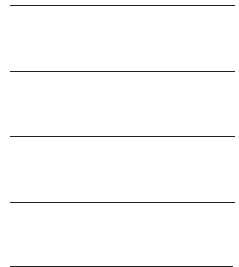


Serial Ports 10



10.1 OVERVIEW

The ADSP-2106x has two independent, synchronous serial ports, SPORT0 and SPORT1, that provide an I/O interface to a wide variety of peripheral devices. Each serial port has its own set of control registers and data buffers. With a range of clock and frame synchronization options, the SPORTs allow a variety of serial communication protocols and provide a glueless hardware interface to many industry-standard data converters and CODECs.

The serial ports can operate at the full clock rate of the processor, providing each with a maximum data rate of n Mbit/s, where n equals the processor clock frequency. Independent transmit and receive functions provide greater flexibility for serial communications. Serial port data can be automatically transferred to and from on-chip memory using DMA block transfers. Each of the serial ports offers a TDM (time division multiplexed) multichannel mode.

Serial port clocks and frame syncs can be internally generated by the ADSP-2106x or received from an external source. The serial ports can operate with little-endian or big-endian transmission formats, with word lengths selectable from 3 to 32 bits. They offer selectable synchronization and transmit modes as well as optional μ -law or A-law companding in hardware.

The serial ports offer the following features and capabilities:

- Independent transmit and receive functions.
- Can transfer data words up to 32 bits in length, either MSB-first or LSB-first.
- Double-buffering of data—both receive and transmit functions have a data buffer register as well as a shift register; the double-buffering provides additional time to service the SPORT.
- A-law and μ -law hardware companding on transmitted and received words.
- Serial clock and frame sync signals can be generated internally, in a wide range of frequencies, or input from an external source.
- Interrupt-driven, single-word transfers to and from on-chip memory controlled by the ADSP-2106x core.

10 Serial Ports

- DMA transfers to and from on-chip memory—each SPORT can automatically receive and/or transmit an entire block of data.
- Chained DMA operations of multiple data blocks.
- Multichannel mode for TDM interfaces—each SPORT can receive and transmit data selectively from channels of a time-division-multiplexed serial bitstream; this mode can be useful for T1 interfaces.

Table 10.1 shows the pins of each serial port:

<i>Function</i>	<i>SPORT0 Pins</i>	<i>SPORT1 Pins</i>
Transmit Data	DT0	DT1
Transmit Clock	TCLK0	TCLK1
Transmit Frame Sync	TFS0	TFS1
Receive Data	DR0	DR1
Receive Clock	RCLK0	RCLK1
Receive Frame Sync	RFS0	RFS1

Table 10.1 Serial Port Pins

A serial port receives serial data on its DR input and transmits serial data on its DT output. It can receive and transmit simultaneously, for full duplex operation.

Serial communications are synchronized to a clock signal—every data bit must be accompanied by a clock pulse. Each serial port can generate or receive its own transmit clock signal (TCLK) and receive clock signal (RCLK). Internally-generated serial clock frequencies are configured in the TDIVx and RDIVx registers.

In addition to the serial clock signal, data may be signalled by a frame synchronization signal. The framing signal can occur either at the beginning of an individual word or at the beginning of a block of words. The configuration of frame synch signals depends upon the type of serial device connected to the ADSP-2106x. Each serial port can generate or receive its own transmit frame sync signal (TFS) and receive frame sync signal (RFS). Internally-generated frame sync frequencies are configured in the TDIVx and RDIVx registers.

Figure 10.1 shows a block diagram of a serial port. Data to be transmitted is written to the TX buffer. The data is (optionally) compressed in hardware, then automatically transferred to the transmit shift register. The data in the shift register is then shifted out on the SPORT's DT pin, synchronous to the TCLK transmit clock. If

Serial Ports 10

framing signals are used, the TFS signal indicates the start of the serial word transmission. The DT pin is always driven, i.e. not tristated, if the serial port is enabled (SPEN=1 in the STCTLx control register), unless it is in multichannel mode and an inactive time slot occurs. (See Section 10.7, “Multichannel Operation.”)

The receive portion of the SPORT shifts in data from the DR pin, synchronous to the RCLK receive clock. If framing signals are used, the RFS signal indicates the beginning of the serial word being received. When an entire word is shifted in, the data is (optionally) expanded, then automatically transferred to the RX buffer.

Note: The ADSP-2106x SPORTs are not UARTs and cannot be used to communicate with an RS-232 device or any other asynchronous communications protocol. One way to implement RS-232-compatible communications with the ADSP-2106x, however, is to use two of the FLAG pins as asynchronous data receive and transmit signals. For an example of how to do this, see the *Software UART* chapter of *Digital Signal Processing Applications Using The ADSP-2100 Family, Volume 2*.

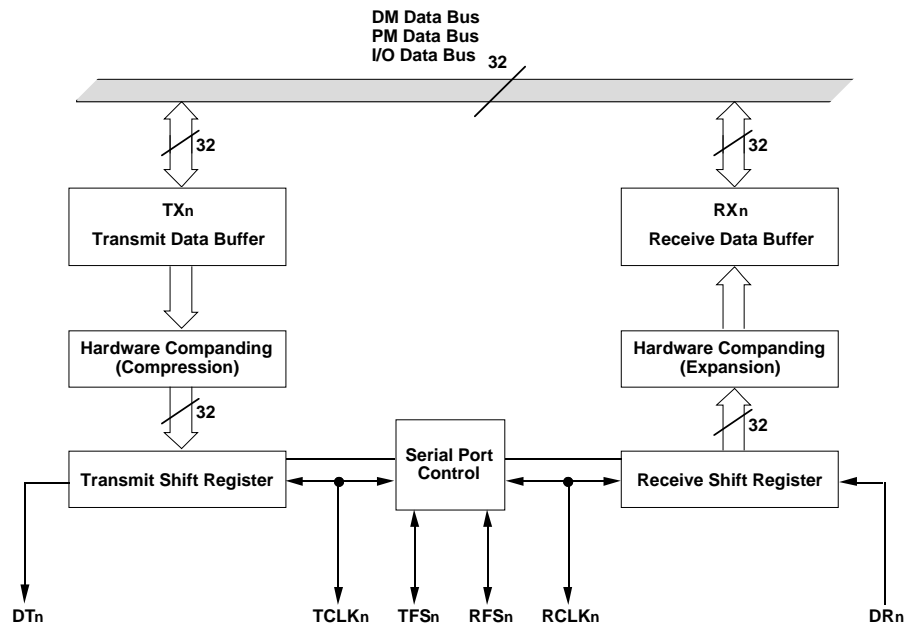


Figure 10.1 Serial Port Block Diagram

10 Serial Ports

10.1.1 SPORT Interrupts

Each serial port has a transmit DMA interrupt and a receive DMA interrupt. When serial port DMA is not enabled, the interrupts occur for each data word transmitted and received. The priority of the serial port interrupts is shown in Table 10.2.

<i>Interrupt Name*</i>	<i>Interrupt</i>	
SPR0I	SPORT0 Receive DMA Channel	Highest Priority
SPR1I	SPORT1 Receive DMA Channel	
SPT0I	SPORT0 Transmit DMA Channel	
SPT1I	SPORT1 Transmit DMA Channel	Lowest Priority

Table 10.2 SPORT Interrupts

* These names are defined in the `def21060.h` include file supplied with the ADSP-21000 Family Development Software.

SPORT Interrupts occur on the second system clock (CLKIN) after the last bit of the serial word is latched in or driven out.

10.2 SPORT RESET

There are two ways to reset the serial ports: a hardware reset using the RESET pin of the processor, and a software reset accomplished by clearing the serial port's enable bit (SPEN) in the STCTLx and SRCTLx control registers. Each method has a different effect on the serial port.

A hardware reset disables the serial ports by clearing the STCTLx and SRCTLx control registers (including the SPEN enable bits) and the TDIVx and RDIVx frame sync divisor registers. Any ongoing operations are aborted.

A software reset of the SPEN enable bit(s) disables the serial port(s) and aborts any ongoing operations. Status bits are also cleared.

The serial ports will be ready to start transmitting or receiving data two CLKIN cycles after they are enabled (in the STCTLx or SRCTLx control register). No serial clocks will be lost from this point on.

Serial Ports 10

10.3 SPORT CONTROL REGISTERS & DATA BUFFERS

The registers used to control and configure the serial ports are part of the IOP register set. Each SPORT has its own set of the following control registers and data buffers:

<i>Register Name*</i>	<i>Function</i>
STCTLx	SPORT Transmit Control Register
TXx	Transmit Data Buffer
TDIVx	Transmit Clock & Frame Sync Divisors
MTCSx	Multichannel Transmit Select
MTCCSx	Multichannel Transmit Compand Select
SRCTLx	SPORT Receive Control Register
RXx	Receive Data Buffer
RDIVx	Receive Clock & Frame Sync Divisors
MRCSx	Multichannel Receive Select
MRCCSx	Multichannel Receive Compand Select
SPATHx	SPORT Path Length (for mesh multiprocessing)
KEYWDx	SPORT Receive Comparison**
KEYMASKx	SPORT Receive Comparison Mask**

* x = 0, 1

** ADSP-21061 only

Table 10.3 (on the next page) shows the memory-mapped address and reset initialization value of each SPORT register. All of the registers are 32 bits wide, except for the 16-bit SPATHx register and location 0x00FF. (Note that for standard, non-mesh-multiprocessing operation of the serial ports, the SPATHx register and location 0x00FF must remain equal to the reset initialization value, 0x0001.)

The SPORT control registers are programmed by writing to the appropriate address in memory. The symbolic names of the registers and individual control bits can be used in ADSP-2106x programs—the `#define` definitions for these symbols are contained in the file `def21060.h` which is provided in the `INCLUDE` directory of the ADSP-21000 Family Development Software. The `def21060.h` file is shown in the *Control/Status Registers* appendix of this manual. All control and status bits in the SPORT registers are active high unless otherwise noted.

