Appendix C

Peripheral Programming for Digital Signal Processors

This appendix introduces the details involved in interfacing the peripherals described in Chapter 3 with DSP processors. We discuss some important issues in the interfacing of peripherals, which include peripheral configuration, address connections of peripheral devices with DSP processors, and steps and coding involved to initialize peripheral devices. In this appendix, we use the Texas Instruments TMS320C5402 DSK as an example in explaining the interface between the DSP processor and peripheral devices.

In the block diagram of the C5402 DSK, shown in Fig. C.1, the C5402 DSP is interfaced to the following peripheral devices:

1. Microphones/loudspeakers via TLC320AD50 analog interface circuits (AICs)
2. A host personal computer via a parallel port
3. An XDS-510 emulator via a JTAG port
4. External memories and universal asynchronous receiver/transmitter (UART) via an EMIF

The C5402 DSP processor offers 16K words of on-chip memory, up to 100 MIPS, two McBSPs, one enhanced 8-bit HPI, and EMIFs to 64K static RAM, 256K flash, a

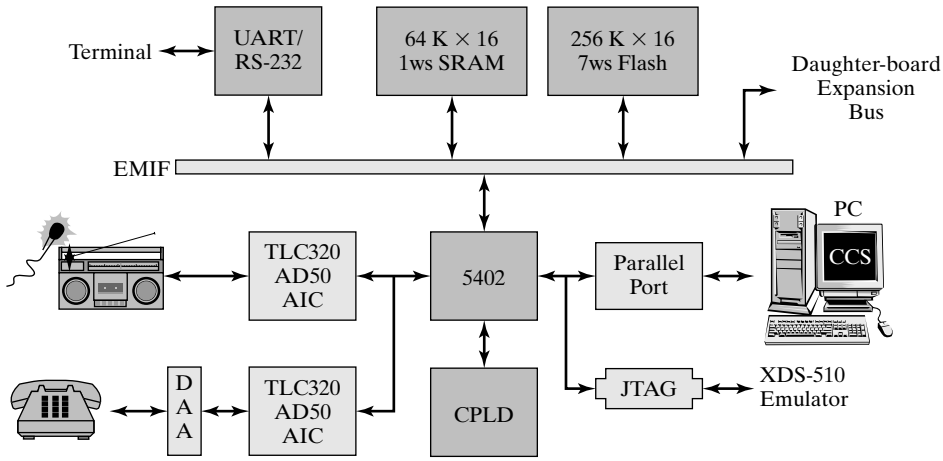

Figure C.1 Block diagram of the TMS320C5402 DSK (reprinted with permission from [1])

UART, and daughter boards. The following sections provide details regarding the CODEC and HPI interfacing on the C5402 DSK.

C.1 INTERFACING WITH THE ANALOG INTERFACE CIRCUIT OR CODER–DECODER VIA DIRECT MEMORY ACCESS

As explained in Chapter 3, DMA and McBSP peripherals provide the mechanism for transferring data in and out of the DSP processor without the intervention of the processor. DMA allows movement of data to and from internal memory, internal peripherals, or external devices in the background of CPU operation. A general explanation of DMA/McBSP was given in Section 3.6.3, and more details are provided in this section.

C.1.1 Initializing a Multichannel Buffered Serial Port

As shown in Fig. C.2, the McBSP provides a full-duplex direct interface to the AIC, the CODEC, and other serial devices. It supports a bit rate of up to half of the CPU clock rate, and it is able to support 8-, 12-, 16-, 20-, 24-, and 32-bit wordlengths. The two McBSPs used in the C5402 processor support up to 128 channels and have internal A-law and μ-law companding circuits. The McBSP also includes independent framing and clocking for receiving and transmitting. Note that the McBSP has double-buffered transmit data registers (XSR and DXR) and triple-buffered receive data registers (RSR, RBR, and DRR).

As shown in Fig. C.2, the McBSP provides several interfacing pins for a group of clocking and frame-synchronization signals: (1) receive serial data (DR), (2) transmit serial data (DX), (3) transmit clock (CLKX), (4) receive clock (CLKR), (5) transmit frame synchronization (FSX), (6) receive frame synchronization (FSR), and (7) external clock (CLKS).

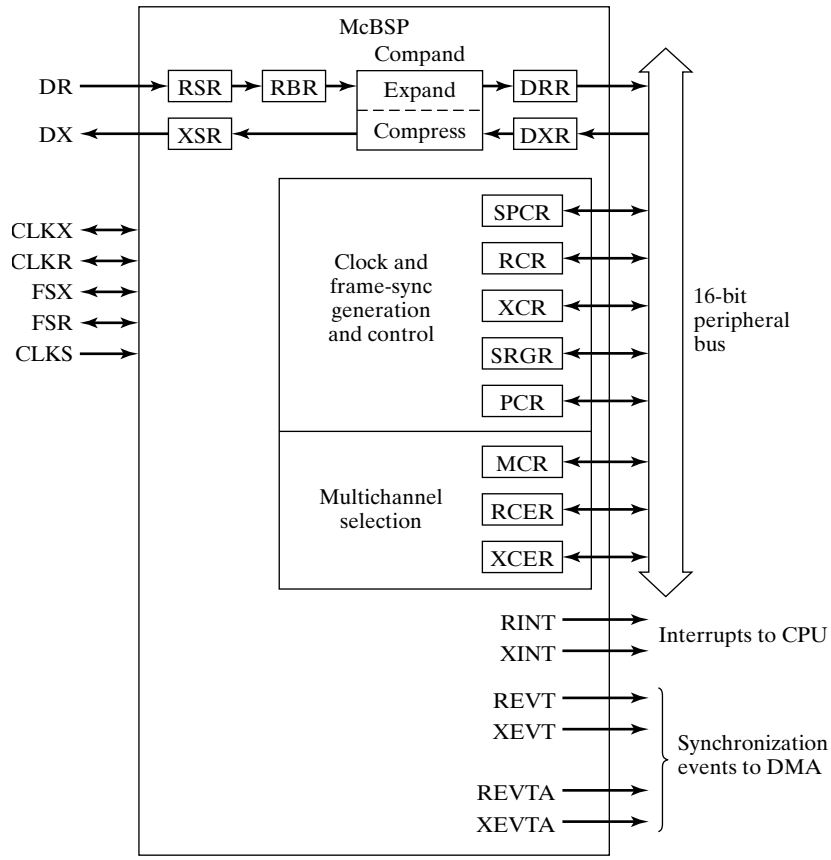

Figure C.2 Block diagram of a McBSP and its interfacing pins (reprinted with permission from [2])

A group of signals in the McBSP is used to provide receive and transmit interrupts (RINT and XINT) to the CPU. Another group that provides event synchronization to the DMA contains (1) receive synchronization event (REVT), (2) transmit synchronization event (XEVT), (3) receive synchronization event A (REVTA), and (4) transmit synchronization event A (XEVTA).

As illustrated in Fig. C.2, serial data from external devices such as a CODEC are received at the DR pin of the McBSP and are shifted into the receive shift register (RSR) before being copied to the receive buffer register (RBR). The data in RBR is then moved to the data receive register (DRR) for a read by either the processor or the DMA controller. Data to be transmitted to an external device is written to the data transmit register (DXR), copied to the transmit shift register (XSR), and shifted out to the DX pin from XSR. Note that there are two sets of these registers, and the second set of registers (RSR2, RBR2, DRR2, DXR2, and XSR2) is not used if the transmit- or receive-data wordlength is 8, 12, or 16 bits.

TABLE C.1 Description of the Pin Control Register (PCR) (reprinted with permission from [2])

Bit	Name	Function	
15–14	Reserved	Reserved	
13	XIOEN	Transmit general purpose I/O mode only when $\overline{\text{XRST}} = 0$ in SPCR	
		XIOEN = 0	DX, FSX and CLKX are configured as serial port pins and do not function as general-purpose I/Os.
		XIOEN = 1	DX pin is a general purpose output. FSX and CLKX are general purpose I/Os. These serial port pins do not perform serial port operation.
12	RIOEN	Receive general purpose I/O mode only when $\overline{\text{RRST}} = 0$ in SPCR	
		RIOEN = 0	DR, FSR, CLKR and CLKS are configured as serial port pins and do not function as general-purpose I/Os.
		RIOEN = 1	DR and CLKS pins are general purpose inputs: FSR and CLKR are general purpose I/Os. These serial port pins do not perform serial port operation. The CLKS pin is affected by a combination of $\overline{\text{RRST}}$ and RIOEN signals of the receiver.
11	FSXM	Transmit Frame-Synchronization Mode	
		FSXM = 0	Frame-synchronization signal derived from an external source
		FSXM = 1	Frame synchronization is determined by the sample rate generator frame-synchronization mode bit FSGM in SRGR2.
10	FSRM	Receive Frame-Synchronization Mode	
		FSRM = 0	Frame-synchronization pulses generated by an external device. FSR is an input pin
		FSRM = 1	Frame synchronization generated internally by sample rate generator. FSR is an output pin except when GSYNC = 1 in SRGR.
9	CLKXM	Transmitter Clock Mode	
		CLKXM = 0	Transmitter clock is driven by an external clock with CLKX as an input pin.
		CLKXM = 1	CLKX is an output pin and is driven by the internal sample rate generator.
		During SPI mode (when CLKSTP is a non-zero value):	
		CLKXM = 0	McBSP is a slave and clock (CLKX) is driven by the SPI master in the system CLKR is internally driven by CLKX.
		CLKXM = 1	McBSP is a master and generates the clock (CLKX) to drive its receive clock (CLKR) and the shift clock of the SPI-compliant slaves in the system.

