TMS320C54x DSP Reference Set

Volume 5: Enhanced Peripherals

Literature Number: SPRU302
June 1999

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

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

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE ("CRITICAL APPLICATIONS"). TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.

Preface

Read This First

About This Manual

The TMS320C54x is a fixed-point digital signal processor (DSP) in the Texas Instruments (TI™) TMS320 family. These devices are characterized by a low-power, enhanced architecture core. This book, the fifth volume of a 5-volume set, provides information about the enhanced peripherals available on some of these devices.

Many device references are shown with an apostrophe (') replacing the usual alphanumeric prefix (ex. '5420 instead of TMS320VC5420). Unless otherwise specified, all references to the '54x in this book apply to the TMS320VC54x.

This user's guide describes the enhanced peripherals available on the '5402, '5410, and '5420 devices, and explains their operations.

The main topics discussed are:

- Host port interface (HPI)
- Multi-channel buffered serial ports (McBSPs)
- Programmable clock generator with a multiple phase-locked loop (PLL)
- DMA controller (DMA)
- Enhanced external input/output interface (EnhXIO)
- Interprocessor FIFO

For all non-enhanced peripheral information related to '54x devices, see *TMS320C54x DSP, CPU and Peripherals, Volume 1,* referenced in the section titled *Related Documents from Texas Instruments*.

Notational Conventions

This book uses the following conventions.

- The TMS320C54x DSP can use either of two forms of the instruction set: a mnemonic form or an algebraic form. This book uses the mnemonic form of the instruction set. For information about the mnemonic form of the instruction set, see *TMS320C54x DSP Reference Set, Volume 2: Mnemonic Instruction Set*. For information about the algebraic form of the instruction set, see *TMS320C54x DSP Reference Set, Volume 3: Algebraic Instruction Set*.

- Program listings and program examples are shown in a `special typeface`.

  Here is a segment of a program listing:

```
STL    A,*AR1+      ;Int_RAM(I)=0
RSBX   INTM         ;Globally enable interrupts
B      MAIN_PG      ;Return to foreground program
```

- Square brackets, [and], identify an optional parameter. If you use an optional parameter, specify the information within the brackets; do not type the brackets themselves.

Information About Cautions

This book contains cautions.

This is an example of a caution statement.

A caution statement describes a situation that could potentially damage your software or equipment.

The information in a caution is provided for your protection. Please read each caution carefully.

Related Documentation from Texas Instruments

The following books provide related documentation for the TM320C54x. To obtain a copy of any of these TI documents, call the Texas Instruments Literature Response Center at (800) 477-8924. When ordering, please identify the book by its title and literature number.

Many of these documents are located on the internet at http://www.ti.com.

TMS320C54x DSP Reference Set, Volume 1: CPU and Peripherals (literature number SPRU131) describes the TMS320C54x 16-bit fixed-point general-purpose digital signal processors. Covered are its architecture, internal register structure, data and program addressing, the instruction pipeline, and on-chip peripherals. Also includes development support information, parts lists, and design considerations for using the XDS510 emulator.

TMS320C54x DSP Reference Set, Volume 2: Mnemonic Instruction Set (literature number SPRU172) describes the TMS320C54x digital signal processor mnemonic instructions individually. Also includes a summary of instruction set classes and cycles.

TMS320C54x DSP Reference Set, Volume 3: Algebraic Instruction Set (literature number SPRU179) describes the TMS320C54x digital signal processor algebraic instructions individually. Also includes a summary of instruction set classes and cycles.

TMS320C54x DSP Reference Set, Volume 4: Applications Guide (literature number SPRU173) describes software and hardware applications for the TMS320C54x digital signal processor. Also includes development support information, parts lists, and design considerations for using the XDS510 emulator.

TMS320C54x DSP Reference Set, Volume 5: Enhanced Peripherals (literature number SPRU302) describes the enhanced peripherals available on the TMS320C54x digital signal processors. Includes the multichannel buffered serial ports (McBSPs), direct memory access (DMA) controller, the HPI-8 and HPI-16 host port interfaces, and the interprocessor.

TMS320C54x, TMS320LC54x, TMS320VC54x Fixed-Point Digital Signal Processors (literature number SPRS039) data sheet contains the electrical and timing specifications for these devices, as well as signal descriptions and pinouts for all of the available packages.

TMS320C54x DSKplus User's Guide (literature number SPRU191) describes the TMS320C54x digital signal processor starter kit (DSK), which allows you to execute custom 'C54x code in real time and debug it line by line. Covered are installation procedures, a description of the debugger and the assembler, customized applications, and initialization routines.

TMS320C54x Assembly Language Tools User's Guide (literature number SPRU102) describes the assembly language tools (assembler, linker, and other tools used to develop assembly language code), assembler directives, macros, common object file format, and symbolic debugging directives for the 'C54x generation of devices.

TMS320C5xx C Source Debugger User's Guide (literature number SPRU099) tells you how to invoke the 'C54x emulator, evaluation module, and simulator versions of the C source debugger interface. This book discusses various aspects of the debugger interface, including window management, command entry, code execution, data management, and breakpoints. It also includes a tutorial that introduces basic debugger functionality.

TMS320C54x Code Generation Tools Getting Started Guide (literature number SPRU147) describes how to install the TMS320C54x assembly language tools and the C compiler for the 'C54x devices. The installation for MS-DOS™, OS/2™, SunOS™, Solaris™, and HP-UX™ 9.0x systems is covered.

TMS320C54x Evaluation Module Technical Reference (literature number SPRU135) describes the 'C54x evaluation module, its features, design details and external interfaces.

TMS320C54x Optimizing C Compiler User's Guide (literature number SPRU103) describes the 'C54x C compiler. This C compiler accepts ANSI standard C source code and produces TMS320 assembly language source code for the 'C54x generation of devices.

TMS320C54x Simulator Getting Started (literature number SPRU137) describes how to install the TMS320C54x simulator and the C source debugger for the 'C54x. The installation for MS-DOS™, PC-DOS™, SunOS™, Solaris™, and HP-UX™ systems is covered.

TMS320 Third-Party Support Reference Guide (literature number SPRU052) alphabetically lists over 100 third parties that provide various products that serve the family of TMS320 digital signal processors. A myriad of products and applications are offered—software and hardware development tools, speech recognition, image processing, noise cancellation, modems, etc.

TMS320C548/C549 Bootloader Technical Reference (literature number SPRU288) describes the process the bootloader uses to transfer user code from an external source to the program memory at power up. (Presently available only on the internet.)

TMS320 DSP Development Support Reference Guide (literature number SPRU011) describes the TMS320 family of digital signal processors and the tools that support these devices. Included are code-generation tools (compilers, assemblers, linkers, etc.) and system integration and debug tools (simulators, emulators, evaluation modules, etc.). Also covered are available documentation, seminars, the university program, and factory repair and exchange.

Technical Articles

A wide variety of related documentation is available on digital signal processing. These references fall into one of the following application categories:

- ❑ General-Purpose DSP
- ❑ Graphics/Imagery
- ❑ Speech/Voice
- ❑ Control
- ❑ Multimedia
- ❑ Military
- ❑ Telecommunications
- ❑ Automotive
- ❑ Consumer
- ❑ Medical
- ❑ Development Support

In the following list, references appear in alphabetical order according to author. The documents contain beneficial information regarding designs, operations, and applications for signal-processing systems; all of the documents provide additional references. Texas Instruments strongly suggests that you refer to these publications.

General-Purpose DSP*:*

1) Chassaing, R., Horning, D.W., *"Digital Signal Processing with Fixed and Floating-Point Processors"*, CoED, USA, Volume 1, Number 1, pages 1-4, March 1991.

2) Defatta, David J., Joseph G. Lucas, and William S. Hodgkiss, *Digital Signal Processing: A System Design Approach*, New York: John Wiley, 1988.

3) Erskine, C., and S. Magar, “Architecture and Applications of a Second-Generation Digital Signal Processor,” *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, USA, 1985.

4) Essig, D., C. Erskine, E. Caudel, and S. Magar, “A Second-Generation Digital Signal Processor,” *IEEE Journal of Solid-State Circuits*, USA, Volume SC-21, Number 1, pages 86-91, February 1986.

5) Frantz, G., K. Lin, J. Reimer, and J. Bradley, “The Texas Instruments TMS320C25 Digital Signal Microcomputer,” *IEEE Microelectronics*, USA, Volume 6, Number 6, pages 10-28, December 1986.

6) Gass, W., R. Tarrant, T. Richard, B. Pawate, M. Gammel, P. Rajasekaran, R. Wiggins, and C. Covington, “Multiple Digital Signal Processor Environment for Intelligent Signal Processing,” *Proceedings of the IEEE, USA*, Volume 75, Number 9, pages 1246-1259, September 1987.

7) Jackson, Leland B., *Digital Filters and Signal Processing*, Hingham, MA: Kluwer Academic Publishers, 1986.

8) Jones, D.L., and T.W. Parks, *A Digital Signal Processing Laboratory Using the TMS32010*, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1987.

9) Lim, Jae, and Alan V. Oppenheim, *Advanced Topics in Signal Processing*, Englewood Cliffs, NJ: Prentice- Hall, Inc., 1988.

10) Lin, K., G. Frantz, and R. Simar, Jr., “The TMS320 Family of Digital Signal Processors,” *Proceedings of the IEEE*, USA, Volume 75, Number 9, pages 1143-1159, September 1987.

11) Lovrich, A., Reimer, J., *“An Advanced Audio Signal Processor”* , Digest of Technical Papers for 1991 International Conference on Consumer Electronics, June 1991.

12) Magar, S., D. Essig, E. Caudel, S. Marshall and R. Peters, “An NMOS Digital Signal Processor with Multiprocessing Capability,” *Digest of IEEE International Solid-State Circuits Conference*, USA, February 1985.

13) Oppenheim, Alan V., and R.W. Schafer, *Digital Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1975 and 1988.

14) Papamichalis, P.E., and C.S. Burrus, “Conversion of Digit-Reversed to Bit-Reversed Order in FFT Algorithms,” *Proceedings of ICASSP 89*, USA, pages 984-987, May 1989.

15) Papamichalis, P., and R. Simar, Jr., “The TMS320C30 Floating-Point Digital Signal Processor,” *IEEE Micro Magazine*, USA, pages 13-29, December 1988.

16) Papamichalis, P.E., "FFT Implementation on the TMS320C30," *Proceedings of ICASSP 88*, USA, Volume D, page 1399, April 1988.

17) Parks, T.W., and C.S. Burrus, *Digital Filter Design*, New York, NY: John Wiley and Sons, Inc., 1987.

18) Peterson, C., Zervakis, M., Shehadeh, N., *"Adaptive Filter Design and Implementation Using the TMS320C25 Microprocessor"* , Computers in Education Journal, USA, Volume 3, Number 3, pages 12-16, July-September 1993.

19) Prado, J., and R. Alcantara, "A Fast Square-Rooting Algorithm Using a Digital Signal Processor," *Proceedings of IEEE*, USA, Volume 75, Number 2, pages 262-264, February 1987.

20) Rabiner, L.R. and B. Gold, *Theory and Applications of Digital Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1975.

21) Simar, Jr., R., and A. Davis, "The Application of High-Level Languages to Single-Chip Digital Signal Processors," *Proceedings of ICASSP 88*, USA, Volume D, page 1678, April 1988.

22) Simar, Jr., R., T. Leigh, P. Koeppen, J. Leach, J. Potts, and D. Blalock, "A 40 MFLOPS Digital Signal Processor: the First Supercomputer on a Chip," *Proceedings of ICASSP 87*, USA, Catalog Number 87CH2396-0, Volume 1, pages 535-538, April 1987.

23) Simar, Jr., R., and J. Reimer, "The TMS320C25: a 100 ns CMOS VLSI Digital Signal Processor," *1986 Workshop on Applications of Signal Processing to Audio and Acoustics*, September 1986.

24) Texas Instruments, *Digital Signal Processing Applications with the TMS320 Family*, 1986; Englewood Cliffs, NJ: Prentice-Hall, Inc., 1987.

25) Treichler, J.R., C.R. Johnson, Jr., and M.G. Larimore, *A Practical Guide to Adaptive Filter Design*, New York, NY: John Wiley and Sons, Inc., 1987.

***Graphics/Imagery**:*

1) Reimer, J., and A. Lovrich, "Graphics with the TMS32020," *WESCON/85 Conference Record*, USA, 1985.

***Speech/Voice**:*

1) DellaMorte, J., and P. Papamichalis, "Full-Duplex Real-Time Implementation of the FED-STD-1015 LPC-10e Standard V.52 on the TMS320C25," *Proceedings of SPEECH TECH 89*, pages 218-221, May 1989.

2) Gray, A.H., and J.D. Markel, *Linear Prediction of Speech*, New York, NY: Springer-Verlag, 1976.

3) Frantz, G.A., and K.S. Lin, "A Low-Cost Speech System Using the TMS320C17," *Proceedings of SPEECH TECH '87*, pages 25-29, April 1987.

4) Papamichalis, P., and D. Lively, "Implementation of the DOD Standard LPC-10/52E on the TMS320C25," *Proceedings of SPEECH TECH '87*, pages 201-204, April 1987.

5) Papamichalis, Panos, *Practical Approaches to Speech Coding*, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1987.

6) Pawate, B.I., and G.R. Doddington, "Implementation of a Hidden Markov Model-Based Layered Grammar Recognizer," *Proceedings of ICASSP 89*, USA, pages 801-804, May 1989.

7) Rabiner, L.R., and R.W. Schafer, *Digital Processing of Speech Signals*, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1978.

8) Reimer, J.B. and K.S. Lin, "TMS320 Digital Signal Processors in Speech Applications," *Proceedings of SPEECH TECH '88*, April 1988.

9) Reimer, J.B., M.L. McMahan, and W.W. Anderson, "Speech Recognition for a Low-Cost System Using a DSP," *Digest of Technical Papers for 1987 International Conference on Consumer Electronics*, June 1987.

***Control**:*

1) Ahmed, I., "16-Bit DSP Microcontroller Fits Motion Control System Application," *PCIM*, October 1988.

2) Ahmed, I., "Implementation of Self Tuning Regulators with TMS320 Family of Digital Signal Processors," *MOTORCON '88*, pages 248-262, September 1988.

3) Ahmed, I., and S. Lindquist, "Digital Signal Processors: Simplifying High-Performance Control," *Machine Design*, September 1987.

4) Ahmed, I., and S. Meshkat, "Using DSPs in Control," *Control Engineering*, February 1988.

5) Allen, C. and P. Pillay, *"TMS320 Design for Vector and Current Control of AC Motor Drives"*, Electronics Letters, UK, Volume 28, Number 23, pages 2188-2190, November 1992.

6) Bose, B.K., and P.M. Szczesny, "A Microcomputer-Based Control and Simulation of an Advanced IPM Synchronous Machine Drive System for Electric Vehicle Propulsion," *Proceedings of IECON '87*, Volume 1, pages 454-463, November 1987.

7) Hanselman, H., "LQG-Control of a Highly Resonant Disc Drive Head Positioning Actuator," *IEEE Transactions on Industrial Electronics*, USA, Volume 35, Number 1, pages 100-104, February 1988.

8) Lovrich, A., G. Troullinos, and R. Chirayil, "An All-Digital Automatic Gain Control," *Proceedings of ICASSP 88*, USA, Volume D, page 1734, April 1988.

9) Matsui, N. and M. Shigyo, *"Brushless DC Motor Control Without Position and Speed Sensors"*, IEEE Transactions on Industry Applications, USA, Volume 28, Number 1, Part 1, pages 120-127, January-February 1992.

10) Meshkat, S., and I. Ahmed, "Using DSPs in AC Induction Motor Drives," *Control Engineering*, February 1988.

11) Panahi, I. and R. Restle, *"DSPs Redefine Motion Control"*, Motion Control Magazine, December 1993.

***Multimedia**:*

1) Reimer, J., *"DSP-Based Multimedia Solutions Lead Way Enhancing Audio Compression Performance"*, Dr. Dobbs Journal, December 1993.

2) Reimer, J., G. Benbassat, and W. Bonneau Jr., *"Application Processors: Making PC Multimedia Happen"*, Silicon Valley PC Design Conference, July 1991.

***Military**:*

1) Papamichalis, P., and J. Reimer, "Implementation of the Data Encryption Standard Using the TMS32010," *Digital Signal Processing Applications*, 1986.

***Telecommunications**:*

1) Ahmed, I., and A. Lovrich, "Adaptive Line Enhancer Using the TMS320C25," *Conference Records of Northcon/86*, USA, 14/3/1-10, September/October 1986.

2) Casale, S., R. Russo, and G. Bellina, "Optimal Architectural Solution Using DSP Processors for the Implementation of an ADPCM Transcoder," *Proceedings of GLOBECOM '89*, pages 1267-1273, November 1989.

3) Cole, C., A. Haoui, and P. Winship, "A High-Performance Digital Voice Echo Canceller on a SINGLE TMS32020," *Proceedings of ICASSP 86*, USA, Catalog Number 86CH2243-4, Volume 1, pages 429-432, April 1986.

4) Cole, C., A. Haoui, and P. Winship, "A High-Performance Digital Voice Echo Canceller on a Single TMS32020," *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, USA, 1986.

5) Lovrich, A., and J. Reimer, "A Multi-Rate Transcoder," *Transactions on Consumer Electronics*, USA, November 1989.

6) Lovrich, A. and J. Reimer, *"A Multi-Rate Transcoder"* , Digest of Technical Papers for 1989 International Conference on Consumer Electronics, June 7-9, 1989.

7) Lu, H., D. Hedberg, and B. Fraenkel, "Implementation of High-Speed Voiceband Data Modems Using the TMS320C25," *Proceedings of ICASSP 87*, USA, Catalog Number 87CH2396-0, Volume 4, pages 1915-1918, April 1987.

8) Mock, P., "Add DTMF Generation and Decoding to DSP- µP Designs," *Electronic Design*, USA, Volume 30, Number 6, pages 205-213, March 1985.

9) Reimer, J., M. McMahan, and M. Arjmand, "ADPCM on a TMS320 DSP Chip," *Proceedings of SPEECH TECH 85*, pages 246-249, April 1985.

10) Troullinos, G., and J. Bradley, "Split-Band Modem Implementation Using the TMS32010 Digital Signal Processor," *Conference Records of Electro/86 and Mini/Micro Northeast*, USA, 14/1/1-21, May 1986.

***Automotive**:*

1) Lin, K., "Trends of Digital Signal Processing in Automotive," *International Congress on Transportation Electronic (CONVERGENCE '88*), October 1988.

***Consumer**:*

1) Frantz, G.A., J.B. Reimer, and R.A. Wotiz, “Julie, The Application of DSP to a Product,” *Speech Tech Magazine*, USA, September 1988.

2) Reimer, J.B., and G.A. Frantz, “Customization of a DSP Integrated Circuit for a Customer Product,” *Transactions on Consumer Electronics*, USA, August 1988.

3) Reimer, J.B., P.E. Nixon, E.B. Boles, and G.A. Frantz, “Audio Customization of a DSP IC,” *Digest of Technical Papers for 1988 International Conference on Consumer Electronics*, June 8-10 1988.

***Medical**:*

1) Knapp and Townshend, “A Real-Time Digital Signal Processing System for an Auditory Prosthesis,” *Proceedings of ICASSP 88*, USA, Volume A, page 2493, April 1988.

2) Morris, L.R., and P.B. Barszczewski, “Design and Evolution of a Pocket-Sized DSP Speech Processing System for a Cochlear Implant and Other Hearing Prosthesis Applications,” *Proceedings of ICASSP 88*, USA, Volume A, page 2516, April 1988.

***Development Support**:*

1) Mersereau, R., R. Schafer, T. Barnwell, and D. Smith, “A Digital Filter Design Package for PCs and TMS320,” *MIDCON/84 Electronic Show and Convention*, USA, 1984.

2) Simar, Jr., R., and A. Davis, “The Application of High-Level Languages to Single-Chip Digital Signal Processors,” *Proceedings of ICASSP 88*, USA, Volume 3, pages 1678-1681, April 1988.

Trademarks

HP-UX is a trademark of Hewlett-Packard Company.

MS-DOS is a registered trademark of Microsoft Corporation.

OS/2 and PC-DOS are trademarks of International Business Machines Corporation.

PAL® is a registered trademark of Advanced Micro Devices, Inc.

Solaris and SunOS are trademarks of Sun Microsystems, Inc.

SPARC is a trademark of SPARC International, Inc., but licensed exclusively to Sun Microsystems, Inc.

Windows is a registered trademark of Microsoft Corporation.

320 Hotline On-line, TI, XDS510, and XDS510WS are trademarks of Texas Instruments Incorporated.

Micro Star is a trademark of Texas Instruments Incorporated.

SPI is a trademark of the Motorola Corporation.

Contents

Figures

Tables

Examples

Chapter 1

Introduction

The ’54x device is a fixed-point digital signal processor (DSP) in the TMS320 family. The central processing unit (CPU), with its modified Harvard architecture, minimizes power consumption and adds a high degree of parallelism. System performance is further enhanced by versatile addressing modes and instruction sets. These and other characteristics allow the ’54x to meet the specific needs of real-time embedded applications such as telecommunications.

All ’54x devices have general-purpose I/O pins (XF and $\overline{\text{BIO}}$), a timer (two on the ’5402), a clock generator, a software-programmable wait-state generator, and a programmable bank-switching module. Different types and quantities of serial ports, host-port interfaces, and clock generators are specific to the various ’54x devices.

This chapter discusses the enhanced peripherals available on the ’5402, ’5410, and ’5420 devices.

Topic **Page**

1.1 Overview of the ’54x Enhanced Peripherals

The sections that follow provide an overview of the enhanced peripherals available on the ’54x.

1.1.1 Multi-channel Buffered Serial Ports (McBSPs)

The’54x family provides high-speed, full-duplex multi-channel buffered serial ports (McBSPs) that allow direct interface to other ’54x devices, codecs, and other devices in a system. The MCBSPs are an enhanced version of the standard serial port interface found on other ’54x devices. Some features of the McBSP include:

- ❏ Double-buffered transmit and triple-buffered receive operation to allow a continuous data stream.
- ❏ Independent framing and clocking for receive and transmit.
- ❏ Multi-channel transmit and receive up to 128 channels.
- ❏ Data sizes including 8, 12, 16, 20, 24, and 32 bits.
- ❏ μ-law and A-law companding.
- ❏ Programmable polarity for both frame synchronization and clocks.
- ❏ Programmable internal clocks and frame synchronization.

For more information on the McBSPs, see Chapter 2.

1.1.2 Direct Memory Access (DMA) Controller

The 6-channel ’54x direct memory access (DMA) controller transfers data between points in the memory map without intervention by the CPU. The DMA allows the following movements of data to occur in the background of CPU operation: data to and from internal program/data memory; internal peripherals such as the McBSP’s; and external memory devices. The DMA has six independent programmable channels allowing six different contexts for DMA operation. For more information on the DMA, see Chapter 3.

1.1.3 Host Port Interfaces (HPI-8 and HPI-16)

There are two enhanced host port interfaces on the ’54x, the HPI-8 and the HPI-16. These are 8-bit and 16-bit parallel ports that provide an interface to a host processor. Information is exchanged between the ’54x and the host processor through ’54x on-chip memory that is accessible to both the host processor and the ’54x. For more details about the operation of the HPI-8 and HPI-16, see Chapters 4 and 5, respectively.

Chapter 2

Multichannel Buffered Serial Port (McBSP)

Depending on the specific device, the 54x digital signal processor provides multiple high-speed, full-duplex, multichannel buffered serial ports (McBSPs) that allow direct interface to other ’54x devices, codecs, and other devices in a system. The ’5402 provides two, the ’5410 three, and the ’5420 six McBSPs. They are based on the standard serial port interface found on other ’54x devices.

This chapter describes the operation of the McBSPs, and includes register definitions and timing diagrams.

Topic **Page**

2.1 McBSP Features

The McBSP is based on the standard serial port interface found on the TMS320C2x, ’C20x, ’C5x, and ’C54x devices. The McBSP provides:

- Full-duplex communication
- Double-buffered transmit and triple-buffered receive data registers, which allow a continuous data stream
- Independent framing and clocking for receive and transmit
- Direct interface to industry-standard codecs, analog interface chips (AICs), and other serially connected A/D and D/A devices
- External shift clock generation, or an internal, programmable-frequency shift clock

In addition, the McBSP has the following capabilities:

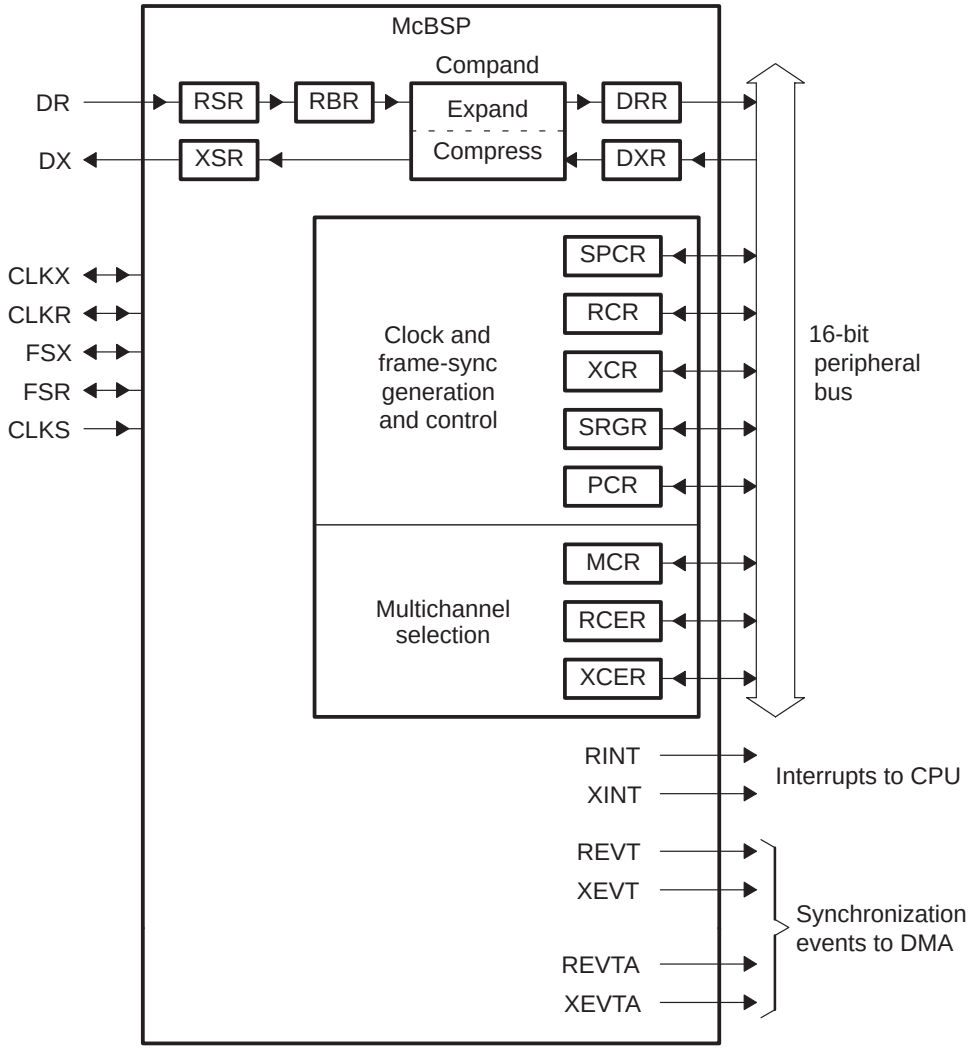
- Direct interface to:
  - T1/E1 framers
  - MVIP switching compatible and ST-BUS compliant devices including:
    - MVIP framers
    - H.100 framers
    - SCSA framers
  - IOM-2 compliant devices
  - AC97 compliant devices (The necessary multi-phase frame-synchronization capability is provided.)
  - IIS compliant devices
  - SPI™ devices
- Multichannel transmit and receive up to 128 channels
- A wide selection of data sizes including 8, 12, 16, 20, 24, and 32 bits

  Note: Data sizes are referred to as *word* or *serial word* throughout this document and can be 8, 12, 16, 20, 24, or 32 bits, in contrast to the true definition of *word* which is 32 bits.
- μ-Law and A-Law companding
- 8-bit data transfers with option of LSB or MSB first
- Programmable polarity for both frame synchronization and data clocks
- Highly programmable internal clock and frame generation

2.2 McBSP General Description

The McBSP consists of a data path and a control path connected to external devices by seven pins as shown in Figure 2–1.

Figure 2–1. McBSP Internal Block Diagram



Data is communicated to devices interfacing the McBSP via the data transmit (DX) pin for transmit and the data receive (DR) pin for receive. Control information in the form of clocking and frame synchronization is communicated via CLKX, CLKR, FSX, and FSR. The '54x communicates with the McBSP through 16-bit-wide control registers accessible via the internal peripheral bus.

The CPU or the DMA controller reads the received data from the data receive register (DRR[1,2]) and writes the data to be transmitted to the data transmit register (DXR[1,2]). Data written to DXR[1,2] is shifted out to DX via the transmit shift register (XSR[1,2]). Similarly, receive data on the DR pin is shifted into the receive shift register (RSR[1,2]) and copied into the receive buffer register (RBR[1,2]). RBR[1,2] is then copied to DRR[1,2], which can be read by the CPU or the DMA controller. This allows simultaneous movement of internal and external data communications.

DRR2, RBR2, RSR2, DXR2, and XSR2 registers are not utilized (written, read, or shifted) if the receive/transmit word length, R/XWDLEN[1,2], is specified for 8-, 12-, or 16-bit mode.

The remaining registers accessible to the CPU configure the control mechanism of the McBSP. These registers are listed in Table 2–2, *McBSP Registers*, on page 2-5. The control block consists of internal clock generation, frame-synchronization signal generation, and their control and multichannel selection. This control block sends notification of important events to the CPU and DMA controller via the two interrupt and four event signals shown in Table 2–3, *McBSP CPU Interrupts and DMA Event Synchronization*, on page 2-6.

Table 2–1. McBSP Interface Signals

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

† I = Input, O = Output, Z = High-impedance

Table 2–2. McBSP Registers

Hex Address			Sub-Address	Acronym	Register Name†	Section
McBSP 0	McBSP 1	McBSP 2				
—	—	—		RBR[1,2]	McBSP receive buffer register 1 and 2	2.2
—	—	—		RSR[1,2]	McBSP receive shift register 1 and 2	2.2
—	—	—		XSR[1,2]	McBSP transmit shift register 1 and 2	2.2
0020	0040	0030	—	DRR2x	McBSP data receive register 2	2.2
0021	0041	0031	—	DRR1x	McBSP data receive register 1	2.2
0022	0042	0032	—	DXR2x	McBSP data transmit register 2	2.2
0023	0043	0033	—	DXR1x	McBSP data transmit register 1	2.2
0038	0048	0034	—	SPSAx	McBSP sub-address register	
0039	0049	0035	0x0000	SPCR1x	McBSP serial port control register 1	2.2.1
0039	0049	0035	0x0001	SPCR2x	McBSP serial port control register 2	2.2.1
0039	0049	0035	0x0002	RCR1x	McBSP receive control register 1	2.2.2
0039	0049	0035	0x0003	RCR2x	McBSP receive control register 2	2.2.2
0039	0049	0035	0x0004	XCR1x	McBSP transmit control register 1	2.2.2
0039	0049	0035	0x0005	XCR2x	McBSP transmit control register 2	2.2.2
0039	0049	0035	0x0006	SRGR1x	McBSP sample rate generator register 1	2.5.1.1
0039	0049	0035	0x0007	SRGR2x	McBSP sample rate generator register 2	2.5.1.1
0039	0049	0035	0x0008	MCR1x	McBSP multichannel register 1	2.6.1
0039	0049	0035	0x0009	MCR2x	McBSP multichannel register 2	2.6.1
0039	0049	0035	0x000A	RCERAx	McBSP receive channel enable register partition A	2.6.3.1
0039	0049	0035	0x000B	RCERBx	McBSP receive channel enable register partition B	2.6.3.1

† RBR[1,2], RSR[1,2], and XSR[1,2] are not directly accessible via the CPU or DMA.

Table 2–2. McBSP Registers (Continued)

Hex Address			Sub-Address	Acronym	Register Name†	Section
McBSP 0	McBSP 1	McBSP 2				
0039	0049	0035	0x000C	XCERAx	McBSP transmit channel enable register partition A	2.6.3.1
0039	0049	0035	0x000D	XCERBx	McBSP transmit channel enable register partition B	2.6.3.1
0039	0049	0035	0x000E	PCRx	McBSP pin control register	2.2.1 & 2.9

† RBR[1,2], RSR[1,2], and XSR[1,2] are not directly accessible via the CPU or DMA.

Table 2–3. McBSP CPU Interrupts and DMA Event Synchronization

Interrupt Name	Description	Section
RINT	Receive interrupt to CPU	2.3.3
XINT	Transmit interrupt to CPU	2.3.3
REVT	Receive synchronization event to DMA	2.3.2.1
XEVT	Transmit synchronization event to DMA	2.3.2.2
REVTA	Receive synchronization eventA to DMA	2.6.4
XEVTA	Transmit synchronization eventA to DMA	2.6.4

2.2.1 Serial Port Configuration

The serial port is configured via the two 16-bit serial port control registers 1 and 2 (SPCR[1,2]) and the Pin Control Register (PCR*).* These registers are shown in Figure 2–2, Figure 2–3 and Figure 2–4, respectively. SPCR[1, 2] and the PCR contain McBSP status information and also bits that can be configured for desired operation. The operation of each bit-field is discussed in the sections listed in Table 2–4, *Serial Port Control Register 1 (SPCR1) Bit-Field Description,* on page 2-7; Table 2–5, *Serial Port Control Register 2 (SPCR2) Bit-Field Description*, on page 2-10; and Table 2–6, *Pin Control Register (PCR) Bit-field Description*, on page 2-12.

In addition to the PCR being used to configure the McBSP pins as inputs or outputs during normal serial port operation, it is used to configure the serial port pins as general purpose inputs or outputs during receiver and/or transmitter reset. This is described in section 2.9, *McBSP Pins as General Purpose I/O*, on page 2-96.