Because the SPORT registers are memory-mapped they cannot be written with data coming directly from memory. They must instead be written

10 Serial Ports

<u>Memory Address</u>	<u>Register Name</u>	<u>Initialization After RESET</u>	<u>Description</u>
0x00E0	STCTL0	0x0000 0000	SPORT0 Transmit Control Register
0x00E1	SRCTL0	0x0000 0000	SPORT0 Receive Control Register
0x00E2	TX0	ni	SPORT0 Transmit Data Buffer
0x00E3	RX0	ni	SPORT0 Receive Data Buffer
0x00E4	TDIV0	ni	SPORT0 Transmit Divisor
0x00E5			<i>reserved</i>
0x00E6	RDIV0	ni	SPORT0 Receive Divisor
0x00E7			<i>reserved</i>
0x00E8	MTCS0	ni	SPORT0 Multichannel Transmit Select
0x00E9	MRCS0	ni	SPORT0 Multichannel Receive Select
0x00EA	MTCCS0	ni	SPORT0 Multichannel Transmit Compand Select
0x00EB	MRCCS0	ni	SPORT0 Multichannel Receive Compand Select
0x00EC	KEYWD	ni	SPORT0 Receive Comparison (ADSP-21061)
0x00ED	KEYMASK	ni	SPORT0 Receive Comparison Mask (ADSP-21061)
0x00EE	SPATH0	0x0001	SPORT0 Path Length (Mesh Multiprocessing)
0x00EF		0x0001	<i>reserved</i>
0x00F0	STCTL1	0x0000 0000	SPORT1 Transmit Control Register
0x00F1	SRCTL1	0x0000 0000	SPORT1 Receive Control Register
0x00F2	TX1	ni	SPORT1 Transmit Data Buffer
0x00F3	RX1	ni	SPORT1 Receive Data Buffer
0x00F4	TDIV1	ni	SPORT1 Transmit Divisor
0x00F5			<i>reserved</i>
0x00F6	RDIV1	ni	SPORT1 Receive Divisor
0x00F7			<i>reserved</i>
0x00F8	MTCS1	ni	SPORT1 Multichannel Transmit Select
0x00F9	MRCS1	ni	SPORT1 Multichannel Receive Select
0x00FA	MTCCS1	ni	SPORT1 Multichannel Transmit Compand Select
0x00FB	MRCCS1	ni	SPORT1 Multichannel Receive Compand Select
0x00FC	KEYWD	ni	SPORT1 Receive Comparison (ADSP-21061)
0x00FD	KEYMASK	ni	SPORT1 Receive Comparison Mask (ADSP-21061)
0x00FE	SPATH1	0x0001	SPORT1 Path Length (Mesh Multiprocessing)
0x00FF		0x0001	<i>reserved</i>

ni= not initialized

Table 10.3 SPORT Register Addresses & Initialization

from (or read into) ADSP-2106x core registers, usually one of the general-purpose universal registers of the register file (R15–R0). The SPORT control registers can also be written or read by external devices, i.e. another ADSP-2106x or a host processor, to set up a serial port DMA operation, for example.

10.3.1 Register Writes & Effect Latency

SPORT register writes are internally completed at the end of the same CLKIN cycle in which they occur. The register will therefore read back the newly written value on the very next cycle. When a read of one of the STCTLx or SRCTLx control registers is immediately followed by a write to that register, however, the write may take two cycles to complete.

Serial Ports 10

After a write to a SPORT register, control and mode bit changes generally take effect in the second CLKIN cycle after the write is completed. The serial ports will be ready to start transmitting or receiving two CLKIN cycles after they are enabled (in the STCTLx or SRCTLx control register). No serial clocks will be lost from this point on.

10.3.2 Transmit & Receive Data Buffers (TX, RX)

TX0 and TX1 are the transmit data buffers for SPORT0 and SPORT1. They are 32-bit buffers which must be loaded with the data to be transmitted; the data is loaded either by the DMA controller or by the program running on the ADSP-2106x core. RX0 and RX1 are the receive data buffers for SPORT0 and SPORT1. They are 32-bit buffers which are automatically loaded from the receive shifter when a complete word has been received. Word lengths of less than 32 bits are right-justified in the receive and transmit buffers.

The TX buffers act like a two-location FIFO because they have a data register plus an output shift register (see Figure 10.1); two 32-bit words may be stored in TX at any one time. When the TX buffer is loaded and any previous word has been transmitted, the buffer contents are automatically loaded into the output shifter. An interrupt is generated when the output shifter has been loaded, signifying that the TX buffer is ready to accept the next word (i.e. the TX buffer is “not full”). This interrupt will not occur if serial port DMA is enabled or if the corresponding mask bit in the IMASK register is set.

The transmit underflow status bit (TUVF) will be set in the transmit control register when a transmit frame synch occurs and no new data has been loaded into TX. The TUVF status bit is “sticky” and is only cleared by disabling the serial port.

The RX buffers act like a three-location FIFO because they have two data registers plus an input shift register. Two complete 32-bit words can be stored in RX while a third word is being shifted in. The third word will overwrite the second if the first word has not been read out (by the ADSP-2106x core or the DMA controller). When this happens, the receive overflow status bit (ROVF) will be set in the receive control register. Almost three complete words can be received without the RX buffer being read before overflow occurs. The overflow status is generated on the last bit of third word. The ROVF status bit is “sticky” and is only cleared by disabling the serial port.

10 Serial Ports

An interrupt is generated when the RX buffer has been loaded with a received word (i.e. the RX buffer is “not empty”). This interrupt will be masked out if serial port DMA is enabled or if the corresponding bit in the IMASK register is set.

10.3.2.1 Reading & Writing RX, TX

If your ADSP-2106x program causes the core processor to attempt a read from an empty RX buffer or a write to a full TX buffer, the access is delayed until the buffer is accessed by the external I/O device. (This delay is called a core processor hang.) If it is not known whether the core processor can access the RX or TX buffer without a hang, the buffer’s *full or empty* status should be read first (in STCTLx or SRCTLx) to determine if the access can be made. To prevent this type of hang condition from occurring, the BHD (Buffer Hang Disable) bit can be set in the SYSCON register.

The status bits in STCTLx and SRCTLx are updated during reads and writes from the core processor even when the serial port is disabled.

The serial port should be disabled when writing to the RX buffer or reading from the TX buffer.

10.3.3 Transmit & Receive Control Registers (STCTL, SRCTL)

The main control registers for each serial port are the transmit control register, STCTLx, and the receive control register, SRCTLx. These registers are defined in Tables 10.4 and 10.5, and are pictured in Figures 10.2 and 10.3. When changing operating modes, a serial port control register should be cleared (i.e. written with all zeros) before the new mode is written to the register.

The Transmit Underflow Status bit (TUVF) is set whenever the TFS signal occurs (from either external or internal source) while the TX buffer is empty. The internally generated TFS may be suppressed whenever TX is empty by clearing the DITFS control bit (DITFS=0).

When DITFS=0, the default, the transmit frame sync signal (TFS) is dependent upon new data being present in the TX buffer—the TFS signal will only be generated for new data. Setting DITFS to 1 selects data-independent frame syncs. This causes the TFS signal to be generated whether or not new data is present, transmitting the contents of the TX buffer regardless. Serial port DMA will typically keep the TX buffer full, and when the DMA operation is complete the last word in TX will be continuously transmitted.

Serial Ports 10

The TXS status bits indicate whether the TX buffer is full (11), empty (00), or partially full (10). To test for space in TX, therefore, test for TXS0 (bit 30) equal to zero. To test for the presence of any data in TX, test for TXS1 (bit 31) equal to one.

<i>Bit(s)</i>	<i>Name</i>	<i>Definition</i>
0	SPEN*	SPORT Enable
1-2	DTYPE	Data Type (data format, companding)
3	SENDN	Serial Word Endian (1=LSB first)
4-8	SLEN	Serial Word Length - 1
9	PACK	Data Word Unpacking (32-bit to 16-bit)
10	ICLK*	Internally Generated Transmit Clock
11	-	<i>reserved</i>
12	CKRE	Data, Frame Sync Sampling on Clock Rising Edge
13	TFSR*	Transmit Frame Sync Required
14	ITFS*	Internally Generated TFS
15	DITFS	Data-Independent TFS
16	LTFS	Active Low TFS
17	LAFS*	Late TFS
18	SDEN	SPORT Transmit DMA Enable
19	SCHEN	SPORT Transmit DMA Chaining Enable
20-23	MFD	Multichannel Frame Delay
24-28	CHNL**	Current Channel Status (read-only)
29	TUVF**	Transmit Underflow Status (sticky, read-only)
30-31	TXS**	TX Buffer Status (read-only) 11=full, 00=empty, 10=partially full

Table 10.4 STCTLx Transmit Control Register Bits

* Must be set to 0 for multichannel operation.

** Status bits are read-only. They are cleared by disabling the serial port (setting SPEN=0). TXS may subsequently change state if the data is read or written by the ADSP-2106x core while the SPORT is disabled.