(*Continued*)

TABLE C.1 (Continued)

Bit	Name	Function
8	CLKRM	Receiver Clock Mode Case 1: Digital loop back mode not set (DLB = 0) in SPCR1 CLKRM = 0 Receive clock (CLKR) is an input driven by an external clock. CLKRM = 1 CLKR is an output pin and is driven by the internal sample rate generator. Case 2: Digital loop back mode set (DLB = 1) in SPCR1 CLKRM = 0 Receive clock (not the CLKR pin) is driven by transmit clock (CLKX) which is based on the CLKXM bit in the PCR. CLKR pin is in high-impedance. CLKRM = 1 CLKR is an output pin and is driven by the transmit clock. The transmit clock is derived based on the CLKXM bit in the the PCR.
7	rsvd	Reseved
6	CLKS_STAT	CLKS pin status. Reflects value on CLKS pin when selected as a general purpose input.
5	DX_STAT	DX pin status. Reflects value driven on to DX pin when selected as a general purpose output.
4	DR_STAT	DR pin status. Reflects value on DR pin when selected as a general purpose input.
3	FSXP	Transmit Frame-Synchronization Polarity FSXP = 0 Frame-synchronization pulse FSX is active high FSXP = 1 Frame-synchronization pulse FSX is active low
2	FSRP	Receive Frame-Synchronization Polarity FSRP = 0 Frame-synchronization pulse FSR is active high FSRP = 1 Frame-synchronization pulse FSR is active low
1	CLKXP	Transmit Clock Polarity CLKXP = 0 Transmit data sampled on rising edge of CLKX CLKXP = 1 Transmit data sampled on falling edge of CLKX
0	CLKRP	Receive Clock Polarity CLKRP = 0 Receive data sampled on falling edge of CLKR CLKRP = 1 Receive data sampled on rising edge of CLKR

In addition to the data registers, there are 15 control registers that configure the operating modes of the McBSP. The McBSP can be configured during the reset state. Detailed definitions and usages of McBSP control registers can be found in [2]. In this section, we briefly describe the functionalities of the following registers:

- McBSP pin control registers
- Serial port control registers
- Transmit control registers and receive control registers
- Multichannel control registers
- Receive channel enable registers and transmit channel enable registers
- Sample rate generator registers

The pin control register (PCR) controls the following modes: (1) configure DX, FSX, and CLKX as general-purpose or serial-port pins; (2) configure DR, FSR, CLKR and CLKS as general-purpose or serial-port pins; (3) transmit frame synchronization derived from an external source or a sample rate generator; (4) receive frame synchronization generated by an external device or a sample rate generator; (5) transmit clock by external clock with CLKX as the input pin or by a sample rate generator with CLKX as the output pin; (6) receive clock by external clock with CLKR as the input pin or by a sample rate generator with CLKR as the output pin; (7) transmit frame-sync (active high or low) and clock polarity (rising edge or falling edge of CLKX); and (8) receive frame-sync and clock polarity (rising edge or falling edge of CLKR). Detailed definitions of the PCR are summarized in Table C.1.

Serial port control register 1 (SPCR1) contains the McBSP receiver-status bits and the main switch to enable or disable the receiver. This register also includes the clock-stop mode bit and receiver-interrupt mode bits. Serial port control register 2 (SPCR2) contains the McBSP transmitter-status bits and the main switch to enable or disable the transmitter. This register also includes the bits to reset the frame-sync generator and the sample-rate generator. Functions of the SPCR1 and SPCR2 are summarized in Tables C.2 and C.3, respectively.

Transmit control registers 1 and 2 (XCR1 and XCR2) configure various parameters of transmit operations, including (1) transmit frame length (from 1 to 128 words per frame), (2) transmit wordlength (for 8, 12, 16, 20, 24, or 32 bits), (3) transmit frame (single or dual phase), (4) transmit companding (none, μ-law, or A-law), (5) transmit frame synchronization, and (6) transmit data delay (0-, 1-, or 2-bit data delay). Tables C.4 and C.5 summarize the XCR1 and XCR2 bit-field descriptions, respectively.

Receive control registers 1 and 2 (RCR1 and RCR2) configure various parameters of receive operations, including (1) receive frame length (from 1 to 128 words per frame), (2) receive wordlength (for 8, 12, 16, 20, 24, or 32 bits), (3) receive frame (single or dual phase), (4) receive companding (none, μ-law, or A-law), (5) receive frame synchronization, and (6) receive data delay (0-, 1-, or 2-bit data delay). Bit-field descriptions similar to those shown in Tables C.4 and C.5 are also applied to RCR1 and RCR2.

TABLE C.2 Description of Serial Port Control Register 1 (SPCR1) (reprinted with permission from [2])

Bit	Name	Function	
15	DLB	Digital Loop Back Mode	
		DLB = 0	Digital loop back mode disabled
		DLB = 1	Digital loop back mode enabled
14–13	RJUST	Receive Sign-Extension and Justification Mode.	
		RJUST = 00	Right-justify and zero-fill MSBs in DRR
		RJUST = 01	Right-justify and sign-extend MSBs in DRR
		RJUST = 10	Left-justify and zero-fill LSBs in DRR
		RJUST = 11	Reserved
12–11	CLKSTP	Clock Stop Mode	
		CLKSTP = 0X	Clock stop mode disabled. Normal clocking for non-SPI mode.
		Various SPI modes when:	
		CLKSTP = 10 and CLKXP = 0	Clock starts with rising edge without delay
		CLKSTP = 10 and CLKXP = 1	Clock starts with falling edge without delay
		CLKSTP = 11 and CLKXP = 0	Clock starts with rising edge with delay
		CLKSTP = 11 and CLKXP = 1	Clock starts with falling edge with delay
10–8	reserved	Reserved	
7	DXENA	DX Enabler	
		DXENA = 0	DX enabler is off
		DXENA = 1	DX enabler is on
6	ABIS	ABIS Mode	
		ABIS = 0	A-bis mode is disabled
		ABIS = 1	A-bis mode is enabled
5–4	RINTM	Receive Interrupt Mode	
		RINTM = 00	RINT driven by RRDY (i.e., end of word) and end of frame in A-bis mode.
		RINTM = 01	RINT generated by end-of-block or end-of-frame in multichannel operation
		RINTM = 10	RINT generated by a new frame synchronization
		RINTM = 11	RINT generated by RSYNCERR

(*Continued*)

TABLE C.2 (Continued)

Bit	Name	Function	
3	RSYNCERR	Receive Synchronization Error	
		RSYNCERR = 0	No synchronization error
		RSYNCERR = 1	Synchronization error detected by McBSP.
2	RFULL	Receive Shift Register (RSR) Full	
		RFULL = 0	RBR is not in overrun condition
		RFULL = 1	DRR is not read, RBR is full and RSR is also full with new word
1	RRDY	Receiver Ready	
		RRDY = 0	Receiver is not ready.
		RRDY = 1	Receiver is ready with data to be read from DRR.
0	$\overline{\text{RRST}}$	Receiver reset. This resets and enables the receiver.	
		$\overline{\text{RRST}}$ = 0	The serial port receiver is disabled and in reset state.
		$\overline{\text{RRST}}$ = 1	The serial port receiver is enabled.

TABLE C.3 Description of Serial Port Control Register 2 (SPCR2) (reprinted with permission from [2])

Bit	Name	Function	
15–10	rsvd	Reserved	
9	FREE	Free Running Mode	
		FREE = 0	Free running mode is disabled
		FREE = 1	Free running mode is enabled
8	SOFT	Soft Bit	
		SOFT = 0	SOFT mode is disabled
		SOFT = 1	SOFT mode is enabled
7	$\overline{\text{FRST}}$	Frame-Sync Generator Reset	
		$\overline{\text{FRST}}$ = 0	Frame-synchronization logic is reset. Frame-sync signal FSG is not generated by the sample-rate generator.
		$\overline{\text{FRST}}$ = 1	Frame-sync signal FSG is generated after (FPER + 1) number of CLKG clocks; i.e., all frame counters are loaded with their programmed values.
6	$\overline{\text{GRST}}$	Sample-Rate Generator Reset	
		$\overline{\text{GRST}}$ = 0	Sample rate generator is reset

(*Continued*)

TABLE C.3 (Continued)

Bit	Name	Function	
		$\overline{\text{GRST}} = 1$	Sample rate generator is pulled out of reset. CLKG is driven as per programmed value in sample rate generator registers (SRGR).
5–4	XINTM	Transmit Interrupt Mode	
		XINTM = 00	XINT driven by XRDY (i.e., end of word) and end of frame in A-bis mode.
		XINTM = 01	XINT generated by end-of-block or end-of-frame in multichannel operation
		XINTM = 10	XINT generated by a new frame synchronization
		XINTM = 11	XINT generated by XSYNCERR
3	XSYNCERR	Transmit Synchronization Error	
		XSYNCERR = 0	No synchronization error
		XSYNCERR = 1	Synchronization error detected by McBSP.
2	$\overline{\text{XEMPTY}}$	Transmit Shift Register (XSR) Empty	
		$\overline{\text{XEMPTY}} = 0$	XSR is empty
		$\overline{\text{XEMPTY}} = 1$	XSR is not empty
1	XRDY	Transmitter Ready	
		XRDY = 0	Transmitter is not ready.
		XRDY = 1	Transmitter is ready for new data in DXR.
0	$\overline{\text{XRST}}$	Transmitter Reset	This resets and enables the transmitter.
		$\overline{\text{XRST}} = 0$	The serial port transmitter is disabled and in reset state.
		$\overline{\text{XRST}} = 1$	The serial port transmitter is enabled.

Multichannel registers 1 and 2 (MCR1 and MCR2) independently select the channels to be used. These registers select the following options: (1) receive and transmit partition A block (each mode selects all-odd blocks of 16 channels), (2) receive and transmit partition B block (each mode selects all-even blocks of 16 channels), (3) receive and transmit current block (each mode selects 1 of the 8 possible blocks), and (4) receive and transmit multichannel selection enable (selects either all 128 channels or select 1 of the 8 blocks, which is enabled by receive partition block registers). Table C.6 shows MCR1 configured as the receiver, while Table C.7 shows MCR2 configured as the transmitter.