Figure 2–2. Serial Port Control Register 1 (SPCR1)

15	14	13	12	11	10	8
DLB	RJUST		CLKSTP		reserved	
RW,+0	RW,+0	RW,+0	RW,+0		R,+0	

7	6	5	4	3	2	1	0
DXENA	ABIS	RINTM		RSYNCERR	RFULL	RRDY	RRST
RW,+0	RW,+0	RW,+0		RW,+0	R,+0	R,+0	RW,+0†

Note: R = Read, W = Write, +0 = Value at reset

† R, +0 means read-only, reset value is 0. RW, +0 means read and write allowed, reset value is 0.

Table 2–4. Serial Port Control Register 1 (SPCR1) Bit-Field Descriptions

Bit	Name	Function		Section
15	DLB	Digital Loop Back Mode		2.5.2.5
		DLB = 0	Digital loop back mode disabled	
		DLB = 1	Digital loop back mode enabled	
14–13	RJUST	Receive Sign-Extension and Justification Mode		2.3.8
		RJUST = 00	Right-justify and zero-fill MSBs in DRR[1,2]	
		RJUST = 01	Right-justify and sign-extend MSBs in DRR[1,2]	
		RJUST = 10	Left-justify and zero-fill LSBs in DRR[1,2]	
		RJUST = 11	Reserved	

Table 2–4. Serial Port Control Register 1 (SPCR1) Bit-Field Descriptions (Continued)

Bit	Name	Function		Section
12–11	CLKSTP	Clock Stop Mode		2.7
		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.		2.3.4.8
		DXENA = 0	DX enabler is off	
		DXENA = 1	DX enabler is on	
6	ABIS	ABIS Mode		2.6.4
		ABIS = 0	A-bis mode is disabled	
		ABIS = 1	A-bis mode is enabled	
5–4	RINTM	Receive Interrupt Mode		2.3.3
		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	
3	RSYNCERR	Receive Synchronization Error		2.3.7.2
		RSYNCERR = 0	No synchronization error	
		RSYNCERR = 1	Synchronization error detected by McBSP.	

Table 2–4. Serial Port Control Register 1 (SPCR1) Bit-Field Descriptions (Continued)

Bit	Name	Function		Section
2	RFULL	Receive Shift Register (RSR[1,2]) Full		2.3.7.1
		RFULL = 0	RBR[1,2] is not in overrun condition	
		RFULL = 1	DRR[1,2] is not read, RBR[1,2] is full and RSR[1,2] is also full with new word	
1	RRDY	Receiver Ready		2.3.2
		RRDY = 0	Receiver is not ready.	
		RRDY = 1	Receiver is ready with data to be read from DRR[1,2].	
0	$\overline{\text{RRST}}$	Receiver reset. This resets and enables the receiver.		2.3.1
		$\overline{\text{RRST}}$ = 0	The serial port receiver is disabled and in reset state.	
		$\overline{\text{RRST}}$ = 1	The serial port receiver is enabled.	

Figure 2–3. Serial Port Control Register 2 (SPCR2)

15	14	13	12	11	10	9	8
reserved†						FREE	SOFT
R,+0						RW,+0	RW,+0

7	6	5	4	3	2	1	0
FRST	GRST	XINTM		XSYNCERR‡	XEMPTY	XRDY	XRST
RW,+0	RW,+0	RW,+0		RW,+0	R,+0	R,+0	RW,+0

Note: R = Read, W = Write, +0 = Value at reset

† Note: This and all reserved bit-fields have NO storage associated with them; however, they are always read as 0.
‡ CAUTION: Writing a 1 to this bit sets the error condition; thus, it is mainly used for testing purposes or if this operation is desired.

Table 2–5. Serial Port Control Register 2 (SPCR2) Bit-Field Descriptions

Bit	Name	Function		Section
15–10	rsvd	Reserved		
9	FREE	Free Running Mode		2.8
		FREE = 0	Free running mode is disabled	
		FREE = 1	Free running mode is enabled	
8	SOFT	Soft Bit		2.8
		SOFT = 0	SOFT mode is disabled	
		SOFT = 1	SOFT mode is enabled	
7	$\overline{\text{FRST}}$	Frame-Sync Generator Reset		2.3.1
		$\overline{\text{FRST}}$ = 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		2.3.1
		$\overline{\text{GRST}}$ = 0	Sample rate generator is reset	
		$\overline{\text{GRST}}$ = 1	Sample rate generator is pulled out of reset. CLKG is driven as per programmed value in sample rate generator registers (SRGR[1,2]).	

Table 2–5. Serial Port Control Register 2 (SPCR2) Bit-Field Descriptions (Continued)

Bit	Name	Function		Section
5–4	XINTM	Transmit Interrupt Mode		2.3.3
		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		2.3.7.2
		XSYNCERR = 0	No synchronization error	
		XSYNCERR = 1	Synchronization error detected by McBSP.	
2	$\overline{\text{XEMPTY}}$	Transmit Shift Register (XSR[1,2]) Empty		2.3.7.4
		$\overline{\text{XEMPTY}}$ = 0	XSR[1,2] is empty	
		$\overline{\text{XEMPTY}}$ = 1	XSR[1,2] is not empty	
1	XRDY	Transmitter Ready		2.3.2
		XRDY = 0	Transmitter is not ready.	
		XRDY = 1	Transmitter is ready for new data in DXR[1,2].	
0	$\overline{\text{XRST}}$	Transmitter reset.	This resets and enables the transmitter.	2.3.1
		$\overline{\text{XRST}}$ = 0	The serial port transmitter is disabled and in reset state.	
		$\overline{\text{XRST}}$ = 1	The serial port transmitter is enabled.	

Figure 2–4. Pin Control Register (PCR)

15	14	13	12	11	10	9	8
reserved		XIOEN	RIOEN	FSXM	FSRM	CLKXM	CLKRM
R,+0		RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0

7	6	5	4	3	2	1	0
reserved	CLKS_STAT	DX_STAT	DR_STAT	FSXP	FSRP	CLKXP	CLKRP
R,+0	R,+0	R,+0	R,+0	RW,+0	RW,+0	RW,+0	RW,+0

Note: R = Read, W = Write, +0 = Value at reset

Table 2–6. Pin Control Register (PCR) Bit-Field Descriptions

Bit	Name	Function		Section
15–14	reserved	Reserved		
13	XIOEN	Transmit general purpose I/O mode only when $\overline{\text{XRST}}$ = 0 in SPCR[1,2]		2.9
		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[1,2]		2.9
		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.	

Table 2–6. Pin Control Register (PCR) Bit-Field Descriptions (Continued)

Bit	Name	Function		Section
11	FSXM	Transmit Frame-Synchronization Mode		2.5.3.3 and 2.9
		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		2.5.3.2 and 2.9
		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 (see section 2.5.1.1).	
9	CLKXM	Transmitter Clock Mode		2.5.2.7 and 2.9
		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.	

Table 2–6. Pin Control Register (PCR) Bit-Field Descriptions (Continued)

Bit	Name	Function	Section
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.	2.5.2.6 and 2.9
7	rsvd	Reserved	
6	CLKS_ STAT	CLKS pin status. Reflects value on CLKS pin when selected as a general purpose input.	2.9
5	DX_STAT	DX pin status. Reflects value driven on to DX pin when selected as a general purpose output.	2.9
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	2.3.4.1 and 2.9
2	FSRP	Receive Frame-Synchronization Polarity FSRP = 0 Frame-synchronization pulse FSR is active high FSRP = 1 Frame-synchronization pulse FSR is active low	2.3.4.1 and 2.9

Table 2–6. Pin Control Register (PCR) Bit-Field Descriptions (Continued)

Bit	Name	Function		Section
1	CLKXP	Transmit Clock Polarity		2.3.4.1 and 2.9
		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		2.3.4.1 and 2.9
		CLKRP = 0	Receive data sampled on falling edge of CLKR	
		CLKRP = 1	Receive data sampled on rising edge of CLKR	

2.2.2 Receive and Transmit Control Registers: RCR[1,2] and XCR[1,2]

The receive and transmit control registers (RCR[1,2] and XCR[1,2]) configure various parameters of the receive and transmit operations, respectively. They are shown in Figure 2–5, *Receive Control Register 1* (*RCR1*), on page 2-16; Figure 2–6, *Receive Control Register 2 (RCR2), on page* 2-17; Figure 2–7, *Transmit Control Register 1 (XCR1)*, on page 2-19; and Figure 2–8, *Transmit Control Register 2 (XCR2),* on page 2-20.

The operation of each bit-field is discussed in the sections listed in Table 2–7, *Receive Control Register 1 (RCR1) Bit-Field Description,* on page 2-16; Table 2–8, *Receive Control Register 2 (RCR2) Bit-Field Description,* on page 2-17; Table 2–9, *Transmit Control Register 1 (XCR1) Bit-Field Description,* on page 2-19; and Table 2–10, *Transmit Control Register 2 (XCR2) Bit-Field Description,* on page 2-20.

Figure 2–5. Receive Control Register 1 (RCR1)

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

Note: R = Read, W = Write, +0 = Value at reset

Table 2–7. Receive Control Register 1 (RCR1) Bit-Field Descriptions

Bit	Name	Function		Section
15	rsvd	Reserved		
14–8	RFRLEN1	Receive Frame Length 1		2.3.4.3
		RFRLEN1 = 000 0000	1 word per frame	
		RFRLEN1 = 000 0001	2 words per frame	
			|	
			|	
		RFRLEN1 = 111 1111	128 words per frame	
7–5	RWDLEN1	Receive Word Length 1		2.3.4.4
		RWDLEN1 = 000	8 bits	
		RWDLEN1 = 001	12 bits	
		RWDLEN1 = 010	16 bits	
		RWDLEN1 = 011	20 bits	
		RWDLEN1 = 100	24 bits	
		RWDLEN1 = 101	32 bits	
		RWDLEN1 = 11X	Reserved	
4–0	rsvd	Reserved		

Figure 2–6. Receive Control Register 2 (RCR2)

15	14 – 8	7 – 5	4 – 3	2	1 – 0
RPHASE	RFRLEN2	RWDLEN2	RCOMPAND	RFIG	RDATDLY
RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0

Note: R = Read, W = Write, +0 = Value at reset

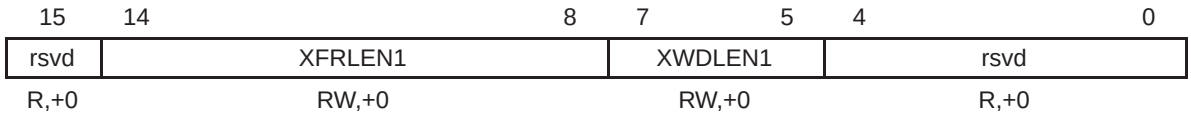
Table 2–8. Receive Control Register 2 (RCR2) Bit-Field Descriptions

Bit	Name	Function		Section
15	RPHASE	Receive Phases		2.3.4.2
		RPHASE = 0	Single-phase frame	
		RPHASE = 1	Dual-phase frame	
14–8	RFRLEN2	Receive Frame Length 2		2.3.4.3
		RFRLEN2 = 000 0000	1 word per frame	
		RFRLEN2 = 000 0001	2 words per frame	
			|	
			|	
		RFRLEN1 = 111 1111	128 words per frame	
7–5	RWDLEN2	Receive Word Length 2		2.3.4.4
		RWDLEN2 = 000	8 bits	
		RWDLEN2 = 001	12 bits	
		RWDLEN2 = 010	16 bits	
		RWDLEN2 = 011	20 bits	
		RWDLEN2 = 100	24 bits	
		RWDLEN2 = 101	32 bits	
		RWDLEN2 = 11X	Reserved	

Table 2–8. Receive Control Register 2 (RCR2) Bit-Field Descriptions (Continued)

Bit	Name	Function		Section
4–3	RCOMPAND	Receive companding mode. Modes other than 00b are only enabled when the appropriate RWDLEN is 000b, indicating 8-bit data.		2.4
		RCOMPAND = 00	No companding, data transfer starts with MSB first.	
		RCOMPAND = 01	No companding, 8-bit data, transfer starts with LSB first.	
		RCOMPAND = 10	Compand using μ-law for receive data.	
		RCOMPAND = 11	Compand using A-law for receive data.	
2	RFIG	Receive Frame Ignore		2.3.6.2
		RFIG = 0	Receive frame-synchronization pulses after the first restarts the transfer.	
		RFIG = 1	Receive frame-synchronization pulses after the first are ignored.	
1–0	RDATDLY	Receive data delay		2.3.4.6
		RDATDLY = 00	0-bit data delay	
		RDATDLY = 01	1-bit data delay	
		RDATDLY = 10	2-bit data delay	
		RDATDLY = 11	Reserved	

Figure 2–7. Transmit Control Register 1 (XCR1)



Note: R = Read, W = Write, +0 = Value at reset

Table 2–9. Transmit Control Register 1 (XCR1) Bit-Field Descriptions

Bit	Name	Function		Section
15	rsvd	Reserved		
14–8	XFRLEN1	Transmit Frame Length 1		2.3.4.3
		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		2.3.4.4
		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		

Figure 2–8. Transmit Control Register 2 (XCR2)

15	14 – 8	7 – 5	4 – 3	2	1 – 0
XPHASE	XFRLEN2	XWDLEN2	XCOMPAND	XFIG	XDATDLY
RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0

Note: R = Read, W = Write, +0 = Value at reset

Table 2–10. Transmit Control Register 2 (XCR2) Bit-Field Descriptions

Bit	Name	Function		Section
15	XPHASE	Transmit Phases		2.3.4.2
		XPHASE = 0	Single-phase frame	
		XPHASE = 1	Dual-phase frame	
14–8	XFRLEN2	Transmit Frame Length 2		2.3.4.3
		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		2.3.4.4
		XWDLEN2 = 000	8 bits	
		XWDLEN2 = 001	12 bits	
		XWDLEN2 = 010	16 bits	
		XWDLEN2 = 011	20 bits	
		XWDLEN2 = 100	24 bits	
		XWDLEN2 = 101	32 bits	
		XWDLEN2 = 11X	Reserved	

Table 2–10. Transmit Control Register 2 (XCR2) Bit-Field Descriptions (Continued)

Bit	Name	Function		Section
4–3	XCOMPAND	Transmit companding mode. Modes other than 00b are only enabled when the appropriate XWDLEN is 000b, indicating 8-bit data.		2.4
		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		2.3.6.2
		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		2.3.4.6
		XDATDLY = 00	0-bit data delay	
		XDATDLY = 01	1-bit data delay	
		XDATDLY = 10	2-bit data delay	
		XDATDLY = 11	Reserved	

2.3 Data Transmission and Reception Flow

As shown in Figure 2–1, *McBSP Internal Block Diagram*, on page 2-3, the receive operation is triple buffered and the transmit operation is double buffered. Receive data arrives on DR and is shifted into RSR[1,2]. Once a full word (8-, 12-, 16-, 20-, 24-, or 32-bit) is received, RSR[1,2] is copied to the receive buffer register, RBR[1,2], only if RBR[1,2] is not full. RBR[1,2] is then copied to DRR[1,2], unless DRR[1,2] is not read by the CPU or DMA.

Transmit data is written by the CPU or DMA to DXR[1,2]. If there is no data in XSR[1,2], the value in DXR[1,2] is copied to XSR[1,2]; otherwise, DXR[1,2] is copied to XSR[1,2] when the last bit of data is shifted out from DX. After transmit frame synchronization, XSR[1,2] begins shifting out the transmit data from DX.

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

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

1) Device reset ($\overline{\text{RS}}$ = 0) places the receiver, transmitter and the sample rate generator in reset. When the device reset is removed ($\overline{\text{RS}}$ = 1), $\overline{\text{GRST}}$ = $\overline{\text{FRST}}$ = $\overline{\text{RRST}}$ = $\overline{\text{XRST}}$ = 0, keeping the entire serial port in the reset state.

2) The serial port transmitter and receiver can be independently reset by the $\overline{\text{RRST}}$ and $\overline{\text{XRST}}$ bits in the serial port control registers. The sample rate generator is reset by the $\overline{\text{GRST}}$ bit in SPCR2.

Table 2–11, *Reset State of McBSP Pins,* on page 2-23 shows the state of McBSP pins when the serial port is reset due to device reset and receiver/transmitter reset ($\overline{\text{XRST}}$ = $\overline{\text{RRST}}$ = $\overline{\text{FRST}}$ = 0).

Table 2–11. Reset State of McBSP Pins

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

- **Device reset or McBSP reset**: When the McBSP is reset in any of the above two ways, the state machine is reset to its initial state. This initial state includes resetting all counters and status bits. The receive status bits include RFULL, RRDY, and RSYNCERR. The transmit status bits include $\overline{\text{XEMPTY}}$, XRDY, and XSYNCERR.

- **Device reset:** When McBSP is reset due to device reset (device pin $\overline{\text{RS}}$ = 0), the entire serial port including the transmitter, receiver, and the sample rate generator is reset. All input-only pins and three-state pins should be in a known state. The output-only pin, DX, is in the high-impedance state. Since the sample rate generator is also reset ($\overline{\text{GRST}}$ = 0), the sample rate generator clock, CLKG, is driven by the divide-by-2 CPU clock, whereas the frame-sync signal, FSG, is not generated. For more information on sample rate generator reset, see section 2.5.1.2, *Sample Rate Generator Reset Procedure*, on page 2-61. When the device is pulled out of reset, the serial port remains in reset condition ([R/X]$\overline{\text{RST}}$ = 0 and $\overline{\text{FRST}}$ = 0) and in this condition the DR and DX pins may be used as general purpose I/O as described in section 2.9, *McBSP Pins as General Purpose I/O*, on page 2-96.

- **McBSP reset:** When the receive and transmitter reset bits, $\overline{\text{RRST}}$ and $\overline{\text{XRST}}$, are written with a zero, the respective portions of the McBSP are reset and activity in the corresponding section of the serial port stops. All input-only pins, such as DR and CLKS, and all other pins that are

configured as inputs, are in a known state. FS(R/X) is driven to its inactive state (same as its polarity bit FS[R/X]) if it is an output. If CLK(R/X) is programmed as an output, it will be driven by CLKG, provided that $\overline{GRST}$ = 1. Lastly, the DX pin will be in the high-impedance state when the transmitter and/or the device is reset. During normal operation, the sample rate generator can be reset by writing a zero to $\overline{GRST}$. $\overline{GRST}$ should be low only when neither the transmitter nor the receiver is using the sample rate generator. In this case, the internal sample rate generator clock (CLKG) and its frame-sync signal (FSG) are driven inactive low. When the sample rate generator is not in the reset state ($\overline{GRST}$ = 1), pins FSR and FSX are in an inactive state when $\overline{RRST}$ = 0 and $\overline{XRST}$ = 0, respectively, even if they are outputs driven by FSG. This ensures that when only one portion of the McBSP is in reset, the other portion can continue operation when $\overline{FRST}$ = 1 and its frame sync is driven by FSG. For more information on sample rate generator reset, see section 2.5.1.2, *Sample Rate Generator Reset Procedure,* on page 2-61.

- **Sample rate generator reset:** As noted earlier, the sample rate generator is reset when the device or its reset bit, $\overline{GRST}$, is written with a zero. In the case of device reset, the sample rate generator clock, CLKG, is driven by a divide-by-2 CPU clock, whereas the frame-sync pulse, FSG, is driven inactive low. If you want to reset the sample rate generator when neither the transmitter nor the receiver is fed by CLKG and FSG, you can program $\overline{GRST}$ in SRGR2 to zero. Here, CLKG and FSG are driven inactive-low. When $\overline{GRST}$ = 1, CLKG comes up running as programmed in SRGR1. Later, if $\overline{FRST}$ = 1, FSG is driven active-high after the programmed frame period (FPER + 1) number of CLKG cycles has elapsed.

After device reset is complete ($\overline{RS}$ = 1), the serial port initialization procedure is as follows:

1) Set $\overline{XRST}$ = $\overline{RRST}$ = $\overline{FRST}$ = 0 in SPCR[1,2]. If coming out of device reset, this step is not required.

2) Program only the McBSP configuration registers (and not the data registers) listed in Table 2–2, *McBSP Registers*, on page 2-5, as required when the serial port is in reset state ($\overline{XRST}$ = $\overline{RRST}$ = $\overline{FRST}$ = 0).

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

4) Set up data acquisition as required such as writing to DXR.

5) Set $\overline{XRST}$ = $\overline{RRST}$= 1 to enable the serial port. Note that the value written to SPCR[1,2] at this time should have only the reset bits changed to 1, and the remaining bit-fields should have the same value as in step 2 above.

6) Set $\overline{\text{FRST}}$ = 1, if internally generated frame sync is required.

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

Alternatively, on either write (steps 1 and 5), the transmitter and receiver may be placed in or taken out of reset individually by modifying the desired bit. Note that the necessary duration of the active-low period of $\overline{\text{XRST}}$ or $\overline{\text{RRST}}$ is at least two bit-clocks (CLKR/CLKX) wide.

The above procedure for reset initialization can be applied in general when the receiver or transmitter has to be reset during its normal operation, and also when the sample rate generator is not used for either operation.

Notes:

(a) The appropriate bit-fields in the serial port configuration registers, SPCR[1,2], PCR, RCR[1,2], XCR[1,2], and SRGR[1,2], should only be modified by the user when the affected portion of the serial port is in reset.

(b) Data Transmit Register, DXR[1,2], should be loaded by the CPU or DMA only when the transmitter is not in reset ($\overline{\text{XRST}}$ = 1). An exception to this rule is during digital loop back mode described in section 2.4.1, *Companding Internal Data*, on page 2-55.

(c) The multichannel selection registers, MCR, XCER[A/B], and RCER[A/B], can be modified at any time as long as they are not being used by the current block in the multichannel selection. See section 2.6.3.2 on page 2-82 for further details in this case.

Example 2–1 shows values in the control registers that reset and configure the transmitter while the receiver is running.

Example 2–1. Resetting and Configuring the Transmitter While Receiver is Running

SPCR1 = 0x0001 SPCR2 = 0x0030	Transmitter reset, transmit interrupt (XINT to CPU) generated by XSYNCERR; receiver is running with RINT driven by RRDY.
PCR = 0x0A00	FSX determined by FSGM in SRGR, transmit clock driven by external source, receive clock continues to be driven by sample rate generator.
SRGR1 = 0x0001 SRGR2 = 0x2000	CPU clock drives the sample rate generator clock (CLKG) after a divide-by-2. A DXR[1,2]-to-XSR[1,2] copy generates the transmit frame-sync signal.
XCR1 = 0x0840 XCR2= 0x8421	Dual-phase frame; phase 1 has eight 16-bit words; phase 2 has four 12-bit words, and 1-bit data delay
SPCR2 = 0x0031	Transmitter taken out of reset

2.3.2 Determining Ready Status

RRDY and XRDY indicate the ready state of the McBSP receiver and transmitter, respectively. Serial port writes and reads may be synchronized by polling RRDY and XRDY, or by using the events to DMA (REVT and XEVT in normal mode, and REVTA and XEVTA in A-bis mode), or by interrupts to CPU (RINT and XINT), which the events generate. Note that reading DRR[1,2] and writing to DXR[1,2] affects RRDY and XRDY.

2.3.2.1 Receive Ready Status: REVT, RINT, and RRDY

RRDY = 1 indicates that the RBR[1,2] contents have been copied to DRR[1,2] and that the data can be read by the CPU or DMA. Once that data has been read by either the CPU or DMA, RRDY is cleared to 0. Also, at device reset or serial port receiver reset ($\overline{\text{RRST}}$ = 0), RRDY is cleared to 0 to indicate no data has yet been received and loaded into DRR[1,2]. RRDY directly drives the McBSP receive event to the DMA (REVT). Also, the McBSP receive interrupt (RINT) to the CPU may be driven by RRDY, if RINTM = 00b in SPCR1.

2.3.2.2 Transmit Ready Status: XEVT, XINT, and XRDY

XRDY = 1 indicates that the DXR[1,2] contents have been copied to XSR[1,2] and that DXR[1,2] is ready to be loaded with a new data word. When the transmitter transitions from reset to non-reset ($\overline{\text{XRST}}$ transitions from 0 to 1), XRDY also transitions from 0 to 1 indicating that DXR[1,2] is ready for new data. Once new data is loaded by the CPU or DMA, XRDY is cleared to 0. However, once this data is copied from DXR[1,2] to XSR[1,2], XRDY transitions again from 0 to 1. Now once again, the CPU or DMA can write to DXR[1,2] although

XSR[1,2] has not been shifted out on DX yet. XRDY directly drives the transmit synchronization event to the DMA (XEVT or XEVTA). In addition, the transmit interrupt (XINT) to the CPU may also be driven by XRDY, if XINTM = 00b in SPCR2.

2.3.3 CPU Interrupts: (R/X)INT

The receive interrupt (RINT) and transmit interrupt (XINT) signals the CPU of changes to the serial port status. Four options exist for configuring these interrupts. The options are set by the receive/transmit interrupt mode bit-field, (R/X)INTM, in SPCR[1,2].

1) (R/X)INTM=00b. Interrupt on every serial word by tracking the (R/X)RDY bits in SPCR[1,2]. Sections 2.3.2.1 and 2.3.2.2 describe the RRDY and XRDY bits.

2) (R/X)INTM=01b. Interrupt after every 16-channel block boundary (in multichannel selection mode) has been crossed within a frame. In any other serial transfer case, this setting is not applicable; and therefore, no interrupts are generated. For details, see section 2.6.3.3, *Update Interrupt,* on page 2-83.

3) (R/X)INTM=10b. Interrupt on detection of frame-synchronization pulses. This generates an interrupt even when the transmitter/receiver is in reset. This is done by synchronizing the incoming frame-sync pulse to the CPU clock and sending it to the CPU via (R/X)INT. This is described in section 2.5.3.4, *Frame Detection for Initialization,* on page 2-68.

4) (R/X)INTM = 11b. Interrupt on frame-synchronization error. Note that if any of the other interrupt modes are selected, (R/X)SYNCERR may be read when servicing the interrupts to detect this condition. See sections 2.3.7.2 and 2.3.7.5 for more detail on synchronization error.

   Note that the last three options listed above are applicable as interrupts to the CPU, and not as events to the DMA.

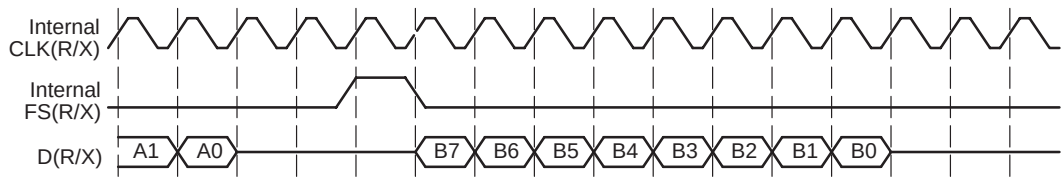
2.3.4 Frame and Clock Configuration

Figure 2–9 shows the typical operation of McBSP clock and frame-sync signals. Serial clocks CLKR, and CLKX define the boundaries between bits for receive and transmit, respectively. Similarly, frame-sync signals FSR and FSX define the beginning of a serial word.

The McBSP allows configuration of various parameters for data frame synchronization. This can be done independently for receive and transmit, which includes the following items:

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

Figure 2–9. Frame and Clock Operation



2.3.4.1 Frame and Clock Operation

Receive and transmit frame-sync pulses can be generated either internally by the sample rate generator (see section 2.5.1, *Sample Rate Generator Clocking and Framing,* on page 2-58) or driven by an external source. The source of frame sync is selected by programming the mode bit, FS(R/X)M, in the PCR. FSR is also affected by the GSYNC bit in SRGR2 (for details, see section 2.5.3.2, *Receive Frame-Sync Selection: DLB, FSRM, GSYNC,* on page 2-67). Similarly, receive and transmit clocks can be selected to be inputs or outputs by programming the mode bit, CLK(R/X)M, in the PCR.

When FSR and FSX are inputs (FSXM=FSRM=0, external frame-sync pulses), the McBSP detects them on the internal falling edge of clock, internal CLKR, and internal CLKX, respectively (see Figure 2–41, *Clock and Frame Generation*, on page 2-57). The receive data arriving at the DR pin is also sampled on the falling edge of internal CLKR. Note that these internal clock signals are either derived from external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that they are driven by the sample rate generator, they are generated (transition to their active state) on the rising edge of internal clock, CLK(R/X). Similarly, data on the DX pin is output on the rising edge of internal CLKX. See section 2.3.4.6, on page 2-34, for further details.

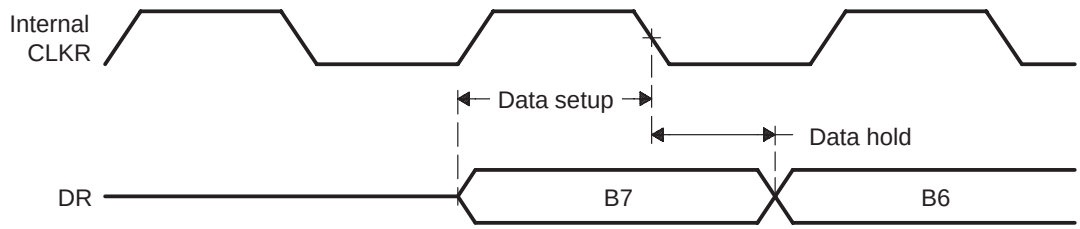
FSRP, FSXP, CLKRP, and CLKXP configure the polarities of FSR, FSX, CLKR, and CLKX signals as shown in Table 2–6, *Pin Control Register (PCR) Bit-Field Description,* on page 2-12. All frame-sync signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP), and FSRP = FSXP = 1, the external active-low frame-sync signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC = 0) is selected, the internal active-high sync signals are inverted, if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin. Figure 2–41, on page 2-57 shows this inversion using XOR gates.

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

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

Note that CLKRP = CLKXP in a system where the same clock (internal or external) is used to clock the receiver and transmitter. The receiver uses the opposite edge as the transmitter to ensure valid setup and hold of data around this edge. Figure 2–10 shows how data, clocked by an external serial device using a rising edge, may be sampled by the McBSP receiver with the falling edge of the same clock.

Figure 2–10. Receive Data Clocking



2.3.4.2 Frame-Synchronization Phases

Frame synchronization indicates the beginning of a transfer on the McBSP. The data stream following frame synchronization may have two phases, phase 1 and phase 2. The number of phases can be selected by the phase bit, (R/X)PHASE, in RCR2 and XCR2. The number of words per frame and bits per word can be independently selected for each phase via (R/X)FRLEN[1,2] and (R/X)WDLEN[1,2] respectively. Figure 2–11 shows an example of a frame where the first phase consists of 2 words of 12 bits each followed by a second phase of three words of 8 bits each. Note that the entire bit stream in the frame is contiguous. There are no gaps either between words or between phases. Table 2–12 shows the bit-fields in the receive/transmit control register (RCR[1,2]/XCR[1,2]) that control the number of words per frame and bits per word for each phase, for both the receiver and transmitter. The maximum number of words per frame is 128 for a single-phase frame and 256 for a dual-phase frame. The number of bits per word can be 8, 12, 16, 20, 24, or 32 bits.

Figure 2–11. Dual-Phase Frame Example

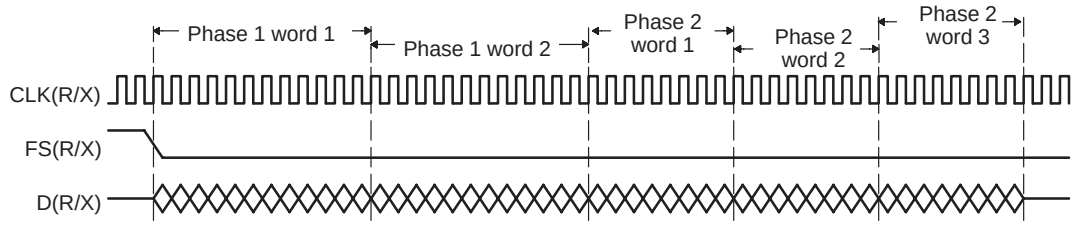


Table 2–12. RCR[1,2]/XCR[1,2] Bit-Fields Controlling Words per Frame and Bits per Word

Serial Port McBSP0/1	Frame Phase	RCR[1,2]/XCR[1,2] Bit-Field Control	
		Words per Frame	Bits per Word
Receive	1	RFRLEN1	RWDLEN1
Receive	2	RFRLEN2	RWDLEN2
Transmit	1	XFRLEN1	XWDLEN1
Transmit	2	XFRLEN2	XWDLEN2

2.3.4.3 Frame Length: (R/X)FRLEN[1,2]

Frame length can be defined as the number of serial words (8-, 12-, 16-, 20-, 24-, or 32-bit) transferred per frame. The length corresponds to the number of words or logical time slots or channels per frame-synchronization signal. The 7-bit (R/X)FRLEN[1,2] field in (R/X)CR[1,2] supports up to 128 words per frame as shown in Table 2–13. (R/X)PHASE = 0 represents a single-phase data frame and (R/X)PHASE = 1 selects a dual phase for the data stream. Note that for a single-phase frame, FRLEN2 is a don't care. The user is cautioned to program the frame length fields with [*w minus 1*], where *w* represents the number of words per frame. For the example in Figure 2–11, (R/X)FRLEN1 = 1 or 0000001b, and (R/X)FRLEN2 = 2 or 0000010b.

Table 2–13. McBSP Receive/Transmit Frame Length (1,2) Configuration

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

2.3.4.4 Word Length: (R/X)WDLEN[1,2]

The 3-bit (R/X)WDLEN[1,2] fields in the receive/transmit control register determine the word length in bits-per-word for the receiver and transmitter for each phase of the frame, as shown in Table 2–12. Table 2–14 shows how the value of these fields selects particular word lengths in bits.

For the example in Figure 2–11 on page 2-30, (R/X)WDLEN1 = 001b, and (R/X)WDLEN2 = 000b.

Notes:

(a) If (R/X)PHASE = 0 indicating a single-phase frame, (R/X)WDLEN2 is not used by the McBSP, and its value is a don't care.

(b) If the specified word length is larger than 16 bits, D(X/R)R2 must be written or read before D(X/R)R1.