10 Serial Ports

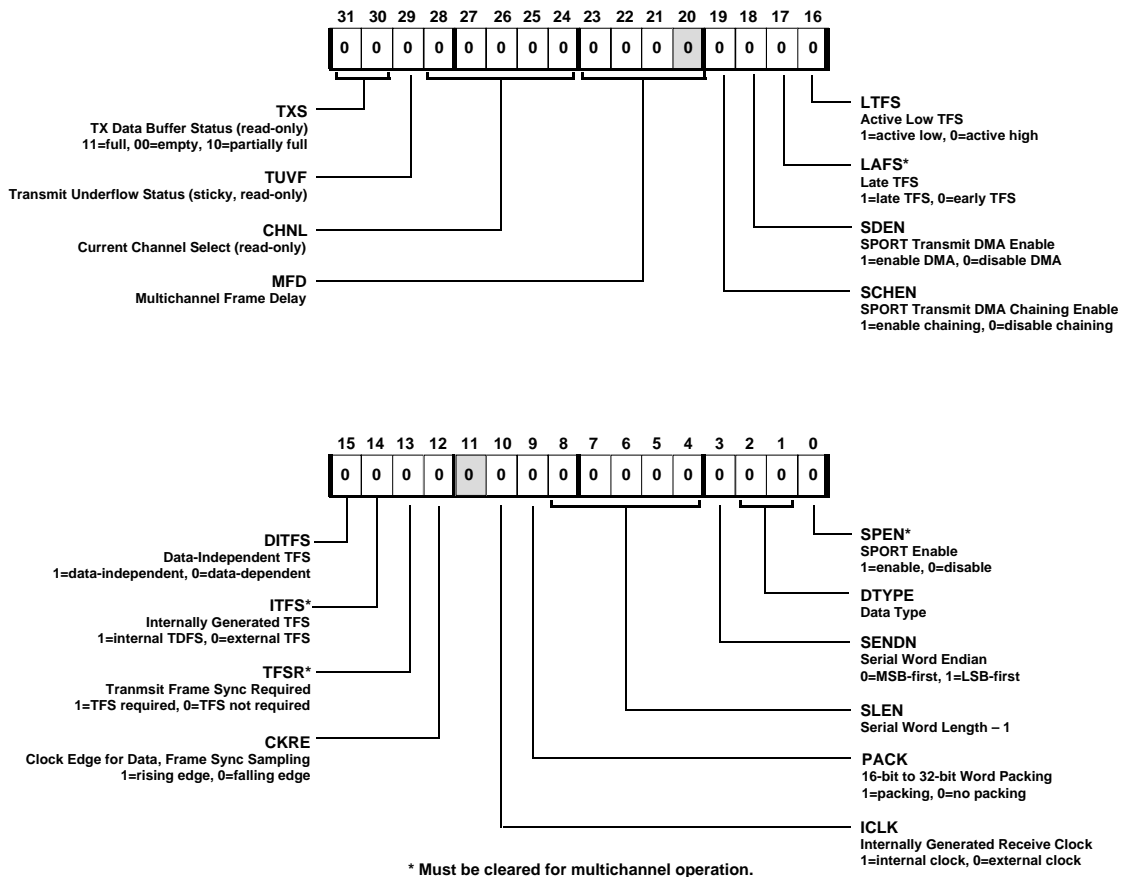


Figure 10.2 STCTL0, STCTL1 Transmit Control Registers

Serial Ports 10

<i>Bit(s)</i>	<i>Name</i>	<i>Definition</i>
0	SPEN*	SPORT Enable
1-2	DTYPE	Data Type (data format, companding)
3	SENDN	Serial Word Endian (1=LSB first)
4-8	SLEN	Serial Word Length - 1
9	PACK	Data Word Packing (16-bit to 32-bit)
10	ICLK	Internally Generated Receive Clock
11	-	<i>reserved</i>
12	CKRE	Data, Frame Sync Sampling on Clock Rising Edge
13	RFSR*	Receive Frame Sync Required
14	IRFS	Internally Generated RFS
15	-	<i>reserved</i>
16	LRFS	Active Low RFS
17	LAFS*	Late RFS
18	SDEN	SPORT Receive DMA Enable
19	SCHEN	SPORT Receive DMA Chaining Enable
20	-	<i>reserved</i>
21	D2DMA*	2-Dimensional DMA Array Enable
22	SPL*	SPORT Loopback (test)
23	MCE	Multichannel Enable
24-28	NCH	Number of Channels - 1 (multichannel operation)
29	ROVF**	Receive Overflow Status (sticky, read-only)
30-31	RXS**	RX Buffer Status (read-only) 11=full, 00=empty, 10=partially full

Table 10.5 SRCTLx Receive Control Register Bits

* Must be cleared for multichannel operation.

** Status bits are read-only. They are cleared by disabling the serial port (setting SPEN=0). RXS may subsequently change state if the data is read or written by the ADSP-2106x core while the SPORT is disabled.

The RXS status bits indicate whether the RX buffer is full (11), empty (00), or partially full (10). To test for space in RX, therefore, test for RXS0 (bit 30) equal to zero. To test for the presence of any data in RX, test for RXS1 (bit 31) equal to one.

The Receive Overflow Status bit (ROVF) is set whenever new data is received while the RX buffer is full; the new data overwrites the existing data.

10 Serial Ports

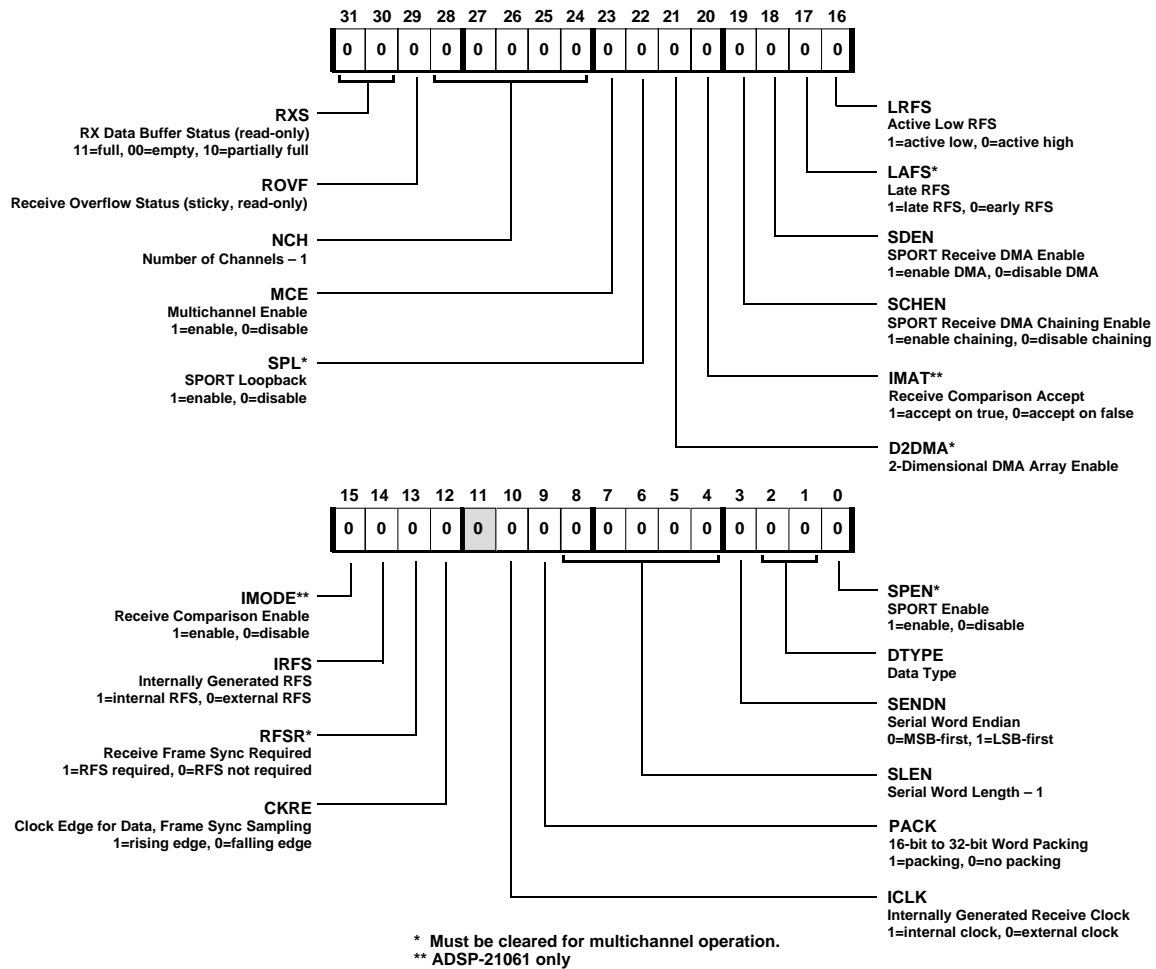


Figure 10.3 SRCTL0, SRCTL1 Receive Control Registers

Serial Ports 10

10.3.4 Clock & Frame Sync Frequencies (TDIV, RDIV)

The TDIVx and RDIVx registers contain divisor values which determine the frequencies for internally generated clocks and frame syncs. These registers are defined in Tables 10.6 and 10.7, and are pictured in Figures 10.4 and 10.5.

<i>Bits</i>	<i>Name</i>	<i>Definition</i>
15-0	TCLKDIV	Transmit Clock Divisor
31-16	TFSDIV	Transmit Frame Sync Divisor

Table 10.6 Transmit Divisor Register Bit Fields

<i>Bits</i>	<i>Name</i>	<i>Definition</i>
15-0	RCLKDIV	Receive Clock Divisor
31-16	RFSDIV	Receive Frame Sync Divisor

Table 10.7 Receive Divisor Register Bit Fields

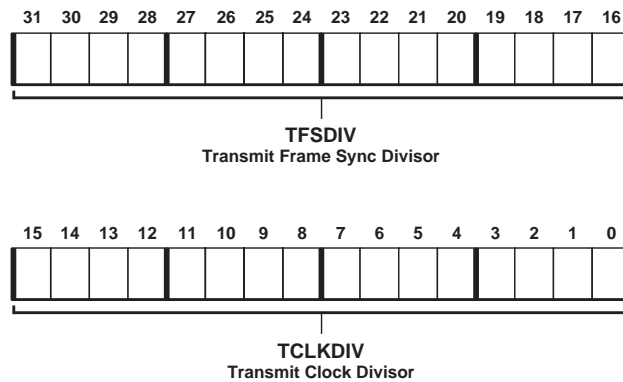


Figure 10.4 TDIV0, TDIV1 Transmit Divisor Registers

10 Serial Ports

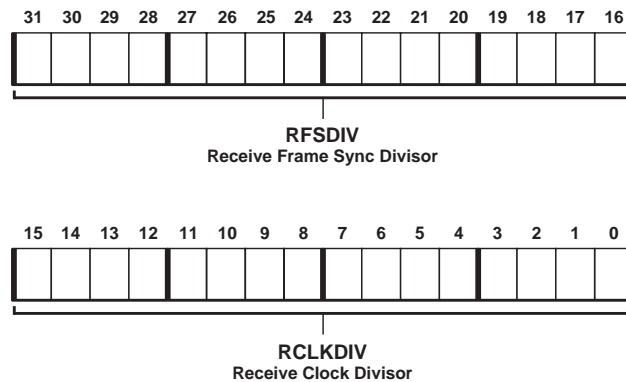


Figure 10.5 RDIV0, RDIV1 Receive Divisor Registers

TCLKDIV and RCLKDIV specify how many times the ADSP-2106x system clock (CLKIN) is divided to generate the transmit and receive clocks. The divisor is a 16-bit value, allowing a wide range of serial clock rates. The following equation is used to calculate the serial clock frequency:

$$\text{serial clock frequency} = \frac{f_{\text{CLKIN}}}{(\text{xCLKDIV} + 1)}$$

The maximum serial clock frequency is equal to the ADSP-2106x system clock frequency, which occurs when xCLKDIV is set to zero.