Receive channel enable register partition A (RCERA) and B (RCERB) are used to enable any of 32 channels for receive. Out of the 32 channels, 16 channels belong to a block in partition A, and the other 16 belong to a block in partition B. The 16-bit RCERA enables or disables each channel in an even-numbered block

TABLE C.4 Partial Bit-Field Description of Transmit Control Register 1 (XCR1) (reprinted with permission from [2])

Bit	Name	Function	
15	rsvd	Reserved	
14–8	XFRLEN1	Transmit Frame Length 1	
		XFRLEN1 = 000 0000	1 word per frame
		XFRLEN1 = 000 0001	2 words per frame
			⋮
		RFRLEN1 = 111 1111	128 words per frame
7–5	XWDLEN1	Transmit Word Length 1	
		XWDLEN1 = 000	8 bits
		XWDLEN1 = 001	12 bits
		XWDLEN1 = 010	16 bits
		XWDLEN1 = 011	20 bits
		XWDLEN1 = 100	24 bits
		XWDLEN1 = 101	32 bits
		XWDLEN1 = 11X	Reserved
4–0	rsvd	Reserved	

TABLE C.5 Bit-Field Description of Transmit Control Register 2 (XCR2) (reprinted with permission from [2])

Bit	Name	Function	
15	XPHASE	Transmit Phases	
		XPHASE = 0	Single-phase frame
		XPHASE = 1	Dual-phase frame
14–8	XFRLEN2	Transmit Frame Length 2	
		XFRLEN2 = 000 0000	1 word per frame
		XFRLEN2 = 000 0001	2 words per frame
			⋮
		XFRLEN1 = 111 1111	128 words per frame
7–5	XWDLEN2	Transmit Word Length 2	
		XWDLEN2 = 000	8 bits
		XWDLEN2 = 001	12 bits
		XWDLEN2 = 010	16 bits

(*Continued*)

TABLE C.5 (Continued)

Bit	Name	Function	
		XWDLEN2 = 011	20 bits
		XWDLEN2 = 100	24 bits
		XWDLEN2 = 101	32 bits
		XWDLEN2 = 11X	Reserved
4–3	XCOMPAND	Transmit companding mode. Modes other than 00b are only enabled when the appropriate XWDLEN is 000b, indicating 8-bit data.	
		XCOMPAND = 00	No companding, data transfer starts with MSB first.
		XCOMPAND = 01	No companding, 8-bit data, transfer starts with LSB first.
		XCOMPAND = 10	Compand using μ-law for transmit data.
		XCOMPAND = 11	Compand using A-law for transmit data.
2	XFIG	Transmit Frame Ignore	
		XFIG = 0	Transmit frame-synchronization pulses after the first restarts the transfer.
		XFIG = 1	Transmit frame-synchronization pulses after the first are ignored.
1–0	XDATDLY	Transmit Data Delay	
		XDATDLY = 00	0-bit data delay
		XDATDLY = 01	1-bit data delay
		XDATDLY = 10	2-bit data delay
		XDATDLY = 11	Reserved

in partition A, while RCERB enables or disables each channel in an odd-numbered block in partition B. Table C.8 describes the RCERA and RCERB registers.

Transmit channel enable register partition A (XCERA) and B (XCERB) operate similarly to the receive channel enable registers, except that they are used to enable any of the 32 channels for transmission. The XCERA and XCERB registers use the same bit descriptions as those shown in Table C.8.

The sample rate generator (SRGR) is composed of a clock divider that generates a programmable data clock (CLKG) and a framing signal (FSG), as shown in Fig. C.3. These internal signals are then used to drive the receive and transmit clocks

TABLE C.6 Multichannel Control Register 1 (MCR1) (reprinted with permission from [2])

Bit	Name	Function	
15–9	rsvd	Reserved	
8–7	RPBBLK	Receive Partition B Block	
		RPBBLK = 00	Block 1. Channel 16 to channel 31
		RPBBLK = 01	Block 3. Channel 48 to channel 63
		RPBBLK = 10	Block 5. Channel 80 to channel 95
		RPBBLK = 11	Block 7. Channel 112 to channel 127
6–5	RPABLK	Receive Partition A Block	
		RPABLK = 00	Block 0. Channel 0 to channel 15
		RPABLK = 01	Block 2. Channel 32 to channel 47
		RPABLK = 10	Block 4. Channel 64 to channel 79
		RPABLK = 11	Block 6. Channel 96 to channel 111
4–2	RCBLK	Receive Current Block	
		RCBLK = 000	Block 0. Channel 0 to channel 15
		RCBLK = 001	Block 1. Channel 16 to channel 31
		RCBLK = 010	Block 2. Channel 32 to channel 47
		RCBLK = 011	Block 3. Channel 48 to channel 63
		RCBLK = 100	Block 4. Channel 64 to channel 79
		RCBLK = 101	Block 5. Channel 80 to channel 95
		RCBLK = 110	Block 6. Channel 96 to channel 111
		RCBLK = 111	Block 7. Channel 112 to channel 127
1	rsvd	Reserved	
0	RMCM	Receive Multichannel Selection Enable	
		RMCM = 0	All 128 channels enabled.
		RMCM = 1	All channels disabled by default. Required channels are selected by enabling RP(A/B)BLK and RCER(A/B) appropriately.

(CLKR/CLKX) and the receive/transmit framing signal (FSR/FSX). The sample rate generator can be programmed to divide either an internal-clock source or an internal clock derived from an external-clock source.

The sample rate generator registers SRGR1 and SRGR2 control the operations of the sample rate generator, including (1) frame width (FWID), which determines the width of the frame-sync pulse (1 to 256 CLKG periods); (2) sample rate generator clock divider (CLKGDV), which generates the required sample rate generator clock frequency (default = 1 and maximum = 255); (3) sample

TABLE C.7 Multichannel Control Register 2 (MCR2) (reprinted with permission from [2])

Bit	Name	Function	
15–9	rsvd	Reserved	
8–7	XPABLK	Transmit Partition A Block	
		XPABLK = 00	Block 0. Channel 0 to channel 15
		XPABLK = 01	Block 2. Channel 32 to channel 47
		XPABLK = 10	Block 4. Channel 64 to channel 79
		XPABLK = 11	Block 6. Channel 96 to channel 111
6–5	XPBBLK	Transmit Partition B Block	
		XPBBLK = 00	Block 1. Channel 16 to channel 31
		XPBBLK = 01	Block 3. Channel 48 to channel 63
		XPBBLK = 10	Block 5. Channel 80 to channel 95
		XPBBLK = 11	Block 7. Channel 112 to channel 127
4–2	XCBLK	Transmit Current Block	
		XCBLK = 000	Block 0. Channel 0 to channel 15
		XCBLK = 001	Block 1. Channel 16 to channel 31
		XCBLK = 010	Block 2. Channel 32 to channel 47
		XCBLK = 011	Block 3. Channel 48 to channel 63
		XCBLK = 100	Block 4. Channel 64 to channel 79
		XCBLK = 101	Block 5. Channel 80 to channel 95
		XCBLK = 110	Block 6. Channel 96 to channel 111
		XCBLK = 111	Block 7. Channel 112 to channel 127
1–0	XMCM	Transmit Multichannel Selection Enable	
		XMCM = 00	All channels enabled without masking (DX is always driven during transmission of data†).
		XMCM = 01	All channels disabled and therefore masked by default. Required channels are selected by enabling XP(A/B)BLK and XCER(A/B) appropriately. Also, these selected channels are not masked and therefore DX is always driven.
		XMCM = 10	All channels enabled, but masked. Selected channels enabled via XP(A/B)BLK and XCER(A/B) are unmasked.
		XMCM = 11	All channels disabled and therefore masked by default. Required channels are selected by enabling RP(A/B)BLK and RCER(A/B) appropriately. Selected channels can be unmasked by RP(A/B)BLK and XCER(A/B). This mode is used for symmetric transmit and receive operation.

TABLE C.8 Receive Channel Enable Register Partitions A and B (RCERA) and (RCERB) (reprinted with permission from [2])

Bit	Name	Function
15–0	RCEA(0.15)	Receive Channel Enable-
		RCEA $n = 0$ Disables reception of nth channel in an even-numbered block in partition A
		RCEA $n = 1$ Enables reception of nth channel in an even-numbered block in partition A

Bit	Name	Function
15–0	RCEB(0.15)	Receive Channel Enable-
		RCEB $n = 0$ Disables reception of nth channel in an even-numbered block in partition B
		RCEB $n = 1$ Enables reception of nth channel in an even-numbered block in partition B

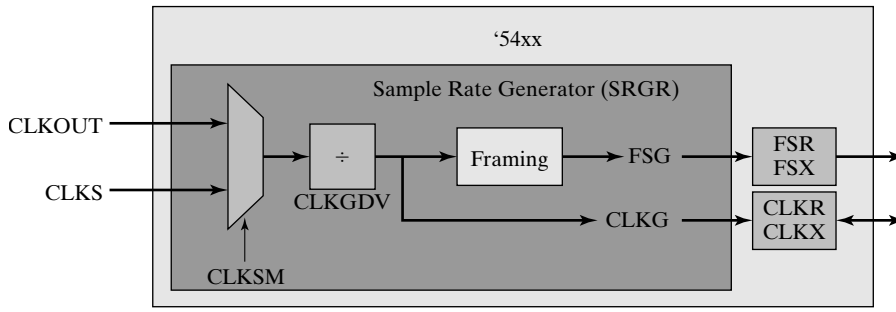

Figure C.3 Internal blocks of the sample rate generator (reprinted with permission from [3])

rate generator clock synchronization (GSYNC); (4) CLKS polarity clock edge select (CLKSP); (5) McBSP sample rate generator clock (CLKSM); (6) sample rate generator transmit frame-synchronization (FSGM); and (7) frame period (FRER), which can be set from 1 to 4,096 CLKG periods. Tables C.9 and C.10 summarize the functions of the SRGR1 and SRGR2 registers, respectively.

Having introduced the McBSP registers, we now examine how to configure clocking and framing signals using these registers. Figure C.4 shows the typical operations of the internal McBSP clock and frame-sync signals, as well as the data bits. The polarities (active high or low) of FSR, FSX, CLKX, and CLKR can be selected by the PCR register.