Table 2–14. McBSP Receive/Transmit Word Length Configuration

(R/X)WDLEN[1,2]	McBSP Word Length (bits)
000	8
001	12
010	16
011	20
100	24
101	32
110	reserved
111	reserved

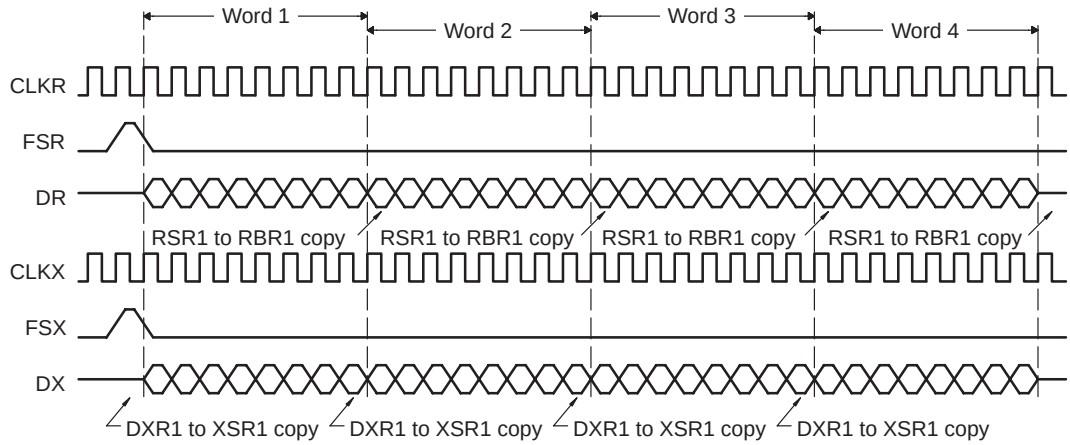
2.3.4.5 Data Packing Using Frame Length and Word Length

The frame length and word length can be manipulated to effectively pack data. For example, consider a situation where four 8-bit words are transferred in a single-phase frame as shown in Figure 2–12. In this case:

- (R/X)FRLEN1 = 0000011b, 4-word frame
- (R/X)PHASE = 0, single-phase frame
- (R/X)FRLEN2 = X
- (R/X)WDLEN1 = 000b, 8-bit word

In this case, four 8-bit data elements are transferred to and from the McBSP by the CPU or DMA. Thus, four reads from DRR1 and four writes to DXR1 are necessary for each frame.

Figure 2–12. Single-Phase Frame of Four 8-Bit Words



The example in Figure 2–12 can also be treated as a data stream of a single-phase frame consisting of one 32-bit data word, as shown in Figure 2–13. In this case:

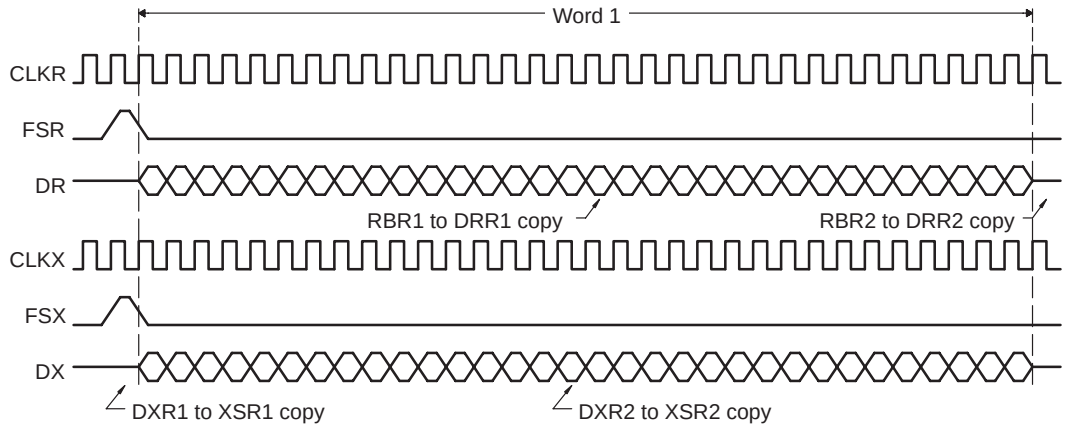
- ❑ (R/X)FRLEN1 = 0b, 1-word frame
- ❑ (R/X)PHASE = 0, single-phase frame
- ❑ (R/X)FRLEN2 = X
- ❑ (R/X)WDLEN1 = 101b, 32-bit word

In this case, two 16-bit data words are transferred to and from the McBSP by the CPU or DMA. Thus, two reads from DRR2 and DRR1 and two writes to DXR2 and DXR1 are necessary for each frame. This results in only one-half the number of transfers compared to the previous case. This manipulation reduces the percentage of bus time required for serial port data movement.

Note:

In this case, D(X/R)R2 must be written or read before D(X/R)R1.

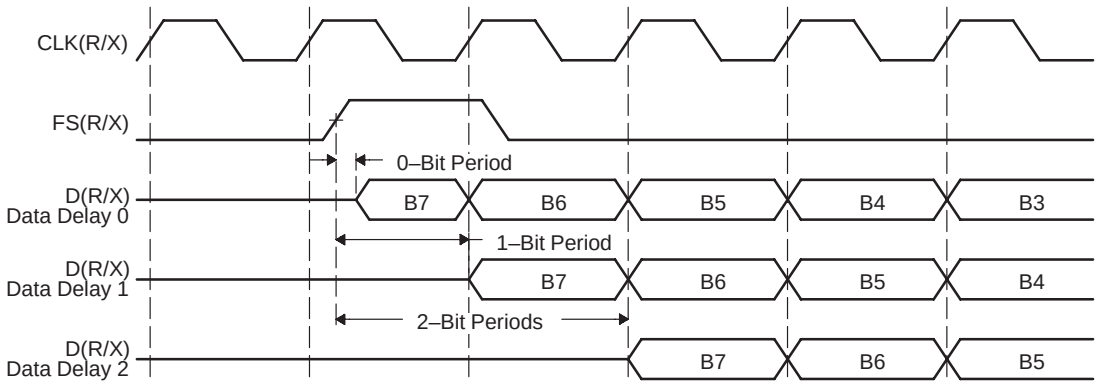
Figure 2–13. Single-Phase Frame of One 32-Bit Word



2.3.4.6 Data Delay: (R/X)DATDLY

The start of a frame is defined by the first clock cycle in which frame synchronization is found to be active. The beginning of actual data reception or transmission with respect to the start of the frame can be delayed if required. This delay is called data delay. RDATDLY and XDATDLY specify the data delay for reception and transmission, respectively. The range of programmable data delay is zero to two bit-clocks ([R/X]DATDLY = 00b –10b), as described in Table 2–7, *Receive Control Register 1 (RCR1) Bit-Field Description*, on page 2-16, and Table 2–8, *Receive Control Register 2 (RCR2) Bit-Field Description*, on page 2-17, and shown in Figure 2–14, *Data Delay*. Typically a 1-bit delay is selected, since data often follows a one-cycle active frame-sync pulse.

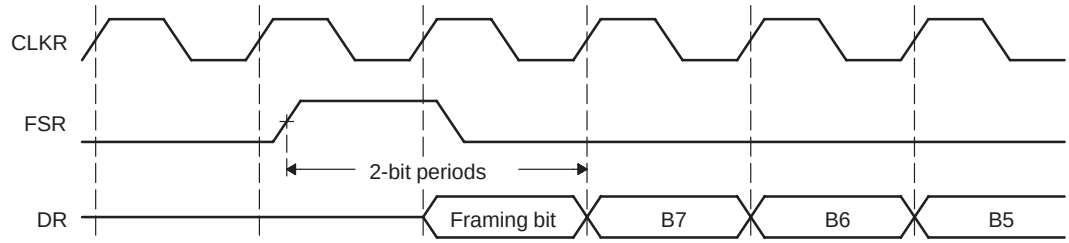
Figure 2–14. Data Delay



Normally, frame-sync pulse is detected or sampled with respect to an edge of serial clock internal CLK(R/X) (see section 2.3.4.1, *Frame and Clock Operation*, on page 2-28). Thus, on the following cycle or later (depending on data delay value), data may be received or transmitted. However, in the case of zero-bit data delay, the data must be ready for reception and/or transmission on the same serial clock cycle. For reception, this problem is solved, since receive data is sampled on the first falling edge of CLKR where an active-high internal FSR is detected. However, data transmission must begin on the rising edge of the internal CLKX clock that generated the frame synchronization. Therefore, the first data bit is assumed to be present in XSR1, and thus DX. The transmitter then asynchronously detects the frame synchronization, FSX, going active high, and immediately starts driving the first bit to be transmitted on the DX pin.

Another common mode is a data delay of two. This configuration allows the serial port to interface to different types of T1 framing devices where the data stream is preceded by a framing bit. During reception of such a stream with data delay of two bits (framing bit appears after one-bit delay and data appears after 2-bit delay), the serial port essentially discards the framing bit from the data stream as shown in Figure 2–15. In transmission, by delaying the first transfer bit, the serial port essentially inserts a blank period (high-impedance period) in place of the framing bit. Here, it is expected that the framing device inserts its own framing bit or that the framing bit is generated by another device. Alternatively, you can pull up or pull down DX to achieve the desired value.

Figure 2–15. Two-Bit Data Delay Used to Discard Framing Bit



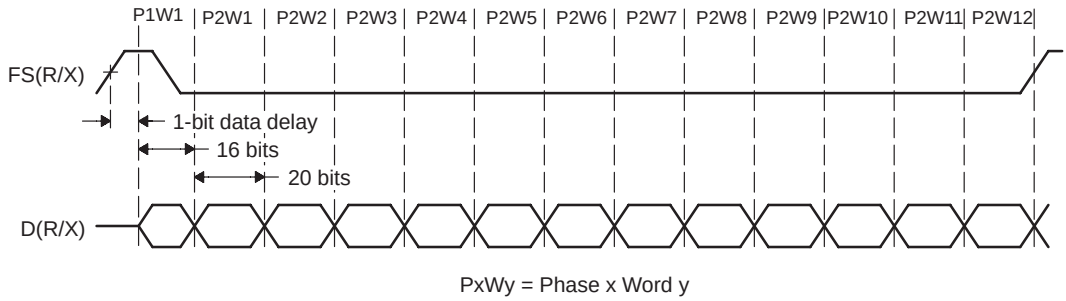
2.3.4.7 Multi-Phase Frame Example: AC97

Figure 2–16 shows an example of the Audio Codec ‘97 (AC97) standard which uses the dual-phase frame feature. The first phase consists of a single 16-bit word. The second phase consists of twelve 20-bit words. The phases are configured as follows:

- ❏ (R/X)PHASE = 1b, dual-phase frame
- ❏ (R/X)FRLEN1 = 0b, 1 word per frame in phase 1

- ❏ (R/X)WDLEN1 = 010b, 16 bits per word in phase 1
- ❏ (R/X)FRLEN2 = 0001011b, 12 words per frame in phase 2
- ❏ (R/X)WDLEN2 = 011b, 20 bits per word in phase 2
- ❏ CLK(R/X)P = 0, receive data sampled on falling edge of internal CLKR; transmit data clocked on rising edge of internal CLKX.
- ❏ FS(R/X)P = 0, active-high frame-sync signals
- ❏ (R/X)DATDLY = 01b, data delay of one bit-clock

Figure 2–16. AC97 Dual-Phase Frame Format

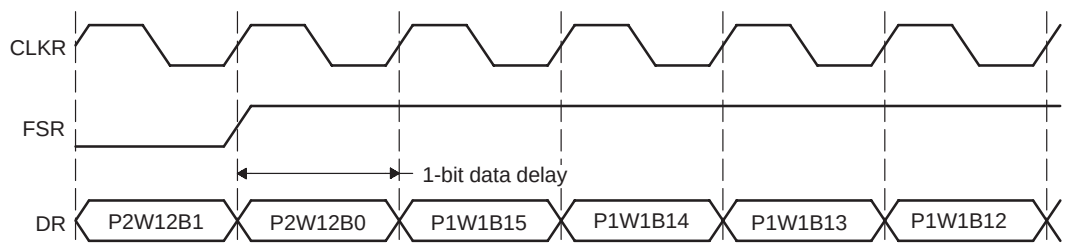


PxWy = Phase x Word y

Figure 2–16 shows the timing of AC97 near frame synchronization. First, notice that the frame-sync pulse itself overlaps the first word. In McBSP operation, the inactive to active transition of the frame-synchronization signal actually indicates frame synchronization. For this reason, frame synchronization may be high for an arbitrary number of bit-clocks. Only after the frame synchronization is recognized to have gone inactive, and then active again, is the next frame synchronization recognized.

Also, notice that there is a one-bit data delay in Figure 2–17. Regardless of the data delay, transmission can occur without gaps. The last bit of the previous (last) word in phase 2 is immediately followed by the first data bit of the first word in phase 1 of the next data frame.

Figure 2–17. AC97 Bit Timing Near Frame-Synchronization Example

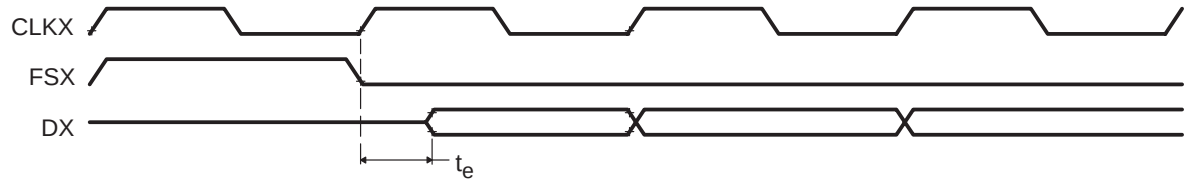


PxWyBz = Phase x Word y Bit z

2.3.4.8 Delay Enable/Disable on the DX Pin

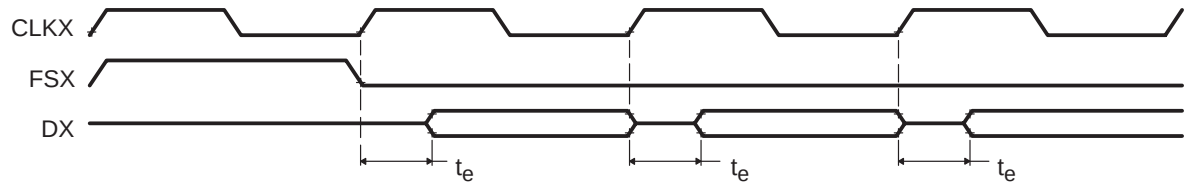
Figure 2–18 and Figure 2–19 show the timing of the DX pin when DXENA bit is set to 1 to enable extra delay for turn-on time. This bit controls the high-impedance (hi-Z) enable on the DX pin, not the data itself; so only the first bit will be delayed in the normal mode. In the A-bis mode, any bit can be delayed since any bit can go from hi-Z to valid. This bit should be set to avoid conflict when tying the DX pins together.

Figure 2–18. DX Enabler in Normal Mode



Note: t_e = extra delay for turn on time with DXENA = 1.

Figure 2–19. DX Enabler in A-bis mode



Note: t_e = extra delay for turn on time with DXENA = 1.

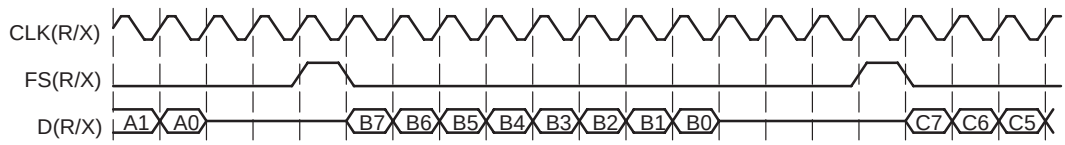
2.3.5 McBSP Standard Operation

During a serial transfer, there are typically periods of serial port inactivity between packets or transfers. The receive and transmit frame-synchronization pulse occurs for every serial transfer. When the McBSP is not in reset state and has been configured for the desired operation, a serial transfer can be initiated by programming (R/X)PHASE = 0, for a single-phase frame with required number of words programmed in (R/X)FRLEN1. The number of words can range from 1 to 128 ([R/X]FRLEN1 = 0x0 to 0x7F). The required serial word length is set in the (R/X)WDLEN1 field in (R/X)CR1. If dual-phase is required for the transfer, RPHASE = 1, (R/X)FRLEN[1,2] can be set to any value between 0x0 to 0x7F, which represents 1 to 128 words.

Figure 2–20 shows an example of a single-phase data frame comprising one 8-bit word. Since the transfer is configured for one data bit delay, the data on the DX and DR pins are available one bit-clock after FS(R/X) goes active. This figure, as well as all others in this section, make the following assumptions:

- ❏ (R/X)FRLEN1 = 0b, 1 word per frame
- ❏ (R/X)PHASE = 0, single-phase frame
- ❏ (R/X)FRLEN2 = X, (R/X)WDLEN2 = X, don't care
- ❏ (R/X)WDLEN1 = 000b, 8-bit word
- ❏ CLK(X/R)P = 0, receive data clocked on falling edge; transmit data clocked on rising edge
- ❏ FS(R/X)P = 0, active-high frame-sync signals
- ❏ (R/X)DATDLY = 01b, one-bit data delay

Figure 2–20. McBSP Standard Operation

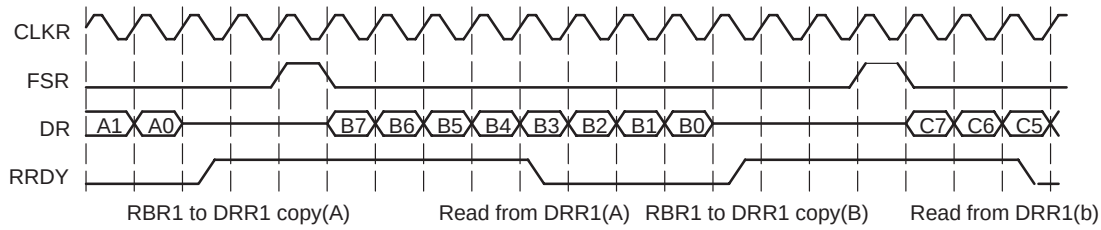


2.3.5.1 Receive Operation

Figure 2–21 shows an example of serial reception. Once receive frame-synchronization (FSR) transitions to its active state, it is detected on the first falling edge of CLKR of the receiver. The data on the DR pin is then shifted into the receive shift register (RSR[1,2]) after the appropriate data delay as set by RDATDLY. The contents of RSR[1,2] is copied to RBR[1,2] at the end of every

word on the rising edge of clock, provided RBR[1,2] is not full with the previous data. Then, an RBR[1,2]-to-DRR[1,2] copy activates the RRDY status bit to 1 on the following falling edge of CLKR. This indicates that the receive data register (DRR[1,2]) is ready with the data to be read by the CPU or DMA. RRDY is deactivated when DRR[1,2] is read by the CPU or DMA.

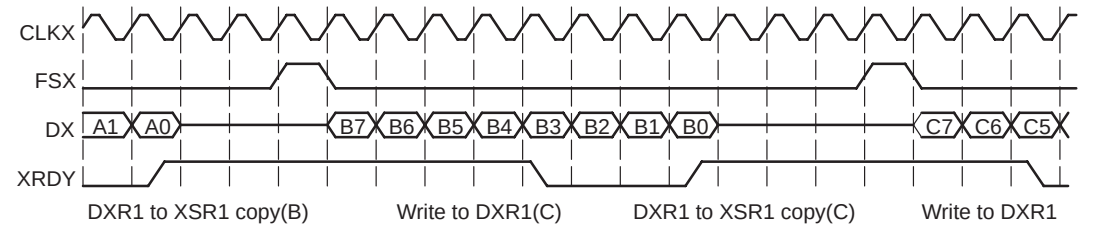
Figure 2–21. Receive Operation



2.3.5.2 Transmit Operation

Once transmit frame synchronization occurs, the value in the transmit shift register, XSR[1,2], is shifted out and driven on the DX pin after the appropriate data delay as set by XDATDLY. XRDY is activated on every DXR[1,2]-to-XSR[1,2] copy on the following falling edge of CLKX, indicating that the data transmit register (DXR[1,2]) is written with the next data to be transmitted. XRDY is deactivated when DXR[1,2] is written by the CPU or DMA. Figure 2–22 shows an example of a serial transmission. See section 2.3.7.4, *Transmit Empty: $\overline{\text{XEMPTY}}$*, on page 2-48, for transmit operation when transmitter is pulled out of reset ($\overline{\text{XRST}}$ = 1).

Figure 2–22. Transmit Operation



2.3.5.3 Maximum Frame Frequency

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

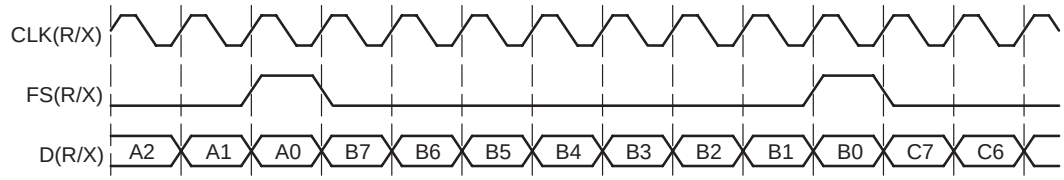
$$\textit{Frame Frequency} = \frac{\textit{Bit–Clock Frequency}}{\textit{Number of Bit–Clocks Between Frame Sync Signals}}$$

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

$$\textit{Maximum Frame Frequency} = \frac{\textit{Bit–Clock Frequency}}{\textit{Number of Bits Per Frame}}$$

Figure 2–23 shows the McBSP operating at maximum packet frequency. At maximum packet frequency, the data bits in consecutive packets are transmitted contiguously with no inactivity between bits. If there is a one-bit data delay as shown, the frame-synchronization pulse overlaps the last bit transmitted in the previous frame.

Figure 2–23. Maximum Frame Frequency Receive/Transmit (R/X)DATDLY = 0)



Effectively, this permits a continuous stream of data; and thus, the frame-synchronization pulses are essentially redundant. Theoretically, only an initial frame-synchronization pulse is required to initiate a multipacket transfer. The McBSP supports operation of the serial port in this fashion by ignoring the successive frame-sync pulses. Data is clocked in to the receiver, or clocked out of the transmitter, on every clock†. The frame ignore bit, (R/X)FIG, in (R/X)CR can be programmed to ignore the successive frame-sync pulses until the desired frame length or number of words is reached. This is explained in section 2.3.6.1, *Data Packing using Frame-Sync Ignore Bits*, on page 2-41 .

2.3.6 Frame-Synchronization Ignore

The McBSP can be configured to ignore transmit and receive frame-synchronization pulses. The (R/X)FIG bit in (R/X)CR2 can be programmed to zero to recognize frame-sync pulses, or set to one to ignore frame-sync pulses. The user can use (R/X)FIG bit to either pack data or ignore unexpected frame-sync pulses. Section 2.3.6.1 explains data packing and McBSP operation on unexpected frame-sync pulses.

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

2.3.6.1 Data Packing Using Frame-Sync Ignore Bits

Section 2.3.4.5, *Data Packing using Frame Length and Word Length,* on page 2-32, describes one method of changing the word length and frame length to simulate 32-bit serial word transfers, thus requiring much less bus bandwidth. This example works when there are multiple words per frame. Now consider the case of the McBSP operating at maximum packet frequency as shown in Figure 2–24. Here, each frame only has a single 8-bit word. This stream takes one read and one write transfer for each 8-bit word. Figure 2–25 shows the McBSP configured to treat this stream as a continuous 32-bit word. In this example, (R/X)FIG is set to 1 to ignore subsequent frames after the first. Only two read- or two write-transfers are needed every 32 bits. This configuration effectively reduces the required bus bandwidth to one-half of the bandwidth needed to transfer four 8-bit words.

Figure 2–24. Maximum Packet Frequency Operation With 8-bit Data

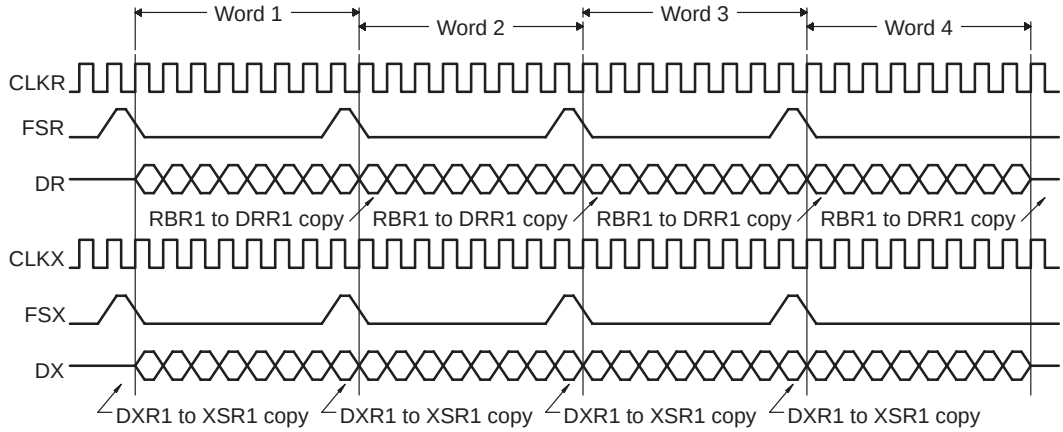
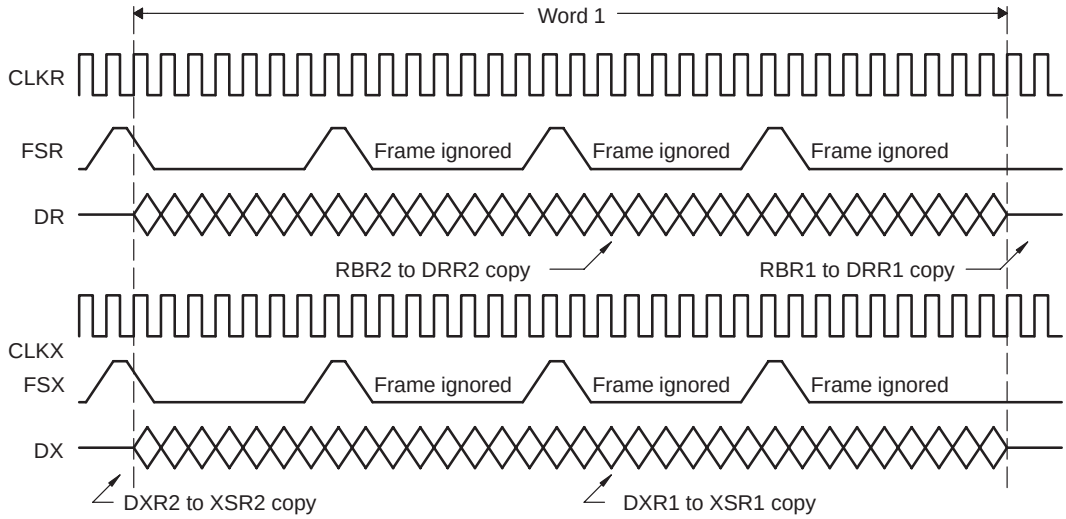


Figure 2–25. Data Packing at Maximum Packet Frequency With (R/X)FIG=1



2.3.6.2 Frame-Sync Ignore and Unexpected Frame-Sync Pulses

The previous section explained how frame ignore bits can be used to pack data and efficiently use the bus bandwidth. (R/X)FIG bit can also be used to ignore unexpected frame-sync pulses. Thus, any frame-sync pulse that occurs one bit-clock earlier than the programmed data delay ([R/X]DATDLY) is considered unexpected. Setting the frame ignore bits to one causes the serial port to ignore these unexpected frame-sync signals.

In reception, if not ignored (RFIG = 0), an unexpected FSR pulse will discard the contents of RSR[1,2] in favor of the new incoming data. Therefore, if RFIG = 0, an unexpected frame-synchronization pulse aborts the current data transfer, sets RSYNCERR in SPCR1 to 1, and begins the transfer of a new data word. For further details, see section 2.3.7.2, *Unexpected Receive Frame Synchronization: RSYNCERR*, on page 2-46. When RFIG = 1, reception continues, ignoring the unexpected frame-sync pulses.

In transmission (if not ignored [XFIG = 0]), an unexpected FSX pulse aborts the present transmission, sets XSYNCERR to 1 in SPCR2 , and re-initiates transmission of the current word that was aborted. For further details, see section 2.3.7.5, *Unexpected Transmit Frame Synchronization: XSYNCERR*, on page 2-50. When XFIG = 1, normal transmission continues with unexpected frame-sync signals ignored.

Figure 2–26. Unexpected Frame Synchronization With (R/X)FIG=0

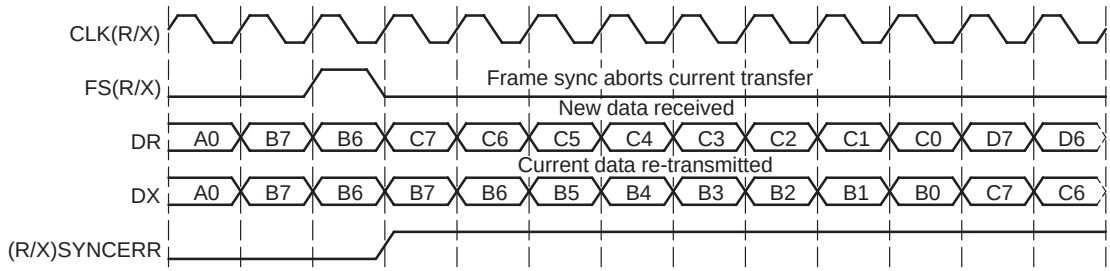
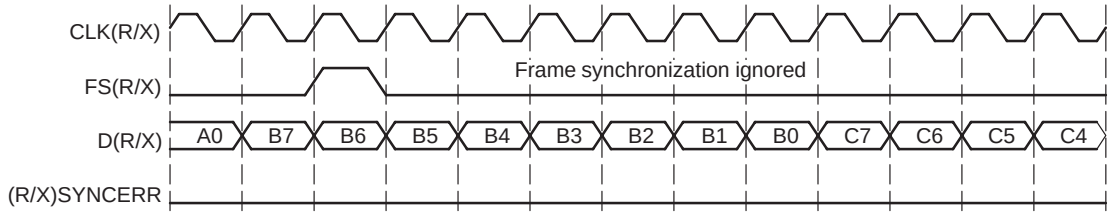


Figure 2–26 shows an example wherein word B is interrupted by an unexpected frame-sync pulse when (R/X)FIG = 0. In the case of reception, the reception of B is aborted (B is lost), and a new data word (C in this example) is received after the appropriate data delay. This condition is a receive synchronization error, and thus sets the RSYNCERR bit. However, for transmission, the transmission of B is aborted, and the same data (B) is retransmitted after the appropriate data delay. This condition is a transmit synchronization error and thus sets the XSYNCERR bit. Synchronization errors are discussed in sections 2.3.7.2 and 2.3.7.5. In contrast, Figure 2–27 shows McBSP operation when unexpected frame-synchronization signals are ignored by setting (R/X)FIG = 1. Here, the transfer of word B is not affected by an unexpected frame synchronization.

Figure 2–27. Unexpected Frame Synchronization With (R/X)FIG = 1



2.3.7 Serial Port Exception Conditions

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

1) **Receive Overrun (RFULL = 1)**. This occurs when DRR[1,2] has not been read since the last RBR[1,2]-to-DRR[1,2] copy. Consequently, a new word in RBR[1,2] will not be transferred to DRR[1,2], and RSR[1,2] is now full with another new word shifted in from DR. Therefore, RFULL indicates an error condition wherein any new data that may arrive at this time on DR will replace the contents in RSR[1,2], and thus, the previous word is lost. RSR[1,2] continues to be overwritten as long as new data arrives on DR and DRR[1,2] is not read.

2) **Unexpected Receive Frame Synchronization (RSYNCERR=1)**. This can occur during reception when RFIG = 0 and an unexpected frame-sync pulse occurs. An unexpected frame-sync pulse is defined as that which occurs RDATDLY minus 1 bit-clock earlier than the first bit of the next associated word. This causes the current data reception to abort and restart. If new data has been copied into RBR[1,2] from RSR[1,2] since the last RBR[1,2]-to-DRR[1,2] copy, this new data in RBR[1,2] will be lost. This is because no RBR[1,2]-to-DRR[1,2] copy occurs as the reception has been restarted.

3) **Transmit Data Overwrite**. Here the user overwrites data in DXR[1,2] before it is copied to XSR[1,2]. The data previously in DXR[1,2] is never transferred on DX, since it never got copied to XSR[1,2].

4) **Transmit Empty ($\overline{\text{XEMPTY}}$ = 0)**. If a new frame-synchronization signal arrives before new data is loaded into DXR[1,2], the old data in DXR[1,2] will be sent again. This will continue for every new frame-sync signal that arrives on the FSX pin until DXR[1,2] is loaded with new data.

5) **Unexpected Transmit Frame Synchronization (XSYNCERR = 1)**. This can occur during transmission when XFIG = 0 and an unexpected frame-sync pulse occurs. Again, an unexpected frame-sync pulse is defined as that which occurs XDATDLY minus 1 bit-clock earlier than the first bit of the next associated word. This causes the current data transmission to abort and restart the current transfer. If new data had been written to DXR[1,2] since the last DXR[1,2]-to-XSR[1,2] copy, the current value in XSR[1,2] will be lost.

These events are described in more detail in the sections that follow.

2.3.7.1 Reception With Overrun: RFULL

RFULL = 1 in SPCR1 indicates that the receiver has experienced overrun and is in an error condition. RFULL is set when all of the following conditions are met:

1) DRR[1,2] has not been read since the last RBR[1,2]-to-DRR[1,2] transfer (RRDY = 1).

2) RBR[1,2] is full and an RBR[1,2]-to-DRR[1,2] copy has not occurred.

3) RSR[1,2] is full and an RSR[1,2]-to-RBR[1,2] transfer has not occurred.

Data arriving on DR is continuously shifted into RSR[1,2]. Once a complete word is shifted into RSR[1,2], an RSR[1,2]-to-RBR[1,2] transfer can occur only if an RBR[1,2]-to-DRR[1,2] copy is complete. Therefore, if DRR[1,2] has not been read by the CPU or DMA since the last RBR[1,2]-to-DRR[1,2] transfer (RRDY = 1), an RBR[1,2]-to-DRR[1,2] copy will not take place until RRDY = 0. At this time, new data arriving on the DR pin is shifted into RSR[1,2] and the previous contents of RSR[1,2] is lost. This data loss occurs because completion of a serial-word reception triggers an RBR[1,2]-to-DRR[1,2] transfer only when RRDY = 0. Note that after the receive portion starts running from reset, a minimum of three words must be received before RFULL is set.

The data loss of the contents in RSR[1,2] can be avoided if DRR[1,2] is read not later than two and one-half cycles before the end of the third word in RSR[1,2].

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

- ❏ Reading DRR[1,2]
- ❏ Resetting the receiver ($\overline{\text{RRST}}$ = 0) or the device

Another frame synchronization is required to restart the receiver.