Use the following equation to determine the value of xCLKDIV to use, given the CLKIN frequency and desired serial clock frequency:

$$\text{xCLKDIV} = \frac{f_{\text{CLKIN}}}{\text{serial clock frequency}} - 1$$

TFSDIV and RFSDIV specify how many transmit or receive clock cycles are counted before generating a TFS or RFS pulse (when the frame sync is internally generated). In this way a frame sync can be used to initiate periodic transfers. The counting of serial clock cycles applies to either internally or externally generated serial clocks.

Serial Ports 10

The formula for the number of cycles between frame synch pulses is:

of serial clock cycles between frame sync assertions = xFSDIV + 1

Use the following equation to determine the value of xFSDIV to use, given the serial clock frequency and desired frame sync frequency:

$$\text{xFSDIV} = \frac{\text{serial clock frequency}}{\text{frame sync frequency}} - 1$$

The frame sync would thus be continuously active if xFSDIV=0. However, the value of xFSDIV should not be less than the serial word length minus one (the value of the SLEN field in the transmit or receive control register), as this may cause an external device to abort the current operation or cause other unpredictable results. If the serial port is not being used, the xFSDIV divisor can be used as a counter for dividing an external clock or for generating a periodic pulse or periodic interrupt. The serial port must be enabled for this mode of operation to work.

10.3.4.1 Maximum Clock Rate Restrictions

Caution should be exercised when operating with externally generated transmit clocks near the frequency of the ADSP-2106x system clock. There is a delay between when the clock arrives at the TCLKx pin and when data is output—this delay may limit the receiver's speed of operation. Refer to the data sheet for exact timing specifications. For reliable operation, it is recommended that full-speed serial clocks only be used when *receiving with an externally generated clock and externally generated frame sync (ICLK=0, IRFS=0)*.

Externally-generated *late* transmit frame syncs also experience a delay from when they arrive to when data is output—this can also limit the maximum serial clock speed. Refer to the data sheet for exact timing specifications.

The serial ports handle word lengths of 3 to 32 bits, but transmitting or receiving words smaller than 7 bits at the full clock rate of the ADSP-2106x may cause incorrect operation when DMA chaining is enabled. Chaining disables the ADSP-2106x's internal I/O bus for several cycles while the new TCB parameters are being loaded. Receive data may be lost (i.e. overwritten) during this period.

10 Serial Ports

10.4 DATA WORD FORMATS

The format of the data words transmitted over the serial ports is configured by the DTYPE, SENDN, SLEN, and PACK bits of the STCTLx and SRCTLx control registers.

10.4.1 Word Length

The serial ports handle word lengths of 3 to 32 bits. The word length is configured in the 5-bit SLEN field in the STCTLx and SRCTLx control registers. The value of SLEN is equal to the word length minus one:

$$\text{SLEN} = \text{Serial Word Length} - 1$$

The SLEN value should not be set to zero or one. Words smaller than 32 bits are right-justified in the RX and TX buffers, residing in the least significant bit positions.

Transmitting or receiving words smaller than 7 bits at the full clock rate of the ADSP-2106x may cause incorrect operation when DMA chaining is enabled. Chaining disables the ADSP-2106x's internal I/O bus for several cycles while the new TCB parameters are being loaded. Receive data may be lost (i.e. overwritten) during this period.

10.4.2 Endian Format

Endian format determines whether the serial word is transmitted MSB-first or LSB-first. Endian format is selected by the SENDN bit in the STCTLx and SRCTLx control registers. When SENDN=0, serial words are transmitted (or received) MSB-first. When SENDN=1, serial words are transmitted (or received) LSB-first.

10.4.3 Data Packing & Unpacking

Received data words of 16 bits or less may be packed into 32-bit words, and 32-bit words being transmitted may be unpacked into 16-bit words. Word packing and unpacking is selected by the PACK bit in the SRCTLx and STCTLx control registers.

When PACK=1 in the receive control register (SRCTLx), two successive words received are packed into a single 32-bit word.

When PACK=1 in the transmit control register (STCTLx), each 32-bit word is unpacked and transmitted as two 16-bit words.

Serial Ports 10

The first 16-bit (or smaller) word is right-justified in bits 15-0 of the packed word, and the second 16-bit (or smaller) word is right-justified in bits 31-16. This applies for both receive (packing) and transmit (unpacking) operations. Companding may be used when word packing or unpacking is being used.

When serial port data packing is enabled, the transmit and receive interrupts are generated for the 32-bit packed words, not for each 16-bit word.

Hint: When 16-bit received data is packed into 32-bit words and stored in normal word space in ADSP-2106x internal memory, the 16-bit words can be read or written with short word space addresses.

10.4.4 Data Type

The DTYPE field of the STCTLx and SRCTLx control registers specifies one of four data formats (*for non-multichannel operation*):

DTYPE	Data Formatting
00	Right-justify, zero-fill unused MSBs
01	Right-justify, sign-extend into unused MSBs
10	Compand using μ -law
11	Compand using A-law

These formats are applied to serial data words loaded into the RX and TX buffers. (TX data words are not actually zero-filled or sign-extended, since only the significant bits are transmitted.)

For multichannel operation, the companding selection and MSB-fill selection is independent:

DTYPE	Data Formatting
x0	Right-justify, zero-fill unused MSBs
x1	Right-justify, sign-extend into unused MSBs
0x	Compand using μ -law
1x	Compand using A-law

Linear transfers will occur if the channel is active but companding is not selected for that channel. Companded transfers will occur if the channel is active and companding is selected for that channel. The multichannel compand select registers, MTCCSx and MRCCSx, are used to specify which transmit and receive channels are companded. See “Channel Selection Registers” in the “Multichannel Operation” section of this chapter.

10 Serial Ports

Transmit sign extension is selected by bit 0 of DTYPE in the STCTLx register and is common to all transmit channels. Receive sign extension is selected by bit 0 of DTYPE in the SRCTLx register and is common to all receive channels. If bit 0 of DTYPE is set, sign extension will occur on selected channels that do not have companding selected. If this bit is not set, the word will contain 0s in the MSBs.

10.4.5 Companding

Companding (*compressing/expanding*) is the process of logarithmically encoding and decoding data to minimize the number of bits that must be sent. The ADSP-2106x serial ports support the two most widely used companding algorithms, A-law and μ -law, performed according to the CCITT G.711 specification. The type of companding can be selected independently for each SPORT. Companding is selected by the DTYPE field of the STCTLx and SRCTLx control registers.

When companding is enabled, the data in the RX0 or RX1 buffer is the right-justified, sign-extended expanded value of the eight LSBs received. Likewise, a write to TX0 or TX1 causes the 32-bit value to be compressed to eight LSBs (sign-extended to the width of the transmit word) before it is transmitted. If the 32-bit value is greater than the 13-bit A-law or 14-bit μ -law maximum, it is automatically compressed to the maximum value.

Because the values in the TX and RX buffers are actually companded in-place, the companding hardware can be used without transmitting (or receiving) any data, for example during testing or debugging. This operation requires a single cycle of overhead, as described below. To compand data in-place, without transmitting, the following sequence of operations should be used:

1. Enable companding in the DTYPE field of the STCTLx transmit control register.
2. Write a 32-bit data word to TX. (*The companding is calculated in this cycle.*)
3. Wait one cycle. A NOP instruction can be used to do this; if a NOP is not inserted, the ADSP-2106x core will be held off for one cycle anyway. (*This allows the serial port companding hardware to reload TX with the companded value.*)
4. Read the 8-bit companded value from TX.

Serial Ports 10

To expand data in-place, the same sequence of operations is used but with RX rather than TX. When expanding data in this way, be sure that the serial word length (SLEN) is set appropriately in the SRCTLx control register.

With companding enabled, interfacing the ADSP-2106x serial port to a codec requires little additional programming effort. If companding is not selected, there are two formats available for received data words of fewer than 32 bits: one that fills unused MSBs with zeros, and another that sign-extends the MSB into the unused bits (see the previous section, “Data Type”).

10.5 CLOCK SIGNAL OPTIONS

Each serial port has a transmit clock signal (TCLKx) and a receive clock signal (RCLKx). The clock signals are configured by the ICLK and CKRE bits of the STCTLx and SRCTLx control registers. Serial clock frequency is configured in the TDIVx and RDIVx registers.

The receive clock pin may be tied to the transmit clock if a single clock is desired for both input and output.

10.5.1 Internal vs. External Clocks

Both transmit and receive clocks can be independently generated internally or input from an external source. The ICLK bit of the STCTLx and SRCTLx control registers determines the clock source.

When ICLK=1, the clock signal is generated internally by the ADSP-2106x and the TCLKx or RCLKx pins will be an output. The clock frequency is determined by the value of the serial clock divisor (TCLKDIV or RCLKDIV) in the TDIVx or RDIVx registers.

When ICLK=0, the clock signal is accepted as an input on the TCLKx or RCLKx pins, and the serial clock divisors in the TDIVx/RDIVx registers are ignored. The externally generated serial clock need not be synchronous with the ADSP-2106x system clock.

