatically upon receiving the XEVT and/or REVT.
10. Upon detection of the second frame sync, DXR is already serviced and the transmitter is ready to transmit the valid data. The receiver is also serviced properly by the ARM.

## 2.8 Interrupt Support

The ASP can send both receive and transmit interrupts to the ARM interrupts controllers. For more information on the interrupt controllers, see the *TMS320DM357 DMSoC ARM Subsystem Reference Guide* (SPRUG25).

### 2.8.1 Interrupt Events and Requests

The RRDY and XRDY bits in SPCR indicate the ready state of the ASP receiver and transmitter, respectively. Writes and reads from the serial port can be synchronized by any of the following methods:

- Polling RRDY and XRDY bits
- Using the events sent to the EDMA controller (REVT and XEVT)
- Using the interrupts to the CPU (RINT and XINT) that the events generate

Reading DRR and writing to DXR affects RRDY and XRDY, respectively.

#### 2.8.1.1 Interrupt Events: RINT and XINT

The receive interrupt (RINT) and transmit interrupt (XINT) signals inform the ARM CPU of changes to the serial port status. Three options exist for configuring these interrupts. These options are set by the receive/transmit interrupt mode bits (RINTM and XINTM) in 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 SPCR.
- (R/X)INTM = 10b. Interrupt on detection of frame synchronization pulses. The associated portion (receiver/transmitter) of the ASP must be out of reset.
- (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 Section 2.4.5.2 and Section 2.4.5.5 for more details on synchronization error.

#### 2.8.1.2 Receive Ready Status: RINT and RRDY

RRDY = 1 indicates that the RBR contents have been copied to DRR and that the data can now be read by either the CPU or the EDMA controller. Once that data has been read by either the CPU, RRDY is cleared to 0. Also, at device reset or serial port receiver reset (RRST = 0), the RRDY bit is cleared to 0 to indicate that no data has been received and loaded into DRR. RRDY directly drives the ASP receive interrupt (RINT) to the CPU if RINTM = 00b (default value) in SPCR.

#### 2.8.1.3 Transmit Ready Status: 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 non-reset (XRST transitions from 0 to 1), XRDY also transitions from 0 to 1 indicating that DXR is ready for new data. Once new data is loaded by the CPU, the XRDY bit is cleared to 0. However, once this data is copied from DXR to XSR, the XRDY bit transitions again from 0 to 1. The CPU can write to DXR although XSR has not yet been shifted out on DX. XRDY directly drives the ASP transmit interrupt (XINT) to the CPU if XINTM = 00b (default value) in SPCR.

**Note:** If the polling method is used to service the transmitter, the CPU should wait for one ASP bit clock (CLKX) before polling again to write the next element in DXR. This is because XRDY transitions occur based on bit clock and not CPU clock. The CPU clock is much faster and can cause false XRDY status, leading to data errors due to over-writes.

### 2.8.2 Interrupt Multiplexing

The RINT and XINT interrupts are not multiplexed. Each interrupt has a dedicated path to the ARM interrupt controllers.

## 2.9 EDMA Event Support

### 2.9.1 Receive Ready Status: REVT and RRDY

RRDY = 1 indicates that the RBR contents have been copied to DRR and that the data can now be read by the EDMA controller. Once that data has been read by the EDMA controller, RRDY is cleared to 0. Also, at device reset or serial port receiver reset (RRST = 0), the RRDY bit is cleared to 0 to indicate that no data has been received and loaded into DRR. RRDY directly drives the ASP receive event to the EDMA controller (via REVT).

For detailed information on using the EDMA to read or write to the ASP, see the *TMS320DM357 DMSoC Enhanced DMA (EDMA) Controller User's Guide* (SPRUG34).

### 2.9.2 Transmit Ready Status: XEVT 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 non-reset (XRST transitions from 0 to 1), XRDY also transitions from 0 to 1 indicating that DXR is ready for new data. Once new data is loaded by the EDMA controller, the XRDY bit is cleared to 0. However, once this data is copied from DXR to XSR, the XRDY bit transitions again from 0 to 1. The EDMA controller can write to DXR although XSR has not yet been shifted out on DX. XRDY directly drives the transmit synchronization event to the EDMA controller (via XEVT).

For detailed information on using the EDMA to read or write to the ASP, see the *TMS320DM357 DMSoC Enhanced DMA (EDMA) Controller User's Guide* (SPRUG34).