Figure 2–28 shows the receive overrun condition. Because serial word A is not read before the reception of serial word B is complete, B is not transferred to DRR1 yet. Another new word C arrives and RSR1 is full with this data. DRR1 is finally read, but not earlier than two and one-half cycles before the end of word C. Therefore, new data D overwrites the previous word C in RSR1. If RFULL is still set after arrival of D, the next word can overwrite it, if DRR is not read in time.

Figure 2–28. Serial Port Receive Overrun

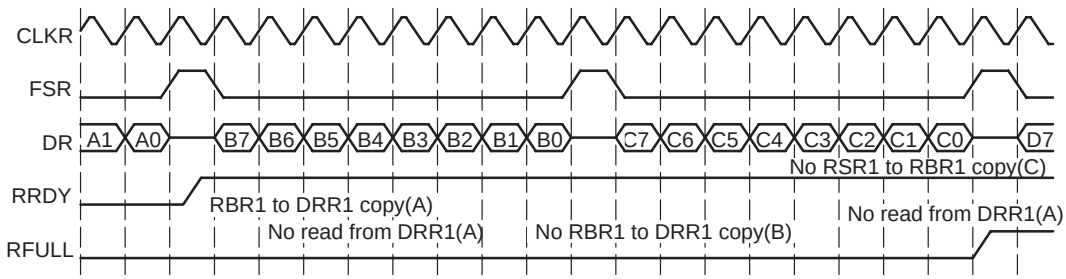
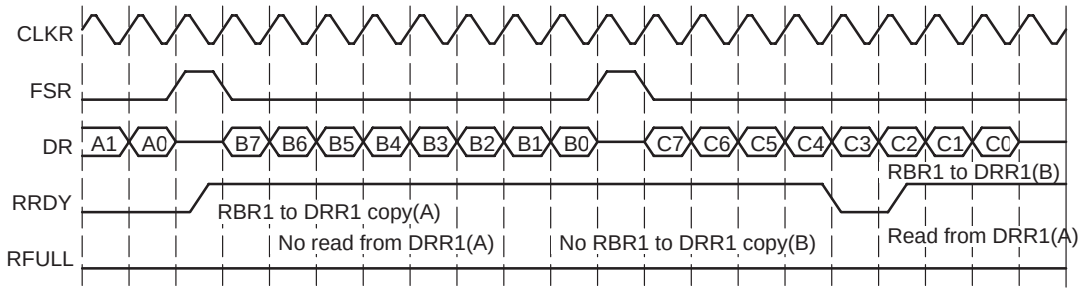


Figure 2–29 shows the case where RFULL is set, but the overrun condition is averted by reading the contents of DRR1 at least two and one-half cycles before the next serial word C is completely shifted into RSR1. This ensures that an RBR1-to-DRR1 copy of data B occurs before the next serial word (C) is transferred from RSR1 to RBR1.

Figure 2–29. Serial Port Receive Overrun Avoided



2.3.7.2 Unexpected Receive Frame Synchronization: RSYNCERR

Figure 2–30 shows the decision tree that the receiver uses to handle all incoming frame-synchronization pulses. The diagram assumes that the receiver has been started, $\overline{\text{RRST}}$ = 1. Unexpected frame-sync pulses can originate from an external source or from the internal sample rate generator. An unexpected frame-sync pulse is defined as a sync pulse which occurs RDATDY bit-clocks earlier than the last transmitted bit of the previous frame. Any one of four cases can occur:

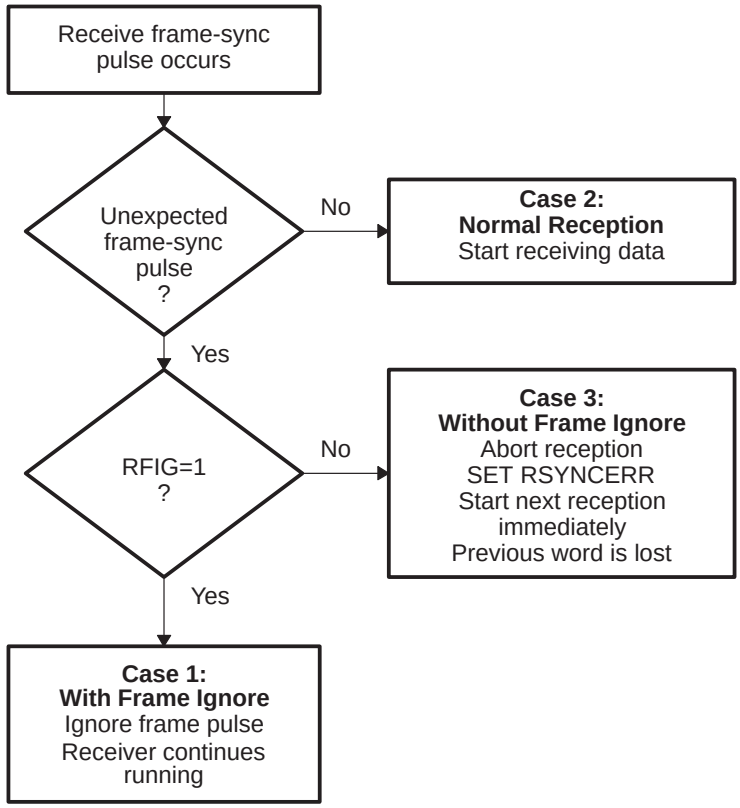
- ❏ Case 1: Unexpected internal FSR pulses with RFIG = 1. This case is discussed in section 2.3.6.2 on page 2-42 and shown in Figure 2–27 on page 2-43. In Case 1, receive frame-sync pulses are ignored and the reception continues.

- ❏ Case 2: Normal serial port reception. There are three possible reasons why a receive might NOT be in progress:
  - ■ The FSR is the first after $\overline{\text{RRST}}$ = 1.
  - ■ The FSR is the first after DRR[1,2] is read clearing an RFULL condition.
  - ■ The serial port is in the interpacket intervals. The programmed data delay (RDATDLY) for reception may start during these interpacket intervals for the first bit of the next word to be received. Thus, at maximum frame frequency, frame synchronization can still be received RDATDLY bit-clocks before the first bit of the associated word.

For Case 2, reception continues normally since these are not unexpected frame-sync pulses.

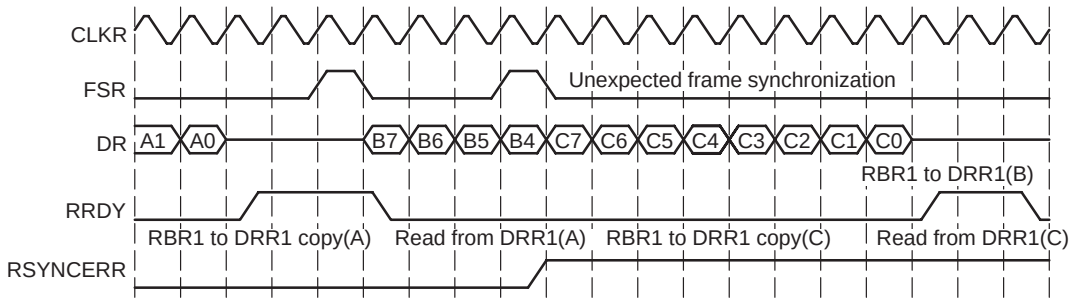
- Case 3: Unexpected receive frame synchronization with RFIG = 0 (unexpected frame not ignored). This case was shown in Figure 2–26 on page 2-43 for maximum frame frequency. Figure 2–31 on page 2-48 shows this case during normal operation of the serial port, with time intervals between packets. An unexpected frame-sync pulse is detected when it occurs at or before RDATDLY minus 1 bit-clock before the last bit of the previous word is received on the DR pin. In both cases, the RSYNCERR bit in SPCR1 is set. RSYNCERR† can be cleared only by a receiver reset or by the user writing a 0 to this bit in SPCR1. You should note that if RINTM = 11b in SPCR1, RSYNCERR drives the receive interrupt (RINT) to the CPU.

Figure 2–30. Response to Receive Frame-Synchronization Pulse



† The RSYNCERR bit in SPCR1 is a read/write bit. Therefore, writing a 1 to it sets the error condition. Typically, writing a 0 is expected.

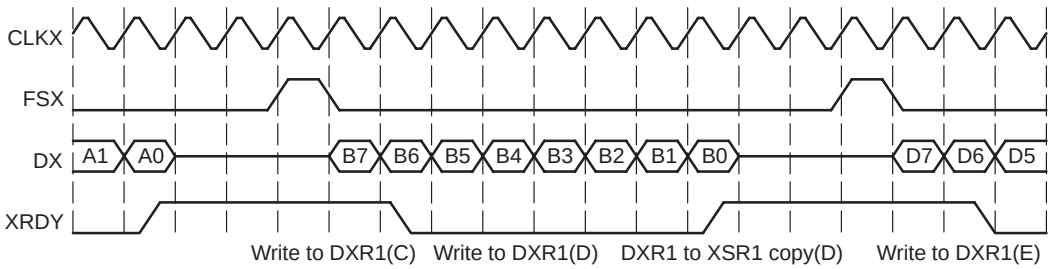
Figure 2–31. Unexpected Receive Synchronization Pulse



2.3.7.3 Transmit with Data Overwrite

Figure 2–32 depicts what happens if the data in DXR1 is overwritten before being transmitted. Initially, the programmer loaded DXR1 with data C. A subsequent write to DXR1 overwrites C with D before it is copied to XSR1. Thus, C is never transmitted on DX. The CPU can avoid data overwrite by polling XRDY before writing to DXR1 or by waiting for an XINT programmed to be triggered by XRDY (XINTM = 00b). The DMA can avoid overwriting by synchronizing data transfers with XEVT.

Figure 2–32. Transmit With Data Overwrite



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

$\overline{\text{XEMPTY}}$ indicates when the transmitter has experienced underflow. Any of the following conditions causes $\overline{\text{XEMPTY}}$ to become active ($\overline{\text{XEMPTY}}$ = 0):

1) During transmission. DXR[1,2] has not been loaded since the last DXR[1,2]-to-XSR[1,2] copy, and all bits of the data word in XSR[1,2] have been shifted out on DX.

2) The transmitter is reset ($\overline{\text{XRST}}$ = 0, or device is reset) and then restarted.

During an underflow condition, the transmitter continues to transmit the old data in DXR[1,2] for every new frame-sync signal that arrives on FSX until a new word is loaded into DXR[1,2] by the CPU or DMA. $\overline{\text{XEMPTY}}$ is deactivated

($\overline{\text{XEMPTY}}$ = 1) when the new word in DXR[1,2] is transferred to XSR[1,2]. In the case of an internal frame generation, the transmitter regenerates a single internal FSX initiated by a DXR[1,2]-to-XSR[1,2] copy (FSXM = 1 in the PCR and FSGM=0 in SRGR2). Otherwise, the transmitter waits for the next frame synchronization.

When the transmitter is taken out of reset ($\overline{\text{XRST}}$ = 1), it is in a transmit ready (XRDY=1) and transmit empty ($\overline{\text{XEMPTY}}$ = 0) condition. If DXR[1,2] is loaded by the CPU or DMA before internal FSX goes active high, a valid DXR[1,2]-to-XSR[1,2] transfer occurs. This allows for the first word of the first frame to be valid even before the transmit frame-sync pulse is generated or detected. Alternatively, if a transmit frame sync is detected before DXR is loaded, zeros will be output on DX.

Figure 2–33 depicts a transmit underflow condition. After B is transmitted, the programmer fails to reload DXR[1,2] before the subsequent frame synchronization. Thus, B is again transmitted on DX. Figure 2–34 shows the case of writing to DXR1 just before a transmit underflow condition that would otherwise occur. After B is transmitted, C is written to DXR1 before the next transmit frame-sync pulse occurs so that C is successfully transmitted on DX, thus averting a transmit empty condition.

Figure 2–33. Transmit Empty

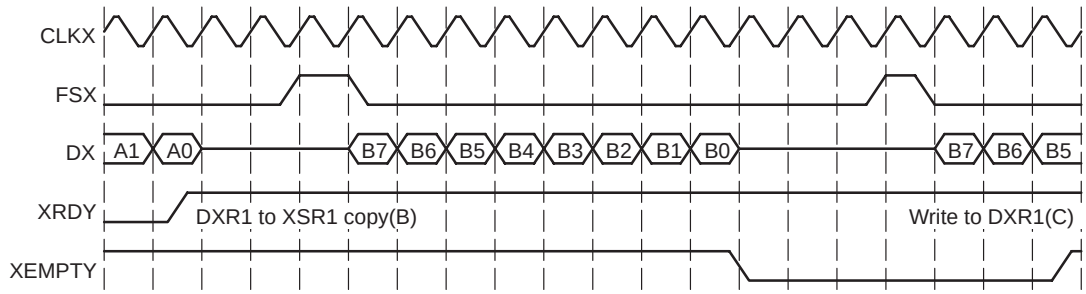
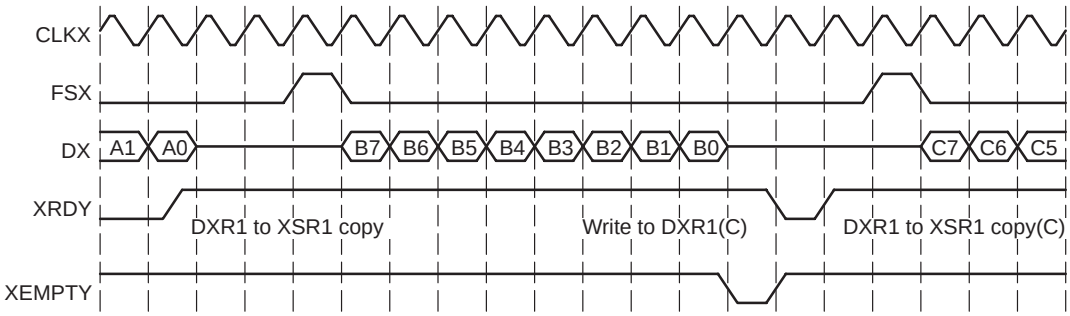


Figure 2–34. Transmit Empty Avoided

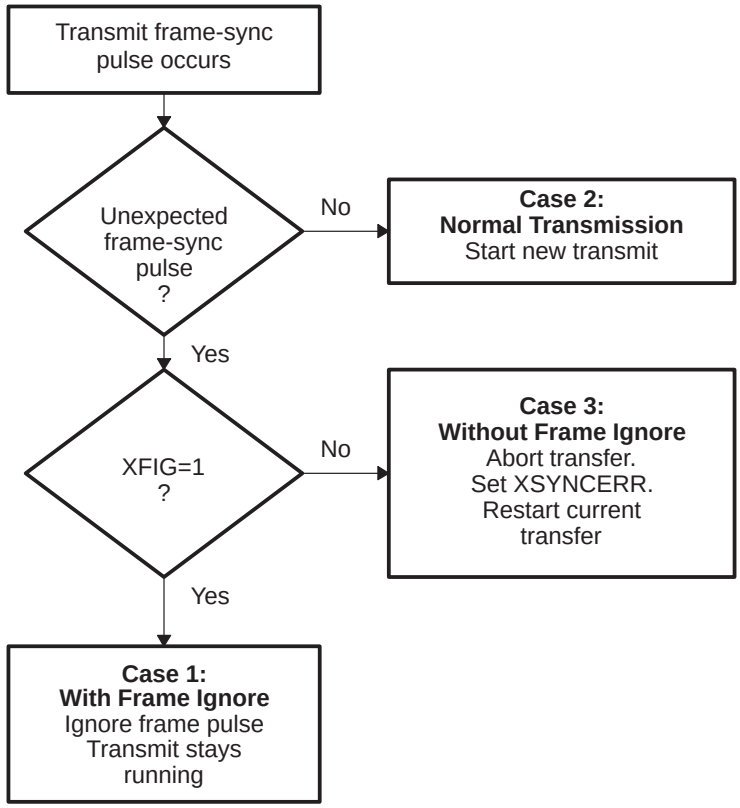


2.3.7.5 Unexpected Transmit Frame Synchronization: XSYNCERR

Figure 2–30 on page 2-47 shows the decision tree that the transmitter uses to handle all incoming frame-synchronization signals. The diagram assumes that the transmitter has been started ($\overline{\text{XRST}}$ = 1). An unexpected transmit frame-sync pulse is defined as a sync pulse which occurs XDATDLY bit-clocks earlier than the last transmit bit of the previous frame. Any one of three cases can occur:

- Case 1: Unexpected FSX pulses with XFIG = 1. This case is discussed in section 2.3.6.2 on page 2-42 and shown in Figure 2–27 on page 2-43.
- Case 2: Normal serial port transmission. This case is discussed in section 2.3.5.2 on page 2-39. You should note that there are two possible reasons why a transmit might NOT be in progress:
  - This FSX pulse is the first after $\overline{\text{XRST}}$ = 1.
  - The serial port is in the interpacket intervals. The programmed data delay (XDATDLY) may start during these interpacket intervals before the first bit of the previous word is transmitted. Thus, if operating at maximum packet frequency, frame synchronization can still be received XDATDLY bit-clocks before the first bit of the associated word.

Figure 2–35. Response to Transmit Frame Synchronization

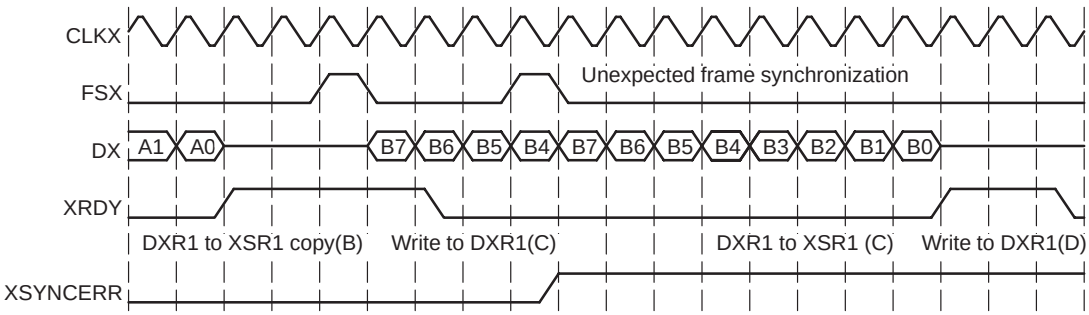


- Case 3: Unexpected transmit frame synchronization with XFIG = 0. The case for subsequent frame synchronization with XFIG = 0 at maximum frame frequency is shown in Figure 2–26. Figure 2–36 shows the case for normal operation of the serial port with interpacket intervals. In both cases, the XSYNCERR bit in SPCR2 is set. XSYNCERR can only be cleared by transmitter reset or by the user writing a 0 to this bit in SPCR2. Note that if XINTM=11b in SPCR2, XSYNCERR drives the receive interrupt (XINT) to the CPU.

Note:

The XSYNCERR bit in SPCR2 is a read/write bit. Therefore, writing a 1 to it sets the error condition. Typically, writing a 0 is expected.

Figure 2–36. Unexpected Transmit Frame-Synchronization Pulse



2.3.8 Receive Data Justification and Sign-Extension: RJUST

RJUST in SPCR1 selects whether data in RBR[1,2] is right or left justified (with respect to the MSB) in DRR[1,2]. If right-justification is selected, RJUST further selects whether the data is sign-extended or zero-filled. Table 2–15 shows the effect various modes of RJUST have on an example 12-bit receive-data value 0xABC. Table 2–16 shows the effect various modes of RJUST have on an example 20-bit receive-data value 0xABCDE.

Table 2–15. Use of RJUST Field With 12-Bit Example Data 0xABC

RJUST	Justification	Extension	Value in DRR2	Value in DRR1
00	Right	Zero-fill MSBs	0x0000	0x0ABC
01	Right	Sign-extend MSBs	0xFFFF	0xFABC
10	Left	Zero-fill LSBs	0x0000	0xABC0
11	Reserved	Reserved	Reserved	Reserved

Table 2–16. Use of RJUST Field With 20-Bit Example Data 0xABCDE

RJUST	Justification	Extension	Value in DRR2	Value in DRR1
00	Right	Zero-fill MSBs	0x000A	0xBCDE
01	Right	Sign-extend MSBs	0xFFFA	0xBCDE
10	Left	Zero-fill LSBs	0xABCD	0xE000
11	Reserved	Reserved	Reserved	Reserved

2.4 μ-LAW/A-LAW Companding Hardware Operation: (R/X)COMPAND

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

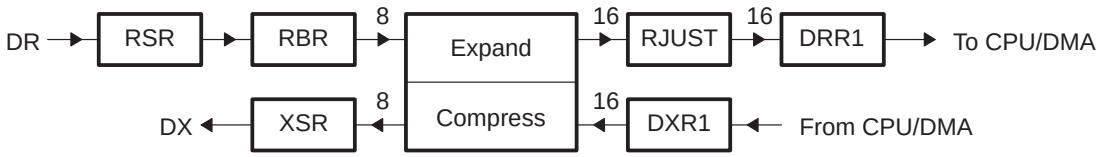
The μ-law and A-law formats encode data into 8-bit code words. Companded data is always 8-bits wide; therefore, the appropriate (R/X)WDLEN[1,2] must be set to 0, indicating 8-bit wide serial data stream. If companding is enabled and either phase of the frame does not have 8-bit word length, then companding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

When companding is used, transmit data is encoded according to specified companding law, and receive data is decoded to 2's complement format. Companding is enabled and the desired format selected by appropriately setting (R/X)COMPAND in (R/X)CR2 as shown in:

- Table 2–7, *Receive Control Register 1 (RCR1) Bit-Field Description*, on page 2-16
- Table 2–8, *Receive Control Register 2 (RCR2) Bit-Field Description*, on page 2-17
- Table 2–9, *Transmit Control Register 1 (XCR1) Bit-Field Description*, on page 2-19
- Table 2–10, *Transmit Control Register 2 (XCR2) Bit-Field Description*, on page 2-20

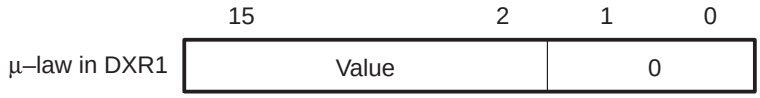
Compression occurs during the process of copying data from DXR1-to-XSR1 and from RBR1-to-DRR1, as shown in Figure 2–37.

Figure 2–37. Companding Flow



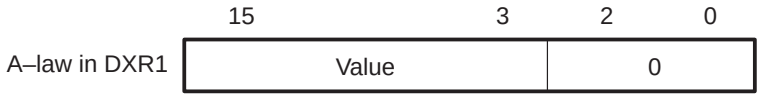
For transmit data to be compressed properly, the data should be left-justified when it is written to DXR1. When using μ-law, the 14 data bits are left-justified in the register, with the remaining two low-order bits filled with zeros as shown in Figure 2–38.

Figure 2–38. μ-Law Transmit Data Companding Format



When using A-law, the 13 data bits are left-justified in the register, with the remaining three low-order bits filled with zeros as shown in Figure 2–39.

Figure 2–39. A-Law Transmit Data Companding Format



For reception, the 8-bit compressed data in RBR1 is expanded to left-justified 16-bit data in DRR1. Note that RJUST is ignored when companding is used.

2.4.1 Companding Internal Data

If the McBSP is otherwise unused (serial port X/R sections are reset), the companding hardware can compand internal data. This can be used to:

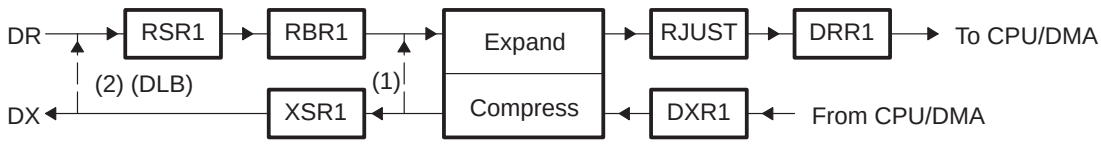
- ❏ Convert linear to the appropriate μ-law or A-law format.
- ❏ Convert μ-law or A-law to the linear format.
- ❏ Observe the quantization effects in companding by transmitting linear data, and compressing and re-expanding this data. This is only useful if both XCOMPAND and RCOMPAND enable the same companding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

Figure 2–40 shows two methods by which the McBSP can compand internal data. Data paths for these two methods are used to indicate:

1) When both the transmit and receive sections of the serial port are reset, DRR1 and DXR1 are internally connected through the companding logic. Values from DXR1 are compressed, as selected by XCOMPAND, and then expanded, as selected by RCOMPAND. Note that RRDY and XRDY bits are not set. However, data is available in DRR1 within four CPU clocks after being written to DXR1. The advantage of this method is its speed. The disadvantage is that there is no synchronization available to the CPU and DMA to control the flow. Note that DRR1 and DXR1 are internally connected if the (X/R)COMPAND bits are set to 1xb, i.e., compand using A-law or μ-law.

2) The McBSP is enabled in digital loop back mode with companding appropriately enabled by RCOMPAND and XCOMPAND. Receive and transmit interrupts (RINT when RINTM = 0 and XINT when XINTM = 0) or synchronization events (REVT and XEVT) allow synchronization of the CPU or DMA to these conversions, respectively. Here, the time for this companding depends on the serial bit rate selected.

Figure 2–40. Companding of Internal Data



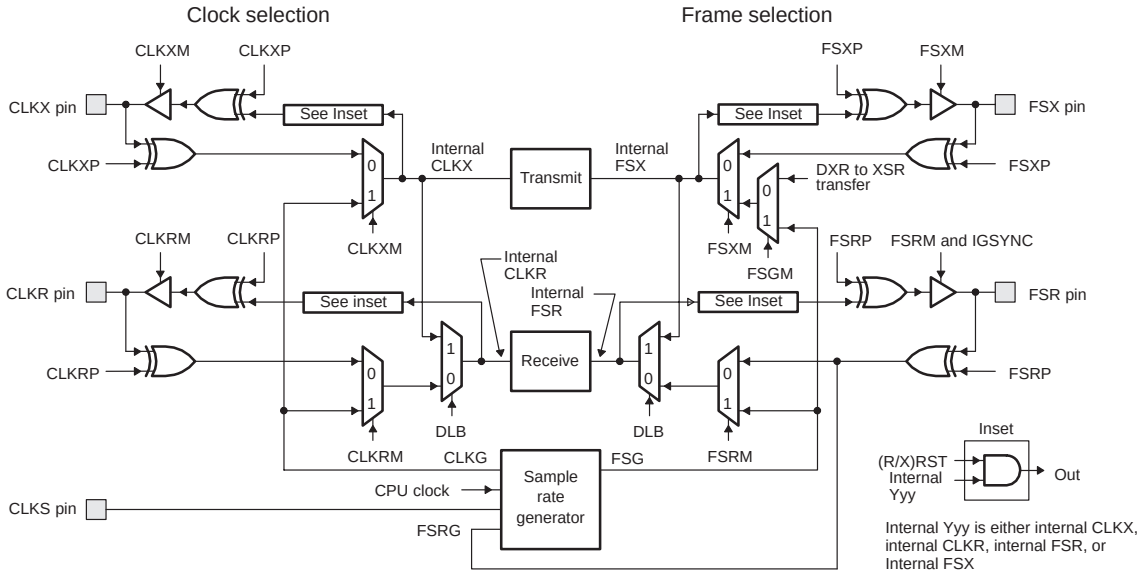
2.4.1.1 Bit Ordering

Normally, all transfers on the McBSP are sent and received with the MSB first. However, certain 8-bit data protocols (that do not use companded data) require the LSB to be transferred first. By setting (R/X)COMPAND = 01b in (R/X)CR2, the bit ordering of 8-bit words is reversed (LSB first) before being sent to the serial port. Similar to companding, this feature is only enabled if the appropriate (R/X)WDLEN[1,2] is set to 0, indicating 8-bit words are to be transferred serially. If either phase of the frame does not have an 8-bit word length, the McBSP assumes the word length is eight bits, and LSB-first ordering is done.

2.5 Programmable Clock and Framing

The McBSP has several means of selecting clocking and framing for both the receiver and transmitter. Clocking and framing can be sent to both portions by the sample rate generator. Each portion can select external clocking and/or framing independently. Figure 2–41 shows a block diagram of the clock and frame selection circuitry. The features that are enabled by this logic are explained in the sections that follow.

Figure 2–41. Clock and Frame Generation



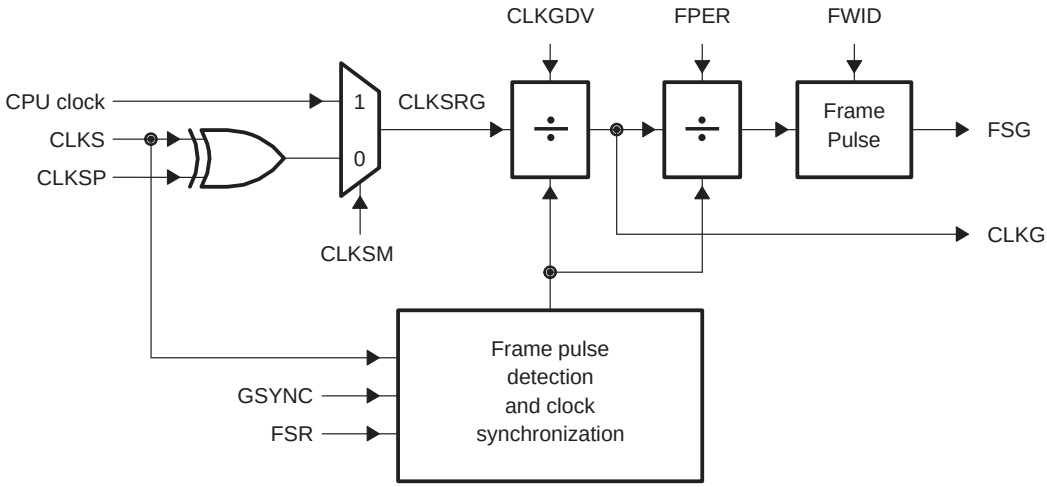
2.5.1 Sample Rate Generator Clocking and Framing

The sample rate generator is composed of a three-stage clock divider that allows programmable data clocks (CLKG) and framing signals (FSG) as shown in Figure 2–42. CLKG and FSG are McBSP internal signals that can be programmed to drive receive and/or transmit clocking (CLKR/X) and framing (FSR/X). The sample rate generator can be programmed to be driven by an internal clock source or an internal clock derived from an external clock source. The three stages of the sample rate generator circuit compute the following:

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

In addition, a frame pulse detection and clock synchronization module allows synchronization of the clock divide down with an incoming frame pulse. The operation of the sample rate generator during device reset is described in section 2.3.1, *Resetting the Serial Port: (R/X)$\overline{RST}$, and $\overline{RESET}$*, on page 2-22.

Figure 2–42. Sample Rate Generator



2.5.1.1 Sample Rate Generator Register (SRGR [1,2])

Sample rate generator registers 1 and 2 (SRGR[1,2]) control the operation of various features of the sample rate generator. These registers and their bit-field descriptions are shown in Figure 2–43 and Table 2–17, and Figure 2–44 and Table 2–18, respectively. The sections that follow these figures and tables describe how you can configure the operation of SRGR using the SRGR[1,2] bit-fields.

Figure 2–43. Sample Rate Generator Register 1 (SRGR1)

15	8	7	0
FWID		CLKGDV	
RW,+0		RW	

Note: R = Read, W = Write, +0 = Value at reset

Table 2–17. Sample Rate Generator Register 1 (SRGR1) Bit-Field Descriptions

Bit	Name	Function	Section
15–8	FWID	Frame Width. This field plus 1 determines the width of the frame-sync pulse, FSG, during its active period. Range: up to 2; 1 to 256 CLKG periods.	2.5.3.1
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.	2.5.2.2

Figure 2–44. Sample Rate Generator Register 2 (SRGR2)

15	14	13	12	11	0
GSYNC	CLKSP	CLKSM	FSGM	FPER	
RW,+0	RW,+0	RW	RW,+0	RW,+0	

Note: R = Read, W = Write, +0 = Value at reset

Table 2–18. Sample Rate Generator Register 2 (SRGR2) Bit-Field Descriptions

Bit	Name	Function	Section
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.	2.5.2.4
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.	2.5.2.3
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.	2.5.2.2

Table 2–18. Sample Rate Generator Register 2 (SRGR2) Bit-Field Descriptions (Continued)

Bit	Name	Function	Section
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.	2.5.3.3
11–0	FPER	Frame Period. This field plus 1 determines when the next frame-sync signal becomes active. Range: 1 to 4096 CLKG periods.	2.5.3.1

2.5.1.2 Sample Rate Generator Reset Procedure

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

1) During device reset, $\overline{\text{GRST}}$ = 0. Otherwise, during normal operation, the sample rate generator can be reset with $\overline{\text{GRST}}$ = 0 in SPCR2, provided CLKG and/or FSG is not used by any portion of the McBSP. If $\overline{\text{GRST}}$ = 0 due to device reset, CLKG is driven by the divide-by-2 CPU clock, and FSG is driven inactive-low. If $\overline{\text{GRST}}$ = 0 as programmed by the user, CLKG and FSG are driven inactive-low. If necessary, set (R/X)$\overline{\text{RST}}$ = 0.
2) Program SRGR[1,2] as required. If necessary, other control registers can be written with desired values, provided the respective portion (R/X) is in reset.
3) Wait two CLKSRG clocks. This ensures proper synchronization internally.
4) Set $\overline{\text{GRST}}$ = 1 to enable the sample rate generator.
5) Wait two CLKG bit-clocks.
6) Pull the receiver and/or transmitter out of reset ([R/X]$\overline{\text{RST}}$ = 1), if required.
7) On the next rising edge of CLKSRG, CLKG transitions to 1 and starts clocking with a frequency equal to (CPU clock/(1+CLKGDV), if CLKSM = 1, or CLKS clock/(1+CLKGDV) if CLKSM = 0).
8) After the required data acquisition set up is done (DXR[1/2]) is loaded with data), $\overline{\text{FRST}}$ can be written with 1 if internally generated frame-sync pulse is required. FSG is generated with an active-high edge after the programmed number of eight CLKG clocks have elapsed.

2.5.2 Data Clock Generation

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

- ❏ The input clock to the sample rate generator can be either the CPU clock or a dedicated external clock input (CLKS).
- ❏ The input clock (CPU clock or external clock CLKS) source to the sample rate generator can be divided down by a programmable value (CLKGDV) to drive CLKG. Regardless of the source to the sample rate generator, the rising edge of CLKSRG (see Figure 2–42) generates CLKG and FSG (also, see section 2.5.2.3).