10 Serial Ports

10.6 FRAME SYNC OPTIONS

Framing signals indicate the beginning of each serial word transfer. The framing signals for each serial port are TFS (transmit frame synchronization) and RFS (receive frame synchronization). A variety of framing options are available; these options are configured in the serial port control registers. The TFS and RFS signals of a serial port are independent and are separately configured in the control registers.

10.6.1 Framed vs. Unframed

The use of frame sync signals is optional in serial port communications. The TFSR (transmit frame sync required) and RFSR (receive frame sync required) control bits determine whether frame sync signals are required. These bits are located in the the STCTLx and SRCTLx control registers.

When TFSR=1 or RFSR=1, a frame sync signal is required for every data word. To allow continuous transmitting from the ADSP-2106x, each new data word must be loaded into the TX buffer before the previous word is shifted out and transmitted. (See “Data-Independent Frame Syncs” in this chapter.)

When TFSR=0 or RFSR=0, the corresponding frame sync signal is not required. A single frame sync is needed to initiate communications but is ignored after the first bit is transferred. Data words are then transferred continuously, unframed. (**Caution:** When DMA is enabled in this mode, with frame syncs not required, DMA requests may be held off by chaining or may not be serviced frequently enough to guarantee continuous unframed data flow.)

Figure 10.6 illustrates framed serial transfers, which have the following characteristics:

- TFSR and RFSR bits in STCTLx, SRCTLx control registers determine framed or unframed mode.
- Framed mode requires a framing signal for every word. Unframed mode ignores framing signal after first word.
- Unframed mode is appropriate for continuous reception.
- Active-low or active-high frame syncs selected with LTFS and LRFS bits of STCTLx, SRCTLx control registers.

Serial Ports 10

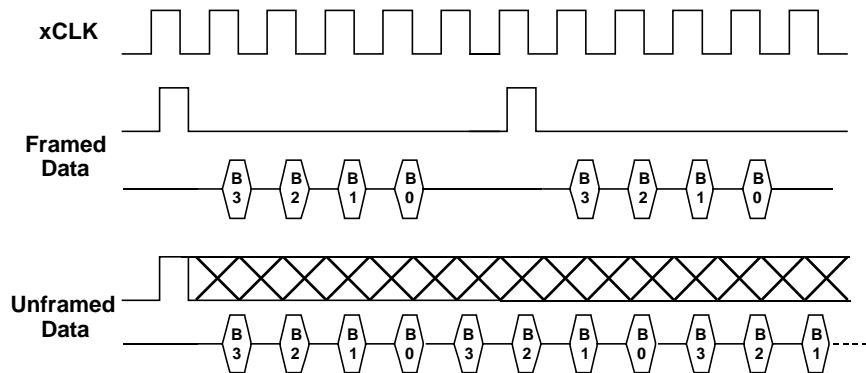


Figure 10.6 Framed vs. Unframed Data

10.6.2 Internal vs. External Frame Syncs

Both transmit and receive frame syncs can be independently generated internally or input from an external source. The ITFS and IRFS bits of the STCTLx and SRCTLx control registers determine the frame sync source.

When ITFS=1 or IRFS=1, the corresponding frame sync signal is generated internally by the ADSP-2106x and the TFSx pin or RFSx pin will be an output. The frequency of the frame sync signal is determined by the value of the frame sync divisor (TFSDIV or RFSDIV) in the TDIVx or RDIVx registers.

When ITFS=0 or IRFS=0, the corresponding frame sync signal is accepted as an input on the TFSx pin or RFSx pins, and the frame sync divisors in the TDIVx/RDIVx registers are ignored.

All of the various frame sync options are available whether the signal is generated internally or externally.

10 Serial Ports

10.6.3 Active Low vs. Active High Frame Syncs

Frame sync signals may be either active high or active low (i.e. inverted). The LTFS and LRFS bits of the STCTLx and SRCTLx control registers determine the frame syncs' logic level.

When LTFS=0 or LRFS=0, the corresponding frame sync signal will be active high.

When LTFS=1 or LRFS=1, the corresponding frame sync signal will be active low.

Active high frame syncs are the default; the LTFS and LRFS bits are initialized to 0 after a processor reset.

10.6.4 Sampling Edge For Data & Frame Syncs

Data and frame syncs can be sampled on either the rising or falling edges of the serial port clock signals. The CKRE bit of the STCTLx and SRCTLx control registers selects the sampling edge.

For transmit data and frame syncs, setting CKRE=1 in STCTLx selects the rising edge of TCLKx. CKRE=0 selects the falling edge. Note that data and frame sync signals will change state on the clock edge that is not selected.

For receive data and frame syncs, setting CKRE=1 in SRCTLx selects the rising edge of RCLKx. CKRE=0 selects the falling edge.

The transmit and receive functions of two serial ports connected together, for example, should always select the same value for CKRE so that any internally generated signals are driven on one edge and any received signals are sampled on the opposite edge.

Serial Ports 10

10.6.5 Early vs. Late Frame Syncs

Frame sync signals can occur during the first bit of each data word (“late”) or during the serial clock cycle immediately preceding the first bit (“early”). The LAFS bit of the STCTLx and SRCTLx control registers configures this option.

When LAFS=0, *early* frame syncs are configured; this is the normal mode of operation. In this mode, the first bit of the transmit data word is available (and the first bit of the receive data word is latched) in the serial clock cycle *after* the frame sync is asserted, and the frame sync is not checked again until the entire word has been transmitted (or received). (In multichannel operation, this is the case when frame delay is 1.)

If data transmission is continuous in early framing mode (i.e. the last bit of each word is immediately followed by the first bit of the next word), then the frame sync signal occurs during the last bit of each word. Internally generated frame syncs are asserted for one clock cycle in early framing mode.

When LAFS=1, *late* frame syncs are configured; this is the alternate mode of operation. In this mode, the first bit of the transmit data word is available (and the first bit of the receive data word is latched) in the *same* serial clock cycle that the frame sync is asserted. (In multichannel operation, this is the case when frame delay is zero.) Receive data bits are latched by serial clock edges, but the frame sync signal is only checked during the first bit of each word. Internally generated frame syncs remain asserted for the entire length of the data word in late framing mode. Externally generated frame syncs are only checked during the first bit.

10 Serial Ports

Figure 10.7 illustrates the two modes of frame signal timing:

- LAFS bits of STCTLx, SRCTLx control registers. LAFS=0 for early frame syncs, LAFS=1 for late frame syncs.
- Early framing: frame sync precedes data by one cycle. Late framing: frame sync checked on first bit only.
- Data transmitted MSB-first (SENDN=0) or LSB-first (SENDN=1).
- Frame sync and clock generated internally or externally.

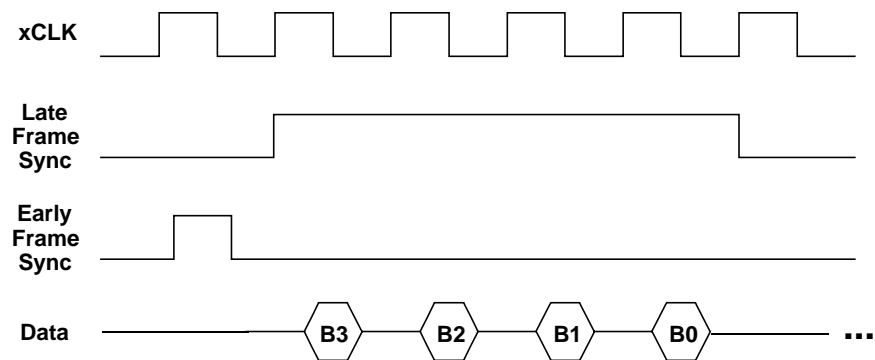


Figure 10.7 Normal vs. Alternate Framing

10.6.6 Data-Independent Transmit Frame Sync

Normally the internally generated transmit frame sync signal (TFS) is output only when the TX buffer has data ready to transmit. The DITFS mode (data-independent transmit frame sync) allows the continuous generation of the TFS signal, with or without new data. The DITFS bit of the STCTLx control register configures this option.

When DITFS=0, the internally generated TFS is only output when a new data word has been loaded into the TX buffer. Once data is loaded into TX, it is not transmitted until the next TFS is generated. This mode of operation allows data to be transmitted only at specific times.

Serial Ports 10

When DITFS=1, the internally generated TFS is output at its programmed interval regardless of whether new data is available in the TX buffer. Whatever data is present in TX will be retransmitted with each assertion of TFS. The TUVF transmit underflow status bit (in the STCTLx control register) will be set when this occurs (i.e. when old data is retransmitted). The TUVF status bit is also set if the TX buffer does not have new data when an externally generated TFS occurs. Note that in this mode of operation, the first internally generated TFS will be delayed until data has been loaded into the TX buffer.

If the internally generated TFS is used, a single write to the TX data register is required to start the transfer.

10.7 MULTICHANNEL OPERATION

The ADSP-2106x serial ports offer a multichannel mode of operation which allows the SPORT to communicate in a time-division-multiplexed (TDM) serial system. In multichannel communications, each data word of the serial bit stream occupies a separate channel—each word belongs to the next consecutive channel so that, for example, a 24-word block of data contains one word for each of 24 channels.

The serial port can automatically select words for particular channels while ignoring the others. Up to 32 channels are available for transmitting or receiving—each SPORT can receive and transmit data selectively from any of the 32 channels. In other words, the SPORT can do any of the following on each channel:

1. transmit data,
2. receive data,
3. transmit and receive data, or
4. do nothing

Data companding and DMA transfers can also be used in multichannel mode.

The DT pin is always driven, i.e. not tristated, if the serial port is enabled (SPEN=1 in the STCTLx control register), unless it is in multichannel mode and an inactive time slot occurs.

Note that (in multichannel mode) the TCLKx pin is always an input and must be connected to its corresponding RCLKx pin.

10 Serial Ports

Figure 10.8 shows example timing for a multichannel transfer, which have the following characteristics:

- Uses TDM method where serial data is sent or received on different channels sharing the same serial bus.
- The number of channels is selected with the NCH bits of SRCTLx:
NCH=(# of channels) - 1.
- Can independently select transmit and receive channels.
- RFS signal start of frame.
- TFS is used as “Transmit Data Valid” for external logic; active only during transmit channels.
- Example: Receive on channels 0 and 2.
Transmit on channels 1 and 2.