**Note:** If the polling method is used to service the transmitter, the CPU should wait for one ASP bit clock (CLKX) before polling again to write the next element in DXR. This is because XRDY transitions occur based on bit clock and not CPU clock. The CPU clock is much faster and can cause false XRDY status, leading to data errors due to over-writes.

## 2.10 Power Management

The ASP can be placed in reduced power modes to conserve power during periods of low activity. The power management of the peripheral is controlled by the processor Power and Sleep Controller (PSC). The PSC acts as a master controller for power management for all of the peripherals on the device.

In order for the ASP to be placed in power-down mode by the PSC, ensure that the XRDY and RRDY flags in the serial port control register (SPCR) are cleared by performing the following steps:

1. Place the ASP in reset by clearing the XRST, RRST, FRST, and GRST bits to 0 in SPCR.
2. If EDMA is being used to service the transmitter and/or the receiver, disable the associated EDMA channels. For detailed information on EDMA usage, see the *TMS320DM357 DMSoC Enhanced DMA (EDMA) Controller User's Guide* (SPRUG34).
3. Switch the ASP clocks and frames to internal clock source:
   a. Set the CLKSM and FSGM bits to 1 in the sample rate generator register (SRGR).
   b. Set the CLKXM, CLKRM, FSXM, and FSRM bits to 1 in the pin control register (PCR).
   c. Clear the SCLKME bit to 0 in PCR.
4. Bring the ASP out of reset by setting the XRST, RRST, and GRST bits to 1 in SPCR.
5. Wait for two CLKSRG cycles for proper internal synchronization.
6. Write a dummy data value to the data transmit register (DXR) in order to clear the first XRDY flag.
7. Wait for at least one ASP bit clock, since once the first dummy data value is internally copied from DXR to XSR, the XRDY flag transitions again from 0 to 1.
8. Write a second dummy data value to DXR in order to clear the second XRDY flag.
9. Check the RRDY flag in SPCR and if set to 1, read the data receive register (DRR) and discard the data to clear the RRDY flag.

10. Place the ASP in power-down mode by issuing the proper PSC commands. For detailed information on power management procedures using the PSC, see the *TMS320DM357 DMSoC ARM Subsystem Reference Guide* (SPRUG25).

**Note:** After waking up the ASP from a power-down mode using the proper PSC commands, remember to reconfigure the SPCR, SRGR, and PCR registers to the clock and frame combination that they were in before entering the power-down sequence and discard the two dummy data values that were used to clear the XRDY flags. If EDMA is used, re-enable the corresponding EDMA channels.

## 2.11 Emulation Considerations

The FREE and SOFT bits are special emulation bits in the serial port control register (SPCR) that determine the state of the ASP when an emulation suspend event occurs in the emulator. An emulation suspend event corresponds to any type of emulator access to the ARM, such as a hardware or software breakpoint, a probe point, or a printf instruction.

Table 16 shows the effects of the FREE and SOFT bits on the response of the ASP to emulation suspend events.

**Table 16. ASP Emulation Modes Selectable With the FREE and SOFT Bits of SPCR**

| FREE | SOFT | ASP Emulation Mode |
|---|---|---|
| 0 | 0 | Immediate stop mode (reset condition). The transmitter and receiver stop immediately in response to an emulation suspend event. |
| 0 | 1 | Soft stop mode. When an emulation suspend event occurs, the transmitter stops after completion of the current word. The receiver is not affected. |
| 1 | 0 or 1 | Free run mode. The transmitter and receiver continue to run when an emulation suspend event occurs. |

# 3 Registers

Table 17 lists the memory-mapped registers for the ASP. See the device-specific data manual for the memory address of these registers. All other register offset addresses not listed in Table 17 should be considerd as reserved locations and the register contents should not be modified.

The ASP control registers are accessible by the ARM CPU. You should halt the ASP before making changes to the serial port control register (SPCR), receive control register (RCR), transmit control register (XCR), and pin control register (PCR). Changes made to these registers without halting the ASP could result in an undefined state.