2.5.2.1 Input Clock Source Mode: CLKSM

The CLKSM bit in SRGR2 selects either the CPU clock (CLKSM = 1) or the external clock input (CLKSM = 0), CLKS, as the source for the sample rate generator input clock. Any divide periods are divide-downs calculated by the sample rate generator and are timed by this input clock selection. When CLKSM = 1, the minimum value of CLKGDV should be 1.

2.5.2.2 Sample Rate Generator Data Bit Clock Rate: CLKGDV

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

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

2.5.2.3 Bit Clock Polarity: CLKSP

External clock (CLKS) is selected to drive the sample rate generator clock divider by selecting CLKSM=0. In this case, the CLKSP bit in SRGR2 selects the edge of CLKS on which sample rate generator data bit-clock (CLKG) and frame-sync signal (FSG) are generated. Since the rising edge of CLKSRG (see Figure 2–42) generates CLKG and FSG, the rising edge of CLKS when CLKSP = 0, or the falling edge of CLKS when CLKSP = 1, causes the transition on the data bit-rate clock (CLKG) and frame sync (FSG).

2.5.2.4 Bit Clock and Frame Synchronization

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

Figure 2–45 and Figure 2–46 show the bit clock and frame-synchronization operation with various polarities of CLKS and FSR. These figures assume FWID = 0, for an FSG one CLKG wide.

Figure 2–45. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 1

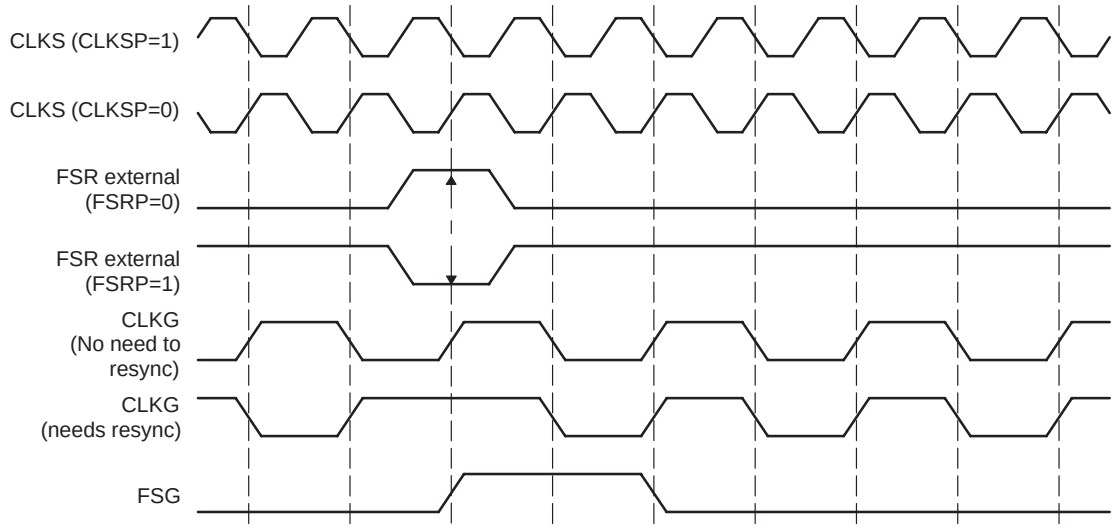
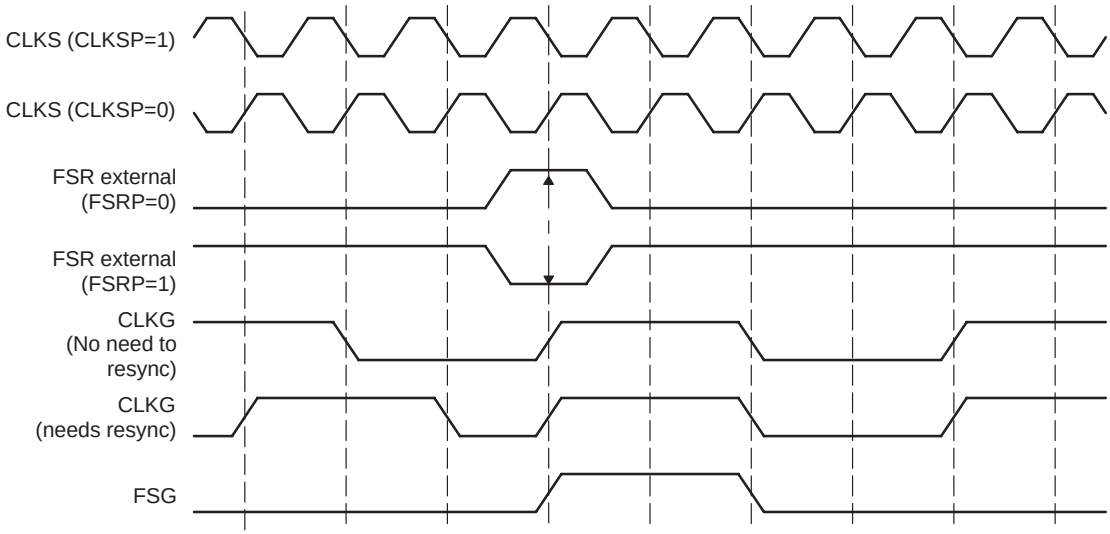


Figure 2–46. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 3



Note that FPER is not programmed since it is determined by the arrival of the next external frame-sync pulse. Figure 2–45 and Figure 2–46 show what happens to CLKG both when it is initially synchronized and GSYNC = 1, and when it is not initially synchronized and GSYNC = 1.

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

1) FSX is programmed to be driven by the sample rate generator frame sync, FSG (FSGM = 1 in SRGM and FSXM = 1 in the PCR). If the input FSR has appropriate timing so that it can be sampled by the falling edge of CLKG, it can be used, instead, by setting FSXM = 0 in the PCR and connecting FSR to FSX externally.

2) The sample rate generator clock should drive the transmit and receive bit clock (CLK[R/X]M = 1 in SPCR[1,2]). Therefore, the CLK(R/X) pin should not be driven by any other driving source.

2.5.2.5 Digital Loop Back Mode: DLB

Setting DLB = 1 in SPCR1 enables digital loop back mode. During DLB mode, DR, FSR, and CLKR are internally connected through multiplexers to DX, FSX, CLKX, respectively, as shown in Figure 2–41. DLB mode allows testing of serial port code with a single DSP device. Figure 2–42 shows the multiplexing of receiver control inputs during digital loop back mode.

2.5.2.6 Receive Clock Selection: DLB, CLKRM

Table 2–19 shows how the digital loop back bit (DLB) and the CLKRM bit in the PCR can select the receiver clock. In digital loop back mode (DLB =1), the transmitter clock drives the receiver. CLKRM determines whether the CLKR pin is an input or an output.

Table 2–19. Receive Clock Selection

DLB in SPCR1	CLKRM in PCR	Source of Receive Clock	CLKR Pin
0	0	CLKR pin acts as an input driven by external clock and inverted as determined by CLKRP before being used.	Input
0	1	Sample Rate Generator Clock (CLKG) drives CLKR.	Output. CLKG inverted as determined by CLKRP before being driven out on CLKR.
1	0	Internal CLKX drives the receive clock internal CLKR as selected and inverted as shown in Table 2–20.	High Impedance
1	1	Internal CLKX drives internal CLKR as selected and inverted as shown in Table 2–20.	Output. CLKR (same as transmit) inverted as determined by CLKRP before being driven out.

2.5.2.7 Transmit Clock Selection: CLKXM

Table 2–20 shows how the CLKXM bit in the PCR selects the transmit clock and indicates whether the CLKX pin is an input or output.

Table 2–20. Transmit Clock Selection

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

2.5.3 Frame-Sync Signal Generation

Similar to data bit clocking, data frame synchronization is also independently programmable for the receiver and transmitter for all data delays. When set to one, the $\overline{\text{FRST}}$ bit in SPCR2 activates the frame-sync generation logic to generate a frame-sync signal, provided FSGM = 1 in SRGR2. The frame-sync programming options are:

- A frame pulse with a programmable period between sync pulses and programmable active width, using the sample rate generator register (SRGR1).
- The transmit portion may trigger its own frame-sync signal generated by a DXR[1,2]-to-XSR[1,2] copy. This causes a frame sync to occur on every DXR1 to XSR1 copy. The data delays can be programmed as required; however, maximum frame frequency cannot be achieved in this method for data delays one and two. This limitation can be overcome by programming the frame ignore bit (R/X)FIG = 1.
- Both the receive and transmit sections may independently select an external frame synchronization on the FSR and FSX pins, respectively.

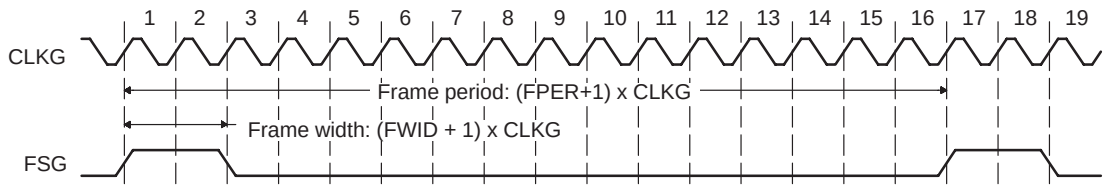
2.5.3.1 Frame Period and Frame Width: FPER and FWID

FPER and FWID are implemented as down-counters. The FPER stage is a 12-bit down-counter that counts down the generated data clocks from 4095 to 0. FPER controls the period of active frame-sync pulses. The FWID stage in the sample rate generator is an 8-bit down counter. The FWID field controls the active width of the frame-sync pulse. Both these counters gets loaded with their respective programmed value in FPER and FWID.

When the sample rate generator comes out of reset, FSG is in its inactive state. Then, when $\overline{\text{FRST}}$ = 1 and FSGM = 1, a frame sync is generated. The frame width value (FWID+1) is counted down on every CLKG cycle until it reaches zero, when FSG goes low. Thus, the value of FWID + 1 determines an active frame pulse width ranging from 1 to 256 data bit-clocks. At the same time, the frame period value (FPER+1) is also counting down. When this value reaches zero, FSG goes high, again indicating a new frame.

It is recommended that FWID be programmed to a value less than WDLEN[1,2].Thus, the value of FPER+1 determines a frame length from 1 to 4096 data bits. When GSYNC = 1, FPER is a don't care value. Figure 2–47 shows a frame of period 16 CLKG periods (FPER = 15 or 00001111b), and a frame with an active width of 2 CLKG periods (FWID = 1).

Figure 2–47. Programmable Frame Period and Width



2.5.3.2 Receive Frame-Sync Selection: DLB, FSRM, GSYNC

Table 2–21 shows how you may select various sources to provide the receive frame-synchronization signal. Note that in digital loop back mode (DLB = 1), the transmit frame-sync signal is used as the receive frame-sync signal, and that DR is connected to DX internally.

Table 2–21. Receive Frame-Synchronization Selection

DLB in SPCR1	FSRM in PCR	GSYNC in SRGR2	Source of Receive Frame Synchronization	FSR Pin
0	0	x	External frame-sync signal drives the FSR input pin. This is then inverted as determined by FSRP before being used as internal FSR.	Input
0	1	0	Internal FSR driven by sample rate generator Frame-Sync signal (FSG), $\overline{\text{FRST}}$ = 1	Output. FSG inverted as determined by FSRP before being driven out on FSR pin.
0	1	1	Internal FSR driven by sample rate generator Frame-Sync signal (FSG), $\overline{\text{FRST}}$ = 1	Input. The external frame-sync input on FSR is used to synchronize CLKG and generate FSG.
1	0	0	Internal FSX drives internal FSR. FSX is selected as shown in Table 2–22.	High Impedance.
1	X	1	Internal FSX drives internal FSR and is selected as shown in Table 2–22.	Input. External FSR not used for frame synchronization but still used to synchronize CLKG and generate FSG, since GSYNC = 1.
1	1	0	Internal FSX drives internal FSR and is selected as shown in Table 2–22.	Output. Receive (same as transmit) frame synchronization inverted as determined by FSRP before being driven out.

2.5.3.3 Transmit Frame-Sync Signal Selection: FSXM, FSGM

Table 2–22 shows how you can select the source of transmit frame-synchronization pulses. The three choices are:

1) External frame-sync input.

2) The sample rate generator frame-sync signal, FSG.

3) A signal that indicates a DXR[1,2]-to-XSR[1,2] copy has been made.

Table 2–22. Transmit Frame-Synchronization Selection

FSXM in PCR	FSGM in SRGR	Source of Transmit Frame Synchronization	FSX Pin
0	x	External frame-sync input on FSX pin. This is inverted by FSXP before being used as internal FSX.	Input.
1	1	Sample rate generator frame-sync signal (FSG) drives internal FSX. $\overline{\text{FRST}}$ = 1	Output. FSG inverted by FSXP before being driven out on FSX pin.
1	0	A DXR[1,2]-to-XSR[1,2] copy activates transmit frame-sync signal.	Output. One bit-clock wide signal inverted as determined by FSXP before being driven out on FSX pin.

2.5.3.4 Frame Detection for Initialization

To facilitate detection of frame synchronization, the receive and transmit CPU interrupts (RINT and XINT) may be programmed to detect frame synchronization by setting RINTM = XINTM=10b in SPCR[1,2]. Unlike other types of serial port interrupts, this mode can operate while the associated portion of the serial port is in reset (such as activating RINT when the receiver is in reset). In this case, FS(R/X)M and FS(R/X)P still select the appropriate source and polarity of frame synchronization. Thus, even when the serial port is in reset state, these signals are synchronized to the CPU clock and then sent to the CPU in the form of RINT and XINT at the point at which they feed the receive and transmit portions of the serial port. Consequently, a new frame-synchronization pulse can be detected, and after this occurs the CPU can take the serial port out of reset safely.

2.5.4 Clocking Examples

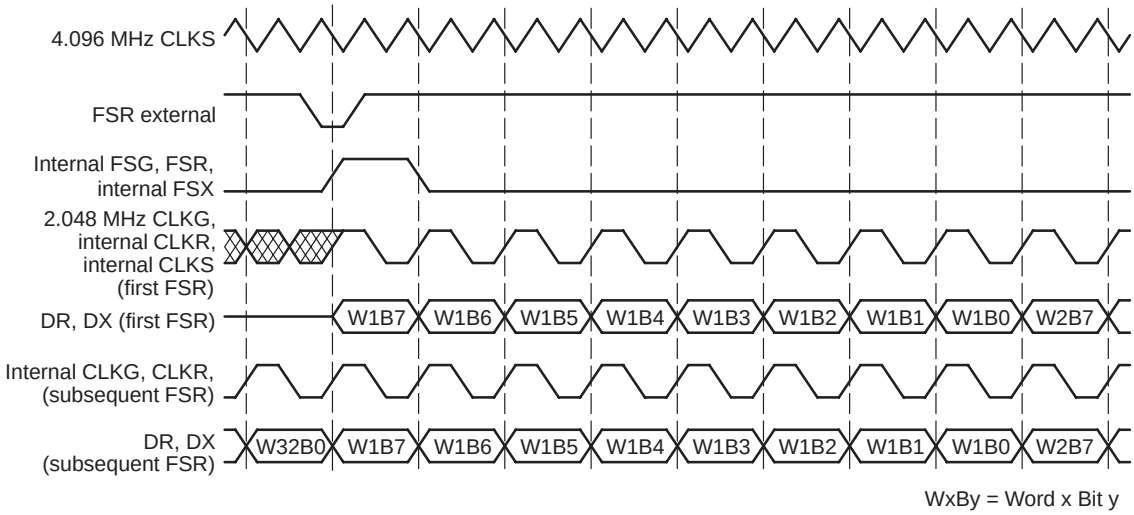
The following sections provide clocking examples:

2.5.4.1 Double-Rate ST-BUS Clock

Figure 2–48 shows McBSP configuration to be compatible with the Mitel ST-Bus. Note that this operation is running at maximum frame frequency.

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

Figure 2–48. ST-BUS and MVIP Example

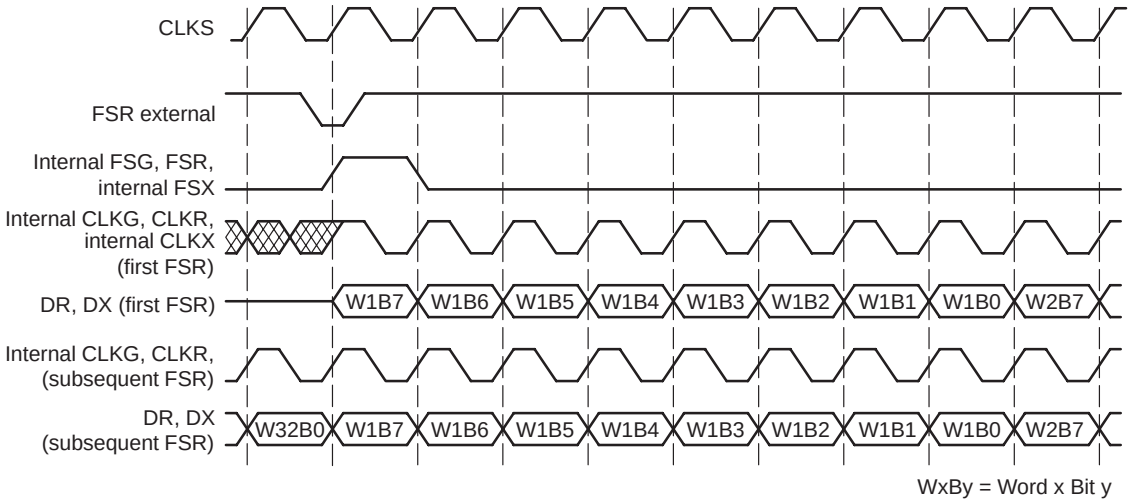


2.5.4.2 Single-Rate ST-BUS Clock

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

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

Figure 2–49. Single-Rate Clock Example



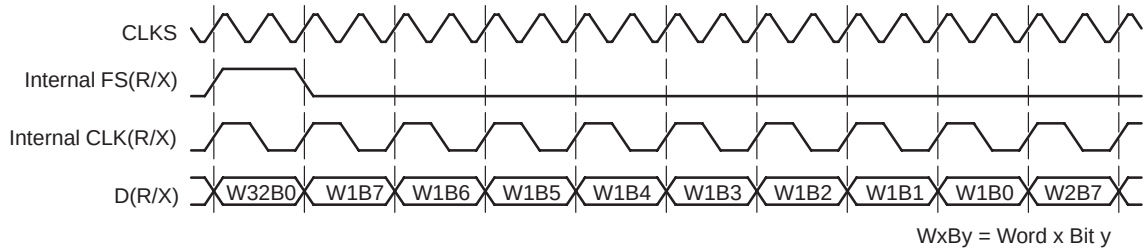
The rising edge of CLKS is used to detect the external FSR. This external frame-sync pulse is used to resynchronize internal McBSP clocks and generate frame-sync for internal use. Note that the internal frame sync is generated so that it is wide enough to be detected on the falling edge of internal clocks.

2.5.4.3 Double-Rate Clock Example

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

- ❑ CLKSP = 0, rising edge of CLKS generates CLKG and thus CLK(R/X).
- ❑ CLKGDV = 1, CLKG and thus internal CLKR and internal CLKX frequency is half CLKS.
- ❑ GSYNC = 0, CLKS drives CLKG. CLKG runs freely and is not resynchronized by FSR.
- ❑ FS(R/X)M = 0, frame synchronization is externally generated. The framing pulse is wide enough to be detected.
- ❑ FS(R/X)P = 0, active-high input frame-sync signal.
- ❑ (R/X)DATDLY = 1, data delay of one bit.

Figure 2–50. Double-Rate Clock Example



2.6 Multichannel Selection Operation

Multiple channels can be independently selected for the transmitter and receiver by configuring the McBSP with a single-phase frame. Each frame represents a time-division multiplexed (TDM) data stream. The number of words per frame represented by (R/X)FRLEN1, denotes the number of channels available for selection.

When using TDM data streams, the CPU may need to process only a few of them. Thus, to save memory and bus bandwidth, multichannel selection allows independent enabling of particular channels for transmission and reception. Up to 32 channels can be enabled in an up-to-128-channel bit-stream.

If a receive channel is not enabled:

- ❏ RRDY is not set to 1 upon reception of the last bit of the word.
- ❏ RBR[1,2] is not copied to DRR[1,2] upon reception of the last bit of the word. Thus, RRDY is not set active. This feature also implies that no interrupts or synchronization events are generated for this word.

If a transmit channel is not enabled:

- ❏ DX is in the high-impedance state.
- ❏ A DXR[1,2]-to-XSR[1,2] transfer is not automatically triggered at the end of serial transmission of the related word.
- ❏ $\overline{\text{XEMPTY}}$ and XRDY similarly are not affected by the end of transmission of the related serial word .

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

2.6.1 Multichannel Operation Control Registers

The following control registers are used in multichannel operation:

1) Multichannel control 1 and 2 (MCR[1,2]) registers

2) Transmit channel enable partition A/B (XCER[A/B]) registers

3) Receive channel enable partition A/B (RCER[A/B]) registers

The use of these registers in controlling multichannel operation is described in the sections that follow.

Figure 2–51. Multichannel Control Register 1 (MCR1)

15 9	8 7	6 5	4 2	1	0
rsvd	RPBBLK	RPABLK	RCBLK	rsvd	RMCM
R,+0	RW,+0	RW,+0	R,+0	R,+0	RW,+0

Note: R = Read, W = Write, +0 = Value at reset

Table 2–23. Multichannel Control Register 1 (MCR1) Bit-Field Descriptions

Bit	Name	Function	Section
15–9	rsvd	Reserved	
8–7	RPBBLK	Receive Partition B Block	2.6.3
		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	2.6.3
		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	

Table 2–23. Multichannel Control Register 1 (MCR1) Bit-Field Descriptions (Continued)

Bit	Name	Function		Section
4–2	RCBLK	Receive Current Block		2.6.3.2
		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		2.6.2
		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.	

Figure 2–52. Multichannel Control Register 2 (MCR2)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD							XPBBLK		XPABLK		XCBLK			XMCM	
R,+0							RW,+0		RW,+0		R,+0			RW,+0	

Note: R = Read, W = Write, +0 = Value at reset

Table 2–24. Multichannel Control Register 2 (MCR2) Bit-Field Descriptions

Bit	Name	Function		Section
15–9	rsvd	Reserved		
8–7	XPABLK	Transmit Partition A Block		2.6.3
		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		2.6.3
		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		2.6.3.2
		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	

Table 2–24. Multichannel Control Register 2 (MCR2) Bit-Field Descriptions (Continued)

Bit	Name	Function		Section
1–0	XMCM	Transmit Multichannel Selection Enable		2.6.2
		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.	

† DX is masked or driven to hi-Z during (a) interpacket intervals, (b) when a channel is masked regardless of whether it is enabled, or (c) when a channel is disabled.

2.6.2 Enabling Multichannel Selection

The multichannel mode can be enabled independently for receive and transmit by setting RMCM = 1 and XMCM to a non-zero value in MCR[1,2], respectively.

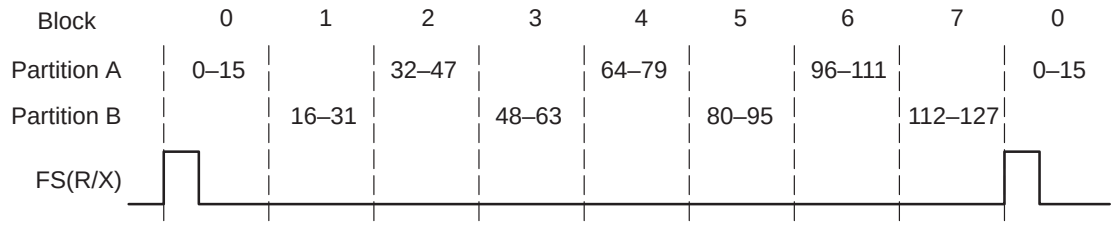
2.6.3 Enabling and Masking of Channels

A total of 32 out of the available 128 channels may be enabled at any given point in time. The 128 channels comprise eight blocks (0 through 7) and each block has 16 contiguous channels. Further, even-numbered blocks 0, 2, 4, and 6 belong to Partition A, and odd-numbered blocks 1, 3, 5, and 7 belong to Partition B.

The number of channels enabled can be updated during the course of a frame to allow any arbitrary group of channels to be enabled. This update is accomplished using an alternating scheme controlling two blocks (one odd-numbered and one even-numbered) of 16 contiguous channels each, at any given time within the frame. One block belongs to Partition A and the other to Partition B.

Any two out of the eight 16-channel blocks may be selected, yielding a total of 32 channels that can be enabled. The blocks are allocated on 16-channel boundaries within the frame as shown in Figure 2–53. (R/X)PABLK and (R/X)PBBLK fields in MCR[1,2] determine the blocks that get selected in partition A and B, respectively. This enabling is performed independently for transmit and receive.

Figure 2–53. Channel Enabling by Blocks in Partition A and B



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

Note:

DX is masked or driven to high-Impedance during (a) interpacket intervals, (b) when a channel is masked regardless of whether it is enabled, or (c) when a channel is disabled.

The following gives a description of the multichannel operation during transmit for various XMCM values:

- XMCM = 00b: The serial port transmits data over the DX pin for as many number of words programmed in XFRLEN1. Thus, DX is driven during transmit.
- XMCM = 01b: Required channels, or only those words that need to be transmitted, are selected via XP(A/B)BLK and XCER(A/B). Therefore, only these selected words will be written to DXR[1,2] and ultimately transmitted. In other words, if XINTM = 00b, which implies that an XINT will be generated for every DXR1-to-XSR1 copy, the number of XINT generated will be equal to the number of channels selected via XCER(A/B) (and NOT equal to XFRLEN1).
- XMCM = 10b: For this case, all channels are enabled which means all the words in a data frame (XFRLEN1) will be written to DXR[1,2] and

DXR[1,2]-to-XSR[1,2] copy occurs at their respective times. However, DX will be driven only for those channels that are selected via XP(A/B)BLK and XCER(A/B), and is placed in the high-impedance state otherwise. In this case, if XINTM = 00b, the number of interrupts generated due to every DXR1-to-XSR1 copy would equal the number of words in that frame (XFRLEN1).

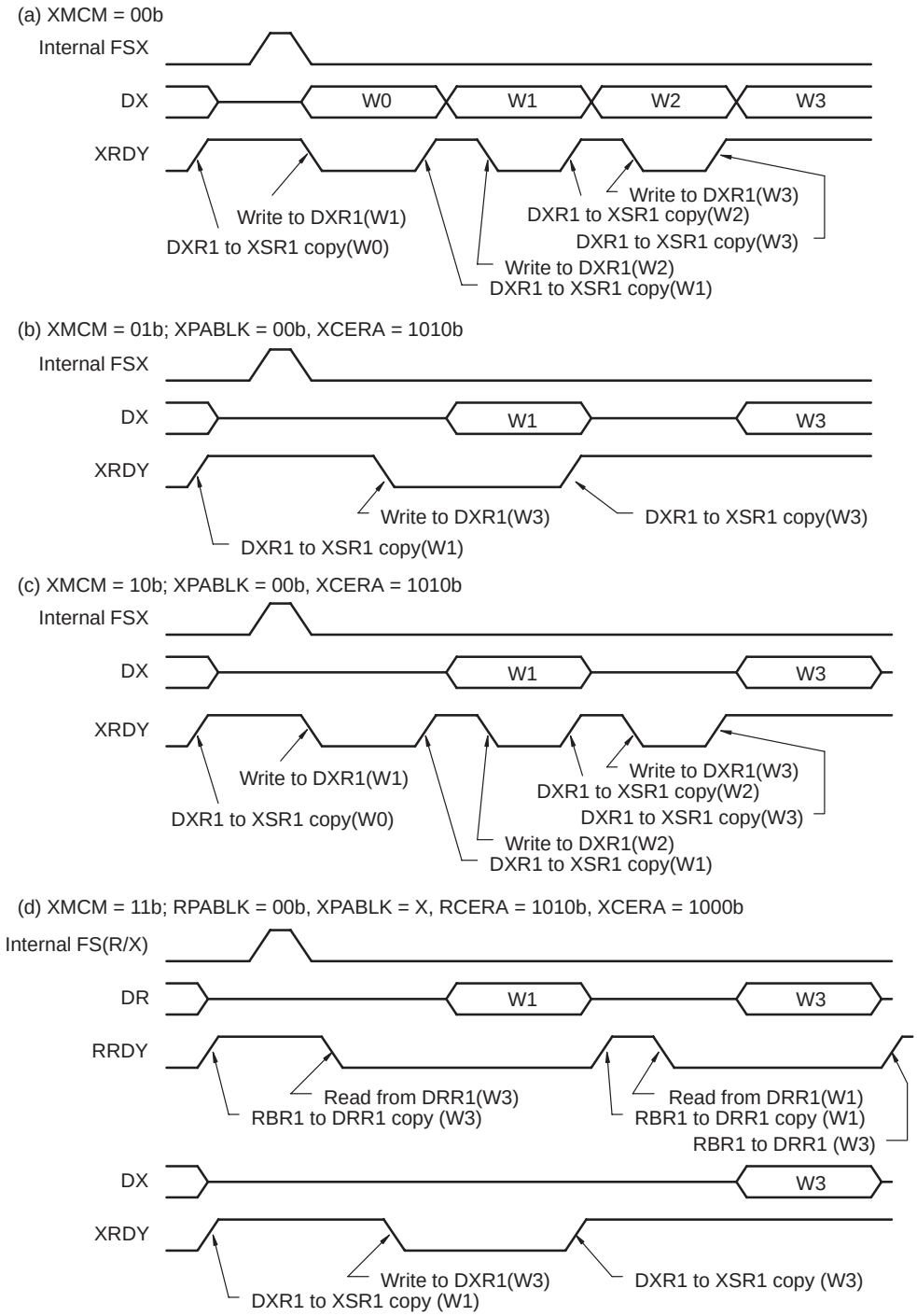
- XMCM = 11b: This mode is basically a combination of XMCM = 01b and 10b cases so that symmetric transmit and receive operation is achieved. For receive, an RBR[1,2]-to-DRR[1,2] copy occurs only for those channels that are selected via RP(A/B)BLK and RCER(A/B). If RINT were to be generated for every RBR[1,2]-to-DRR[1,2] copy, it would occur as many times as the number of channels selected in RCER(A/B) (and NOT the number of words programmed in RFRLEN1). For transmit, the same block that is used for reception is used in order to maintain symmetry and thus XP(A/B)BLK is a don't care. DXR[1,2] is loaded and DXR[1,2]-to-XSR[1,2] copy occurs for all the channels that are enabled via RP(A/B)BLK. However, DX will be driven only for those channels that are selected via XCER(A/B). Note that the channels enabled in XCER(A/B) can only be a subset, or same as, those selected in RCER(A/B). Therefore, if XINTM = 00b, transmit interrupts to the CPU would be generated as many times as the number of channels selected in RCER(A/B) (not XCER[A/B]).

Figure 2–54 shows the activity on the McBSP pins for all the previously described modes with the following conditions:

- (R/X)PHASE = 0, single-phase frame for multichannel selection enabled
- FRLEN1 = 011b, 4-word frame
- WDLEN1 = 000b, 8-bit word

Please note that in the following illustrations, the arrows showing where the various events occur are only sample indications. Wherever possible, there is a time window in which these events can occur.

Figure 2–54. XMCM Operation



2.6.3.1 Channel Enable Registers: (R/X)CER(A/B)

The Receive Channel Enable Partition A and B (RCER[A/B]) and Transmit Channel Enable Partition A and B (XCER[A/B]) registers are used to enable any of the 32 channels for receive and transmit, respectively. 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. They are shown in Figure 2–55, Figure 2–56, Figure 2–57, and Figure 2–58.

(R/X)CERA and (R/X)CERB register fields shown in Table 2–25, Table 2–26, Table 2–27, and Table 2–28 enable channels within the 16-channel-wide blocks in partitions A and B, respectively. The (R/X)PABLK and (R/X)PBBLK fields in the MCR select which 16-channel blocks get selected.

Figure 2–55. Receive Channel Enable Register Partition A (RCERA)

15	14	13	12	11	10	9	8
RCEA15	RCEA14	RCEA13	RCEA12	RCEA11	RCEA10	RCEA9	RCEA8
RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0
7	6	5	4	3	2	1	0
RCEA7	RCEA6	RCEA5	RCEA4	RCEA3	RCEA2	RCEA1	RCEA0
RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0

Note: R = Read, W = Write, +0 = Value at reset

Table 2–25. Receive Channel Enable Register Partition A (RCERA) Bit-Field Descriptions

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

Figure 2–56. Receive Channel Enable Register Partition B (RCERB)

15	14	13	12	11	10	9	8
RDEB15	RCEB14	RCEB13	RCEB12	RCEB11	RCEB10	RCEB9	RCEB8
RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0
7	6	5	4	3	2	1	0
RCEB7	RCEB6	RCEB5	RCEB4	RCEB3	RCEB2	RCEB1	RCEB0
RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0

Note: R = Read, W = Write, +0 = Value at reset

Table 2–26. Receive Channel Enable Register Partition B (RCERB) Bit-Field Descriptions

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

Figure 2–57. Transmit Channel Enable Register Partition A (XCERA)

15	14	13	12	11	10	9	8
XCEA15	XCEA14	XCEA13	XCEA12	XCEA11	XCEA10	XCEA9	XCEA8
RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0
7	6	5	4	3	2	1	0
XCEA7	XCEA6	XCEA5	XCEA4	XCEA3	XCEA2	XCEA1	XCEA0
RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0

Note: R = Read, W = Write, +0 = Value at reset

Table 2–27. Transmit Channel Enable Register Partition A (XCERA) Bit-Field Descriptions

Bit	Name	Function	
15–0	XCEA(0:15)	Transmit Channel Enable-	
		XCEA *n* = 0	Disables transmission of *n*th channel in an even-numbered block in partition A
		XCEA *n* = 1	Enables transmission of *n*th channel in an even-numbered block in partition A

Figure 2–58. Transmit Channel Enable Register Partition B (XCERB)

15	14	13	12	11	10	9	8
XCEB15	XCEB14	XCEB13	XCEB12	XCEB11	XCEB10	XCEB9	XCEB8
RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0
7	6	5	4	3	2	1	0
XCEB7	XCEB6	XCEB5	XCEB4	XCEB3	XCEB2	XCEB1	XCEB0
RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0	RW,+0

Note: R = Read, W = Write, +0 = Value at reset

Table 2–28. Transmit Channel Enable Register Partition B (XCERB) Bit-Field Descriptions

Bit	Name	Function	
15–0	XCEB(0:15)	Transmit Channel Enable-	
		XCEB *n* = 0	Disables transmission of *n*th channel in an even-numbered block in partition B
		XCEB *n* = 1	Enables transmission of *n*th channel in an even-numbered block in partition B

2.6.3.2 Changing Channel Selection

Using the multichannel selection feature, a static group of 32 channels may be enabled, and will remain enabled with no CPU intervention until this allocation requires modification. An arbitrary number, group, or all of the words/channels within a frame can be accessed by updating the block allocation registers during the course of the frame, in response to the end-of-block interrupts (see section 2.6.3.3, *Update Interrupts*).