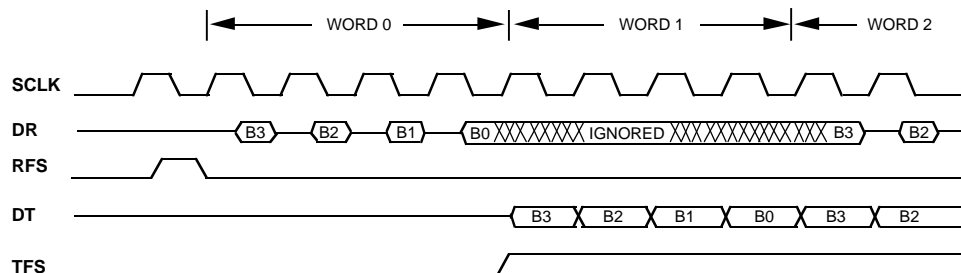


Figure 10.8 Multichannel Operation

10.7.1 Frame Syncs In Multichannel Mode

All receiving and transmitting devices in a multichannel system must have the same timing reference. The RFS signal is used for this reference, indicating the start of a block (or frame) of multichannel data words.

When multichannel mode is enabled on a SPORT, both the transmitter and receiver use RFS as a frame sync. This is true whether RFS is generated internally or externally. The RFS signal is used to synchronize the channels and restart each multichannel sequence. RFS assertion occurs the beginning of the channel 0 data word.

TFS is used as a *transmit data valid* signal which is active during transmission of an enabled word. Since the serial port's DTx pin is tristated when the time slot is not active, the TFS signal specifies

Serial Ports 10

whether or not DTx is being driven by the ADSP-2106x. The ADSP-2106x drives TFS in multichannel mode whether or not ITFS is cleared. After the TX transmit buffer is loaded, transmission begins and the TFS signal is generated. When serial port DMA is being used, this may happen several cycles after the multichannel transmission is enabled. If a deterministic start time is required, the TX buffer should be preloaded.

Note: TFS is normally left unconnected in multichannel mode, and the RFS pins of the serial port(s) are usually connected together.

10.7.2 Multichannel Control Bits In STCTL, SRCTL

The STCTLx and SRCTLx control registers contain several bits used to enable and configure multichannel operations.

10.7.2.1 Multichannel Enable

Multichannel mode is enabled by setting the MCE bit in the SRCTLx control register.

When MCE=1, multichannel operation is enabled.

When MCE=0, all multichannel operations are disabled.

Multichannel operation is activated three cycles after MCE is set. Internally generated frame sync signals activate four cycles after MCE is set.

Setting the MCE bit enables multichannel operation for both receive *and* transmit sides of the SPORT. A transmitting SPORT must therefore be in multichannel mode if the receiving SPORT is in multichannel mode.

10.7.2.2 Number Of Channels

The number of channels used in multichannel operation is selected by the 5-bit NCH field in the SRCTLx control register. NCH should be set to the actual number of channels minus one:

$NCH = \text{Number of Channels} - 1$

10.7.2.3 Current Channel Indicator

The 5-bit CHNL field in the STCTLx control register indicates which channel is currently selected during multichannel operation. This field is a read-only status indicator. CHNL(4:0) increments modulo NCH(4:0) as each channel is serviced.

10 Serial Ports

10.7.2.4 Multichannel Frame Delay

The 4-bit MFD field in the STCTLx control register specifies a delay between the frame sync pulse and the first data bit in multichannel mode. The value of MFD is the number of serial clock cycles of the delay. Multichannel frame delay allows the processor to work with different types of T1 interface devices.

A value of zero for MFD causes the frame sync to be concurrent with the first data bit. The maximum value allowed for MFD is 15. A new frame sync may occur before data from the last frame has been received, because blocks of data occur back to back.

A multichannel frame delay of at least one should be used when the ADSP-2106x is generating frame syncs for the multichannel system and the serial clock of the system is equal to CLKIN (the processor clock). If MFD is not set to at least one, the master ADSP-2106x in a multiprocessing system will not recognize the first frame sync after multichannel operation is enabled. All succeeding frame syncs will be recognized normally, however.

10.7.3 Channel Selection Registers

Specific channels can be individually enabled or disabled to select which words are received and transmitted during multichannel communications. Data words from the enabled channels are received or transmitted, while disabled channel words are ignored. Up to 32 channels are available for transmitting and up to 32 channels for receiving.

The multichannel selection registers are used to enable and disable individual channels. The registers for each serial port are as follows:

<i>Register Name</i>	<i>Function</i>
MTCSx	Multichannel Transmit Select—specifies the active transmit channels
MRCSx	Multichannel Receive Select—specifies the active receive channels
MTCCSx	Multichannel Transmit Compand Select—specifies which active transmit channels are companded
MRCCSx	Multichannel Receive Compand Select—specifies which active receive channels are companded

Each register has 32 bits, corresponding the 32 channels. Setting a bit enables that channel so that the serial port will select its word from the multiple-word block of data (for either receive or transmit). For example, setting bit 0 selects word 0, setting bit 12 selects word 12, and so on.

Serial Ports 10

Setting a particular bit to 1 in the MTCSx register causes the serial port to transmit the word in that channel's position of the data stream. Clearing the bit to 0 in the MTCSx register causes the serial port's DT (data transmit) pin to tristate during the time slot of that channel.

Setting a particular bit to 1 in the MRCSx register causes the serial port to receive the word in that channel's position of the data stream; the received word is loaded into the RX buffer. Clearing the bit to 0 in the MRCSx register causes the serial port to ignore the data.

Companding may be selected on a per-channel basis. The MTCCSx and MRCCSx registers are used to specify companding for any active channels. Setting a bit to 1 in these registers causes the data to be companded. A-law or μ -law companding is selected with the DTYPE bit 1 in the STCTLx and SRCTLx control registers.

10.7.4 SPORT Receive Comparison Registers

On the ADSP-21061, two sets of registers aid multiprocessor communications when using multichannel mode (MCE=1) through the serial ports. These 32-bit registers are the Receive Comparison (KEYWDx) registers and the Receive Comparison Mask (KEYMASKx) registers.

The KEYWD0 or KEYWD1 register stores the pattern to be matched with the incoming data. The corresponding KEYMASK0 or KEYMASK1 register specifies which of the bits in the received data should be compared. Setting a KEYMASKx bit (=1) masks the corresponding bit in the KEYWDx register, disabling its comparison.

The processor receiving the data compares it with the data in the KEYWDx register. Depending on the comparison results, the received data is accepted or ignored. If accepted, the receiver requests—based on the setting to the SRCTL register—a DMA transfer to internal memory or generates an interrupt.

In addition to the MCE setting, the following bits in the SRCTL register control the operation of Receive Comparison:

<u>IMODE</u> <u>(Bit 15)</u>	<u>IMAT</u> <u>(Bit 20)</u>	<u>Operation</u>
0	x	Receive comparison disabled
1	0	Accept receive data if the KEYWD comparison is false
1	1	Accept receive data if the KEYWD comparison is true

10 Serial Ports

When receive comparison is enabled, companding is disabled on the transmitter and receiver. The MTCCSx register, which selects multichannel companding when receive comparison is disabled, determines whether the DSP performs a KEYWD comparison for the enabled received channels. If the MTCCSx bit for a particular channel is '0,' the processor does not perform a comparison and always accepts the receive data on that channel. If the MTCCSx bit for a particular channel is '1,' the processor performs the comparison and accepts (or rejects) the receive data, depending on the result of the comparison and IMAT setting in the SRCTLx register.

The receive comparison feature lets the ADSP-21061's SPORTS generate a DMA request or an interrupt when the received data matches a specified condition on a specified channel in multichannel mode. Without this feature, the SPORT would interrupt the processor every time data was received and the processor would be required to check if the data was meant for it or not. It is possible that most of the time the data being sent is not meant for the processor. With the receive comparison feature, the SPORT on a particular processor can be programmed to interrupt only on messages meant for that processor.

As a receive comparison example, consider four ADSP-21061s (A, B, C, and D) which use SPORT0 (in multichannel mode) for interprocessor communication. Channels 0, 1, 2, and 3 are used respectively by A, B, C, and D to transmit control information between the processors. Channels 4 through 10, 11 through 17, 18 through 24, and 25 through 31 are used respectively by A, B, C, and D to transmit data.

Because channels 0 through 3 are used to send control information between the processors, the comparisons for incoming data is enabled only for these channels. Initially, channels 4 through 31 may have receive disabled. For this example, consider communication between processors A and B only. The key word for comparison is programmable; in this example, processor B can check for the key word "START TRANSMIT TO B",

Serial Ports 10

Processor B can check for this key word as follows:

1. Set the KEYWD register to "START TRANSMIT TO B"
2. Clear bits 31:16 of the KEYMASK register to 0 and set the other bits to 1
This step enables comparison only for bits 31:16. So, assume that the code for "START TRANSMIT TO B" only uses bits 31:16 and bits 15:0 indicate the source of the transmission and the data channels.
3. Set bits 15 and 20 of the SRCTL register to 1
This step enables the SPORT to generate an interrupt or DMA request only if the incoming data matches the KEYWD.
4. Set bits 0 through 3 of the Transmit Compand Channel Selector register to 1 and clear the remaining bits to 0
This step enables comparison only on channels 0 through 3.

Until it receives the "START TRANSMIT TO B" keyword, processor B ignores all transmissions that it receives. When processor A wants to send data to B, it sends the "START TRANSMIT TO B" keyword on channel 0. When receive comparison on processor B recognizes the "START TRANSMIT TO B" keyword, the SPORT interrupts processor B. Then, processor B analyzes the remaining 16-bits, determining that the source is processor A and the data is on channels 4 through 10. Because processor A is using channels 4 through 10 to transmit data, processor B enables receive channels 4 through 10 and sends a "READY TO RECEIVE DATA" message to processor A, using channel 1. After processor A receives this message, it sends the data on channels 4 through 10. If the transfer protocol uses a fixed number of bytes in each message, processor B could send back a checksum message to processor A after receiving A's message, confirming that the data transferred accurately.