Figure C.5 shows an example of a dual-phase frame consisting of 2 words and 3 words in phases 1 and 2, respectively. The wordlength in phase 1 is 12 bits and in phase 2 is 8 bits. Using Tables C.4 and C.5, we can specify the preceding dual-phase

TABLE C.9 Sample Rate Control Register 1 (SRGR1) (reprinted with permission from [2])

Bit	Name	Function
15–8	FWID	Frame Width. This field plus 1 determines the width of the frame-syncpulse, FSG, during its active period. Range: up to 2; 1 to 256 CLKG periods.
7–0	CLKGDV	Sample Rate Generator Clock Divider This value is used as the divide-down number to generate the required sample rate generator clock frequency. Default value is 1.

TABLE C.10 Sample Rate Control Register 2 (SRGR2) (reprinted with permission from [2])

Bit	Name	Function
15	GSYNC	Sample Rate Generator Clock Synchronization Only used when the external clock (CLKS) drives the sample rate generator clock (CLKSM = 0). GSYNC = 0 The sample rate generator clock (CLKG) is free running. GSYNC = 1 The sample rate generator clock (CLKG) is running. But CLKG is resynchronized and frame-sync signal (FSG) is generated only after detecting the receive frame-synchronization signal (FSR). Also, frame period, FPER, is a don't care because the period is dictated by the external frame-sync pulse.
14	CLKSP	CLKS Polarity Clock Edge Select Only used when the external clock CLKS drives the sample rate generator clock (CLKSM = 0). CLKSP = 0 Rising edge of CLKS generates CLKG and FSG. CLKSP = 1 Falling edge of CLKS generates CLKG and FSG.
13	CLKSM	McBSP Sample Rate Generator Clock Mode CLKSM = 0 Sample rate generator clock derived from the CLKS pin. CLKSM = 1 Sample rate generator clock derived from CPU clock.
12	FSGM	Sample Rate Generator Transmit Frame-Synchronization Mode Used when FSXM = 1 in the PCR. FSGM = 0 Transmit frame-sync signal (FSX) due to DXR [1,2]-to-XSR[1,2] copy. When FSGM = 0, FPR and FWID are ignored. FSGM = 1 Transmit frame-sync signal driven by the sample rate generator frame-sync signal, FSG.
11–0	FPER	Frame Period. This field plus 1 determines when the next frame-sync signal becomes active. Range: 1 to 4096 CLKG periods.

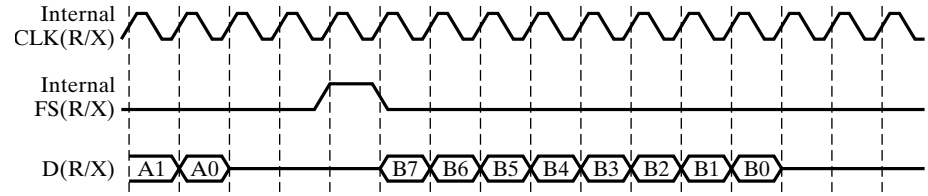

Figure C.4 Typical operations of clock, frame-sync, and data signals (reprinted with permission from [2])

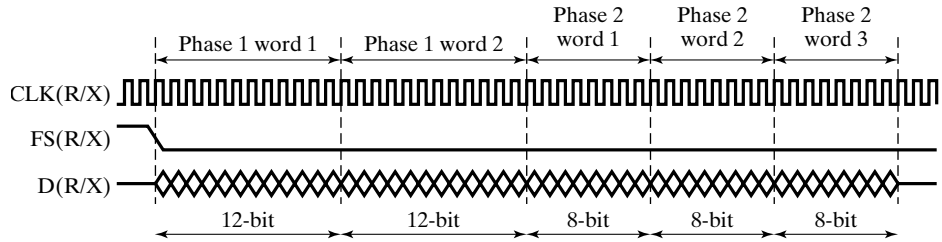

Figure C.5 Dual-phase example with 12 bits per word and 2 words per phase in phase 1 and with 8 bits per word and 3 words per phase in phase 2 (reprinted with permission from [2])

frame in the following registers:

- X/RPHASE = 1, dual-phase frame
- X/RFRLEN1 = 000 0001, 2 words for phase 1
- X/RFRLEN2 = 000 0010, 3 words for phase 2
- X/RWDLEN1 = 001, 12 bits per word for phase 1
- X/RWDLEN2 = 000, 8 bits per word for phase 2

The maximum number of words per frame is 128 for a single-phase frame and 256 for a dual-phase frame. Altogether, the maximum number of bits per frame is $128 \times 32 = 4{,}096$ for a single-phase frame and 8,192 for a dual-phase frame, assuming there are no gaps between words and frames. The frame frequency is determined by the period between frame-sync signals. Therefore, the maximum frame frequency is the bit-clock frequency divided by the number of bits per frame.

In addition, there are five serial-port events that may result in system error. They are summarized as follows:

1. A receive overrun error occurs when DRR has not been read since the last data move from RBR to DRR.
2. An unexpected receiver frame sync error happens when reset frame ignore (RFIG) = 0 and when an unexpected frame sync pulse occurs during reception.
3. A transmit data overwrite error occurs when the user overwrites data in DXR before it has been copied to XSR.

4. A transmit empty error occurs when a new frame sync signal arrives before the new data is loaded into the DXR. In this case, the old data in DXR is sent again.
5. An unexpected transmit frame sync error occurs when transmit frame ignore (XFIG) = 0 and when an unexpected frame sync pulse occurs during transmission.

In general, the following steps show how to transmit and receive data using the McBSP:

Step 1: Reset the McBSP using either the device reset RS = 0, or independently reset serial-port transmit (XRST) = 0 and/or receiver reset (RRST) = 0.

Step 2: Wait until the device reset is complete (RS = 1), then initialize the serial port using the following steps:

- **a.** Set XRST = RRST = frame-sync generator reset (FRST) = 0 in the SPCR1 and SPCR2 registers, which disables both the transmitter and the receiver and puts them in reset state.
- **b.** Program the McBSP configuration registers, including SPCR1, SPCR2, RCR1, RCR2, XCR1, XCR2, SPGR1, SPGR2, MCR1, MCR2, RCERA, RCERB, XCERA, XCERB, and PCR.
- **c.** Wait for a 2-bit clock period to ensure proper internal synchronization.
- **d.** Set up data acquisition such as writing to DXR.
- **e.** Enable the serial port by setting XRST = RRST = 1 in the SPCR1 and SPCR2 registers.
- **f.** Set FRST = 1 if an internal generated frame sync is required.
- **g.** Wait for a 2-bit clock period to ensure the receiver and transmitter become active.

Step 3: Ensure the serial port has been initialized, and then determine whether the McBSP is ready by polling the receiver ready (RRDY) and transmitter ready (XRDY) signals, the DMA events (REVT and XEVT in normal mode), or the interrupts to CPU (RINT and XINT). In the case of the receiver, the condition RRDY = 1 indicates that the data in the RBR register have been copied to the DRR register, which implies that data can be read by the CPU or DMA. Once the data has been read, RRDY = 0. In the case of the transmitter, XRDY = 1 indicates that the DXR register contents have been copied to the XSR register. DXR is now ready to be loaded with a new data word. Once data is loaded, XRDY = 1.

C.1.2 Initializing Direct Memory Access

The DMA controller is used to transfer data between the McBSP and the internal memory of the processor without CPU intervention. For example, DMA can transfer

data from an external device to internal memory while the CPU is processing data located at another memory section, as shown in Fig. C.6. At the same time, a data sample from the processor can be sent out to an external device via another DMA channel. Six programmable DMA channels are available on the C54x processor. These channels allow six different DMA operations with programmable priorities. In addition to moving data between external peripherals and internal memory through the McBSP, DMA also allows data transfer between internal memory, internal peripherals, and HPI peripherals. The HPI interface is discussed in Section C.2.

There are several transfer operations in DMA:

1. A read transfer occurs when the DMA reads a data element from a source located in memory or from a peripheral device.
2. A write transfer occurs when the DMA writes a data element during the preceding read transfer to the destination located in memory or to a peripheral device.
3. An element transfer combines the preceding read and write transfers for a single data element.
4. A frame transfer allows each of the six DMA channels in a C54x processor to program the number of elements per frame independently. A frame transfer is completed when the DMA moves all of the elements in a single frame.
5. A block transfer allows each of the six DMA channels in a C54x processor to program the number of frames per block independently. A block transfer is completed when the DMA moves all of the frames defined in the block.

Similar to other peripherals, DMA must be configured before operation. The DMA controller has a group of memory-mapped registers that use a subaddressing scheme. Register subaddressing is a technique used in multiplexing a set of registers to a single location in the memory map. Therefore, a large set of registers can be mapped into a small memory space, as shown in Fig. C.7. It shows a subbank address

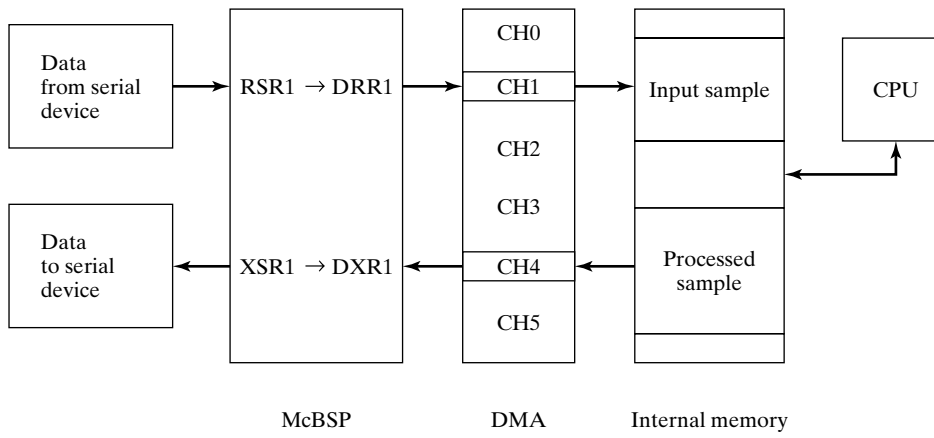

Figure C.6 DMA reading/writing data from/to a serial device to/from internal memory

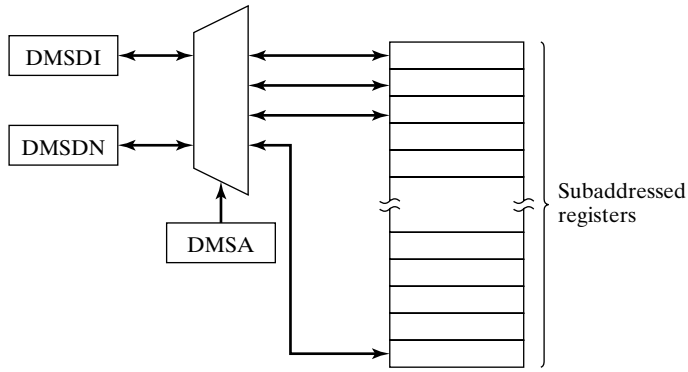

Figure C.7 Register subaddressing in DMA (reprinted with permission from [2])

register (DMSA), which directs the multiplexer to connect the subbank access registers (DMSDI and DMSDN) to the actual subaddressed registers. The difference between DMSDI and DMSDN is that DMSDI is incremented automatically after each access, while DMSDN is used for single-register access without modifying the subaddress.