Note:

When changing the selection, the user must be careful not to affect the currently selected block.

The currently selected block is readable through the RCBLK and XCBLK fields in MCR1 for receive and MCR2 for transmit, respectively. The associated channel enable register cannot be modified if it is selected by the appropriate (R/X)P(A/B)BLK register to point toward the current block. Similarly, the (R/X)PABLK and (R/X)PBBLK fields in MCR[1,2] cannot be modified while pointing to, or being changed to point to, the currently selected block. Note that if the total number of channels is 16 or less, the current partition is always pointed to. In this case, only a reset of the serial port can change enabling.

2.6.3.3 Update Interrupts

At the end of every 16-channel block boundary during multichannel operation, the receive interrupt (RINT) or transmit interrupt (XINT) to the CPU is generated if RINTM = 01b or XINTM = 01b in SPCR[1,2], respectively. This interrupt indicates that a new partition has been crossed. You can then check the current partition and change the selection of blocks in the A and/or B partitions, if they do not point to the current block. These interrupts are two CPU-clock long, active-high pulses. If RINTM = XINTM = 01b when (R/X)MCM = 0 (non-multichannel operation), interrupts are not generated.

2.6.4 A-bis interface functionality (available on ’5410 only)

In the A-bis mode (ABIS = 1), the McBSP can receive and transmit up to 1024 bits on a PCM link. The receive section can extract all 1024 bits from a 1024-bit PCM frame according to a given receiving pattern, and generate an interrupt to the CPU when DRR is compacted with a 16-useful-bit word, or when a receive frame is ended. In addition, the transmit section can expand up to 1024 bits into a 1024-bit PCM frame at a specific position, according to a given transmitting pattern, and generate an interrupt when a 16-bit word is transmitted or a transmit frame is ended. In this mode, a word length must be specified as 16-bit mode (WDLEN1 = 010b); otherwise, the A-bis mode is undetermined.

In A-bis mode, the channel enable registers, (R/X)CER(A/B), have a different function than in normal multichannel operation. Instead of indicating which channels will be enabled, in A-bis mode these registers indicate which bits in the data stream will be enabled. A 1 in a given position in the (R/X)CER(A/B) register enables a corresponding bit in the receive or transmit data stream.

On the receiver, bits that are not enabled in the RCER(A/B) registers are ignored and are not compacted in the receiver. Bits that are enabled are received and compacted into a useful data word. When 16 useful-data bits have been received, the received word is copied from RSR1 to DRR1 and the McBSP generates an interrupt to the CPU. RCERA and RCERB alternate specifying the receive masking pattern for each of the 16 receive clocks. Figure 2–59 shows an example bit sequence for the receiver.

Figure 2–59. A-bis Mode Receive Operation

RCERA	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1																
RCERB																	0	0	1	0	0	1	1	1	0	0	1	0	0	1	1	1
DR pin†	1	0	1	1	0	1	0	1	0	0	1	1	0	1	0	1	0	1	1	1	0	1	1	0	1	1	0	1	1	0	1	1
DRR1‡	–	0	–	1	–	1	–	1	–	0	–	1	0	–	–	1	–	–	1	–	–	1	1	0	–	–	0	–	–	0	1	1

75E3h

† Data arriving on the DR pin
‡ Received data in DRR1

On the transmitter, only bits that are enabled in XCER(A/B) are transmitted out from the DX pin. Bits that are not enabled are not transmitted and the DX pin is in the high-impedance state during that clock cycle. XCERA and XCERB alternate specifying the receive masking pattern for each 16 receive clocks. When 16 useful bits have been shifted out, the McBSP generates an interrupt to the CPU. Figure 2–60 shows an example bit sequence for the transmitter.

Figure 2–60. A-bis Mode Transmit Operation (z Indicates High-Impedance)

XCERA	0	1	1	1	1	1	0	0	0	1	1	0	0	1	1	1																
XCERB																	0	0	1	1	0	0	0	1	1	0	0	0	1	1	0	0
DXR1§	1	0	1	1	0	1	0	1	0	0	0	1	1	1	1	1	1	0	1	1	0	1	1	0	1	1	0	0	0	0	0	0
DX pin¶	z	0	1	1	0	1	z	z	z	0	0	z	z	1	1	1	z	z	1	1	z	z	z	0	1	z	z	z	0	0	z	z

§ Data written to DXR1
¶ Data transmitted on the DX pin

Two special events, XEVTA (transmit A-bis event) and REVTA (receive A-bis event), can be used by the DMA to load patterns into the (R/X)CER(A/B) registers. This capability is used for bit sequences that are longer than the 32 bits covered by (R/X)CER(A/B). These two events are generated every 16 CLKR/CLKX cycles.

As an example, the following gives a description of the A-bis operation on a 256-bit PCM link:

- ABIS bit is enabled in SPCR1.
- The initial pattern of bits that must be enabled is loaded into the (R/X)CER(A/B) registers.
- (R/X)PHASE = 1111b, single-phase frame.
- FRLEN1 = 1111b, 16-word frame.
- WDLEN1 = 010b, 16-bit words (required for A-bis mode).

Two DMA channels (one for transmit, one for receive) are used to update the pattern selection data to (R/X)CER(A/B) as the operation proceeds. One 16-word block in memory contains the bit pattern selections for the receiver. Sixteen words of 16-bits each contain the entire receive selection pattern for the 256-bit PCM link. On each REVTA event, the DMA copies new receive selection pattern data from memory to RCER(A/B) and automatically toggles its destination pointer from RCERA to RCERB, or vice versa, as necessary. In other words, the DMA channel is initially set to RCERA as a destination. After the first access to RCERA, the destination automatically toggles to RCERB. After the next RCERB access, the destination automatically toggles back to RCERA. Since the toggling between RCERA and RCERB is handled automatically, you do not need to configure the DMA controller to modify the destination address by other means. As the A-bis transfer proceeds, the receiver alternates using RCERA and RCERB to specify the enable pattern for each group of 16 serial port clocks.

Another 16-word block in memory contains the bit pattern selections for the transmitter. Sixteen words of 16-bits each contain the entire transmit selection pattern for the 256-bit PCM link. On each XEVTA event, the DMA copies the new transmit selection pattern data from memory to XCER(A/B) and automatically toggles its destination pointer from XCERA to XCERB, or vice versa, as necessary. In other words, the DMA channel is initially set to XCERA as a destination. After the first access to XCERA, the destination automatically toggles to XCERB. After the next XCERB access, the destination automatically toggles back to XCERA. Since the toggling between XCERA and XCERB is handled automatically, you do not need to configure the DMA controller to modify the destination address by other means. As the A-bis transfer proceeds, the receiver alternates. The transmitter alternates using XCERA and XCERB to specify the enable pattern for each group of 16 serial port clocks.

The result is operation of a continuous 256-bit sequence with a unique selection pattern for both the transmitter and the receiver.

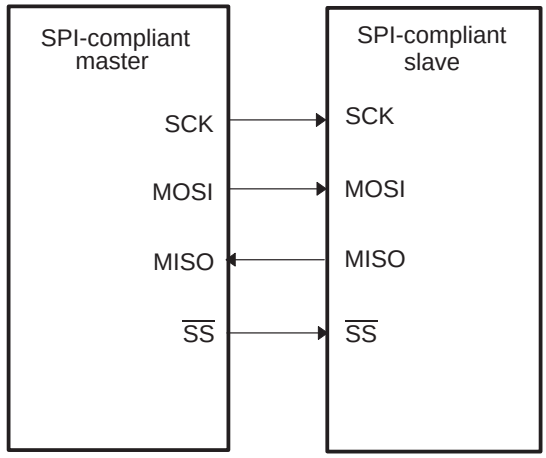
2.7 SPI Protocol: McBSP Clock Stop Mode

The SPI protocol is a master-slave configuration, with one master device and one or more slave devices. The interface consists of the following four signals:

- ❏ Serial data input (also referred to as Master In – Slave Out, or MISO)
- ❏ Serial data output (also referred to as Master Out – Slave In, or MOSI)
- ❏ Shift-clock (also referred to as SCK)
- ❏ Slave-enable signal (also referred to as $\overline{SS}$)

A typical SPI interface with a single slave device is shown in Figure 2–61.

Figure 2–61. Typical SPI Interface

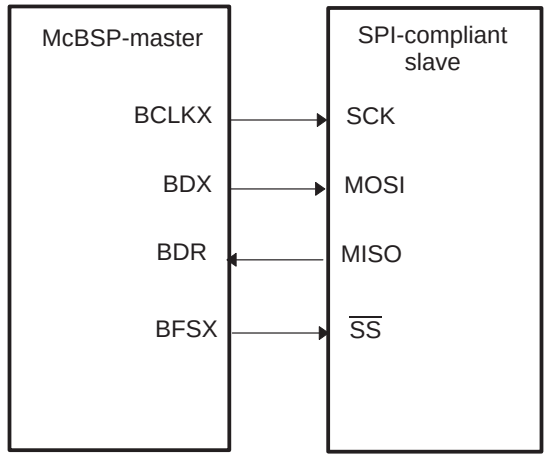


The master device controls the flow of communication by providing shift-clock and slave-enable signals. The slave-enable signal is an optional active-low signal that enables the serial data input and output of the slave device (device not outputting clock). In the absence of a dedicated slave-enable signal, communication between the master and slave is determined by the presence or absence of an active shift-clock. In such a configuration, the slave device must remain enabled at all times, and multiple slaves cannot be used.

The clock stop mode of the McBSP provides compatibility with the SPI protocol. When the McBSP is configured in clock stop mode, the transmitter and receiver are internally synchronized, so that the McBSP functions as an SPI master or slave device. The transmit clock signal (BCLKX) corresponds to the serial clock signal (SCK) of the SPI protocol, while the transmit frame-synchronization signal (BFSX) is used as the slave-enable signal ($\overline{SS}$). The receive clock signal (BCLKR) and receive frame-synchronization signal (BFSR), are not used in the clock stop mode, since these signals are internally connected to their transmit counterparts, BCLKX and BFSX.

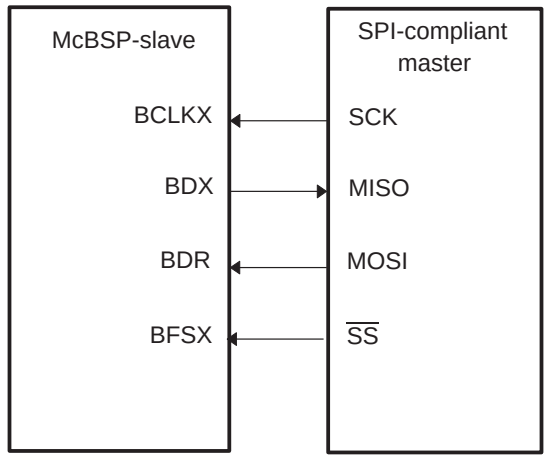
When the McBSP is configured as a master, the transmit output signal (BDX) is used as the MOSI signal of the SPI protocol, and the receive input signal (BDR) is used as the MISO signal. An SPI interface with the McBSP used as the master is shown in Figure 2–62.

Figure 2–62. SPI Configuration: McBSP as the Master



Similarly, when the McBSP is configured as a slave, BDX is used as the MISO signal and BDR is used as the MOSI signal. An SPI interface with the McBSP used as a slave is shown in Figure 2–63.

Figure 2–63. SPI Configuration: McBSP as the Slave



2.7.1 Clock Stop Mode Configuration and Signal Descriptions

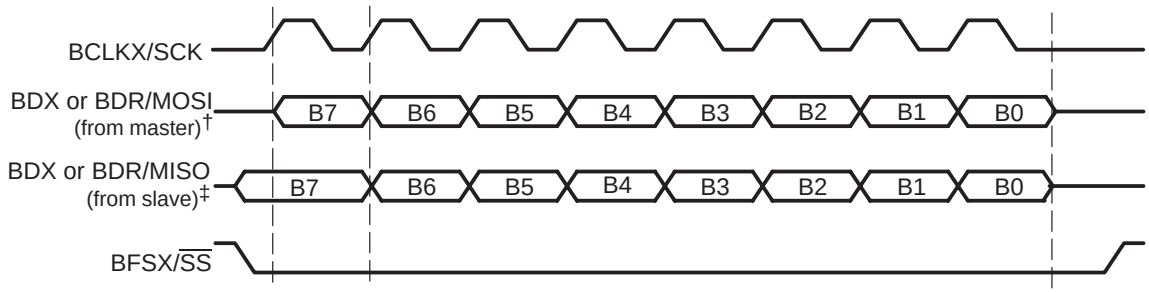
The CLKSTP bit-field of serial port control register 1 (SPCR1) and the CLKXP bit of the pin configuration register (PCR) are used to configure the clock stop mode. The CLKSTP bit-field enables clock stop mode and selects one of two possible timing variations, while the CLKXP bit configures the polarity of the BCLKX signal. Together, these bits provide four possible clock-stop mode configurations, as shown in Table 2–29.

Table 2–29. Clock-Stop Mode Configurations

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

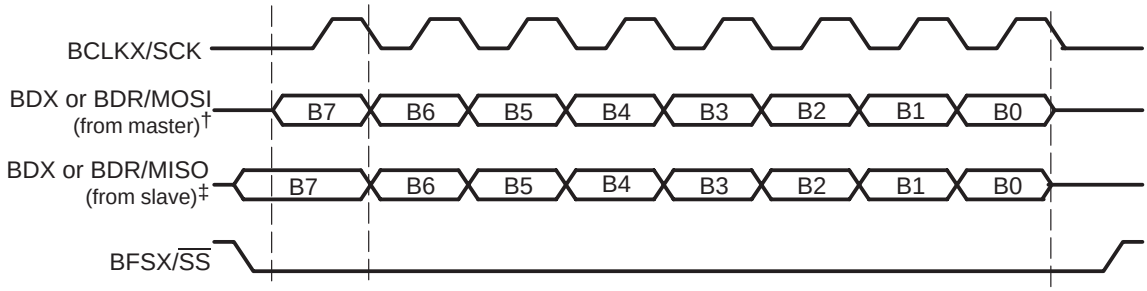
The CLKXM bit-field of the PCR designates the McBSP as a master or a slave by configuring the BCLKX signal in output mode (master), or input mode (slave). The timing diagrams for the four possible clock stop mode configurations are shown in Figure 2–64 through Figure 2–67.

Figure 2–64. SPI Transfer With CLKSTP = 10b, and CLKXP = 0



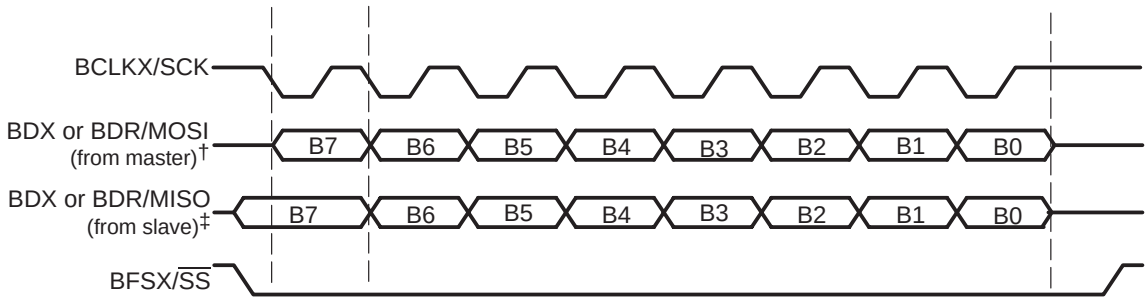
† If the McBSP is the SPI master (CLKXM = 1), MOSI = BDX. If the McBSP is the SPI slave (CLKXM = 0), MOSI = BDR.
‡ If the McBSP is the SPI master (CLKXM = 1), MISO = BDR. If the McBSP is the SPI slave (CLKXM = 0), MISO = BDX.

Figure 2–65. SPI Transfer With CLKSTP = 11b, and CLKXP = 0



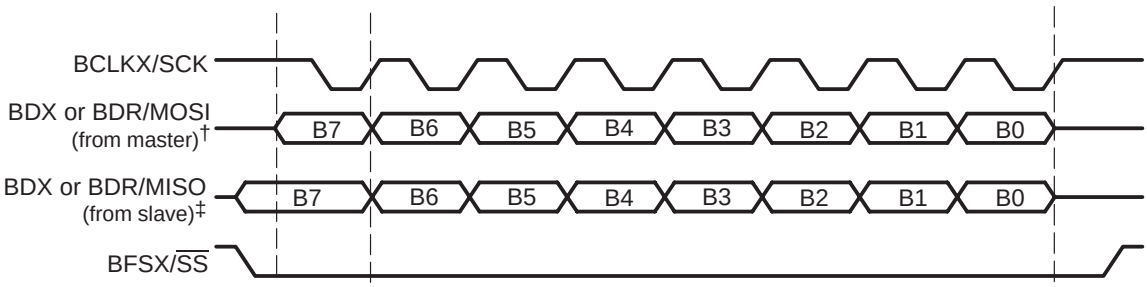
† If the McBSP is the SPI master (CLKXM = 1), MOSI = BDX. If the McBSP is the SPI slave (CLKXM = 0), MOSI = BDR.
‡ If the McBSP is the SPI master (CLKXM = 1), MISO = BDR. If the McBSP is the SPI slave (CLKXM = 0), MISO = BDX.

Figure 2–66. SPI Transfer With CLKSTP = 10b, and CLKXP = 1



† If the McBSP is the SPI master (CLKXM = 1), MOSI = BDX. If the McBSP is the SPI slave (CLKXM = 0), MOSI = BDR.
‡ If the McBSP is the SPI master (CLKXM = 1), MISO = BDR. If the McBSP is the SPI slave (CLKXM = 0), MISO = BDX.

Figure 2–67. SPI Transfer With CLKSTP = 11b, and CLKXP = 1



† If the McBSP is the SPI master (CLKXM = 1), MOSI=BDX. If the McBSP is the SPI slave (CLKXM = 0), MOSI = BDR.
‡ If the McBSP is the SPI master (CLKXM = 1), MISO=BDR. If the McBSP is the SPI slave (CLKXM = 0), MISO = BDX.

Notice that the frame-synchronization signal used in clock stop mode is active throughout the entire transmission. The frame-synchronization waveform is described in more detail in the next two sections. Although the timing diagrams show 8-bit transfers, the packet length can be set to 8-, 12-, 16-, 20-, 24-, or 32-bits per packet. The packet length is selected by the receive word length (RWDLEN1) in receive control register 1 (RCR1), and the transmit word length (XWDLEN1) in transmit control register 1 (XCR1). Note that for clock stop mode, the values of RWDLEN1 and XWDLEN1 must be set to the same value since the McBSP transmit and receive circuits are synchronized to a single clock. The bit-fields required to configure the McBSP as an SPI device are shown in Table 2–30.

Table 2–30. Register Bit Values for SPI Mode Configuration

Bit-Field	Value	Description	Register	Reference
CLKSTP	1xb	Enables clock stop mode, and selects one of two timing variations.	SPCR1	Table 2–4, page 2-7
CLKXP	0 or 1	Configures the BCLKX signal polarity.	PCR	Table 2–6, page 2-12
CLKXM	0 or 1	Configures the BCLKX signal as an input (slave) or an output (master).	PCR	Table 2–6, page 2-12
RWDLEN1	000–101b	Configures the receive packet length. Must be equal to XWDLEN1.	RCR1	Table 2–7, page 2-16
XWDLEN1	000–101b	Configures the transmit packet length. Must be equal to RWDLEN1.	XCR1	Table 2–9, page 2-19

2.7.2 McBSP Operation as an SPI Master

When the McBSP functions as the SPI master, it controls the transmission of data by producing the serial clock signal. The clock signal on the BCLKX pin is only enabled during packet transfers. When packets are not being transferred, the BCLKX pin remains high or low depending on the polarity used. An example of an SPI interface with the McBSP used as the master is shown in Figure 2–62. For SPI master operation, the BCLKX pin must be configured as an output. The sample rate generator is then used to derive the BCLKX signal from the CPU clock. The clock stop mode internally connects the BCLKX pin to the BCLKR signal so that no external signal connection is required on the BCLKR pin, and both the transmit and receive circuits are clocked by the master clock (BCLKX).

The McBSP can also provide a slave-enable signal ($\overline{SS}$) on the BFSX pin. If a slave-enable signal is required, the BFSX pin must be configured as an out-

put, and the frame generator must be configured so that a frame-synchronization pulse is generated each time a packet is transmitted. The data delay parameters of the McBSP (XDATDLY and RDATDLY) must be set to 1 for proper SPI master operation, since data delay values of 0 or 2 are undefined during this mode. The polarity of the BFSX pin is programmable high or low; however, in most cases the pin should be configured active-low.

In this configuration, the frame generator bit fields (FPER and FWID) in the sample rate generator registers are overridden, and custom frame-synchronization waveforms are not allowed. The resulting waveform produced on the BFSX pin is shown in Figure 2–64 though Figure 2–67. The signal becomes active before the first bit of a packet transfer, and remains active until the last bit of the packet is transferred. After the packet transfer is complete, the BFSX signal returns to the inactive state.

It is important to note that even if multiple words are consecutively transferred, the BCLKX signal is always stopped and the BFSX signal returns to the inactive state after a packet transfer. When consecutive packet transfers are performed, this leads to a minimum idle time of two bit-periods between each packet transfer. The register bit values required to configure the McBSP as a master are listed in Table 2–31.

Table 2–31. Register Bit Values for SPI Master Operation

Bit-Field	Value	Description	Register	Reference
CLKXM	1	Configures the BCLKX pin as an output.	PCR	Table 2–6, page 2-12
CLKSM	1	Sample rate clock derived from CPU clock.	SRGR2	Table 2–18, page 2-60
CLKGDV	1–255	Defines the divide factor for the sample rate clock.	SRGR1	Table 2–17, page 2-59
FSXM	1	Configures the BFSX pin as an output.	PCR	Table 2–6, page 2-12
FSGM	0	The BFSX signal is activated for each packet transfer.	SRGR2	Table 2–18, page 2-60
FSXP	1	Configures the BFSX pin as active-low.	PCR	Table 2–6, page 2-12
XDATDLY	01b	Provides correct setup time on the BFSX signal.	XCR2	Table 2–10, page 2-20
RDATDLY	01b	Provides correct setup time on the BFSX signal.	RCR2	Table 2–8, page 2-17

2.7.3 McBSP Operation as an SPI Slave

When the McBSP is used as an SPI slave, the master clock and slave-enable signals are generated externally by a master device. Accordingly, the BCLKX and BFSX pins must be configured as inputs. The BCLKX pin is internally connected to the BCLKR signal, so that both the transmit and receive circuits of the McBSP are clocked by the external master clock. The BFSX pin is also internally connected to the BFSR signal, and no external signal connections are required on the BCLKR and BFSR pins. An example of an SPI interface with the McBSP used as the slave is shown in Figure 2–63.

Although the BCLKX signal is generated externally by the master and is asynchronous to the McBSP, the sample rate generator of the McBSP must be enabled for proper SPI slave operation. The sample rate generator should be programmed to its maximum rate of half the CPU clock rate. The internal sample rate clock is then used to synchronize the McBSP logic to the external master clock and slave-enable signals.

The McBSP requires an active edge of the slave-enable signal on the BFSX input for each transfer. This means that the master device must assert the slave-enable signal at the beginning of each transfer, and deassert the signal after the completion of each packet transfer; the slave-enable signal cannot remain active between transfers. The data delay parameters of the McBSP must be set to 0 for proper SPI slave operation, and values of 1 or 2 are undefined during this mode of operation. The register bit values required to configure the McBSP as a slave are listed in Table 2–32.

Table 2–32. Register Bit Values for SPI Slave Operation

Bit Field	Value	Description	Register	Reference
CLKXM	0	Configures the BCLKX pin as an input.	PCR	Table 2–6, page 2-12
CLKSM	1	Sample rate clock derived from CPU clock.	SRGR2	Table 2–18, page 2-60
CLKGDV	1	Select a divide factor of 2 for the sample rate clock.	SRGR1	Table 2–17, page 2-59
FSXM	0	Configures the BFSX pin as an input.	PCR	Table 2–6, page 2-12
FSGM	0	The BFSX signal is activated for each packet transfer.	SRGR2	Table 2–18, page 2-60
FSXP	1	Configures the BFSX pin as active-low.	PCR	Table 2–6, page 2-12
XDATDLY	0	Must be 0 for SPI slave operation.	XCR2	Table 2–10, page 2-20
RDATDLY	0	Must be 0 for SPI slave operation.	RCR2	Table 2–8, page 2-17

2.7.4 McBSP Initialization for SPI Mode

The operation of the McBSP during device reset, transmitter reset, or receiver reset is described in section 2.3.1, *Resetting the Serial Port*, on page 2-22. To configure the McBSP for SPI master or slave operation, the following steps must be performed:

1) Set the transmitter reset bit ($\overline{\text{XRST}}$) in serial port control register 2 (SPCR2) to zero, to reset the transmitter. Set the receiver reset bit ($\overline{\text{RRST}}$) in serial port control register 1 (SPCR1) to zero, to reset the receiver.

   ($\overline{\text{XRST}}$=$\overline{\text{RRST}}$=0)

2) Program the McBSP register fields shown in Table 2–30. Also program the register fields shown in Table 2–31 or Table 2–32, depending on whether the McBSP is the master or the slave. All other McBSP register fields can be programmed to their default values.

3) Set the sample rate generator reset bit ($\overline{\text{GRST}}$) in serial port control register 2 (SPCR2) to one, to release the sample rate generator from reset. Note that the value written to SPCR2 should have only the $\overline{\text{GRST}}$ bit changed to one, and the remaining bit-fields should have the same values written in Step 2.

   ($\overline{\text{GRST}}$=1)

4) Wait for the duration of two sample rate generator clock-periods for the McBSP logic to stabilize.

5) Depending on whether the CPU or DMA controller services the McBSP transmit and receive buffers, either Step(a) or Step (b) should be performed. If the CPU services the McBSP buffers, Step (a) should be performed, or if the DMA controller services the McBSP buffers, Step (b) should be performed.

   a) *CPU services the McBSP buffers*. Set the $\overline{\text{XRST}}$ and $\overline{\text{RRST}}$ bit-fields to one, to enable the transmitter and receiver. Note that the values written to SPCR1 and SPCR2 should have only the reset bits changed to one, and the remaining bit-fields should have the same values written in Step 2.

      ($\overline{\text{XRST}}$=$\overline{\text{RRST}}$=1)

b) *DMA controller services the McBSP buffers*. First, configure the DMA controller and enable the channels that service the McBSP buffers. Then set the $\overline{\text{XRST}}$ and $\overline{\text{RRST}}$ bit-fields to one, to enable the transmitter and receiver. Note that the values written to SPCR1 and SPCR2 should have only the reset bits changed to one, and the remaining bit-fields should have the same values written in Step 2.

($\overline{\text{XRST}}$=$\overline{\text{RRST}}$=1)

6) Wait for the duration of two sample rate generator clock-periods for the McBSP logic to stabilize.

2.8 Emulation FREE and SOFT Bits

FREE and SOFT are special emulation bits that determine the state of the serial port clock when a breakpoint is encountered in the high-level language debugger. If the FREE bit is set to one, then upon a software breakpoint, the clock continues to run (free runs) and data still shifts out. When FREE = 1, the SOFT bit is a *don't care.* If the FREE bit is cleared to zero, then the SOFT bit takes effect. If the SOFT bit is cleared to zero, then the clock stops immediately, thus aborting a transmission. If the SOFT bit is set to one and a transmission is in progress, the transmission continues until completion of the transfer, and then the clock halts. These options are listed in Table 2–33.

The receiver-side functions in a similar fashion. Note that if an option other than *immediate stop* (SOFT = FREE = 0) is chosen, the receiver continues running and an overflow error is possible.

Table 2–33. McBSP Clock Configuration

FREE	SOFT	McBSP Clock Configuration
0	0	Immediate stop, clocks are stopped. (Reset value)
0	1	Transmitter stops after completion of the current word. The receiver is not affected.
1	don't care	Free run.

2.9 McBSP Pins as General Purpose I/O

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

1) The related portion (transmitter or receiver) of the serial port is in reset; (R/X)$\overline{\text{RST}}$ = 0 in SPCR[1,2].

2) General purpose I/O is enabled for the related portion of the serial port; (R/X)IOEN = 1 in the PCR.

Figure 2–4, *Pin Control Register (PCR)*, on page 2-12, has bits that configure each of the McBSP pins as general purpose inputs or outputs. Table 2–34 shows how this is achieved. In the case of FS(R/X), FS(R/X)M=0 configures the pin as an input and FS(R/X)M = 1 configures that pin as an output. When configured as an output, the value driven on FS(R/X) is the value stored in FS(R/X)P. If configured as an input, FS(R/X)P becomes a read-only bit that reflects the status of that signal. CLK(R/X)M and CLK(R/X)P work similarly for CLK(R/X). When the transmitter is selected as general purpose I/O, the value of the DX_STAT bit in the PCR is driven onto DX. DR is always an input and its value is held in the DR_STAT bit in the PCR. To configure CLKS as a general purpose input, both the transmitter and receiver must be in reset state and (R/X)IOEN = 1, because CLKS is always an input to the McBSP and affects both transmit and receive operations.

Table 2–34. Configuration of Pins as General Purpose I/O

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

2.10 McBSP Operation in Power-Down Mode

’54x devices offer several power-down modes which allow all or part of the device to enter a dormant state, and consequently, dissipates considerably less power than when running normally. Power-down modes may be invoked in several ways, including executing the IDLE instruction or driving the $\overline{\text{HOLD}}$ input low with the HM status bit set to one. The McBSP, like other peripherals, can take the CPU out of IDLE using a transmit or receive interrupt.

When in IDLE1 or HOLD modes, the McBSP continues to operate normally with no restrictions.

In IDLE2 or IDLE3 modes, the internal device clocks provided to the peripherals are stopped. Consequently, some limitations are placed on operating the McBSP in these modes. If external clock and frame-sync are provided, the McBSP can continue to operate, and receive and transmit interrupts can be used to exit the IDLE state. If either clocks or frame-syncs are internal (provided by the clock and frame-sync generator), the McBSP will stop in IDLE2/3 because there are no internal clocks available to operate the clock and frame-sync generator.

In IDLE2/3, the internal clocks to the McBSP and the DMA controller are started automatically when a transfer begins, and stopped after the transfer is completed. This feature provides additional power savings since neither module is active when no activity is occurring.

In an autobuffering configuration, the McBSP and the DMA can be operated such that the CPU remains in the IDLE state until a half- or full-buffer of data has been transferred. The McBSP receive and transmit interrupts should be disabled to avoid interrupt generation on the reception/transmission of each word. The DMA should be configured in autobuffering (ABU) mode and DMA transfers should be triggered on McBSP receive and transmit events. The DMA should then be configured to interrupt the CPU based on completion of half-buffers or full-buffers, whichever is desired. Since the CPU remains in the IDLE state until an interrupt occurs, an entire block of data can be transferred without causing the CPU to exit the IDLE state. For more information on the DMA operation, see Chapter 3, *Direct Memory Access (DMA) Controller.*

2.11 McBSP Programming Example Code

McBSP programming example code is presented in Chapter 3 as follows: Example 3–19, *McBSP Data Transfer in ABU Mode*, is shown on page 3-52; Example 3–20, *McBSP Data Transfer in Double-Word Mode*, is shown on page 3-54; and Example 3–21, *McBSP to Data Memory Transfer With Data Sorting*, is shown on page 3-56.

Chapter 3

Direct Memory Access (DMA) Controller

The direct memory access (DMA) controller is available on selected members of the ’54x family of devices. This chapter describes the configuration and operation of the DMA available on the ’5402, ’5410, and ’5420 devices.

Topic **Page**

3.1 DMA Overview

The direct memory access (DMA) controller transfers data between regions in the memory map without intervention by the CPU. The DMA allows movement to and from internal memory, internal peripherals, or external devices to occur in the background of CPU operation. The DMA has six independent programmable channels allowing six different contexts for DMA operation. The DMA controller also services requests from the host port interface peripherals (HPI-8 or HPI-16) to utilize the DMA bus. Throughout this chapter, the abbreviation HPIx is used to denote either the HPI-8 or HPI-16.

The terms used in discussing DMA operations are defined as follows:

- **Read transfer**. The DMA reads a data element from a source location in memory. The source may be memory or a peripheral device, and may be located in program, data, or I/O space.
- **Write transfer**. The DMA writes the data elements that were read during a read transfer to its destination in memory location. The destination may be memory or a peripheral device, and may be located in program, data, or I/O space.
- **Element transfer**. The combined read and write transfer for a single data element.
- **Frame transfer**. Each DMA channel has an independently programmable number of elements per frame. In completing a frame transfer, the DMA moves all elements in a single frame.
- **Block Transfer**. Each DMA channel also has an independently programmable number of frames per block. In completing a block transfer, the DMA moves the total number of frames defined for the block.

The DMA has the following features:

- **Background operation**. The DMA operates independently of the CPU.
- **Six channels**. The DMA can keep track of the context of six independent block transfers.
- **Host port interface access**. The DMA provides a path to the memory for host port interface data without intervention from the CPU. This allows the host port interface access to a greater memory space (the entire on-chip memory) than ever before.
- **Multiframe transfer**. Each block transfer can consist of multiple frames of a programmable size. See section 3.2.3.3, *DMA Sync Event and Frame Count Registers*, on page 3-13.
- **Programmable priority**. Each channel can be programmed independently for high or low priority. See section 3.2.2, *DMA Channel Priority and Enable Control Register*, on page 3-7.