**Table 17. ASP Registers**

| Offset | Acronym | Register Name | Section |
|---|---|---|---|
| - | RBR[1] | Receive buffer register | - |
| - | RSR[1] | Receive shift register | - |
| - | XSR[1] | Transmit shift register | - |
| 0h | DRR[2][3] | Data receive register | Section 3.1 |
| 4h | DXR[3] | Data transmit register | Section 3.2 |
| 8h | SPCR | Serial port control register | Section 3.3 |
| Ch | RCR | Receive control register | Section 3.4 |
| 10h | XCR | Transmit control register | Section 3.5 |
| 14h | SRGR | Sample rate generator register | Section 3.6 |
| 24h | PCR | Pin control register | Section 3.7 |

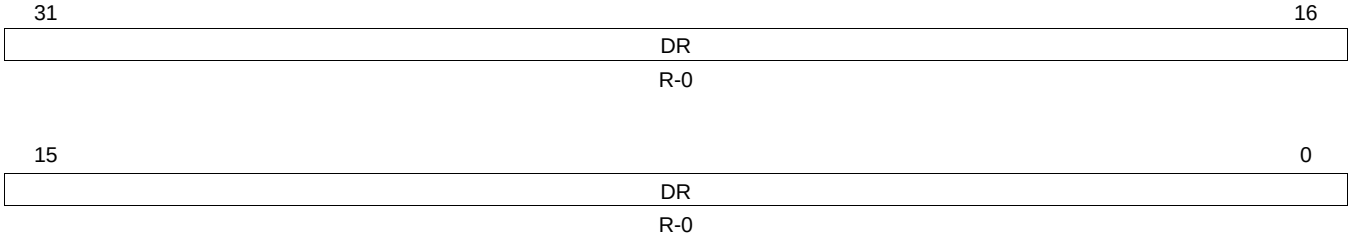
(1) The RBR, RSR, and XSR are not directly accessible via the CPUs or the EDMA controller.
(2) The CPUs and EDMA controller can only read this register; they cannot write to it.
(3) The DRR and DXR are accessible via the CPUs or the EDMA controller.

## 3.1 Data Receive Register (DRR)

The data receive register (DRR) contains the value to be written to the data bus. The DRR is shown in Figure 33 and described in Table 18.

See the device-specific data manual for the memory address of these registers. Both the CPUs and the EDMA can access DRR in all the memory-mapped locations. An access to *any* EDMA bus location is equivalent to an access to DRR of the corresponding ASP.

**Figure 33. Data Receive Register (DRR)**




LEGEND: R = Read only; -*n* = value after reset

**Table 18. Data Receive Register (DRR) Field Descriptions**

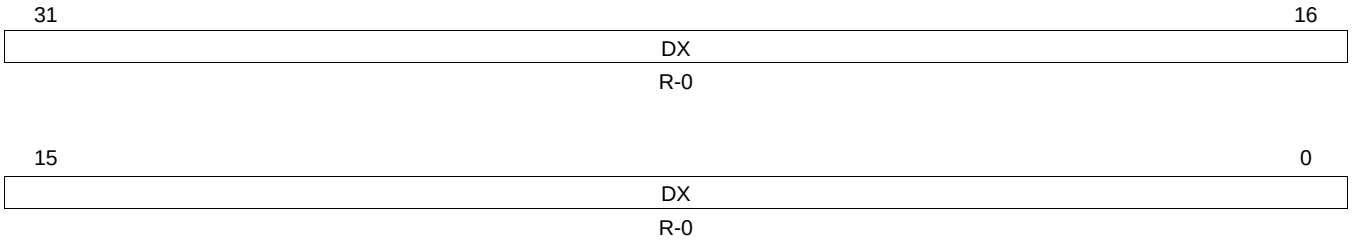
| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | DR | 0-FFFF FFFFh | Data receive register value to be written to the data bus. |

## 3.2 Data Transmit Register (DXR)

The data transmit register (DXR) contains the value to be loaded into the data transmit shift register (XSR). The DXR is shown in Figure 34 and described in Table 19.

See the device-specific data manual for the memory address of these registers. DXR is accessible via the peripheral bus and via the EDMA bus. Both the CPUs and the EDMA can access DXR in all the memory-mapped locations.

**Figure 34. Data Transmit Register (DXR)**




LEGEND: R = Read only; -*n* = value after reset

**Table 19. Data Transmit Register (DXR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-0 | DX | 0-FFFF FFFFh | Data transmit register value to be loaded into the data transmit shift register (XSR). |

## 3.3 Serial Port Control Register (SPCR)

The serial port is configured via the serial port control register (SPCR) and the pin control register (PCR). The SPCR contains ASP status control bits. The SPCR is shown in Figure 35 and described in Table 20.