Table C.11 lists the addresses, subaddresses, and functions of DMA registers. Only the first four registers listed in Table C.11 are directly addressed, while the rest of the registers are subaddressed. For example, to program DMA channel 0 registers, the value 0x00 (subaddress) is written to the DMSA register. The first value written to the DMSDI register is directed to the DMSCR0 register, the second value written to DMSDI is automatically directed to the DMDST0 register, and so on. However, if the value is written to the DMSDN register, the subaddress is not autoincremented, and only the particular register (specified in the DMSA register) is modified. A detailed description of the DMA register can be found in [2].

The DMA channel priority and enable control register (DMPREC) (located at address 54h) selects the DMA channels, controls the multiplexed interrupts, and controls the channel priorities. Because the DMA controller resets each enable bit after a block transfer, DMPREC can also be polled to determine whether the block transfer for a given channel has been completed. Each DMA channel can be assigned with only two priorities (low or high), where the high-priority channel is serviced before the low-priority channel. When multiple channels with high priority are selected, these channels are serviced in a round-robin fashion from low to high channel number. Detailed bit-field descriptions of the DMPREC are given in Table C.12.

In addition, the DMPREC register controls how DMA interrupts are assigned in the interrupt vector and IMR (interrupt mask register)/IMF (interrupt mask flag) registers. Due to the limited number of interrupts available in the C54x memory map, some DMA interrupts are multiplexed with other peripheral interrupts. Therefore, the interrupt multiplex control bits (INTOSEL) field in the DMPREC register provides a means of selecting the desired DMA channel interrupt. An example for the C5402 processor is displayed in Table C.13, which shows the bit value of the INTOSEL to select the DMA interrupt for channels 1 to 3.

TABLE C.11 DMA Registers (reprinted with permission from [2])

Address	SubAddress	Name	Function
54h	—	DMPREC	Channel Priority and Enable Control Register
55h	—	DMSA	Subbank Address Register
56h	—	DMSDI	Subbank Access Register With Autoincrement
57h	—	DMSDN	Subbank Access Register Without Autoincrement
—	00h	DMSRC0	Channel 0 Source Address Register
—	01h	DMDST0	Channel 0 Destination Address Register
—	02h	DMCTR0	Channel 0 Element Count Register
—	03h	DMSFC0	Channel 0 Sync Select and Frame Count Register
—	04h	DMMCR0	Channel 0 Transfer Mode Control Register
—	05h	DMSRC1	Channel 1 Source Address Register
—	06h	DMDST1	Channel 1 Destination Address Register
—	07h	DMCTR1	Channel 1 Element Count Register
—	08h	DMSFC1	Channel 1 Sync Select and Frame Count Register
—	09h	DMMCR1	Channel 1 Transfer Mode Control Register
—	0Ah	DMSRC2	Channel 2 Source Address Register
—	0Bh	DMDST2	Channel 2 Destination Address Register
—	0Ch	DMCTR2	Channel 2 Element Count Register
—	0Dh	DMSFC2	Channel 2 Sync Select and Frame Count Register
—	0Eh	DMMCR2	Channel 2 Transfer Mode Control Register
—	0Fh	DMSRC3	Channel 3 Source Address Register
—	10h	DMDST3	Channel 3 Destination Address Register
—	11h	DMCTR3	Channel 3 Element Count Register
—	12h	DMSFC3	Channel 3 Sync Select and Frame Count Register
—	13h	DMMCR3	Channel 3 Transfer Mode Control Register
—	14h	DMSRC4	Channel 4 Source Address Register
—	15h	DMDST4	Channel 4 Destination Address Register
—	16h	DMCTR4	Channel 4 Element Count Register
—	17h	DMSFC4	Channel 4 Sync Select and Frame Count Register
—	18h	DMMCR4	Channel 4 Transfer Mode Control Register
—	19h	DMSRC5	Channel 5 Source Address Register
—	1Ah	DMDST5	Channel 5 Destination Address Register
—	1Bh	DMCTR5	Channel 5 Element Count Register
—	1Ch	DMSFC5	Channel 5 Sync Select and Frame Count Register

(*Continued*)

TABLE C.11 (Continued)

Address	SubAddress	Name	Function
—	1Dh	DMMCR5	Channel 5 Transfer Mode Control Register
—	1Eh	DMSRCP	Source Program Page Address (all channels)
—	1Fh	DMDSTP	Destination Program Page Address (all channels)
—	20h	DMIDX0	Element Address Index Register 0
—	21h	DMIDX1	Element Address Index Register 1
—	22h	DMFRI0	Frame Address Index Register 0
—	23h	DMFRI1	Frame Address Index Register 1
—	24h	DMGSA	Global Source Address Reload Register
—	25h	DMGDA	Global Destination Address Reload Register
—	26h	DMGCR	Global Element Count Reload Register
—	27h	DMGFR	Global Frame Count Reload Register

TABLE C.12 DMA Channel Priority and Enable Control Register (DMPREC) (reprinted with permission from [2])

Bit	Name	Reset Value	Function
15	FREE	0	This bit controls the behavior of the DMA controller during emulation. When FREE = 0, DMA transfers are suspended when the emulator stops. When FREE = 1, DMA transfers continue even during emulation stop.
14	RSVD	0	Reserved. Values written to this field have no effect.
13	DPRC[5]	0	DMA channel 5 priority control bit. DPRC[5] = 1 High priority DPRC[5] = 0 Low priority
12	DPRC[4]	0	DMA channel 4 priority control bit. DPRC[4] = 1 High priority DPRC[4] = 0 Low priority
11	DPRC[3]	0	DMA channel 3 priority control bit. DPRC[3] = 1 High priority DPRC[3] = 0 Low priority
10	DPRC[2]	0	DMA channel 2 priority control bit. DPRC[2] = 1 High priority DPRC[2] = 0 Low priority

(*Continued*)

TABLE C.12 (Continued)

Bit	Name	Reset Value	Function
9	DPRC[1]	0	DMA channel 1 priority control bit. DPRC[1] = 1 High priority DPRC[1] = 0 Low priority
8	DPRC[0]	0	DMA channel 0 priority control bit. DPRC[0] = 1 High priority DPRC[0] = 0 Low priority
7–6	INTOSEL	0	Interrupt multiplex control bits. The INTOSEL bits control how the DMA interrupts will be assigned in the interrupt vector table and IMR/IMF registers. The effects of this field on the operation are device-specific
5	DE[5]	0	DMA channel 5 enable bit. DE[5] = 1 Enables DMA channel 5 DE[5] = 0 Disables DMA channel 5
4	DE[4]	0	DMA channel 4 enable bit. DE[4] = 1 Enables DMA channel 4 DE[4] = 0 Disables DMA channel 4
3	DE[3]	0	DMA Channel 3 enable bit. DE[3] = 1 Enables DMA channel 3 DE[3] = 0 Disables DMA channel 3
2	DE[2]	0	DMA Channel 2 enable bit. DE[2] = 1 Enables DMA channel 2 DE[2] = 0 Disables DMA channel 2
1	DE[1]	0	DMA channel 1 enable bit. DE[1] = 1 Enables DMA channel 1 DE[1] = 0 Disables DMA channel 1
0	DE[0]	0	DMA channel 0 enable bit. DE[0] = 1 Enables DMA channel 0 DE[0] = 0 Disables DMA channel 0

The other three direct-addressed registers are DMSA (located at 55h), DMSDI (located at 56h), and DMSDN (located at 57h). The rest of the DMA registers are subaddressed and are grouped under (1) channel-context registers and (2) DMA system registers.

TABLE C.13 Multiplex Interrupt Assignments for the C5402 Processor (reprinted with permission from [2])

Interrupt Number (IMR/IFR#)	INTOSEL [1:0] Value 00b	01b	10b	11b
7	Timer 1 interrupt	Timer 1 interrupt	DMA Channel 1 Interrupt	Reserved
10	McBSP 1 RINT	DMA Channel 2 Interrupt	DMA Channel 2 Interrupt	Reserved
11	McBSP 1 XINT	DMA Channel 3 Interrupt	DMA Channel 3 Interrupt	Reserved

As shown in Table C.11, each DMA channel has a set of five channel-context registers, which configure the source address, destination address, element count, sync event/frame count, and transfer-mode control. These issues are discussed as follows:

1. The source address in registers (DMSRCn) (n = 0 to 5) defines the address of the data being read, while the destination address in DMDSTn registers defines the address of the data being written.
2. Channel element count registers (DMCTRn) keep track of the number of DMA element transfers. The number of elements to be initialized in this register is one less than the desired number of element transfers (e.g., initialize to 9 for 10 elements to be transferred), and it is initialized as a 16-bit unsigned number. This counter autodecrements with every transfer and reloads with the original count value after the last element in each frame is reached.
3. The DMA sync event and frame count registers (DMSFCn) are used to determine (1) the sync event to trigger DMA transfers (e.g., McBSP receive and transmit events, timer interrupt, or external interrupt), (2) the transfer wordlength (16 or 32 bits), and (3) the number of frames to be transferred (initialized as one less than the desired number of frames). The maximum number of frames supported is 256. Table C.14 shows the bit-field descriptions of the DMSFCn registers. A detailed listing of the DMA sync event for different processors can be found in [2].
4. The transfer mode control register (DMMCRn) controls the transfer mode of the channel. Transfer modes include (1) DMA autoinitialization (on/off), (2) DMA interrupt generation mask bit, (3) DMA interrupt generation mode bit, (4) DMA transfer counter mode control bit, (5) DMA source address transfer index mode bit, which determines postincrement or postdecrement of the DMA source address by a certain offset index, (6) DMA source address space select bit (program, data, or I/O space), (7) DMA destination address transfer index mode bit (on how destination address modification after transfer), and (8) DMA destination address space select bit (program, data, or I/O space). Table C.15 shows a detailed listing of the DMMCRn register.