- **Programmable address generation**. Each channel's source and destination address registers can have configurable indexes for each read and write transfer. The address may remain constant, increment, decrement, or be adjusted by a programmable value. These programmable values allow a separate index for the last transfer in a frame from the preceding transfers, and can be used to achieve data sorting (discussed later in this chapter).
- **Full address range**. The DMA can access the full extended address range implemented on the device. The DMA access extends to the following:
  - On-chip memory
  - On-chip peripherals
  - External memory (selected devices). Access to certain areas of the memory map may be restricted depending on the device. For more information, see Chapter 3, *Memory*, in the the user guide titled *TMS320C54x DSP, CPU and Peripherals, Reference Set Volume 1* (literature number SPRU131).
- **Programmable width transfers**. Each channel can be independently configured to transfer in single-word mode (16-bit), or double-word mode (32-bit). See section 3.2.3.3**,** *DMA Sync Event and Frame Count Registers*, on page 3-13.
- **Autoinitialization**. Once a block transfer is complete, a DMA channel may automatically re-initialize itself for the next block transfer. See section 3.2.3.4, *Transfer Mode Control Register,* on page 3-17.
- **Event synchronization**. Each element transfer may be initiated by selected events. See section 3.2.3.3, *DMA Sync Event and Frame Count Registers*, on page 3-13.
- **Interrupt generation**. On completion of each frame transfer or block transfer, each DMA channel can send an interrupt to the CPU. See section 3.2.2.2, *Multiplexed Interrupt Control*, on page 3-9.

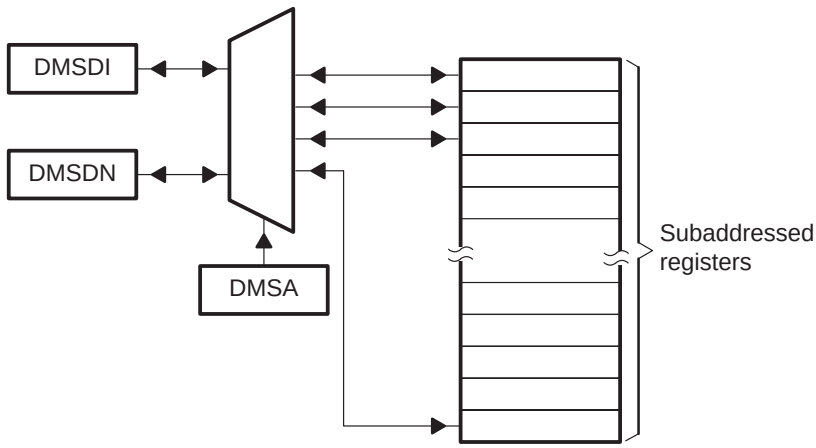
3.2 DMA Operation and Configuration

The DMA configuration and operation is achieved by a set of memory-mapped control registers. This scheme is similar to that used by other ’54x device peripherals. The DMA registers are memory-mapped using a register subaddressing scheme.

3.2.1 Register Subaddressing

Register subaddressing involves multiplexing a set of registers to a single location in the memory map. A subbank addressing register is used to control the multiplexer. A subbank data register is used to actually read or write data to the desired subaddressed register. A visual representation of this arrangement is shown in Figure 3–1. In this way, a large number of registers can be mapped into a small memory space.

Figure 3–1. Register Subaddressing



To access a specific subaddressed register, first the desired subaddress is written to the subbank address register (DMSA). This directs the multiplexer to connect the subbank access registers (DMSDI and DMSDN) to the desired physical location. When an access (read or write) is made to the subbank data registers, the data is moved to or from the desired embedded data register as previously specified in the subbank address register.

The DMA controller provides two types of subbank access registers. The first, DMSDI, is a subbank access register with automatic increment of the subaddress. This means that the subaddress will be incremented automatically after each access. If the entire set of DMA configuration registers needs to be configured, DMSDI provides a convenient means to set the subbank address once and then postincrement it after each access until the entire set of registers has been configured. If a single register access is desired, or if there is a need to prevent modification of the subaddress, DMSDN is used. DMSDN provides access to the subaddressed register without modifying the subaddress. In this way, the entire register set used to program the DMA is contained in four memory-mapped register locations.

Examples of the use of subaddressing are shown in Example 3–1 and Example 3–2.

Example 3–1. Using Register Subaddressing Without Autoincrement

The following code sample illustrates how the DMA channel 5 source address register is initialized with the value 1000h.

```
DMSA      .set   55h       ;subbank address register location
DMSDN     .set   57h
DMSRC5    .set   19h
STM       DMSRC5,DMSA      ;initialize the subbank address
                              ;register to point to DMSRC5
STM       #1000h,DMSDN     ;write 1000h to DMSRC5
```

Example 3–2. Using Register Subaddressing With Autoincrement

The following code sample illustrates how DMA channel 5 context registers can be initialized. The values written to the registers in this example are arbitrary and do not represent a particular configuration.

```
DMSA      .set   55h          ;set register locations
DMSDI     .set   56h
DMSRC5    .set   19h
DMDST5    .set   1Ah
DMCTR5    .set   1Bh
DMSFC5    .set   1Ch
DMMCR5    .set   1Dh
STM       DMSRC5,DMSA         ;initialize the subbank address
                              ;register to point to DMSRC5
STM       #1000h,DMSDI        ;write 1000h to DMSRC5
```

Example 3–2 Using Register Subaddressing With Autoincrement (Continued)

```
STM     #2000h,DMSDI    ;write 2000h to DMDST5
STM     #0010h,DMSDI    ;write 10h to DMCTR5
STM     #0002h,DMSDI    ;write 2h to DMSFC5
STM     #0000h,DMSDI    ;write 0h to DMMCR5
```

The DMA registers are shown in Table 3–1. Only the channel priority and enable control register (DMPREC) is directly addressed. All other DMA registers are subaddressed.

Table 3–1. DMA Registers

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

Table 3–1. DMA Registers (Continued)

Address	SubAddress	Name	Function
—	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
—	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

3.2.2 DMA Channel Priority and Enable Control Register

The channel priority and enable control (DMPREC) register controls several functions of the overall operation of the DMA system including:

- ❏ Selective enabling of each of the DMA channels
- ❏ Control of the multiplexed interrupts
- ❏ Control of channel priorities

The structure of DMPREC is shown in Figure 3–2. DMPREC is located at address 0054h in data space and is not subaddressed. Following reset, DMPREC is initialized to 0000h.

Figure 3–2. DMA Channel Priority and Enable Control (DMPREC) Register

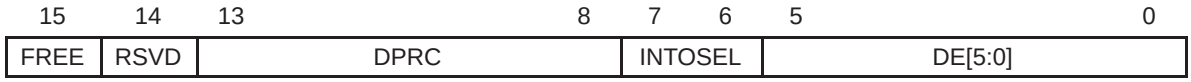


Table 3–2. DMA Channel Priority and Enable Control (DMPREC) Register Bit/Field Descriptions

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
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 (refer to Table 3–3, Table 3–4, and Table 3–5.)
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

Table 3–2. DMA Channel Priority and Enable Control (DMPREC) Register Bit/Field Descriptions (Continued)

Bit	Name	Reset Value	Function
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

3.2.2.1 DMA Channel Enable Control

Each of the six DMA channels can be independently enabled through the DE field in the DMPREC. Bits 0–5 in this register correspond to each of the six DMA channels, with bit 0 representing channel 0, bit 1 representing channel 1, etc. A logic 1 enables the associated channel, and a logic 0 disables the channel.

Each enable bit can be reset by the DMA controller upon completion of a block transfer. DMPREC can be polled to determine when a block transfer on a given channel is complete. If the DMA controller and the CPU both attempt to change the state of the DE field bits, the DMA controller has priority.

3.2.2.2 Multiplexed Interrupt Control

DMA events can trigger interrupts to the CPU upon completion of a transfer. Due to a limited number of interrupts in the ’54x memory map, some DMA interrupts are multiplexed with other peripheral interrupts on the device. The interrupt functions available are controlled by the INTOSEL field (bits 7 and 6) in DMPREC. The INTOSEL field can configure up to 6 interrupts to be assigned to DMA interrupts, depending on the device being used.

The DMA interrupt assignments available on each device are shown in Table 3–3, Table 3–4, and Table 3–5. Following reset, the interrupts are configured as in the 00b column in each table.

Table 3–3. Multiplexed Interrupt Assignments for the ’5402

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

Table 3–4. Multiplexed Interrupt Assignments for the ’5410

Interrupt Number (IMR/IFR #)	INTOSEL [1:0] Value 00b	01b	10b	11b
4	McBSP 0 RINT	McBSP 0 RINT	McBSP 0 RINT	Reserved
5	McBSP 0 XINT	McBSP 0 XINT	McBSP 0 XINT	Reserved
6	McBSP 2 RINT	McBSP 2 RINT	DMA Channel 0 Interrupt	Reserved
7	McBSP 2 XINT	McBSP 2 XINT	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
12	DMA Channel 4 Interrupt	DMA Channel 4 Interrupt	DMA Channel 4 Interrupt	Reserved
13	DMA Channel 5 Interrupt	DMA Channel 5 Interrupt	DMA Channel 5 Interrupt	Reserved

Table 3–5. Multiplexed Interrupt Assignments for the ’5420 (each subsystem)

Interrupt Number (IMR/IFR #)	INTOSEL [1:0] Value 00b	01b	10b	11b
4	McBSP 0 RINT	McBSP 0 RINT	McBSP 0 RINT	Reserved
5	McBSP 0 XINT	McBSP 0 XINT	McBSP 0 XINT	Reserved
6	McBSP 2 RINT	McBSP 2 RINT	DMA Channel 0 Interrupt	Reserved
7	McBSP 2 XINT	McBSP 2 XINT	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
12	DMA Channel 4 Interrupt	DMA Channel 4 Interrupt	DMA Channel 4 Interrupt	Reserved
13	DMA Channel 5 Interrupt	DMA Channel 5 Interrupt	DMA Channel 5 Interrupt	Reserved

3.2.2.3 DMA Channel Priority Control

Each DMA channel can be independently assigned low or high priority. All high priority DMA channels are serviced before any low priority channels are serviced. When multiple channels are enabled and assigned the same priority level, each of the enabled channels are serviced in a circular pattern.

Priority level for each channel is set in the 6-bit DPRC field of the DMPREC. Each bit position is associated with a DMA channel. Bits 8 through 13 are associated with channels 0 though 5, respectively. A logic 0 in the bit position assigns low priority to the associated channel. A logic 1 assigns high priority to the associated channel.

3.2.2.4 Emulation Control for DMA Functions

The FREE field of DMPREC controls the behavior of the DMA controller when the emulator is stopped. If FREE is cleared, DMA transfers are suspended when the emulator stops. If FREE is set, DMA transfers continue even if the emulator stops.

3.2.3 Channel-Context Registers

Each DMA channel has a set of 5 channel-context registers which configure the operation for only that channel. These five registers are:

- ❏ Source address register
- ❏ Destination address register
- ❏ Element count register
- ❏ Sync event / frame count register
- ❏ Transfer mode control register

The operation and effects of each of these registers is discussed in detail in the sections that follow.

3.2.3.1 Source and Destination Address Registers

Each DMA element transfer involves a read followed by a write. The address of the data to be read is called the source address, and the address where the data will be written is called the destination address. Two channel-context registers per channel store the source and destination addresses: the DMA channel *n* source address register (DMSRCn) and the DMA channel *n* destination address register (DMDSTn). Each is a 16-bit register and contains the least significant 16 bits of the extended address of the source or destination location.

The source and destination address registers are initialized prior to starting the DMA transfer in software, and updated automatically during transfers by the DMA controller. If necessary, the CPU can read either of these addresses during transfers to monitor the status of the transfer. These registers can also be written by the CPU during transfers. Changes to the address registers made during block transfers are effective immediately and will affect the progress of the transfer. Care should be taken to prevent unintentional writes to these registers during transfers.

The locations of the source and destination address registers for each channel are shown in Table 3–1.

3.2.3.2 Element Count Register

Each DMA Channel has a 16-bit element counter that keeps track of the number of DMA transfers to be performed. The element counter is initialized with the 16-bit unsigned number contained in the DMA channel element count register (DMCTRn) that represents the number of elements to be transferred. The element count register should be initialized with one less than the desired number of element transfers. For example, if DMA channel 2 is to move nine data-elements, DMCTR2 should be initialized to eight.

In normal multi-frame transfer mode, the DMCTRn is automatically decremented after each transfer by the DMA controller. When the last element in each frame is reached, the element counter is reloaded with the original contents of the DMCTRn, which has been stored in a shadow register.

In autobuffering (ABU) mode, the contents of the DMCTRn contains the buffer size. The DMCTRn is not decremented during transfers in this mode. ABU mode is included in the DMA controller to perform the same function as the autobuffering units of the ’54x buffered serial port (BSP). The McBSP, in conjunction with the DMA in ABU mode, can mimic the BSP ABU operation. ABU mode is discussed in more detail in section 3.2.3.5, *Addressing Modes*, on page 3-21.

If the CPU and the DMA controller attempt to modify the element count register at the same time, the CPU has priority.

The locations of the element count registers for each channel are shown in Table 3–1.

3.2.3.3 DMA Sync Event and Frame Count Registers

The DMA sync event and frame count register (DMSFCn) serves three purposes:

- ❏ Determines which synchronization events will be used to trigger DMA transfers
- ❏ Determines the transfer word size (16-bit or 32-bit)
- ❏ Determines the number of frames to be transferred

The structure of the DMSFCn is shown in Figure 3–3. The register locations of the DMA sync event and frame count registers for each channel are shown in Table 3–1.

Figure 3–3. DMA Sync Event and Frame Count (DMSFCn) Register

15 12	11	10 8	7 0
DSYN[3:0]	DBLW	rsvd	Frame Count

Table 3–6. DMA Sync Event and Frame Count (DMSFCn) Register Bit/Field Descriptions

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. See the *DMA Synchronization Events* section following this table for available sync events.
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. See the *Frame Count* section following Table 3–9 for more details.

DMA Synchronization Events

DMA element transfers can be initiated by a variety of events including McBSP receive or transmit events, timer interrupt events, and external interrupt events. A DMA channel can also run unsynchronized (not associated with any event). When a DMA channel is synchronized to a particular event, each element transfer waits for the occurrence of that event. If the DMA channel is not synchronized to any event, the transfers proceed as rapidly as possible.

If sync events are used, one sync event is required for each transfer. In single-word mode, one sync event initiates one 16-bit single-word transfer. In double-word mode, one sync event initiates one 32-bit double-word transfer (as two 16-bit transfers).

The types of sync events available depend on the ’54x device. Not all of the sync events mentioned above are available on all devices. Table 3–6, Table 3–7, Table 3–8, and Table 3–9 show the available sync events on each device. The sync event is chosen by loading the DSYN field of the DMSFCn with the appropriate value as shown in the Tables. Only one sync event can be chosen for each DMA channel. The DSYN field is set to 0000h (no sync event) on reset.

Table 3–7. DMA Sync Event Options on the '5402

DSYN [3:0] value	DMA Synchronization Mode
0000	No sync event (nonsynchronized operation)
0001	McBSP 0 receive event (REVT0)
0010	McBSP 0 transmit event (XEVT0)
0011	Reserved
0100	Reserved
0101	McBSP 1 receive event (REVT1)
0110	McBSP 1 transmit event (XEVT1)
0111	Reserved
1000	Reserved
1001	Reserved
1010	Reserved
1011	Reserved
1100	Reserved
1101	Timer 0 interrupt event
1110	External interrupt 3 (INT3) event
1111	Timer 1 interrupt event

Table 3–8. DMA Sync Event Options on the ’5410

DSYN [3:0] value	DMA Synchronization Mode
0000	No sync event (nonsynchronized operation)
0001	McBSP 0 receive event (REVT0)
0010	McBSP 0 transmit event (XEVT0)
0011	McBSP 2 receive event (REVT2)
0100	McBSP 2 transmit event (XEVT2)
0101	McBSP 1 receive event (REVT1)
0110	McBSP 1 transmit event (XEVT1)
0111	McBSP 0 receive event – ABIS mode (REVTA0)
1000	McBSP 0 transmit event – ABIS mode (XEVTA0)
1001	McBSP 2 receive event – ABIS mode (REVTA2)
1010	McBSP 2 transmit event – ABIS mode (XEVTA2)
1011	McBSP 1 receive event – ABIS mode (REVTA1)
1100	McBSP 1 transmit event – ABIS mode (XEVTA1)
1101	Timer interrupt event
1110	External interrupt 3 (INT3) event
1111	Reserved

Table 3–9. DMA Sync Event Options on the ’5420 (each subsystem)

DSYN [3:0] value	DMA Synchronization Mode
0000	No sync event (nonsynchronized operation)
0001	McBSP 0 receive event (REVT0)
0010	McBSP 0 transmit event (XEVT0)
0011	McBSP 2 receive event (REVT2)
0100	McBSP 2 transmit event (XEVT2)
0101	McBSP 1 receive event (REVT1)
0110	McBSP 1 transmit event (XEVT1)
0111	FIFO receive buffer not empty event
1000	FIFO transmit buffer not full event
1001	Reserved
1010	Reserved
1011	Reserved
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

Double-Word Mode

DMA transfers can be configured as single-word 16-bit transfers, or double-word 32-bit transfers. In double-word mode, each 32-bit transfer is considered one element. Double-word transfers are implemented as two back-to-back 16-bit transfers, and administered by the DMA controller (including appropriate address modification) without any intervention from the CPU. Double-word mode is available with any of the address indexing modes. For a detailed discussion of address indexing modes, see section 3.2.3.5, *Addressing Modes,* on page 3-21.

In double-word mode, the address of the second transfer is always the address of the first transfer with the LSB inverted. If the first transfer address is an even address, the second transfer will be to the next higher address. If the first transfer address is an odd address, the second transfer will be directed to the next lower address.

Double-word mode is configured by setting the DBLW bit in the DMSFCn. On reset, DBLW is cleared (single-word mode).

Frame Count

The frame count is an 8-bit field in the DMSFCn that specifies the number of frames to be included in a block transfer. The frame count should be initialized to one less than the desired number of frames. For example, if the desired number of frames to be transferred in a block is eight, the frame count field should be initialized to seven. If only one frame is desired, the frame count field should be set to zero. On reset, the frame count field is set to zero (single frame operation). The maximum number of frames is 256.

The frame count field is decremented by the DMA controller upon the completion of each frame. Once the last frame is transferred, the frame count can be reloaded with the contents of the DMA global frame count register (DMGFR), if autoinitialization mode is enabled. For more information on autoinitialization, see section 3.2.3.6, *Autoinitialization*, on page 3-27.

The element count and frame count can be used together to allow up to 65536 transfers. The total number of transfers is the product of the element count and the frame count.

If the CPU and the DMA controller attempt to modify the frame count field at the same time, the CPU has priority.

3.2.3.4 Transfer Mode Control Register

The DMA transfer mode control register (DMMCRn) is a 16-bit register that controls the transfer mode for the channel. The structure of the DMMCRn is shown in Figure 3–4, *DMA Transfer Mode Control (DMMCRn) Register*, on page 3-17. Function descriptions of the register fields are shown in Table 3–10 on page 3-17, and discussed in the sections that follow.

Figure 3–4. DMA Transfer Mode Control (DMMCRn) Register

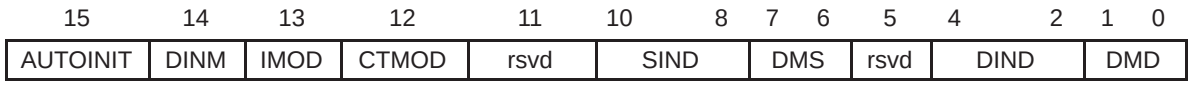


Table 3–10. DMA Transfer Mode Control (DMMCRn) Register Bit/Field Descriptions

Bit	Name	Reset Value	Function
15	AUTOINIT	0	DMA autoinitialization mode bit. AUTOINIT = 0 Autoinitialization is disabled AUTOINIT = 1 Autoinitialization is enabled

Table 3–10. DMA Transfer Mode Control (DMMCRn) Register Bit/Field Descriptions (Continued)

Bit	Name	Reset Value	Function
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.
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.

Table 3–10. DMA Transfer Mode Control (DMMCRn) Register Bit/Field Descriptions (Continued)

Bit	Name	Reset Value	Function
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

Source and Destination Address Selection and Modification

Two fields in the DMMCRn, DMS and DMD, control the selection of the source and destination address spaces for DMA transfers. DMS specifies whether the source data comes from program space, data space, or I/O space. DMD specifies whether the destination data goes to program space, data space, or I/O space. Bit configurations for each of these modes is shown in Table 3–10.

In addition, the DMMCRn controls how addresses are modified for the source and destination. The SIND field controls how the source addresses are indexed as transfers proceed. The DIND field controls how destination addresses are indexed as transfers proceed. The following options are available for address modification:

- ❏ No modification (address remains constant for each transfer)
- ❏ Postincrement by 1
- ❏ Post-decrement by 1
- ❏ Postincrement by the offset value contained in the element index register 0 (DMIDX0)
- ❏ Postincrement by the offset value contained in the element index register 1 (DMIDX1)
- ❏ Postincrement by the offset values contained in the element index register 0 (DMIDX0) and the frame index register 0 (DMFRI0)
- ❏ Postincrement by the offset values contained in the element index register 0 (DMIDX1) and the frame index register 0 (DMFRI1)

The latter four modes utilize the element and frame index registers. The element index registers (DMIDX0 and DMIDX1) are used to index the source and destination addresses during transfers. The index contained in these registers is used to modify the source or destination address following the transfer of each element. DMIDX0 and DMIDX1 are not associated with either the source or destination address. Either register can be used to index the source, destination, or both.

The frame index registers (DMFRI0 and DMFRI1) are used to index the source and destination addresses following completion of blocks (or frames) of element transfers. When both element and frame indexes are used, the address is modified by the element index after each transfer, and then modified by the frame index at the end of each frame. This capability can be used to implement circular buffers and sorting functions (see the related sections that follow). DMFRI0 and DMFRI1 are not associated with either the source or destination address. Either register can be used to index the source, destination, or both.

DMMCRn is a channel-context register, so each channel can be configured separately. DMIDX0, DMIDX1, DMFRI0 and DMFRI1 are not channel-context registers. These registers configure index options for the entire DMA system.

3.2.3.5 Addressing Modes

The CTMOD field in DMMCRn controls the operation of the DMA transfer counter for each channel. In multiframe mode, the element and frame indexes are used to modify the source and destination addresses following each transfer. This mode is convenient for transferring data formatted as frames or blocks. ABU mode is used to implement autobuffering functions. In this mode, the element counter for each channel (DMCTRn) represents the buffer size and is not modified during the transfers. Although the McBSP is mentioned here as an example of a source/destination, any source/destination can be used.

Multiframe Mode

In multiframe mode, the data transfers can be structured as multiple elements in a frame, and multiple frames in a block. The element index registers (DMIDX0, DMIDX1) are used to modify the addresses of elements within a frame, and the frame-index registers (DMFRI0, DMFRI1) are used to modify addresses following the completion of frames. The number of elements transferred per frame is determined by the channel element count (DMCTRn) register and the number of frames transferred per block is determined by the channel-frame count (frame count field of the DMSFCn register). The two counters are decremented following each element or frame transfer, respectively.

The element count is an unsigned 16-bit integer. The element count register should be initialized with one less than the desired number of elements to be transferred. The number of elements to be transferred per frame can be between 1 (0000h) and 65536 (FFFFh).

The frame count is an unsigned 8-bit integer. The frame count register should be initialized with one less than the desired number of frames to be transferred. The number of frames to be transferred per block can be between 1 (00h) and 256 (FFh). The total number of elements to be transferred is called the *block size*, and is the product of the frame count and the element count.

The element counter is decremented following each transfer. If all elements in a frame have been transferred, the element counter is reloaded with the original value (via a shadow register) and the frame count is decremented. If the last element in the last frame has been transferred, both the element counter and the frame counter will contain 0000h and will remain unmodified, unless the autoinitialization mode is enabled (AUTOINIT = 1). For more information on the autoinitialization mode, see section 3.2.3.6, on page 3-27.

All of the address indexing modes mentioned in the *Source and Destination Address Selection and Modification* section are valid in multiframe mode, and

the indexing methods can be applied to the source and destination addresses separately. The following describes the operation of each of the addressing modes during multiframe operation:

- **No modify.** The address is not modified following transfers, and maintains its initialized value.
- **Postincrement.** The address is incremented by 1 following each transfer. The address is incremented by 2 if double-word operation is selected.
- **Post-decrement.** The address is decremented by 1 following each transfer. The address is decremented by 2 if double-word operation is selected.
- **Postincrement With Index Offsets (DMIDX0 or DMIDX1).** The address is modified by the amount in the referenced element index register. The element index register contains a 16-bit signed (2s complement value). Following a transfer, the index value is added (or subtracted) from the current address to generate the next address. In double-word mode, the address is modified by twice the value of the index.
- **Postincrement With Index Offset (DMIDX0 and DMFRI0, or DMIDX1 and DMFRI1), or Sorting Mode.** If the current transfer is not the last element in the frame, the address is modified by the contents of the referenced element index register. If the current element is the last element in the frame, the address in modified by the contents of the frame index register. The frame-index registers contain a 16-bit signed value. This mode is useful for sorting data such as the information in a T1 frame.

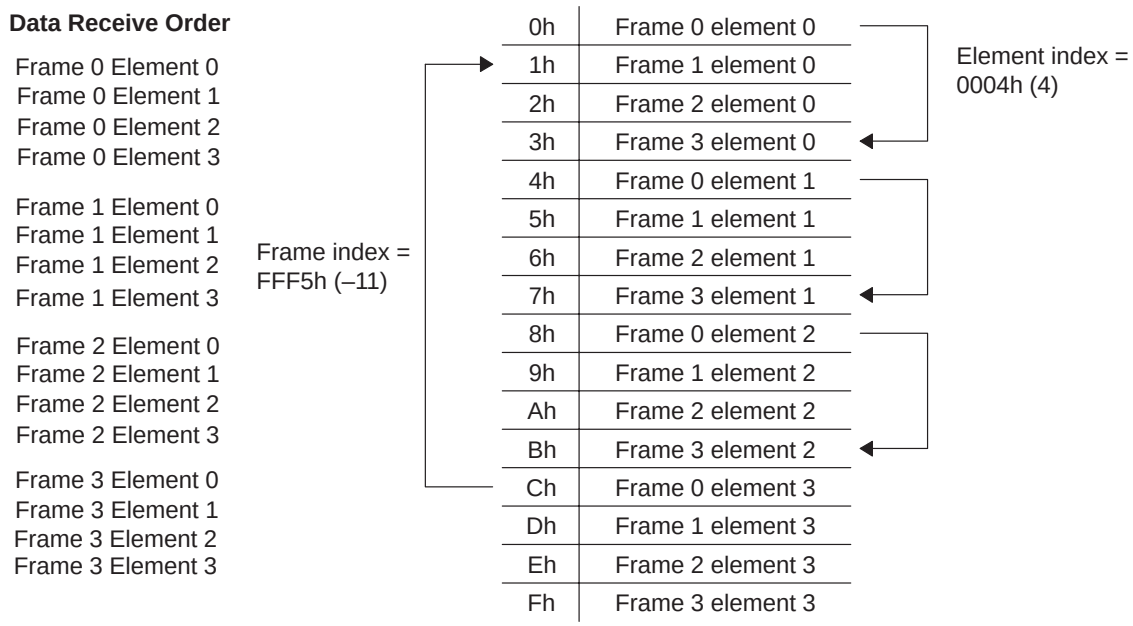
Example 3–3. Data Sorting by Address Modification

For this example, assume a data stream where the data is structured as four elements per frame, and four frames per block. The source data arrives in the order shown on the left in Figure 3–5. It may be convenient to store or process the data sorted by element number instead of frame number. This is achieved by selecting sorting mode for the address modification, setting the destination element index register to increment by four after each transfer, and setting the elements and frame index to decrement by 11 after each frame. When the entire block of 16 transfers has been completed, the data will be sorted by element number instead of frame number as shown in Figure 3–5.

To configure the element index to increment by four, the signed value 0004h is stored in the element index register. To configure the frame index to decrement by 11, the signed value 0FFF5h (-11 decimal) is stored in the frame index register.

The source and destination addresses are not updated after the last transfer in the block. They contain the last source and destination address. The element and frame counters both contain 0000h after the last transfer. The configuration of the DMA registers required to implement this example is shown in Example 3–21, *McBSP to Data Memory Transfer with Data Sorting*, on page 3-56.

Figure 3–5. Data Sorting Example



ABU Mode

ABU, or autobuffering mode, provides automatically controlled circular buffer capability for DMA transfers. ABU can be used to duplicate the autobuffering function implemented on the buffered serial port (BSP) using the McBSP in conjunction with the DMA controller.

In this mode, either the source or destination is configured in autobuffering mode, and the other location is specified as an unmodified address. For example, a circular buffer for McBSP receive data requires the DMA to be configured with the data receive register of the McBSP as the source location, and the circular buffer controlled by the DMA as the destination location.

During ABU mode operation, the element count register contains the buffer size. The frame count register does not have a function in ABU mode. When the address reaches the end of the buffer, it wraps back to the beginning automatically. The number of transfers is not specified in this mode, so the address wraps indefinitely until the DMA channel is disabled. The element count register is interpreted as a 16-bit unsigned integer and valid buffer sizes range from 0002h to 0FFFFh. The buffer can be any size in this range and is not limited to powers of two.

The buffer has a minimum address, called the base address, and a maximum address. The difference between the base address and the maximum address is the buffer size. Although the buffer size is not limited to powers of two, the base address must be based on powers of two. The required location depends on the buffer size. The base address must be located on an address boundary that corresponds to one power-of-two higher than the most significant bit position of the buffer size. The address boundaries for all available buffer sizes are shown in Table 3–11. Circular buffers cannot cross 64k address boundaries. Don't care values in the table are indicated by X.

Table 3–11. ABU Buffer Examples

ABU Buffer Size [hexadecimal (decimal)]		Buffer Base Address [binary]				
0002h–0003h	(2–3)	XXXX	XXXX	XXXX	XX00	b
0004h–0007h	(4–7)	XXXX	XXXX	XXXX	X000	b
0008h–000Fh	(8–15)	XXXX	XXXX	XXXX	0000	b
0010h–001Fh	(16–31)	XXXX	XXXX	XXX0	0000	b
0020h–003Fh	(32–63)	XXXX	XXXX	XX00	0000	b
0040h–007Fh	(64–127)	XXXX	XXXX	X000	0000	b
0080h–00FFh	(128–255)	XXXX	XXXX	0000	0000	b
0100h–01FFh	(256–511)	XXXX	XXX0	0000	0000	b
0200h–03FFh	(512–1023)	XXXX	XX00	0000	0000	b
0400h–07FFh	(1024–2047)	XXXX	X000	0000	0000	b
0800h–0FFFh	(2048–4095)	XXXX	0000	0000	0000	b
1000h–1FFFh	(4096–8191)	XXX0	0000	0000	0000	b
2000h–3FFFh	(8192–16383)	XX00	0000	0000	0000	b
4000h–7FFFh	(16384–32767)	X000	0000	0000	0000	b
8000h–7FFFh	(32768–65535)	0000	0000	0000	0000	b

Example 3–4. ABU Buffer Size Examples

- For a buffer size of eight (decimal), the next higher power of two is 16, so the buffer base address must be aligned with 16 word boundaries (ex. 0000h, 0010h, 0020h).
- For a buffer size of 5 (decimal), the next higher power of two is 8, so the buffer base address must be aligned with 8 word boundaries (ex. 0000h, 0008h, 0010h, 0018h, 0020h).
- For a buffer size of 200 (decimal), the next higher power of two is 256, so the buffer base address must be aligned with 256 word boundaries (ex. 0000h, 0100h, 0200h, 0300h).

The DMA transfer always starts at the addresses specified in the channel source or destination address registers. The address register can point to any location inside the defined range of the buffer. After each access, the appropriate address register is modified according to the specified address indexing mode. A limited set of address indexing modes are available for ABU mode operation. Table 3–12 shows the modes available for ABU operation.

To configure the ABU mode properly, one side of the transfer (the source or destination, but not both) must be configured with no address modification (SIND/DIND=000b). The other side of the transfer must be configured with a modified addressing mode. All legal combinations of SIND and DIND for ABU mode are shown in Table 3–12. Use of combinations other than those shown may cause unpredictable behavior.

Table 3–12. ABU Address Indexing Modes

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)		

As transfers proceed, the address in the circular buffer can be incremented or decremented by one, or modified by the contents of one of the element index registers (DMIDX0 or DMIDX1). If the DMA channel is configured for double-word mode, the address is modified either by two, or by twice the index-offset value. If the calculated next address for the buffer is greater than the maximum buffer size, the address wraps to the base address. If a greater-than-one index is used, the appropriate offset from the base address is calculated.

You should note that in double-word mode, the address of the second transfer is always the address of the first transfer with the LSB inverted, regardless of whether the maximum address has been crossed. The buffer wrap address is always calculated with regard to the first transfer address of the double-word transfer.

Example 3–5 shows the buffer wrap address calculated for a single-word transfer with indexed addressing. Example 3–6 shows the buffer wrap address calculated for a double-word transfer with indexed addressing.

Example 3–5. Wrap Address Calculation for a Single-Word Transfer With Indexed Addressing

Buffer location:	0580h–059Eh
Current Address:	059Eh
Index:	2
Next Address:	0581h

Example 3–6. Wrap Address Calculation for a Double-Word Transfer With Indexed Addressing

Buffer location:	0580h–059Eh
Current Address:	059Eh
Index:	2 (a total of + 4 since double-word mode)
Next Address:	0583h

Negative indexes can also be specified. In this case, the base address and the maximum address calculation remains the same as described above. The address is modified according to the negative address index. When the base address is crossed, the address wraps to the maximum address. If an index of less than -1 is being used, the appropriate offset from the maximum address is calculated.

In ABU mode, interrupts can be configured to occur when the buffer is half full or full. A detailed explanation of the interrupt operation is provided in section 3.2.3.7, *Interrupt Generation*, on page 3-27.