**Figure 35. Serial Port Control Register (SPCR)**

| 31 | 26 | 25 | 24 |
|---|---|---|---|
| Reserved | | FREE | SOFT |
| R-0 | | R/W-0 | R/W-0 |

| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| FRST | GRST | XINTM | | XSYNCERR | XEMPTY | XRDY | XRST |
| R/W-0 | R/W-0 | R/W-0 | | R/W-0 | R-0 | R-0 | R/W-0 |

| 15 | 14 | 13 | 12 | 8 |
|---|---|---|---|---|
| DLB | RJUST | | Reserved | |
| R/W-0 | R/W-0 | | R-0 | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved(1) | Reserved | RINTM | | RSYNCERR | RFULL | RRDY | RRST |
| R/W-0 | R-0 | R/W-0 | | R/W-0 | R-0 | R-0 | R/W-0 |

LEGEND: R = Read only; R/W = Read/ Write; -*n* = value after reset

(1) If writing to this field, always write the default value of 0 to ensure proper ASP operation.

**Table 20. Serial Port Control Register (SPCR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-26 | Reserved | 0 | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect. |
| 25 | FREE | | Free-running enable mode bit. This bit is used in conjunction with SOFT bit to determine state of serial port clock during emulation halt. |
| | | 0 | Free-running mode is disabled. During emulation halt, SOFT bit determines operation of ASP. |
| | | 1 | Free-running mode is enabled. During emulation halt, serial clocks continue to run. |
| 24 | SOFT | | Soft bit enable mode bit. This bit is used in conjunction with FREE bit to determine state of serial port clock during emulation halt. This bit has no effect if FREE = 1. |
| | | 0 | Soft mode is disabled. Serial port clock stops immediately during emulation halt, thus aborting any transmissions. |
| | | 1 | Soft mode is enabled. During emulation halt, serial port clock stops after completion of current transmission. |
| 23 | FRST | | Frame-sync generator reset bit. |
| | | 0 | Frame-synchronization logic is reset. Frame-sync signal (FSG) is not generated by the sample-rate generator. |
| | | 1 | Frame-sync signal (FSG) is generated after (FPER + 1) number of CLKG clocks; that is, all frame counters are loaded with their programmed values. |
| 22 | GRST | | Sample-rate generator reset bit. |
| | | 0 | Sample-rate generator is reset. |
| | | 1 | Sample-rate generator is taken out of reset. CLKG is driven as per programmed value in sample-rate generator register (SRGR). |
| 21-20 | XINTM | 0-3h | Transmit interrupt (XINT) mode bit. |
| | | 0 | XINT is driven by XRDY (end-of-word). |
| | | 1h | Reserved |
| | | 2h | XINT is generated by a new frame synchronization. |
| | | 3h | XINT is generated by XSYNCERR. |

**Table 20. Serial Port Control Register (SPCR) Field Descriptions (continued)**

| Bit | Field | Value | Description |
|---|---|---|---|
| 19 | XSYNCERR | | Transmit synchronization error bit. Writing a 1 to XSYNCERR sets the error condition when the transmitter is enabled (XRST = 1). Thus, it is used mainly for testing purposes or if this operation is desired. |
| | | 0 | No synchronization error is detected. |
| | | 1 | Synchronization error is detected. |
| 18 | XEMPTY | | Transmit shift register empty bit. |
| | | 0 | XSR is empty. |
| | | 1 | XSR is not empty. |
| 17 | XRDY | | Transmitter ready bit. |
| | | 0 | Transmitter is not ready. |
| | | 1 | Transmitter is ready for new data in DXR. |
| 16 | XRST | | Transmitter reset bit resets or enables the transmitter. |
| | | 0 | Serial port transmitter is disabled and in reset state. |
| | | 1 | Serial port transmitter is enabled. |
| 15 | DLB | | Digital loop back mode enable bit. |
| | | 0 | Digital loop back mode is disabled. |
| | | 1 | Digital loop back mode is enabled. |
| 14-13 | RJUST | 0-3h | Receive sign-extension and justification mode bit. |
| | | 0 | Right-justify and zero-fill MSBs in DRR. |
| | | 1h | Right-justify and sign-extend MSBs in DRR. |
| | | 2h | Left-justify and zero-fill LSBs in DRR. |
| | | 3h | Reserved |
| 12-8 | Reserved | 0 | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect. |
| 7 | Reserved | 0 | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect. If writing to this field, always write the default value of 0 to ensure proper ASP operation. |
| 6 | Reserved | 0 | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect. |
| 5-4 | RINTM | 0-3h | Receive interrupt (RINT) mode bit. |
| | | 0 | RINT is driven by RRDY (end-of-word). |
| | | 1h | Reserved |
| | | 2h | RINT is generated by a new frame synchronization. |
| | | 3h | RINT is generated by RSYNCERR. |
| 3 | RSYNCERR | | Receive synchronization error bit. Writing a 1 to RSYNCERR sets the error condition when the receiver is enabled (RRST = 1). Thus, it is used mainly for testing purposes or if this operation is desired. |
| | | 0 | No synchronization error is detected. |
| | | 1 | Synchronization error is detected. |
| 2 | RFULL | | Receive shift register full bit. |
| | | 0 | RBR is not in overrun condition. |
| | | 1 | DRR is not read, RBR is full, and RSR is also full with new word. |
| 1 | RRDY | | Receiver ready bit. |
| | | 0 | Receiver is not ready. |
| | | 1 | Receiver is ready with data to be read from DRR. |
| 0 | RRST | | Receiver reset bit resets or enables the receiver. |
| | | 0 | The serial port receiver is disabled and in reset state. |
| | | 1 | The serial port receiver is enabled. |