TABLE C.14 DMA Sync Event and Frame Count Registers (DMSFCn) (reprinted with permission from [2])

Bit	Name	Reset Value	Function
15–12	DSYN[3:0]	0	DMA sync event. Specifies which sync event is used to initiate DMA transfers for the corresponding DMA channel.
11	DBLW	0	Double-word mode. DBLW = 0 Single-word mode. Each element is 16 bits. DBLW = 1 Double-word mode. Each element is 32 bits.
10–8	rsvd	0	Reserved Values written to this field have no effect.
7–0	Frame Count	0	Frame count. Specifies the total number of frames to be transferred.

TABLE C.15 DMA Transfer Mode Control Register (DMMCRn) (reprinted with permission from [2])

Bit	Name	Reset Value	Function
15	AUTOINIT	0	DMA autoinitialization mode bit. AUTOINIT = 0 Autoinitialization is disabled AUTOINIT = 1 Autoinitialization is enabled
14	DINM	0	DMA interrupt generation mask bit. DINM = 0 No interrupt generated DINM = 1 Interrupts generated based on IMOD bit
13	IMOD	0	DMA interrupt generation mode bit. In ABU mode (CTMOD = 1): IMOD = 0 Interrupt at buffer full only IMOD = 1 Interrupt at half buffer full and buffer full In multiframe mode (CTMOD = 0): IMOD = 0 Interrupt at completion of block transfer IMOD = 1 Interrupt at end of frame and end of block
12	CTMOD	0	DMA Transfer Counter Mode Control Bit. CTMOD = 0 Multiframe mode CTMOD = 1 ABU mode
11	Reserved	0	Reserved. Values written to this field have no effect

(*Continued*)

TABLE C.15 (Continued)

Bit	Name	Reset Value	Function
10–8	SIND	0	DMA source address transfer index mode bit. SIND = 000 No modification SIND = 001 Postincrement SIND = 010 Post-decrement SIND = 011 Postincrement with index offset (DMIDX0) SIND = 100 Postincrement with index offset (DMIDX1) SIND = 101 Postincrement with index offset (DMIDX0 and DMFRI0) SIND = 110 Postincrement with index offset (DMIDX1 and DMFRI1) SIND = 111 Reserved
7–6	DMS	0	DMA source address space select bit. DMS = 00 Program space DMS = 01 Data space DMS = 10 I/O space DMS = 11 Reserved
5	Reserved	0	Reserved. Values written to this field have no effect.
4–2	DIND	0	DMA destination address transfer index mode bit. DIND = 000 No modification DIND = 001 Postincrement DIND = 010 Post-decrement DIND = 011 Postincrement with index offset (DMIDX0) DIND = 100 Postincrement with index offset (DMIDX1) DIND = 101 Postincrement with index offset (DMIDX0 and DMFRI0) DIND = 110 Postincrement with index offset (DMIDX1 and DMFRI1) DIND = 111 Reserved
1–0	DMD	0	DMA Destination Address Space Select Bit. DMD = 00 Program space DMD = 01 Data space DMD = 10 I/O space DMD = 11 Reserved

As shown in Table C.15, the DMA transfer counter mode control bit (CTMOD) in the DMMCRn register can be configured in either multiframe mode or autobuffering (ABU) mode:

1. Multiframe mode (CTMOD = 0) is commonly used in a frame- or block-formatted data transfer. Element and frame indexes are used to modify the source and destination addresses after every element/frame transfer. The element counter register (DMCTRn) is automatically decremented after each DMA transfer until the last element in each frame is reached. The element is then reloaded with the original content of DMCTRn, which is a 16-bit integer. Therefore, the number of elements to be transmitted per frame is between 1 and 65,536. The frame count in the DMSFCn register is an unsigned 8-bit integer that limits the number of frames to be transferred per block from 1 to 256. The total number of elements to be transmitted is called the block size, which is defined as the frame count multiplied by the element count.

   Indexing to the source and destination addresses is determined by the DMA source address transfer index mode bit (SIND) and DMA destination address transfer index mode bit (DIND) options in Table C.15. The common post-increment or post-decrement operation increases or decreases the address by 1 (or 2) for each word (or double word) transfer. In addition to these options, post-increment by a different element index can be specified in the element address index registers 1 and 2 (DMIDX0 and DMIDX1), which contain a 16-bit signed number for the source, destination, or both. In addition, the frame address index registers (DMFRI0 and DMFRI1) are used to index the source and destination addresses after the completion of blocks or frames of data transfers. This feature is useful in sorting data by address modification, as illustrated in [2].

   When the DMA autoinitialization mode bit (AUTOINIT) of the DMMCRn register is enabled (as shown in Table C.15), it reinitializes the channel-context registers after the block transfer has completed. The following registers are modified after the completion of the block transfer: DMSRCn = DMGSA (global source address), destination address register (DMDSTn) = DMGDA (global destination address), DMCTRn = DMGCR (global element count), and DMSFCn = DMGFR (global frame count). Note that autoinitialization is only available in multiframe mode.

2. ABU mode (CTMOD = 1) is used for autobuffering functions, such as a circular buffer. The element counter DMCTRn represents the buffer size and is not modified during transmission. The frame count register does not function in ABU mode. When the address reaches the end of the buffer, it wraps back to the beginning automatically. The working operation in ABU mode is to configure either the source or destination in ABU mode, while the other location is left unmodified, as shown in Table C.16.

   The element count register (DMCTRn) contains a 16-bit unsigned integer that represents a valid buffer size from 2 (0002h) to 65,535 (FFFFh). The buffer size is the difference between the base address and the maximum

TABLE C.16 ABU Address Index Modes (reprinted with permission from [2])

SIND	Address Index Mode	DIND	Address Index Mode
000	No modification	001	Postincrement
		010	Postdecrement
		011	Postincrement with index offset (DMIDX0)
		100	Postincrement with index offset (DMIDX1)
001	Postincrement	000	No modification
010	Postdecrement		
011	Postincrement with index offset (DMIDX0)		
100	Postincrement with index offset (DMIDX1)		

address. Note that the base address of the buffer must be based on a power of two. For example, if a buffer size of 2,048 is required, the buffer base address must be aligned with 4,096-word boundaries such as 0000h, 1000h, 2000h, 3000h, etc.

There are four DMA block transfer interrupt generation modes. These modes can be selected by specifying CTMOD, DIND, and interrupt generation mode bit (IMOD) of the DMMCRn registers, as shown in Table C.17. It is noted that in the ABU mode, interrupts can be generated either when the entire buffer has been transferred or when the buffer has been half-transferred. In multiframe mode, interrupts are only generated either at the end of the frame or at the end of the block.

The remaining registers shown in Table C.11 are DMA system registers. They consist of (1) source/destination program page address registers (DMSRCP and DMDSTP), (2) element address index registers (DMIDX0 and DMIDX1), which

TABLE C.17 DMA Block Transfer Interrupt Generation Modes (reprinted with permission from [2])

MODE	CTMOD	DINM	IMOD	Interrupt Generation
ABU	1	1	0	At buffer full only
ABU	1	1	1	At half buffer full and buffer full
Multiframe	0	1	0	At block transfer complete
Multiframe	0	1	1	At end of frame and end of block
Either	X	0	X	No interrupt generated

contain the index offset value to add to the current address after every transfer, (3) frame address index registers (DMFRI0 and DMFRI1), which are used to modify the current address if the current element is the last element of the frame, (4) global source/destination address reload registers (DMGSA and DMGDA), and (5) global element and frame count reload registers (DMGCR and DMGFR). These registers configure index options for the entire DMA system.

An important performance benchmark in a DMA transfer is its latency. A 16-bit DMA transfer is always composed of a read followed by a write. DMA latency depends on the source and destination locations and on whether these locations are internal or external. For external locations, there is a need to consider interface conditions such as wait states and bank-switching cycles. For the C5402 processor, only internal-to-internal transfers (e.g., from a memory-mapped register to on-chip DARAM) are supported, and they take four CPU clock cycles (two for read and two for write). If the C5402 processor is operating at a clock rate of 100 MHz, the maximum transfer rate is 100M/4 = 25M words per second or 12.5M double words per second. If more than one DMA channel is selected with high priority, the data transfer rate is reduced by the number of high-priority channels. For example, if three DMA channels are assigned as high priority, the data transfer rate is 25M/3 = 8.33M words per second. Low-priority channels share the transfer rate only after the completion of high-priority channels.

We illustrate setting the DMA controller for interface to the audio CODEC in the following section.