10.8 TRANSFERRING DATA BETWEEN SPORTS AND MEMORY

Transmit and receive data can be transferred between the ADSP-2106x serial ports and on-chip memory in one of two ways, with single-word transfers or with DMA block transfers. Both methods are interrupt-driven, using the same internally generated interrupts.

10 Serial Ports

When serial port DMA is *not* enabled in the STCTLx or SRCTLx control registers, the SPORT generates an interrupt every time it has received a data word or has started to transmit a data word. SPORT DMA provides a mechanism for receiving or transmitting an entire block of serial data before the interrupt is generated. The ADSP-2106x's on-chip DMA controller handles the DMA transfer, allowing the processor core to continue running until the entire block of data is transmitted or received. Service routines can then operate on the block of data rather than on single words, significantly reducing overhead.

10.8.1 DMA Block Transfers

The ADSP-2106x's on-chip DMA controller allows automatic DMA transfers between internal memory and the two serial ports. There are four DMA channels for serial port operations—each SPORT has one channel for receiving data and one for transmitting data. The serial port DMA channels are numbered as follows:

- DMA Channel 0 – SPORT0 Receive
- DMA Channel 1 – SPORT1 Receive (or Link Buffer 0)
- DMA Channel 2 – SPORT0 Transmit
- DMA Channel 3 – SPORT1 Transmit (or Link Buffer 1)

Note that channels 1 and 3 are shared between SPORT1 and link buffers 0 and 1. The SPORT DMA channels are assigned higher priority than all other DMA channels (i.e. for link ports and the external port) because of their relatively low service rate and their inability to hold off incoming data. Having higher priority causes the SPORT DMA transfers to be performed first when multiple DMA requests occur in the same cycle.

Although the DMA transfers are always performed with 32-bit words, the serial ports can handle word sizes from 3 to 32 bits. If the serial words are 16 bits or smaller, they can be packed into 32-bit words for each DMA transfer; this is configured by the PACK bit of the STCTLx and SRCTLx control registers. When serial port data packing is enabled (PACK=1), the transmit and receive interrupts are generated for the 32-bit packed words, not for each 16-bit word.

The following sections present an overview of serial port DMA operations; some additional details are covered in the *DMA* chapter of this manual.

Serial Ports 10

10.8.1.1 SPORT DMA Channel Setup

Each SPORT DMA channel has an enable bit (SDEN) in the STCTLx and SRCTLx control registers of the two serial ports. When DMA is not enabled for a particular channel, the SPORT generates an interrupt every time it has received a data word or has started to transmit a data word (see “Single-Word Transfers” later in this chapter). Each channel also has a DMA chaining enable bit (SCHEN) in the control registers (see “SPORT DMA Chaining” later in this chapter).

A serial port DMA channel is set up by writing a set of memory buffer parameters to the SPORT DMA parameter registers. The II, IM, and C registers must be loaded with a starting address for the buffer, an address modifier, and a word count, respectively. The programming of these registers can be done from the ADSP-2106x core processor or from an external processor.

Once serial port DMA is set up and enabled, data words received in the RX buffer are automatically transferred to the buffer in internal memory. Likewise, when the serial port is ready to transmit data, a word is automatically transferred from memory to the TX buffer. These transfers continue until the entire data buffer is received or transmitted (i.e. when the count register reaches zero).

When the count register of an active DMA channel reaches zero, the corresponding interrupt is generated.

10.8.1.2 SPORT DMA Parameter Registers

A DMA channel consists of a set of parameter registers that implement a data buffer in internal memory, plus hardware used by the serial port to request DMA service. The parameter registers for each SPORT DMA channel are shown in Tables 10.8 and 10.9. These registers are part of the memory-mapped IOP register set of the ADSP-2106x.

The DMA channels operate in a similar fashion as the ADSP-2106x’s Data Address Generators (DAGs). Each channel has an index register (II) and a modify register (IM) which are used to set up a data buffer in internal memory. The index register must be initialized with a starting address for the data buffer. After each serial I/O word is transferred to or from the SPORT, the DMA controller adds the modify value to the index register to generate the address for the next DMA transfer. The modify value in the IM register is a signed integer, which allows both incrementing and decrementing.

10 Serial Ports

Each DMA channel has a count register (C) which must be initialized with a word count to be transferred. The count register is decremented after each DMA transfer on that channel; when the count reaches zero, the interrupt for that channel is generated and the channel is automatically disabled.

Each SPORT DMA channel also has a chain pointer register (CP), a general-purpose register (GP), and two registers used for two-dimensional array addressing in mesh multiprocessing applications (DA, DB). The CP register is used in chained DMA operations, as described below in “SPORT DMA Chaining”, and the GP register can be used for any purpose. The DA and DB registers may be used as general-purpose registers in standard, non-mesh-multiprocessing DMA operations.

<u>Register</u>	<u># of Bits</u>	<u>Function</u>
Iix	17	Index (starting address for data buffer)
IMx	16	Index Modifier (address increment)
Cx	16	Count (number of words to transfer)
CPx	18*	Chain Pointer (address of next set of buffer parameters)
GPx	17	General-Purpose or 2D DMA
DBx	16	General-Purpose or 2D DMA
DAx	16	General-Purpose or 2D DMA

Table 10.8 Parameter Registers For Each SPORT DMA Channel

* Lower 17 bits contains memory address of the next set of parameters for chained DMA operations. Most significant bit (bit 17) is the PCI bit (Program-Controlled Interrupts), which determines whether the DMA interrupts occur at the completion of each DMA sequence.

Serial Ports 10

Memory

<u>Address</u>	<u>Register</u>	<u>Channel Number & Function</u>
0x0060	II0	DMA Channel 0 – SPORT0 Receive
0x0061	IM0	DMA Channel 0 – SPORT0 Receive
0x0062	C0	DMA Channel 0 – SPORT0 Receive
0x0063	CP0	DMA Channel 0 – SPORT0 Receive
0x0064	GP0	DMA Channel 0 – SPORT0 Receive
0x0065	DB0	DMA Channel 0 – SPORT0 Receive
0x0066	DA0	DMA Channel 0 – SPORT0 Receive
0x0067	<i>reserved</i>	
0x0068	II1	DMA Channel 1 – SPORT1 Receive (or Link Buffer 0)
0x0069	IM1	DMA Channel 1 – SPORT1 Receive (or Link Buffer 0)
0x006A	C1	DMA Channel 1 – SPORT1 Receive (or Link Buffer 0)
0x006B	CP1	DMA Channel 1 – SPORT1 Receive (or Link Buffer 0)
0x006C	GP1	DMA Channel 1 – SPORT1 Receive (or Link Buffer 0)
0x006D	DB1	DMA Channel 1 – SPORT1 Receive (or Link Buffer 0)
0x006E	DA1	DMA Channel 1 – SPORT1 Receive (or Link Buffer 0)
0x006F	<i>reserved</i>	
0x0070	II2	DMA Channel 2 – SPORT0 Transmit
0x0071	IM2	DMA Channel 2 – SPORT0 Transmit
0x0072	C2	DMA Channel 2 – SPORT0 Transmit
0x0073	CP2	DMA Channel 2 – SPORT0 Transmit
0x0074	GP2	DMA Channel 2 – SPORT0 Transmit
0x0075	DB2	DMA Channel 2 – SPORT0 Transmit
0x0076	DA2	DMA Channel 2 – SPORT0 Transmit
0x0077	<i>reserved</i>	
0x0078	II3	DMA Channel 3 – SPORT1 Transmit (or Link Buffer 1)
0x0079	IM3	DMA Channel 3 – SPORT1 Transmit (or Link Buffer 1)
0x007A	C3	DMA Channel 3 – SPORT1 Transmit (or Link Buffer 1)
0x007B	CP3	DMA Channel 3 – SPORT1 Transmit (or Link Buffer 1)
0x007C	GP3	DMA Channel 3 – SPORT1 Transmit (or Link Buffer 1)
0x007D	DB3	DMA Channel 3 – SPORT1 Transmit (or Link Buffer 1)
0x007E	DA3	DMA Channel 3 – SPORT1 Transmit (or Link Buffer 1)
0x007F	<i>reserved</i>	

Table 10.9 SPORT DMA Parameter Registers

10.8.1.3 SPORT DMA Chaining

In chained DMA operations, the ADSP-2106x automatically sets up another DMA transfer when the contents of the current buffer have been transmitted (or received). The chain pointer register (CP) is used to point to the next set of buffer parameters stored in memory. The ADSP-2106x's DMA controller automatically downloads these buffer parameters to set up the next DMA sequence. Refer to the *DMA* chapter of this manual for details on how to set up chaining parameters in memory.

10 Serial Ports

DMA chaining occurs independently for the transmit and receive channels of each serial port. Each SPORT DMA channel has a chaining enable bit (SCHEN) in the STCTLx and SRCTLx control registers. This bit must be set to 1 to enable chaining. Writing all zeros to the address field of the chain pointer register (CP) also disables chaining.

10.8.2 Single-Word Transfers

Individual data words may also be transmitted and received by the serial ports, with interrupts occurring as each 32-bit word is transmitted or received. When a serial port is enabled and DMA is disabled (in the STCTLx or SRCTLx control registers), the SPORT DMA interrupts will be generated in this way—whenever a complete 32-bit word has been received in the RX buffer, or whenever the TX buffer is not full. Single-word interrupts can be used to implement interrupt-driven I/O on the serial ports.

Whenever the ADSP-2106x core's program reads a word from a serial port's RX buffer or writes a word to its TX buffer, the buffer's *full/empty* status should first be checked in order to avoid hanging the ADSP-2106x core. (This can also happen to an external device, for example a host processor, when it is reading or writing a serial port buffer.) The *full/empty* status can be read in the RXS bits of the SRCTLx register or the TXS bits of the STCTLx register. Reading from an empty RX buffer or writing to a full TX buffer causes the ADSP-2106x (or external device) to hang, waiting for the status to change. To prevent this hang condition from occurring, the BHD (Buffer Hang Disable) bit should be set in the SYSCON register.