## 3.4 Receive Control Register (RCR)

The receive control register (RCR) configures parameters of the receive operations. The RCR is shown in Figure 36 and described in Table 21.

**Figure 36. Receive Control Register (RCR)**

| 31 | 30 24 | 23 21 | 20 19 | 18 | 17 16 |
|---|---|---|---|---|---|
| RPHASE | RFRLEN2 | RWDLEN2 | RCOMPAND | RFIG | RDATDLY |
| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |

| 15 | 14 8 | 7 5 | 4 | 3 0 |
|---|---|---|---|---|
| Reserved | RFRLEN1 | RWDLEN1 | RWDREVRS | Reserved |
| R-0 | R/W-0 | R/W-0 | R/W-0 | R-0 |

LEGEND: R = Read only; R/W = Read/Write; -*n* = value after reset

**Table 21. Receive Control Register (RCR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31 | RPHASE | | Receive phases bit. |
| | | 0 | Single-phase frame |
| | | 1 | Dual-phase frame |
| 30-24 | RFRLEN2 | 0-7Fh | Specifies the receive frame length (number of words) in phase 2. |
| | | 0 | 1 word in phase 2 |
| | | 1h | 2 words in phase 2 |
| | | 2h | 3 words in phase 2 |
| | | ... | ... |
| | | 7Fh | 128 words in phase 2 |
| 23-21 | RWDLEN2 | 0-7h | Specifies the receive word length (number of bits) in phase 2. |
| | | 0 | Receive word length is 8 bits. |
| | | 1h | Receive word length is 12 bits. |
| | | 2h | Receive word length is 16 bits. |
| | | 3h | Receive word length is 20 bits. |
| | | 4h | Receive word length is 24 bits. |
| | | 5h | Receive word length is 32 bits. |
| | | 6h-7h | Reserved |
| 20-19 | RCOMPAND | 0-3h | Receive companding mode bit. Modes other than 00 are only enabled when RWDLEN1/2 bit is 000 (indicating 8-bit data). |
| | | 0 | No companding, data transfer starts with MSB first. |
| | | 1h | No companding, 8-bit data transfer starts with LSB first. |
| | | 2h | Compand using μ-law for receive data. |
| | | 3h | Compand using A-law for receive data. |
| 18 | RFIG | | Receive frame ignore bit. |
| | | 0 | Receive frame-synchronization pulses after the first pulse restarts the transfer. |
| | | 1 | Receive frame-synchronization pulses after the first pulse are ignored. |
| 17-16 | RDATDLY | 0-3h | Receive data delay bit. |
| | | 0 | 0-bit data delay |
| | | 1h | 1-bit data delay |
| | | 2h | 2-bit data delay |
| | | 3h | Reserved |
| 15 | Reserved | 0 | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect. |

**Table 21. Receive Control Register (RCR) Field Descriptions (continued)**

| Bit | Field | Value | Description |
|---|---|---|---|
| 14-8 | RFRLEN1 | 0-7Fh | Specifies the receive frame length (number of words) in phase 1. |
| | | 0 | 1 word in phase 1 |
| | | 1h | 2 words in phase 1 |
| | | 2h | 3 words in phase 1 |
| | | ... | ... |
| | | 7Fh | 128 words in phase 1 |
| 7-5 | RWDLEN1 | 0-7h | Specifies the receive word length (number of bits) in phase 1. |
| | | 0 | Receive word length is 8 bits. |
| | | 1h | Receive word length is 12 bits. |
| | | 2h | Receive word length is 16 bits. |
| | | 3h | Receive word length is 20 bits. |
| | | 4h | Receive word length is 24 bits. |
| | | 5h | Receive word length is 32 bits. |
| | | 6h-7h | Reserved |
| 4 | RWDREVRS | | Receive 32-bit bit reversal enable bit. |
| | | 0 | 32-bit bit reversal is disabled. |
| | | 1 | 32-bit bit reversal is enabled. 32-bit data is received LSB first. RWDLEN1/2 bit should be set to 5h (32-bit operation); RCOMPAND bit should be set to 1h (transfer starts with LSB first); otherwise, operation is undefined. |
| 3-0 | Reserved | 0 | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect. |