C.1.3 An Example

We investigate a simple digital loop-back example in Fig. C.8 using AD50 AIC, where the analog signal from the sound card is first passed to the ADC inside the AIC. The serial output from the ADC is then sent to the DRR1 register of the McBSP1 and transferred to the internal memory (DARAM) of the C5402 processor

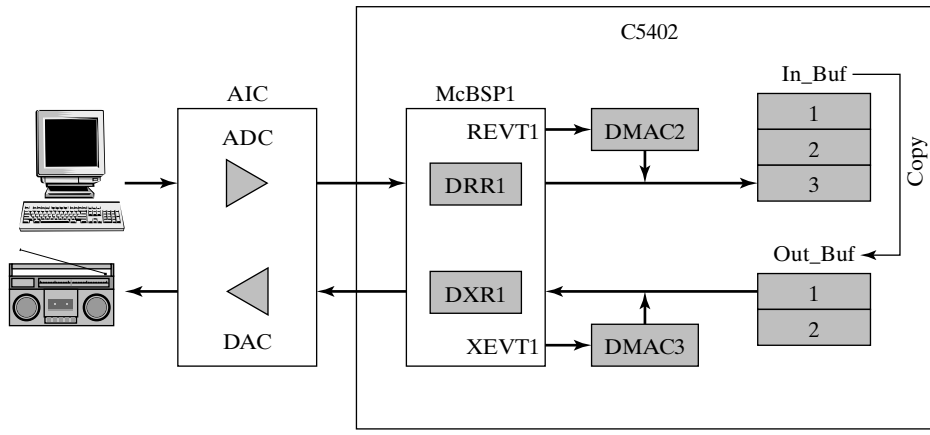

Figure C.8 Data transfer from the ADC/DAC to/from the McBSP and to/from on-chip memory via DMA channels 2 and 3 (reprinted with permission from [1])

via channel 2 of the DMA. A triple buffer is used in the **In_Buf** buffer in DARAM. This data is then copied to the **Out_Buf** buffer organized as a double buffer in DARAM and is output to the DXR1 register of the McBSP1 via channel 3 of the DMA. The DXR1 is then output to the DAC inside the AIC. The analog output is sent to the loudspeaker for playback. In this digital loop-back process, interrupt (REVT1) from the McBSP1 is used to trigger the DMA when complete data are received.

In order to start the digital loop-back process, a C program is written to initialize the DMA and the AIC and to wait for the interrupt from the McBSP. An ISR is activated once the interrupt is detected. In the case of digital loop back, the ISR contains a sequence of code that stores data in multiple buffers. Since the ISR contains the most time-critical code, such as signal-processing algorithms and memory management, it is normally written in assembly.

Both the AIC and the McBSP/DMA need to be configured and initialized before the digital loop-back operation can be performed.

Setting up the Analog Interface Circuit

The C54x on-board peripheral library provides high-level support for DSP applications running on the C54x DSK. This library enables the user to develop applications that can control and operate the peripherals of the DSK platform. Some of the supports include on-board peripheral initialization, power down (reset), register access, and data movement.

The AD50 application programming interfaces (APIs) configure the on-board AIC, including its input and output gains, sampling frequency, settings for the internal registers of the AIC, and digital or analog loop back capability. In addition, the API provides functions to open, close, and reset the AIC, read from and write to the AIC, etc. A detailed listing of the AD50 API can be found in the TMS320C5402 DSK online help [1]. For example, to set up AD50 AIC parameters on the C5402 DSK, we can apply the below C code. A detailed listing of CODEC driver routines can be found in `codec.h` [1].

```
hHandset = codec_open(HANDSET_CODEC);      /* Open handset CODEC */

/* Set codec parameters */
codec_dac_mode(hHandset, CODEC_DAC_15BIT); /* DAC in 15-bit mode */
codec_adc_mode(hHandset, CODEC_ADC_15BIT); /* ADC in 15-bit mode */
codec_ain_gain(hHandset, CODEC_AIN_6dB);   /* 6 dB gain on analog
                                              input to ADC       */
codec_aout_gain(hHandset, CODEC_AOUT_MINUS_6dB);  /* -6 dB gain on
                                        analog output from DAC  */
codec_sample_rate(hHandset,SR_16000);    /* 16 kHz sampling rate */
```

In this case, the 15-bit mode of the ADC and DAC is being used. It represents 15 bits of data plus 1 bit to indicate whether the register data comes from the master or slave device if the read bit is set. In the preceding program, the input to the ADC is being amplified by 6 dB, while the DAC output is being attenuated by 6 dB. The sampling frequency of the ADC/DAC is set to 16 kHz.

Initializing the Direct Memory Access/Multichannel Buffered Serial Port

The initialization program for the DMA/McBSP that follows is used to configure DMA channel 2. The variables and functions listed in this code can be found in the header file `dma54xx.h` for the C5402 processor [1].

```
/* Clear IFR */
INTR_CLR_FLAG(DMAC2);

/* Reset all DMA channels */
dma_reset_all();

/* Initialize DMA channel 2 */
dmsefc = ((DSYNC_REVT1 <<12));      /* DSYNC_REVT1 = 0101b */
dmmcr = ((AUTOINIT_ENABLE << 15) | (DINM_ENABLE << 14) |
(IMOD_HALFBLOCK <<13) | (CTMOD_DEC <<12) | (INDEXMODE_NOMOD << 8) |
(SPACE_DATA << 6) | (INDEXMODE_INC << 2) | (SPACE_DATA));
dmctr = 0xFF;
src_addr = DRR1_ADDR(HANDSET_CODEC);
dst_addr = (unsigned int) &buffer;  /* &buffer is the start
                                       address of input buffer */
dma_init(DMA_CH2, dmsefc, dmmcr, dmctr, SPACE_DATA, src_addr,
SPACE_DATA, dst_addr);

/* Set number of frames for channel 2 */
DMA_FRAMECOUNT(DMA_CH2, 2);

/* Initialize DMA channel 3 */
dmsefc = ((DSYNC_REVT1 <<12));      /* DSYNC_REVT1 = 0101b */
dmmcr = ((AUTOINIT_DISABLE << 15) | (DINM_ENABLE << 14) |
(IMOD_HALFBLOCK <<13) | (CTMOD_DEC <<12) | (INDEXMODE_INC << 8) |
(SPACE_DATA << 6) | (INDEXMODE_NOMOD << 2) | (SPACE_DATA));
dmctr = 0xFF;
dst_addr = DXR1_ADDR(HANDSET_CODEC);
src_addr = (unsigned int) &buffer_out;  /* &buffer is the start
                                           address of input buffer */
dma_init(DMA_CH3, dmsefc, dmmcr, dmctr, SPACE_DATA, src_addr,
SPACE_DATA, dst_addr);

/* Set number of frames for channel 3 */
DMA_FRAMECOUNT(DMA_CH3, 1);

/* Set up global autoinit registers for DMA CH2 input */
dmgsa = src_addr;
dmgda = dst_addr;
dmgcr = 0xFF;
dmgfr = 2;

/* Set up global priority and enable control register for CH2 */
dmprec = ((HIGH_PRIORITY << 10) | (INTSEL_01 << 6));
dmsrcp = SPACE_DATA;
dmdstp = SPACE_DATA;
dmidx0 = 0;
```

```
dmidx1 = 0;
dmfri0 = 0;
dmfri1 = 0;

dma_global_init(dmprec, dmsrcp, dmdstp, dmidx0, dmidx1, dmfri0,
dmfri1, dmgsa, dmgda, dmgcr, dmgfr);

/* Enable channel 2 and 3 */
DMA_ENABLE(DMA_CH2);
DMA_ENABLE(DMA_CH3);

/* prime the serial port to begin input buffer stream */
temp = *(volatile u16*)DRR1_ADDR(HANDSET_CODEC);

/* Enable DMAC2 interrupt */
INTR_ENABLE(DMAC2);

/* Enable global interrupts */
INTR_GLOBAL_ENABLE;

/* Endless loop waiting for DMAC2 interrupt */
   for(;;);
```

The global interrupt is first disabled by clearing the IFR register, and all DMA channels are reset before initialization. Two initialization steps are involved in the DMA channel: (1) local initialization of the individual DMA channel and (2) global initialization and prioritization of the DMA channel.

The local initialization is set up by defining the DMA synchronization (sync) event in the DMSFCn register, as described in Table C.14. The DMA sync mode (DSYN) selected in the example (DSYN = 0101) is based on the McBSP 1 receive event (REVT1) of the C5402 processor. The double-word mode (DBLW) is set to the reset value (DBLW = 0) of single word (16 bits), and the frame count is set to three frames by using the DMA_FRAMECOUNT (DMA_CH2, 2) API, which is initialized to one less than the actual frame count. Using the DMA transfer mode control register (DMMCRn) in Table C.15, we can interpret the initialization as follows: (1) DMA autoinitialization is enabled; (2) the DMA interrupt is generated based on the CTMOD bit = 0, which is a multiframe mode, and DMA interrupts are at end of the frame/block (IMOD = 1); (3) the DMA source address transfer index is not modified; (4) the DMA source address used is data space; (5) the DMA destination address transfer index is post-increment; and (6) the DMA destination address used is data space. The next three local registers to be set up are the DMA channel 2 source address (DMSRC2), destination address (DMDST2), and element count register (DMCTR2). DMSRC2 is set to the DRR1 address of the McBSP1, while DMDST2 is set to the start address of the buffer (&buffer). DMCTR2 is set to 256 elements (specified as 0xFF).

Similarly, we can also set up local registers for channel 3 of the DMA. However, there are some differences in configuring channel 3 (transmit) and channel 2 (receive). DMA autoinitialization is disabled in channel 3 since its source address needs to be updated in the ISR. In channel 3, the source address transfer index is post-incremented, while the destination address transfer index is not modified after

every data transfer. The element count register still remains at 256, and the frame count is now set to 2, as shown in Fig. C.8.

In the preceding program, the global autoinit registers for all DMA channels are set as follows: (1) DMGSA = DRR11 address (0x41) of the McBSP; (2) DMGDA = start address of the buffer; (3) DMA DMGCR = 256 elements; (4) DMGFR = 3 frames; (5) DMPREC selects channel 2 as high priority, while all other channels are of low priority, and DMA channel 2 interrupt is assigned to the interrupt vector as interrupt number IMR/IFR #10; (6) DMSRCP = DMDSTP = data space address; and (7) all of the DMA element and frame address index registers (DMIDX0, DMIDX1, DMFRI0, and DMFRI1) are set to an index offset value of 0 since the element and frame are automatically post-incremented by 1.

After the DMA/McBSP and AIC have been initialized, the selected DMA channels, interrupts, and global interrupts can all be enabled. In the digital loopback process, the processor can go into an endless loop while waiting for the DMA channel 2 interrupt. Once interrupted, the ISR activates, and the processor stores the incoming data from the ADC to the **In_Buf** buffer and arranges the data in a triple-buffer fashion, as shown in Fig. C.8. Once the input buffer is filled up, it is copied to one of the available output buffers (**Out_Buf**) and transmitted to the DXR11 (0x43) of the McBSP via channel 3 of the DMA before being passed to the DAC for analog playback. The DMPREC register can be polled to determine when a block transfer on DMA channel 3 is completed.