Multiple interrupts can occur if both SPORTs transmit or receive data in the same cycle. Any interrupt can be masked out in the IMASK register; if the interrupt is later enabled in IMASK, the corresponding interrupt latch bit in IRPTL must be cleared in case the interrupt has occurred in the meantime.

When serial port data packing is enabled (PACK=1 in the STCTLx or SRCTLx control registers), the transmit and receive interrupts are generated for the 32-bit packed words, not for each 16-bit word.

10.9 SPORT LOOPBACK

When the SPL bit (SPORT loopback) is set in the SRCTLx receive control register, the serial port is configured in an internal loopback connection. The loopback configuration allows the serial ports to be tested internally.

Serial Ports 10

When loopback is configured, the DRx, RCLKx, and RFSx signals of the receive section of the SPORT are internally connected to the DTx, TCLKx, and TFSx signals of the transmit section. The DTx, TCLKx, and TFSx signals are active and are available at their respective pins, while the DRx, RCLKx, and RFSx pins are ignored by the ADSP-2106x.

Only transmit clock and transmit frame sync options may be used in loopback mode—you must ensure that the serial port is set up correctly in the STCTLx and SRCTLx control registers. Multichannel mode is not allowed.

10.10 SPORT PIN DRIVER CONCERNS

The ADSP-2106x has very fast drivers on all output pins including the serial ports. If connections on the data, clock, or frame sync lines are longer than six inches, you should consider using a series termination for strip lines on point-to-point connections. This may be necessary even when using low-speed serial clocks, because of the edge rates.

10.11 SPORT PROGRAMMING EXAMPLES

There are three ways to control serial port communications and memory-to-SPORT data transfers: single-word transfers under core processor control with no interrupts, single-word transfers under core processor control with interrupts, and DMA transfers with interrupts. The three examples presented below illustrate each of these methods. In each example, the SPORT0 is used to transmit eight 32-bit words from a data buffer in internal memory.

Any of the three control schemes may be used in multichannel mode and with any of the serial clock and frame sync options.

10.11.1 Single-Word Transfers Without Interrupts

The ADSP-2106x processor core will stall (i.e. hang) when it attempts to write data to a full TX buffer or read data from an empty RX buffer. This provides a very simple method of controlling the SPORT—placing the instruction that writes data to TX or reads data from RX in a loop. Program execution will stall at this instruction, until the SPORT is ready to transmit new data or has received new data.

Listing 10.1 shows the code for this example, which sets up a loop to transmit data out of SPORT0. Although this technique provides a very simple programming solution, it prevents the ADSP-2106x processor core from handling any other tasks while waiting for the serial port. The interrupt-driven technique described in the following section alleviates this.

Serial Ports 10

10.11.2 Single-Word Transfers With Interrupts

While the non-interrupt-driven solution of the previous example provides a very simple control scheme, it prevents the ADSP-2106x processor core from handling any additional tasks while it is stalled. In most real-time applications, the DSP must process data *while* new data is being received. It may also need to perform background tasks between data transfers.

In most systems, therefore, the DSP processor must be able to continue executing its program at all times. Using the serial port receive and transmit interrupts allows this to happen, by interrupting the core processor only when a new data word has been received or when a new data word can be transmitted. The interrupt service routine then performs the data transfer between internal memory and the serial port's TX or RX buffer.

Listing 10.2 below shows the code for this example. Note that the interrupt used is the *SPORT0 Transmit DMA Channel* interrupt (SPT0I)—when serial port DMA is disabled, this interrupt becomes a single-word transmit interrupt.

```
/*
_____
ADSP-2106x Interrupt-Driven SPORT Transmit Example:

This example uses interrupts to notify the core when new data
is required for serial port 0 transmit. The buffer "source"
is transmitted.
_____ */

#define N 8
#include "def21060.h" /* Use symbolic register names */

.segment/dm dm32_b1; /* Data segment name described in arch file.*/
.var source[N]= 0x11111111, 0x22222222, 0x33333333, 0x44444444,
                0x55555555, 0x66666666, 0x77777777, 0x88888888;
.endseg;

.segment/pm rst_svc; /* Reset vector from arch file.*/
    nop; /* First location is used for booting.*/
    jump start;
.endseg;

.segment/pm spt0_svc; /* Sport0 TX interrupt vector.*/
    jump s0tx;
.endseg;
```

(listing continues on next page)

10 Serial Ports

```
/*_____Main routine_____*/
.segment/pm pm48_lb0; /* Main code segment from arch file.*/
start: r0=0x00270007; /* TDIV0 Register: TCLKDIV=7,TFSDIV=39 */
      dm(TDIV0)=r0; /* sclock=CLKIN/8, framerate=sclock/20 */

      r0=0x000064f1; /* STCTL0 Register: */
      dm(STCTL0)=r0; /* SPEN=1,(SPORT enabled)*/
                    /* SLEN=15 (16-bit word)*/
                    /* ICLK=1, (internal tx clock)*/
                    /* TFSR=1, (require TFS)*/
                    /* ITFS=1, (internal TFS)*/
                    /* DITFS=0,(data dependent FS)*/
                    /* all other bits = 0 */

      b0=source; /* Pointer to source; i0=b0 automatically.*/
      l0=@source;

      bit set imask SPT0I; /* Enable Sport0 TX interrupt.*/
      bit set model IRPTEN; /* Global interrupt enable.*/

      r0=dm(i0,1); /* Write first value into TX0 to kick off sport.*/
      dm(TX0)=r0;

wait:  idle; /* Wait for SPORT0 TX interrupts.*/
      jump wait;

/*_____SPORT0 Transmit Interrupt Routine_____*/
s0tx:  rti (db);
      r0=dm(i0,1); /* Get data from source buffer */
      dm(TX0)=r0; /* Write transmit register */
.endseg;
```

Listing 10.2 Interrupt-Driven SPORT Control (Single-Word Transfers)

Serial Ports 10

10.11.3 DMA Transfers With Interrupts

This example shows how to use the ADSP-2106x's on-chip DMA controller to handle serial port I/O. The DMA controller performs the data transfers between internal memory and the SPORTs, providing the most efficient way to handle input and output of multiple-word blocks of data. Once it has been set up, the DMA controller operates independently from the ADSP-2106x processor core. It interrupts core execution only when an entire block of data has been received (or transmitted). This frees the core to continue with other tasks.

Listing 10.3 shows the code for this example, which uses the serial port's loopback mode. The program first sets up the SPORT1 DMA channels by loading values into the DMA parameter registers, then writes to the SRCTL1 and STCTL1 registers and waits to be interrupted.

```
/*  
-----  
ADSP-2106x DMA-Driven SPORT Loopback Example:  
  
This example sets up a SPORT DMA transfer and receive for serial  
port 1 in the loopback mode. The buffer "source" is DMAed out of  
the sport. The loopback mode internally attaches DT1, TFS1, and  
TCLK1 to DR1, RFS1, and RCLK1. The receive DMA places the data  
in the buffer "destination".  
-----*/  
  
#define N 8  
#include "def21060.h" /* Use symbolic register names */  
  
.segment/dm dm32_b1; /* Data segment name described in arch. file.*/  
.var source[N]= 0x11111111, 0x22222222, 0x33333333, 0x44444444,  
0x55555555, 0x66666666, 0x77777777, 0x88888888;  
.var destination[N];  
.endseg;  
  
.segment/pm rst_svc; /* Reset vector from arch. file.*/  
nop; /* First location is used for booting.*/  
jump start;  
.endseg;  
  
.segment/pm spr1_svc; /* SPORT1 rx interrupt vector.*/  
jump slrx;  
.endseg;
```

(listing continues on next page)

10 Serial Ports

```
/*_____main routine_____*/

.segment/pm pm48_1b0;      /* Main code segment from arch. file*/

start: r0=source;
      dm(II3)=r0;          /* Set DMA tx index to start of source buffer*/
      r0=destination;
      dm(III1)=r0;        /* Set DMA rx index to start of destination buffer*/
      r0=1;
      dm(IM3)=r0;         /* Set DMA modify (stride) to 1.*/
      dm(IM1)=r0;
      r0=@source;
      dm(C3)=r0;          /* Set DMA count to length of data buffer*/
      dm(C1)=r0;

      r0=0x004421f1;      /* SRCTL1 Register:          */
      dm(SRCTL1)=r0;      /* SPEN=1, (SPORT1 enabled)*/
                          /* SLEN=31, (32-bit word)*/
                          /* RFSR=1, (require RFS)*/
                          /* SDEN=1, (rx DMA enable)*/
                          /* SPL=1, (loop back DT to DR & TFS to RFS)*/

      r0=0x00270007;      /* TDIV0 Register: TCLKDIV=7, TFSDIV=39*/
      dm(TDIV1)=r0;       /* sclock=CLKIN/8, framerate=sclock/2 0 */

      r0=0x000465f1;      /* STCTL1 Register:          */
      dm(STCTL1)=r0;      /* SPEN=1, (SPORT1 enabled)*/
                          /* SLEN=31, (32-bit word)*/
                          /* ICLK=1, (internal tx clock)*/
                          /* TFSR=1, (require TFS)*/
                          /* ITFS=1, (internal TFS)*/
                          /* DITFS=0, (data dependent FS), all other bits=0*/
                          /* SDEN=1, (tx dma enable), this kicks it off*/

      bit set imask SPR1I; /* Enable SPORT1 rx interrupt*/
      bit set model IRPTEN; /* Global interrupt enable*/

wait:  idle;               /* Wait for SPORT1 rx interrupt*/
      jump wait;          /* Will end up here after entire DMA complete*/

/*_____SPORT1 Receive Interrupt Routine_____*/

slrx:  rti;                /* This interrupt will occur only once*/

.endseg;
```

Listing 10.3 SPORT DMA Example