## 3.5 Transmit Control Register (XCR)

The transmit control register (XCR) configures parameters of the transmit operations. The XCR is shown in Figure 37 and described in Table 22.

**Figure 37. Transmit Control Register (XCR)**

| 31 | 30 24 | 23 21 | 20 19 | 18 | 17 16 |
|---|---|---|---|---|---|
| XPHASE | XFRLEN2 | XWDLEN2 | XCOMPAND | XFIG | XDATDLY |
| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |

| 15 | 14 8 | 7 5 | 4 | 3 0 |
|---|---|---|---|---|
| Reserved | XFRLEN1 | XWDLEN1 | XWDREVRS | Reserved |
| R-0 | R/W-0 | R/W-0 | R/W-0 | R-0 |

LEGEND: R = Read only; R/W = Read/Write; -*n* = value after reset

**Table 22. Transmit Control Register (XCR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31 | XPHASE | | Transmit phases bit. |
| | | 0 | Single-phase frame |
| | | 1 | Dual-phase frame |
| 30-24 | XFRLEN2 | 0-7Fh | Specifies the transmit frame length (number of words) in phase 2. |
| | | 0 | 1 word in phase 2 |
| | | 1h | 2 words in phase 2 |
| | | 2h | 3 words in phase 2 |
| | | ... | ... |
| | | 7Fh | 128 words in phase 2 |
| 23-21 | XWDLEN2 | 0-7h | Specifies the transmit word length (number of bits) in phase 2. |
| | | 0 | Transmit word length is 8 bits. |
| | | 1h | Transmit word length is 12 bits. |
| | | 2h | Transmit word length is 16 bits. |
| | | 3h | Transmit word length is 20 bits. |
| | | 4h | Transmit word length is 24 bits. |
| | | 5h | Transmit word length is 32 bits. |
| | | 6h-7h | Reserved |
| 20-19 | XCOMPAND | 0-3h | Transmit companding mode bit. Modes other than 00 are only enabled when XWDLEN1/2 bit is 000 (indicating 8-bit data). |
| | | 0 | No companding, data transfer starts with MSB first. |
| | | 1h | No companding, 8-bit data transfer starts with LSB first. |
| | | 2h | Compand using μ-law for transmit data. |
| | | 3h | Compand using A-law for transmit data. |
| 18 | XFIG | | Transmit frame ignore bit. |
| | | 0 | Transmit frame-synchronization pulses after the first pulse restarts the transfer. |
| | | 1 | Transmit frame-synchronization pulses after the first pulse are ignored. |
| 17-16 | XDATDLY | 0-3h | Transmit data delay bit. |
| | | 0 | 0-bit data delay |
| | | 1h | 1-bit data delay |
| | | 2h | 2-bit data delay |
| | | 3h | Reserved |
| 15 | Reserved | 0 | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect. |

**Table 22. Transmit Control Register (XCR) Field Descriptions (continued)**

| Bit | Field | Value | Description |
|---|---|---|---|
| 14-8 | XFRLEN1 | 0-7Fh | Specifies the transmit frame length (number of words) in phase 1. |
| | | 0 | 1 word in phase 1 |
| | | 1h | 2 words in phase 1 |
| | | 2h | 3 words in phase 1 |
| | | ... | ... |
| | | 7Fh | 128 words in phase 1 |
| 7-5 | XWDLEN1 | 0-7h | Specifies the transmit word length (number of bits) in phase 1. |
| | | 0 | Transmit word length is 8 bits. |
| | | 1h | Transmit word length is 12 bits. |
| | | 2h | Transmit word length is 16 bits. |
| | | 3h | Transmit word length is 20 bits. |
| | | 4h | Transmit word length is 24 bits. |
| | | 5h | Transmit word length is 32 bits. |
| | | 6h-7h | Reserved |
| 4 | XWDREVRS | | Transmit 32-bit bit reversal feature enable bit. |
| | | 0 | 32-bit bit reversal is disabled. |
| | | 1 | 32-bit bit reversal is enabled. 32-bit data is transmitted LSB first. XWDLEN1/2 bit should be set to 5h (32-bit operation); XCOMPAND bit should be set to 1h (transfer starts with LSB first); otherwise, operation is undefined. |
| 3-0 | Reserved | 0 | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect. |

## 3.6 Sample Rate Generator Register (SRGR)

The sample rate generator register (SRGR) controls the operation of various features of the sample rate generator. The SRGR is shown in Figure 38 and described in Table 23.