More examples on interfacing different CODECs to C5402 DSP processors are given in [4, 5]. Readers can refer to these application notes for more detailed information on and code for the interface.

C.2 INTERFACE WITH THE HOST PROCESSOR VIA THE HOST PORT INTERFACE

As introduced in Chapter 3, HPI is a dedicated parallel interface for host-to-DSP processor communications. Unlike the parallel bus that is dedicated to memory accesses, the HPI functions as a slave to the host processor, which can be a microcontroller or another DSP processor. In the latest C54x processor family, the HPI is referred to as the enhanced HPI, which comes in either an 8-bit or 16-bit interface. The HPI-8 is the 8-bit version that transfers 8 bits of data between the host and DSP, while the HPI-16 allows a 16-bit word to be transferred. In a 100 MHz processor, the HPI-8 and HPI-16 have a throughput of 21 Mbit/sec and 33 Mbit/sec, respectively. These throughput values are accurate only if no other DMA channel is active.

The C5402 processor has one HPI-8. Data is exchanged between the C54x processor and the host processor through the C54x processor's on-chip memory, which is accessible to both the host and C54x processors. For a 16-bit data exchange, two 8-bit reads/writes from/to the memory must be performed.

The HPI-8 uses the DMA bus to gain access to on-chip memory in the processor. As shown in Fig. C.9, the HPI has a dedicated port on the DMA controller and makes a request to use the DMA bus. The DMA controller completes the current DMA transfer in progress before granting the HPI-8 access to the DMA bus, and all

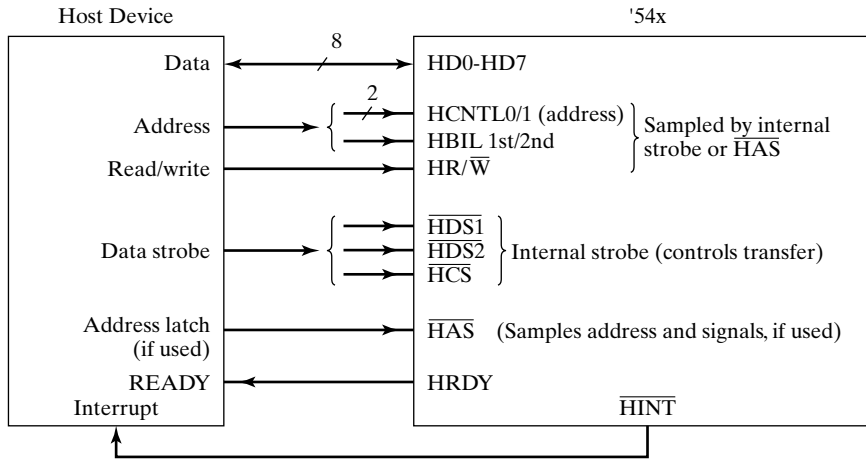

Figure C.9 Block diagram of linking the host processor to the C54x (reprinted with permission from [2])

TABLE C.18 HPI-8 Input-Control Signals and Functions (reprinted with permission from [2])

HCNTL1	HCNTL0	Description
0	0	Host can read from or write to the HPI control register, HPIC.
0	1	Host can read from or write to the HPI data latches. HPIA is automatically postincremented each time a read is performed, and preincremented each time a write is performed.
1	0	Host can read from or write to the address register, HPIA. This register points to the '54x on-chip RAM.
1	1	Host can read from or write to the HPI data latches. HPIA is not affected.

pending DMA channel transfers are halted. The DMA channel only resumes its activities after the completion of HPI-8 access. Therefore, the HPI-8 can be considered as the seventh DMA channel with the highest priority over all six DMA channels.

As shown in Fig. C.9, the HD0-7 is the data bus of the HPI-8. Two control inputs (HCNTL0/1) are used to select internal HPI-8 registers with different options: HPI control register (HPIC), HPI address register (HPIA), or the HPI data register (HPID). Table C.18 shows the HPI-8 input-control signals and their functions.

The HPIC contains the control and status bits for HPI-8 operations and can be accessed directly by both the host and C54x processors. The HPIA serves as a pointer to C54x on-chip memory. The HPIA can also be configured in autoincrement mode (HCNTL1 = 1 and HCNTL0 = 0) for consecutive transfers. The HPID contains the data to be transferred to/from the address specified in the HPIA. Since the HPI is a slave interface, only the host has direct access to the HPIA and HPID.

In addition, the host byte identification input line (HBIL) indicates whether the first or second byte is transferred, while the HR/W strobe signal indicates the read or write access. Two host data strobe signals (HDS1 and HDS2) are used to control the transfer of data across the interface. The HCS is the host chip select line that enables the HPI-8. The host address strobe (HAS) is used only when communicating with host devices that have a multiplexed bus for address and data. The HPI ready (HRDY) pin provides a means to let the host know if the HPI-8 is ready for a new transaction. Finally, the HPI-8 includes an interrupt (HINT) pin to interrupt the host. This HINT pin can be set and cleared by writing to the HINT bit in the HPIC register.

In this section, we briefly highlight the sequence on how HPI-8 access can be carried out. The host must first initialize the HPIC register by specifying the byte order bit (BOB) in the HPIC. If BOB = 1, the first byte of a transfer is least significant; otherwise, the first byte is most significant. The host can then write to the HPIA register with the correct byte alignment. On-chip memory is automatically read, and the contents at the given address are transferred to the two 8-bit data latches, which are the first and second bytes of the HPID register. The host then must perform a read of the HPID to retrieve the data in the latches. In the case of write access to the HPI, the first byte data latch is overwritten by the data coming from the host, while the HBIL pin is low. The second byte data latch, in turn, is overwritten by the data coming from the host while the HBIL pin is high. After the write access, the byte in the data latches is transferred as one 16-bit word to the on-chip RAM at the address specified by the HPIA register. Note that when autoincrement is enabled, data read causes a post-increment of the HPIA, and the data write causes a pre-increment of the HPIA.

There are two types of interrupts between the host and the C54x processor: (1) the host interrupts the C54x processor via DSPINT (DSP CPU interrupt), and (2) the C54x processor interrupts the host using HINT. The host device writes a 1 to the DSPINT bit in the HPIC register for interrupting the C54x processor. Continue interrupts from the host can be generated without the need to write 0 to the DSPINT bit. The C54x writes a 1 to the HINT bit in the HPIC register for interrupting the host processor. When HINT = 1, the HINT output is driven low. When HINT = 0, the output is driven high. The host can clear the interrupt by writing a 1 to the HINT bit.

Other topics regarding HPI transfer operations, resetting the HPI, and HPI performance considerations can be found in [2].

C.3 FURTHER EXPLORATION

Several other important peripheral programming issues and internal DSP hardware setups are necessary for implementing a complete system. The user can refer to the application notes from Texas Instruments, which can be downloaded from the websites listed in Appendix E. Several important implementation issues include the following:

1. Bootloading is used to transfer boot-up program code from external memory into internal memory following powerup. This bootloading feature in the DSP

processor eliminates the need for mask programming in the processor's internal ROM. There are different ways to download code into the processor's memory. These boot modes include the parallel-port boot, serial-port boot, I/O boot, HPI boot, etc. A detailed description can be found in [6].

2. The UART controller allows data to be transmitted without a clock signal to the receiver. The UART performs a parallel-to-serial conversion on data received from the DSP processor and performs a serial-to-parallel conversion on data received from the external device. The data packet is preceded by a start bit, which alerts the receiver that data is about to be sent. It is followed immediately by data bits starting from the LSB to the MSB. The transmitter may add a parity bit for simple error checking at the receiver. A stop bit terminates the complete data packet. Refer to [7, 8] for more detailed information on the UART implementation.
3. Interfacing a DSP processor to external memory can be carried out by using external (address and data) buses and other control lines, as shown in [9, 10].

C.4 TRENDS IN PERIPHERAL PROGRAMMING

Texas Instruments supports a new chip support library [11], which is a collection of functions, macros, and symbols used to configure and control on-chip peripherals. The chip support library consists of a standard protocol (data type, macro, and function) for programming on-chip peripherals and provides a powerful GUI that generates peripheral register values and C files for peripheral initialization. In addition, it provides a convenient platform for managing multiple peripheral resources and peripherals with multiple channels.

Moreover, the DSP/BIOS driver development kit is available for simplifying the development of device drivers for numerous DSP peripherals. The driver development kit complements the chip support library by providing drivers for sophisticated peripherals, such as a multimedia card, video port, CODEC, etc. Driver development kit drivers also use the chip support library for peripheral initialization and control. For more detailed information, please refer to [12].

SUGGESTED READINGS

1 Texas Instruments. *C5402 DSK One-Day Workshop*. Version 2.11, 2001.
2 Texas Instruments. *TMS320C54x DSP: Enhanced Peripherals*. Volume 5, SPRU302, 1999.
3 Texas Instruments. *TMS320C54x DSP Design Workshop*. Version 4.21, 2001.
4 Texas Instruments. *Interfacing AC97 Codec to TMS320C5402*. SPRA777, 2001.
5 Texas Instruments. *Interfacing the TLV320AIC10/11 Codec to the TMS320C5402 DSP*. SLAA109, 2000.
6 Texas Instruments. *TMS320VC5402 and TMS320UC5402 Bootloader*. SPRA618A, 2002.
7 Texas Instruments. *Programming the C5404/5406/5407 UART Peripheral*. SPRA023A, 2002.
8 Texas Instruments. *Implementing a Software UART on the TMS320C54x with the McBSP and DMA*. SPRA 661A, 2000.

9 Texas Instruments. *TMS320C54x Interface with SDRAM*. SPRA531, 1999.
10 Texas Instruments. *Connecting TMS320C54x DSP with Flash Memory*. SPRA585, 1999.
11 Texas Instruments. *TMS320C54x Chip Support Library API Reference Guide*. SPRU420B, 2002.
12 *http://www.ti.com/driverdevkit*.