**Figure 38. Sample Rate Generator Register (SRGR)**

| 31 | 30 | 29 | 28 | 27 | 16 |
|---|---|---|---|---|---|
| Reserved | | CLKSM | FSGM | FPER | |
| R-0 | | R/W-1 | R/W-0 | R/W-0 | |

| 15 | 8 | 7 | 0 |
|---|---|---|---|
| FWID | | CLKGDV | |
| R/W-0 | | R/W-1 | |

LEGEND: R = Read only; R/W = Read/Write; -*n* = value after reset

**Table 23. Sample Rate Generator Register (SRGR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-30 | Reserved | 0 | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect. |
| 29 | CLKSM | | Sample rate generator input clock mode bit. The sample rate generator can accept an input clock signal and divide it down according to CLKGDV to produce an output clock signal, CLKG. The frequency of CLKG is:<br>*CLKG frequency = Input clock frequency/(CLKGDV + 1)*<br>CLKSM is used in conjunction with the SCLKME bit in the serial port control register (SPCR) to determine the source for the input clock.<br>An ARM reset selects the ASP internal input clock as the input clock and forces the CLKG frequency to 1/2 the ASP internal input clock frequency. |
| | | 0 | The input clock for the sample rate generator is taken from the CLKR pin when selected as shown below (when the SCLKME bit in SPCR = 1):<br>**CLKSM** / **SCLKME** / **Input Clock for Sample Rate Generator**<br>0 / 0 / Reserved<br>0 / 1 / Signal on CLKR pin |
| | | 1 | The input clock for the sample rate generator is taken from the ASP internal input clock or from the CLKX pin, depending on the value of the SCLKME bit in SPCR:<br>**CLKSM** / **SCLKME** / **Input Clock for Sample Rate Generator**<br>1 / 0 / ASP internal input clock<br>1 / 1 / Signal on CLKX pin |
| 28 | FSGM | | Sample-rate generator transmit frame-synchronization mode bit is only used when FSXM = 1 in PCR. |
| | | 0 | Transmit frame-sync signal (FSX) is generated on every DXR-to-XSR copy. When FSGM = 0, FWID bit and FPER bit are ignored. |
| | | 1 | Transmit frame-sync signal (FSX) is driven by the sample-rate generator frame-sync signal (FSG). |
| 27-16 | FPER | 0-FFFh | Frame period value plus 1 specifies when the next frame-sync signal becomes active. Range is 1 to 4096 sample-rate generator clock (CLKG) periods. |
| 15-8 | FWID | 0-FFh | Frame width value plus 1 specifies the width of the frame-sync pulse (FSG) during its active period. |
| 7-0 | CLKGDV | 0-FFh | Sample-rate generator clock (CLKG) divider value is used as the divide-down number to generate the required sample-rate generator clock frequency. |

## 3.7 Pin Control Register (PCR)

The serial port is configured via the serial port control register (SPCR) and the pin control register (PCR). The PCR contains ASP status control bits. The PCR is shown in Figure 39 and described in Table 24.

**Figure 39. Pin Control Register (PCR)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| R-0 | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| Reserved | | Reserved(1) | Reserved(1) | FSXM | FSRM | CLKXM | CLKRM |
| R-0 | | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SCLKME | Reserved(1) | Reserved | Reserved | FSXP | FSRP | CLKXP | CLKRP |
| R/W-0 | R-0 | R/W-0 | R-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |

LEGEND: R = Read only; R/W = Rear/Write; -*n* = value afer reset

(1) If writing to this field, always write the default value of 0 to ensure proper ASP operation.

**Table 24. Pin Control Register (PCR) Field Descriptions**

| Bit | Field | Value | Description |
|---|---|---|---|
| 31-14 | Reserved | 0 | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect. |
| 13-12 | Reserved | 0 | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect. If writing to this field, always write the default value of 0 to ensure proper ASP operation. |
| 11 | FSXM | | Transmit frame-synchronization mode bit. |
| | | 0 | Frame-synchronization signal is derived from an external source. |
| | | 1 | Frame-synchronization signal is determined by FSGM bit in SRGR. |
| 10 | FSRM | | Receive frame-synchronization mode bit. |
| | | 0 | Frame-synchronization signal is derived from an external source. FSR is an input pin. |
| | | 1 | Frame-synchronization signal is generated internally by the sample-rate generator. FSR is an output pin. |
| 9 | CLKXM | | Transmit clock mode bit |
| | | 0 | CLKX is an input pin and is driven by an external clock. |
| | | 1 | CLKX is an output pin and is driven by the internal sample-rate generator. |
| 8 | CLKRM | | Receive clock mode bit |
| | | | **Digital loop back mode is disabled (DLB = 0 in SPCR):** |
| | | 0 | CLKR is an input pin and is driven by an external clock. |
| | | 1 | CLKR is an output pin and is driven by the internal sample-rate generator. |
| | | | **Digital loop back mode is enabled (DLB = 1 in SPCR):** |
| | | 0 | Receive clock (not the CLKR pin) is driven by transmit clock (CLKX) that is based on CLKXM bit. CLKR pin is in high-impedance state. |
| | | 1 | CLKR is an output pin and is driven by the transmit clock. The transmit clock is based on CLKXM bit. |

**Table 24. Pin Control Register (PCR) Field Descriptions (continued)**

| Bit | Field | Value | Description |
|---|---|---|---|
| 7 | SCLKME | | Sample rate generator input clock mode bit. The sample rate generator can produce a clock signal, CLKG. The frequency of CLKG is:<br>*CLKG frequency = Input clock frequency/(CLKGDV + 1)*<br>SCLKME is used in conjunction with the CLKSM bit in the sample rate generator register (SRGR) to select the input clock.<br>An ARM reset selects the ASP internal input clock as the input clock and forces the CLKG frequency to 1/2 the ASP internal input clock frequency. |
| | | 0 | The input clock for the sample rate generator is taken from the ASP internal input clock as shown below (when the CLKSM bit in SRGR = 1):<br>**SCLKME** / **CLKSM** / **Input Clock for Sample Rate Generator**<br>0 / 0 / Reserved<br>0 / 1 / ASP internal input clock |
| | | 1 | The input clock for the sample rate generator is taken from the CLKR pin or from the CLKX pin, depending on the value of the CLKSM bit in SRGR:<br>**SCLKME** / **CLKSM** / **Input Clock for Sample Rate Generator**<br>1 / 0 / Signal on CLKR pin<br>1 / 1 / Signal on CLKX pin |
| 6 | Reserved | 0 | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect. If writing to this field, always write the default value of 0 to ensure proper ASP operation. |
| 5-4 | Reserved | 0 | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect. |
| 3 | FSXP | | Transmit frame-synchronization polarity bit. |
| | | 0 | Transmit frame-synchronization pulse is active high. |
| | | 1 | Transmit frame-synchronization pulse is active low. |
| 2 | FSRP | | Receive frame-synchronization polarity bit. |
| | | 0 | Receive frame-synchronization pulse is active high. |
| | | 1 | Receive frame-synchronization pulse is active low. |
| 1 | CLKXP | | Transmit clock polarity bit. |
| | | 0 | Transmit data sampled on rising edge of CLKX. |
| | | 1 | Transmit data sampled on falling edge of CLKX. |
| 0 | CLKRP | | Receive clock polarity bit. |
| | | 0 | Receive data sampled on falling edge of CLKR. |
| | | 1 | Receive data sampled on rising edge of CLKR. |

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

| **Products** | | **Applications** | |
|---|---|---|---|
| Amplifiers | amplifier.ti.com | Audio | www.ti.com/audio |
| Data Converters | dataconverter.ti.com | Automotive | www.ti.com/automotive |
| DSP | dsp.ti.com | Broadband | www.ti.com/broadband |
| Clocks and Timers | www.ti.com/clocks | Digital Control | www.ti.com/digitalcontrol |
| Interface | interface.ti.com | Medical | www.ti.com/medical |
| Logic | logic.ti.com | Military | www.ti.com/military |
| Power Mgmt | power.ti.com | Optical Networking | www.ti.com/opticalnetwork |
| Microcontrollers | microcontroller.ti.com | Security | www.ti.com/security |
| RFID | www.ti-rfid.com | Telephony | www.ti.com/telephony |
| RF/IF and ZigBee® Solutions | www.ti.com/lprf | Video & Imaging | www.ti.com/video |
| | | Wireless | www.ti.com/wireless |

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265