3.2.3.6 Autoinitialization

The AUTOINIT field of the DMMCRn controls the autoinitialization capability for each DMA channel. When autoinitialization is enabled (AUTOINIT = 1), the channel-context registers are re-initialized upon the completion of a block transfer. Autoinitialization mode eliminates the need for the CPU to intervene when block transfers are completed. When autoinitialization occurs, the following channel-context registers are modified:

- DMSRCn is loaded with the contents of DMGSA, the global source address register. DMGSA contains a 16-bit source address.
- DMDSTn is loaded with the contents of DMGDA, the global destination address register. DMGDA contains a 16-bit destination address.
- DMCTRn is loaded with the contents of DMGCR, the global element count register. DMGCR contains a 16-bit unsigned element count value.
- The frame count field of the DMSFCn is loaded with the contents of DMGFR, the global frame count register. DMGFR contains an 8-bit unsigned frame count value. The upper 8 bits of the DMGFR are reserved and have no effect. The reserved bits are always read as zeros.

Autoinitialization is only available in multiframe mode (CTMOD = 0).

3.2.3.7 Interrupt Generation

Two fields in the DMMCRn (DINM and IMOD) control how interrupts are handled during DMA transfers. DINM is used to enable or disable a generation of interrupts based on the completion of part or all of a transfer. If DINM = 0, interrupts are disabled and no interrupt generation occurs. If DINM = 1, interrupt generation is enabled and occurs based on the configuration of the IMOD bit. The interrupt mask register (IMR) and the INTM bit in the CPU still control whether an interrupt from the DMA is serviced. DINM enables the ability of the DMA to generate an interrupt. For more information on operation of the IMR, INTM, and interrupt processing, see the user's guide titled *TMS320C54x DSP, CPU and Peripherals Reference Set, Volume 1* (literature number SPRU131).

If interrupt generation is enabled (DINM=1), then the IMOD field in DMMCRn controls when the interrupt is generated within a block transfer. The available operation modes are shown below in Table 3–13.

Table 3–13. DMA Block Transfer Interrupt Generation Modes

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

In multiframe mode, interrupts can be generated either at the end of each frame (and the end of each block of frames), or only at the end of the entire block. The end of a frame occurs when the frame count equals zero. The end of a block occurs when the element count and the frame count both equal zero.

In ABU mode (CTMOD = 1), interrupts can be generated either when the entire buffer has been transferred or when each half of the buffer has been transferred. The half-buffer option can be used to emulate the operation of the buffered serial port (BSP) on the ’548 and ’549 devices.

The logic that generates the interrupt is based on the calculated next address. The determination of the half-buffer point depends on the ’54x DSP being used. The half-buffer interrupt generation for each device is described in detail in the sections that follow.

3.2.3.8 ’5402/’5420 Half-Buffer Interrupt Generation in ABU Mode

On the ’5402 and ’5420, the half-buffer interrupt is generated when the next address is equal to or greater than the halfway point in the buffer (if positive indexing is used), or when the next address is less than the halfway point when negative indexing is being used. The interrupt point for odd and even size buffers differ accordingly. The full-buffer interrupt is generated when the next address wraps back to the base address (positive indexing), or back to the maximum address (negative indexing). The examples that follow illustrate when the interrupts are generated in each case.

Example 3–7. ’5402/’5420 ABU Interrupt Example – Even Size Buffer With +1 Index

Buffer size = 8 Half-buffer = 4

Current Address	Next Address	Interrupt
0000h	0001h	
0001h	0002h	
0002h	0003h	
0003h	0004h	Interrupt generated
0004h	0005h	
0005h	0006h	
0006h	0007h	
0007h	0000h	Interrupt generated (wrap)

Example 3–8. ’5402/’5420 ABU Interrupt Example – Odd Size Buffer With +1 Index

Buffer size = 7 Half-buffer = 3

Current Address	Next Address	Interrupt
0000h	0001h	
0001h	0002h	
0002h	0003h	Interrupt generated
0003h	0004h	
0004h	0005h	
0005h	0006h	
0006h	0000h	Interrupt generated (wrap)

Example 3–9. ’5402/’5420 ABU Interrupt Example – Even Size Buffer With –1 Index

Buffer size = 8 Half-buffer = 4

Current Address	Next Address	Interrupt
0007h	0006h	
0006h	0005h	
0005h	0004h	
0004h	0003h	Interrupt generated
0003h	0002h	
0002h	0001h	
0001h	0000h	
0000h	0007h	Interrupt generated (wrap)

Example 3–10. ’5402/’5420 ABU Interrupt Example – Odd Size Buffer With –1 Index

Buffer size = 7 Half-buffer = 3

Current Address	Next Address	Interrupt
0006h	0005h	
0005h	0004h	
0004h	0003h	
0003h	0002h	Interrupt generated
0002h	0001h	
0001h	0000h	
0000h	0006h	Interrupt generated (wrap)

3.2.3.9 ’5410 Half-Buffer Interrupt Generation in ABU Mode

On the ’5410, the half-buffer interrupt is generated when the next address is greater than the halfway point in the buffer if positive indexing is used, or when the next address is less than the halfway point when negative indexing is used. For odd and even size buffers, the interrupt point differs accordingly. The full-buffer interrupt is generated when the next address wraps back to the base address (positive indexing), or back to the maximum address (negative indexing). The examples that follow illustrate when interrupts are generated in each case.

Example 3–11. ’5410 ABU Interrupt Example – Even Size Buffer With +1 Index

Buffer size = 8 Half-buffer = 4

Current Address	Next Address	Interrupt
0000h	0001h	
0001h	0002h	
0002h	0003h	
0003h	0004h	
0004h	0005h	Interrupt generated
0005h	0006h	
0006h	0007h	
0007h	0000h	Interrupt generated (wrap)

Example 3–12. ’5410 ABU Interrupt Example – Odd Size Buffer With +1 Index

Buffer size = 7 Half-buffer = 3

Current Address	Next Address	Interrupt
0000h	0001h	
0001h	0002h	
0002h	0003h	
0003h	0004h	Interrupt generated
0004h	0005h	
0005h	0006h	
0006h	0000h	Interrupt generated (wrap)

Example 3–13. ’5410 ABU Interrupt Example – Even Size Buffer With –1 Index

Buffer size = 8 Half-buffer = 4

Current Address	Next Address	Interrupt
0007h	0006h	
0006h	0005h	
0005h	0004h	
0004h	0003h	Interrupt generated
0003h	0002h	
0002h	0001h	
0001h	0000h	
0000h	0007h	Interrupt generated (wrap)

Example 3–14. ’5410 ABU Interrupt Example – Odd Size Buffer With –1 Index

Buffer size = 7 Half-buffer = 3

Current Address6	Next Address	Interrupt
0006h	0005h	
0005h	0004h	
0004h	0003h	
0003h	0002h	Interrupt generated
0002h	0001h	
0001h	0000h	
0000h	0006h	Interrupt generated (wrap)

3.3 Extended Addressing

The DMA controller has the ability to perform transfers to and from the extended program memory space. Two subaddressed registers are employed to provide this capability: the DMA source program page address register (DMSRCP) and the DMA destination program page address register (DMDSTP). These registers contain the extended program page number for the source and destination locations, respectively. In each of these registers, the least significant seven bits are used to store the extended address page of the source and destination addresses, as shown in Figure 3–6 and Figure 3–7. Following reset, DMSRCP and DMDSTP are initialized to 0000h, or program page zero. The reserved bit locations are always read as zeros.

DMSRCP and DMDSTP are not channel specific. These registers control the source and destination address pages in program memory for all DMA channels. In addition, the program page addresses stored in these registers are not modified during source and destination address modifications. Consequently, program space transfers cannot cross 64K page boundaries. If a program page boundary is crossed during a transfer, the next transfer wraps to the same page.

Not all ’54x devices support 128 pages of extended program memory. You should consult the device-specific data sheet for detailed information on extended memory support.

Figure 3–6. DMA Source Program Page Address Register (DMSRCP)

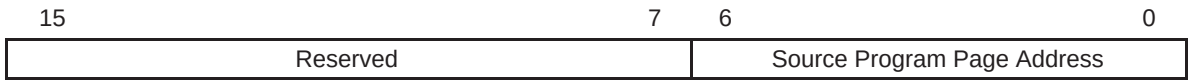
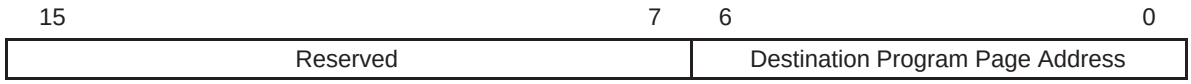


Figure 3–7. DMA Destination Program Page Address Register (DMDSTP)



3.4 DMA Memory Maps

This section provides DMA memory maps for the ’5402, ’5410, and ’5420 devices.

3.4.1 ’5402 DMA Memory Map

The ’5402 DMA supports internal accesses only. The DMA memory map for the ’5402 is shown in Table 3–14. Memory regions marked as reserved (shaded) are not accessible through the ’5402 DMA. The DMA memory map is not affected by the MP/$\overline{\text{MC}}$, DROM, or OVLY bits.

Table 3–14. ’5402 DMA Memory Map

	Address Range (Hex)		Description
Program space	00 0000	0F FFFF	Reserved
Data space	0000	001F	Reserved
	0020		McBSP0 data receive register (DRR20)
	0021		McBSP0 data receive register (DRR10)
	0022		McBSP0 data transmit register (DXR20)
	0023		McBSP0 data transmit register (DXR10)
	0024	003F	Reserved
	0040		McBSP1 data receive register (DRR21)
	0041		McBSP1 data receive register (DRR11)
	0042		McBSP1 data transmit register (DXR21)
	0043		McBSP1 data transmit register (DXR11)
	0044	005F	Reserved
	0060	007F	Scratchpad RAM
	0080	3FFF	DARAM
	4000	FFFF	Reserved
I/O space	0000	FFFF	Reserved

3.4.2 ’5410 DMA Memory Map

The ’5410 supports DMA accesses to internal and external memory. The DMA memory map for the ’5410 is shown in Table 3–15. Memory regions marked as reserved (shaded) are not accessible through the ’5410 DMA. The DMA memory map is not affected by the MP/$\overline{\text{MC}}$, DROM, or OVLY bits.

Table 3–15. ’5410 DMA Memory Map

	Address Range (Hex)		Description
Program space	00 0000	00 007F	Reserved
	00 0080	00 BFFF	External memory
	00 C000	00 FFFF	On-chip ROM
	10 0000	01 7FFF	External memory
	01 8000	01 FFFF	On-chip SARAM2
	02 0000	7F FFFF	External memory
Data space	0000	001F	Reserved
	0020		McBSP0 data receive register (DRR20)
	0021		McBSP0 data receive register (DRR10)
	0022		McBSP0 data transmit register (DXR20)
	0023		McBSP0 data transmit register (DXR10)
	0024	002F	Reserved
	0030		McBSP2 data receive register (DRR22)
	0031		McBSP2 data receive register (DRR12)
	0032		McBSP2 data transmit register (DXR22)
	0033		McBSP2 data transmit register (DXR12)
	0034	0035	Reserved
	0036		RCERA2
	0037		XCERA2
	0038	0039	Reserved
	003A		RCERA0
	003B		XCERA0

Table 3–15. ’5410 DMA Memory Map (Continued)

	Address Range (Hex)		Description
	003C	003F	Reserved
	0040		McBSP1 data receive register (DRR21)
	0041		McBSP1 data receive register (DRR11)
	0042		McBSP1 data transmit register (DXR21)
	0043		McBSP1 data transmit register (DXR11)
	0044	0049	Reserved
	004A		RCERA1
	004B		XCERA1
	004C	005F	Reserved
	0060	7FFF	On-chip RAM
	8000	FFFF	External memory
I/O space	0000	FFFF	External memory

3.4.3 ’5420 DMA Memory Map

The ’5420 DMA supports internal accesses only. The DMA memory map for the ’5420 is shown in Table 3–16. Memory regions marked as reserved (shaded) are not accessible through the ’5420 DMA. The entire I/O space on both subsystems is devoted to the interprocessor FIFO; consequently, the ’5420 DMA does not have access to the external I/O ports. The DMA memory map is not affected by the MP/$\overline{\text{MC}}$, DROM, or OVLY bits.

Table 3–16. ’5420 DMA Memory Map

	Address Range (Hex)		Description
Program space	00 0000	00 001F	Reserved
	00 0020		McBSP0 data receive register (DRR20)
	00 0021		McBSP0 data receive register (DRR10)
	00 0022		McBSP0 data transmit register (DXR20)
	00 0023		McBSP0 data transmit register (DXR10)
	00 0024	00 002F	Reserved
	00 0030		McBSP2 data receive register (DRR22)
	00 0031		McBSP2 data receive register (DRR12)
	00 0032		McBSP2 data transmit register (DXR22)
	00 0033		McBSP2 data transmit register (DXR12)
	00 0034	00 003F	Reserved
	00 0040		McBSP1 data receive register (DRR21)
	00 0041		McBSP1 data receive register (DRR11)
	00 0042		McBSP1 data transmit register (DXR21)
	00 0043		McBSP1 data transmit register (DXR11)
	00 0044	00 005F	Reserved
	00 0060	00 3FFF	On-chip DARAM 0
	00 4000	00 7FFF	On-chip SARAM 1
	00 8000	00 FFFF	On-chip SARAM 2
	01 0000	01 005F	Reserved
	01 0060	01 3FFF	On-chip DARAM 0
	01 4000	01 7FFF	On-chip SARAM 1
	01 8000	01 FFFF	On-chip SARAM 3
	02 0000	02 005F	Reserved
	02 0060	02 3FFF	On-chip DARAM 0
	02 4000	02 7FFF	On-chip SARAM 1
	02 8000	02 EFFF	Reserved
	02 F000	02 FFFF	On-chip SARAM 4

Table 3–16. ’5420 DMA Memory Map (Continued)

	Address Range (Hex)		Description
	03 0000	03 005F	Reserved
	03 0060	03 3FFF	On-chip DARAM 0
	03 4000	03 7FFF	On-chip SARAM 1
	03 8000	03 FFFF	Reserved
Data space	0000	001F	Reserved
			McBSP0 data receive register (DRR20)
	0021		McBSP0 data receive register (DRR10)
	0022		McBSP0 data transmit register (DXR20)
	0023		McBSP0 data transmit register (DXR10)
	0024	002F	Reserved
	0030		McBSP2 data receive register (DRR22)
	0031		McBSP2 data receive register (DRR12)
	0032		McBSP2 data transmit register (DXR22)
	0033		McBSP2 data transmit register (DXR12)
	0034	003F	Reserved
	0040		McBSP1 data receive register (DRR21)
	0041		McBSP1 data receive register (DRR11)
	0042		McBSP1 data transmit register (DXR21)
	0043		McBSP1 data transmit register (DXR11)
	0044	005F	Reserved
	0060	007F	On-chip scratchpad RAM
	0080	3FFF	On-chip DARAM 0
	4000	7FFF	On-chip SARAM 1
	8000	FFFF	On-chip SARAM 2
I/O space	0000	FFFF	DMA FIFO for interprocessor communications

3.5 DMA Transfer Latency

All 16-bit DMA transfers are composed of a read followed by a write. The time to complete this activity depends on the source and destination locations, external interface conditions (such as wait states and bank-switching cycles), the number of active DMA channels, and the activity level of the host port interface (HPIx).

A single DMA transfer from an internal (on-chip) source location to an internal destination location requires four CPU clock cycles. The read portion of the transfer takes two cycles and the write portion of the transfer takes two cycles. If either the read, the write, or both attempt to access an external (off-chip) location, the total transfer time is extended by the number of cycles necessary to complete the access on that interface. This time determines the maximum transfer rate for transfers from/to a given source/destination. All ’54x devices support internal-to-internal transfers of four cycles. Therefore, the maximum transfer rate for a channel is the CPU clock rate divided by four. For example, a processor operating with a CPU clock rate of 100 MHz can support a maximum of 25M 16-bit transfers per second (25 Mwords per second), or 12.5M double-word (32-bit) transfers per second. A summary of maximum DMA transfer cycle times is shown in Table 3–17. Accesses to on-chip memory or memory-mapped registers are both considered internal. For devices that do not support external accesses through the DMA, the table entries are marked not applicable (N/A).

Table 3–17. DMA Transfer Cycle Times

DMA Source Location	DMA Destination Location	’5402 Transfer Cycles	’5410 Transfer Cycles	’5420 Transfer Cycles
Internal	Internal	4	4	4
Internal	External	N/A	5	N/A
External	Internal	N/A	5	N/A
External	External	N/A	6	N/A

On transfers that involve an external access, the number of cycles required to complete the transfer is affected by conditions on the external interface including software and hardware wait states, additional cycles due to bank-switching, and the speed of the interface if CLKOUT division is available. For example, on the ’5410, CLKOUT (and the external parallel interface) can be configured to operate as slow as one-fourth the speed of the CPU clock. In this case, external DMA transfers also slow down accordingly. The conditions present in the actual application should be considered when estimating the maximum DMA transfer rate involving external accesses. The ’5410 column in

Table 3–17 assumes the external interface is configured with no hardware or software wait states and CLKOUT is in divide-by-1 mode (CLKOUT is equal to CPU clock).

The transfer rate on a particular channel is affected by the activity on the other channels. Since all high-priority channels are serviced first in a circular pattern, the data rate on a particular channel is governed by the number of other channels that are active at the same priority level. So while the high-priority channels are still active, the maximum data rate on each high-priority channel is divided by the total number of high-priority channels. The transfer rate on the low-priority channels is zero until there is no high priority activity. Example 3–15 illustrates such a situation.

Example 3–15. DMA Channel Transfer Rate Example

Assume the ’54x processor is running at 100 MHz CPU clock. Assume the following DMA channel configuration:

- ❏ DMA channel 1 is configured for 100 high priority internal-to-internal single-word transfers.
- ❏ DMA channel 2 is configured for 200 high priority internal-to-internal single-word transfers.
- ❏ DMA channel 3 is configured for 50 low priority internal-to-internal single-word transfers.
- ❏ DMA channel 4 is configured for 75 low priority internal-to-internal single-word transfers.
- ❏ DMA channels 5 and 6 are disabled and HPI-8/-16 is inactive.

Assuming channels 1, 2, 3, and 4 are enabled at the same time and none of the channels wait for sync events, channels 1 and 2 are serviced first and channels 3 and 4 wait until channels 1 and 2 are completed. The transfer rates on each of the channels proceed as follows.

While channels 1 and 2 are still active:

- ❏ DMA channel 1 data transfer rate = 25MWps/2 active channels = 12.5 MWps
- ❏ DMA channel 2 data transfer rate = 25MWps/2 active channels = 12.5 MWps
- ❏ DMA channel 3 data transfer rate = 0 due to priority
- ❏ DMA channel 4 data transfer rate = 0 due to priority

When channel 1 is completed but channel 2 is still active:

- ❏ DMA channel 1 data transfer rate = 0 (completed)
- ❏ DMA channel 2 data transfer rate = 25MWps /1 active channel = 25 MWps
- ❏ DMA channel 3 data transfer rate = 0 due to priority
- ❏ DMA channel 4 data transfer rate = 0 due to priority

When channels 1 and 2 are completed, channels 3 and 4 become active:

- ❏ DMA channel 1 data transfer rate = 0 (completed)
- ❏ DMA channel 2 data transfer rate = 0 (completed)
- ❏ DMA channel 3 data transfer rate = 25MWps/2 active channels = 12.5 MWps
- ❏ DMA channel 4 data transfer rate = 25MWps/2 active channels = 12.5 MWps

When channels 1, 2, and 3 are completed but channel 4 is still active:

- ❏ DMA channel 1 data transfer rate = 0 (completed)
- ❏ DMA channel 2 data transfer rate = 0 (completed)
- ❏ DMA channel 3 data transfer rate = 0 (completed)
- ❏ DMA channel 4 data transfer rate = 25MWps/1 active channels = 25 MWps

3.6 Enhanced Host Port Interface Access Through the DMA Controller

The enhanced host port interface (HPI-8 or HPI-16, depending on the DSP) uses the DMA bus to gain access to on-chip memory. The HPI has a dedicated port on the DMA controller and makes requests to use the DMA bus. The DMA controller arbitrates these requests based on the current activity. When the HPIx makes a request, the current DMA transfer in progress is completed and the DMA controller grants the HPIx access to the DMA bus. All pending high- and low-priority transfers are suspended until the HPIx releases the bus. When the HPI releases the bus, the pending DMA transfers continue as before. For this reason, HPIx activity affects the transfer rates on the DMA channels.

Double-word transfers are not interrupted by an HPIx request. Both 16-bit transfers are completed before the DMA controller grants access to the HPIx.

Although the HPIx has a dedicated port on the DMA controller, it does not have a dedicated channel. No channel context programming is required for the HPIx to use the DMA bus. The HPIx is operated normally by the host, while access to the internal buses is arbitrated by the DMA controller. Due to this arrangement, use of the HPIx does not consume any of the available DMA channels.

For more information on the operation of the enhanced host port interfaces (HPI-8 and HPI-16), refer to Chapter 4, *Enhanced 8-Bit Host Port Interface (HPI-8)*, and Chapter 5, *Enhanced 16-Bit Host Port Interface (HPI-16).*

3.7 Interprocessor FIFO Communication on the ’5420

The ’5420 has two CPU subsystems that are independent of each other, and can operate separately with complete functionality.

One method of exchanging data between the two subsystems uses the DMA controller in conjunction with an on-chip first-in, first-out (FIFO) unit. The FIFO is mapped to the DMA I/O space. Any access to DMA I/O space on the ’5420 writes to, or reads from, this FIFO. All addresses are mapped to the same FIFO location. For this reason, the ’5420 DMA does not access I/O space in the same manner as the other ’54x devices.

For more information on ’5420 subsystem communication methods, see Chapter 6, *Interprocessor Communications.*

3.8 DMA Operation in Power-Down Mode

The ’54x devices offer several power-down modes that allow part or all of the device to enter a dormant state and dissipate less power than when running normally. Power-down modes can be invoked in several ways, including executing the IDLE instruction, and driving the HOLD input low with the HM status bit set to one. The DMA, like the other peripherals, can take the CPU out of IDLE using interrupts.

The DMA continues to operate while in all IDLE modes (IDLE1/2/3). When the DMA is not running (not performing a transfer), the DMA controller automatically stops its internal clocks to conserve power. When a transfer needs to be performed, the clocks are switched on, the transfer is completed, and the clocks are switched off again. This clock management operates in each of the three IDLE modes.

In IDLE1 and IDLE2, the clock source to the DMA comes from the same source used to generate the CPU and system clocks (either the PLL or a divided version of the X2/CLKIN). In IDLE3, the PLL is stopped, and therefore, cannot be used as a clock source for the DMA. In this case, X2/CLKIN is used directly to clock the DMA controller. For this reason, a clock must always be present at the X2/CLKIN pin for the DMA to operate in IDLE 3 mode.

When X2/CLKIN is used to clock the DMA in IDLE 3 mode, it is not multiplied or divided. As a result, the speed of DMA transfers in IDLE 3 mode is affected accordingly.

3.9 Programming Examples

Below are several DMA controller programming examples. Each example contains a description of the desired operation and the code used to set up the DMA controller. Only the configuration code is included – additional code such as interrupt service routines are not included. For clarity, the code sections refer to DMA register names. The addresses for these registers can be identified in assembly code as shown on the facing page.

```
************************************************
*
*  54x Register Definitions for the DMA Controller
*
************************************************
DMPREC .set 0054h   ;Channel Priority and Enable Control Register
DMSA   .set 0055h   ;Sub-bank Address Register
DMSDI  .set 0056h   ;Sub-bank Data Register with autoincrement
DMSDN  .set 0057h   ;Sub-bank Data Register without modification
DMSRC0 .set 00h     ;Channel 0 Source Address Register
DMDST0 .set 01h     ;Channel 0 Destination Address Register
DMCTR0 .set 02h     ;Channel 0 Element Count Register
DMSFC0 .set 03h     ;Channel 0 Sync Select and Frame Count Register
DMMCR0 .set 04h     ;Channel 0 Transfer Mode Control Register
DMSRC1 .set 05h     ;Channel 1 Source Address Register
DMDST1 .set 06h     ;Channel 1 Destination Address Register
DMCTR1 .set 07h     ;Channel 1 Element Count Register
DMSFC1 .set 08h     ;Channel 1 Sync Select and Frame Count Register
DMMCR1 .set 09h     ;Channel 1 Transfer Mode Control Register
DMSRC2 .set 0Ah     ;Channel 2 Source Address Register
DMDST2 .set 0Bh     ;Channel 2 Destination Address Register
DMCTR2 .set 0Ch     ;Channel 2 Element Count Register
DMSFC2 .set 0Dh     ;Channel 2 Sync Select and Frame Count Register
DMMCR2 .set 0Eh     ;Channel 2 Transfer Mode Control Register
DMSRC3 .set 0Fh     ;Channel 3 Source Address Register
DMDST3 .set 10h     ;Channel 3 Destination Address Register
DMCTR3 .set 11h     ;Channel 3 Element Count Register
DMSFC3 .set 12h     ;Channel 3 Sync Select and Frame Count Register
DMMCR3 .set 13h     ;Channel 3 Transfer Mode Control Register
DMSRC4 .set 14h     ;Channel 4 Source Address Register
DMDST4 .set 15h     ;Channel 4 Destination Address Register
DMCTR4 .set 16h     ;Channel 4 Element Count Register
DMSFC4 .set 17h     ;Channel 4 Sync Select and Frame Count Register
DMMCR4 .set 18h     ;Channel 4 Transfer Mode Control Register
DMSRC5 .set 19h     ;Channel 5 Source Address Register
DMDST5 .set 1Ah     ;Channel 5 Destination Address Register
DMCTR5 .set 1Bh     ;Channel 5 Element Count Register
DMSFC5 .set 1Ch     ;Channel 5 Sync Select and Frame Count Register
DMMCR5 .set 1Dh     ;Channel 5 Transfer Mode Control Register
DMSRCP .set 1Eh     ;Source Program Page Address
DMDSTP .set 1Fh     ;Destination Program Page Address
DMIDX0 .set 20h     ;Element Address Index Register 0
DMIDX1 .set 21h     ;Element Address Index Register 1
DMFRI0 .set 22h     ;Frame Address Index Register 0
DMFRI1 .set 23h     ;Frame Address Index Register 1
DMGSA  .set 24h     ;Global Source Address Reload Register
DMGDA  .set 25h     ;Global Destination Address Reload Register
DMGCR  .set 26h     ;Global Element Count Reload Register
DMGFR  .set 27h     ;Global Frame Count Reload Register
```

Example 3–16. Program Memory to Data Memory Transfer Without Autoincremented Subaddressing

This example transfers a block of data from program space to data space via direct memory access. The DMA controller is configured to perform 1000h single-word transfers with both the source and destination addresses incremented by one after each element transfer. The transfers are not associated with any sync event (free-running). After the block transfer is complete (autoinitialization is off), the DMA channel is disabled.

Transfer mode:	Multi-frame mode
Source address:	18000h in program space
Destination address:	03000h in data space
Transfer size:	1000h single (16-bit) words
Sync event:	None (free-running)
Channel use:	DMA channel #0

```
*********************************
       stm    DMSRCP,DMSA         ;set source program page to 1
       stm    #1h,DMSDN
       stm    DMSRC0,DMSA         ;set source program address to 8000
       stm    #8000h,DMSDN        ; (lower 16-bit of 18000h)
       stm    DMDST0,DMSA         ;set destination address to 3000
       stm    #3000h,DMSDN
       stm    DMCTR0 ,DMSA        ;set for 1000h transfers
       stm    #(1000h-1) ,DMSDN
       stm    DMSFC0 ,DMSA
       stm    #0000000000000000b ,DMSDN
              ;0000~~~~~~~~~~~~ (DSYN)         No sync event
              ;~~~~0~~~~~~~~~~~ (DBLW)         Single-word mode
              ;~~~~~000~~~~~~~~                Reserved
              ;~~~~~~~~00000000 (Frame Count)  Frame Count=0h (one frame)
       stm    DMMCR0 ,DMSA
       stm    #0000000100000101b ,DMSDN
              ;0~~~~~~~~~~~~~~~ (AUTOINIT)  Autoinitialization disabled
              ;~0~~~~~~~~~~~~~~ (DINM)   Interrupts masked
              ;~~0~~~~~~~~~~~~~ (IMOD)   N/A
              ;~~~0~~~~~~~~~~~~ (CTMOD)  Multi-frame mode
              ;~~~~0~~~~~~~~~~~          Reserved
              ;~~~~~001~~~~~~~~ (SIND)   Post increment source address
              ;~~~~~~~~00~~~~~~ (DMS)    Source in program space
              ;~~~~~~~~~~0~~~~~          Reserved
              ;~~~~~~~~~~~001~~ (DIND)   Post increment destination address
              ;~~~~~~~~~~~~~~01 (DMD)    Destination in data space

       stm    #0000000100000001b ,DMPREC
              ;0~~~~~~~~~~~~~~~ (FREE)    DMA stops on emulation stop
              ;~0~~~~~~~~~~~~~~           Reserved
              ;~~0~~~~~~~~~~~~~ (DPRC[5]) Channel 5 low  priority
              ;~~~0~~~~~~~~~~~~ (DPRC[4]) Channel 4 low  priority
              ;~~~~0~~~~~~~~~~~ (DPRC[3]) Channel 3 low  priority
              ;~~~~~0~~~~~~~~~~ (DPRC[2]) Channel 2 low  priority
              ;~~~~~~0~~~~~~~~~ (DPRC[1]) Channel 1 low  priority
              ;~~~~~~~1~~~~~~~~ (DPRC[0]) Channel 0 high priority
              ;~~~~~~~~00~~~~~~ (INTOSEL) N/A
              ;~~~~~~~~~~0~~~~~ (DE[5])   Channel 5 disabled
              ;~~~~~~~~~~~0~~~~ (DE[4])   Channel 4 disabled
              ;~~~~~~~~~~~~0~~~ (DE[3])   Channel 3 disabled
              ;~~~~~~~~~~~~~0~~ (DE[2])   Channel 2 disabled
              ;~~~~~~~~~~~~~~0~ (DE[1])   Channel 1 disabled
              ;~~~~~~~~~~~~~~~1 (DE[0])   Channel 0 enabled
*********************************
```

Example 3–17. Program Memory to Data Memory Transfer Using Autoincremented Subaddressing

This example performs the same transfer as the previous example but uses autoincremented subaddressing (the DSMBAI register) during the setup. This subaddressing scheme allows the channel-context registers to be configured using a single instruction instead of two, saving time and memory.

Transfer mode:	Multi-frame mode
Source address:	18000h in program space
Destination address:	03000h in data space
Transfer size:	1000h single (16-bit) words
Sync event:	None (free-running)
Channel use:	DMA channel #0

```
**********************************
       stm    DMSRCP,DMSA          ;set source program page to 1
       stm    #1h,DMSDN

       stm    DMSRC0,DMSA          ;set source program address to 8000 and
       stm    #8000h,DMSDI         ; point DMSA to next sub-address (DMDST0)

       stm    #3000h,DMSDI         ;set destination address to 3000 and
                                   ; point DMSA to next sub-address (DMCTR0)

       stm    #(1000h–1),DMSDI     ;set for 1000h transfers and point
                                   ; DMSA to next sub-address (DMSFC0)

       stm    #00000h,DMSDI        ;configure DMSFC0 and point DMSA
                                   ; to next sub-address (DMMCR0)

       stm    #00105h,DMSDI        ;configure DMMCR0 and point DMSA
                                   ; to next sub-address (DMSRC0)

       stm    #00101h,DMPREC       ;configure DMPREC

**********************************
```

Example 3–18. Program Memory to Data Memory Transfer With Autoinitialization

This example transfers a block of data from program space to data space via direct memory access. The DMA controller is configured to perform 1000h single-word transfers per block, with both the source and destination addresses incremented by one after each element transfer. The transfers are not associated with any sync event (free-running). When the first block transfer is completed, the DMA channel is autoinitialized to begin again using the contents of the global reload registers (DMGSA, DMGDA, DMGCR, DMGFR). The global reload registers direct the destination to 02000h in data space, instead of 03000h, on the first block transfer. The global reload registers can also be used to autoinitialize the identical conditions as the first block transfer.

Transfer mode:	Multi-frame mode
Initial source address:	18000h in program space
Initial destination address:	03000h in data space
Initial transfer size:	1000h single (16-bit) words (1 frame, 1000h elements)
Autoinitialize source address:	18000h in program space
Autoinitialize destination address:	02000h in data space
Autoinitialize element count:	1000h single (16-bit) words
Autoinitialize frame count:	000h (1 frame)
Sync event:	None (free-running)
Channel use:	DMA channel #4

```
**********************************
       stm    DMSRCP,DMSA          ;set source program page to 1
       stm    #1h,DMSDN
       stm    DMSRC4,DMSA          ;set source program address to 18000h
       stm    #8000h,DMSDN
       stm    DMDST4,DMSA          ;set destination address to 3000h
       stm    #3000h,DMSDN
       stm    DMCTR4 ,DMSA         ;set for 1000h transfers
       stm    #(1000h-1) ,DMSDN
       stm    DMSFC4 ,DMSA
       stm    #0000000000000000b ,DMSDN
              ;0000~~~~~~~~~~~~ (DSYN)          No sync event
              ;~~~~0~~~~~~~~~~~ (DBLW)          Single-word mode
              ;~~~~~000~~~~~~~~                 Reserved
              ;~~~~~~~~00000000 (Frame Count)   Frame Count = 0h (one frame)
```

```
       stm    DMMCR4 ,DMSA
       stm    #1000000100000101b ,DMSDN
              ;1~~~~~~~~~~~~~~~ (AUTOINIT)  Autoinitialization enabled
              ;~0~~~~~~~~~~~~~~ (DINM)   Interrupts masked
              ;~~0~~~~~~~~~~~~~ (IMOD)   N/A
              ;~~~0~~~~~~~~~~~~ (CTMOD)  Multi-frame mode
              ;~~~~0~~~~~~~~~~             Reserved
              ;~~~~~001~~~~~~~~ (SIND)   Post increment source address
              ;~~~~~~~~00~~~~~~ (DMS)    Source in program space
              ;~~~~~~~~~~0~~~~~            Reserved
              ;~~~~~~~~~~~001~~ (DIND)   Post increment destination address
              ;~~~~~~~~~~~~~~01 (DMD)    Destination in data space

       stm    DMGSA,DMSA          ;set global source address to 8000h
       stm    #8000h,DMSDN        ; (lower 16-bits of 18000h in program space)
       stm    DMGDA,DMSA          ;set global destination address to 2000h
       stm    #2000h,DMSDN
       stm    DMGCR,DMSA          ;set global element count to move 1000h words
       stm    #(1000h–1),DMSDN
       stm    DMGFR,DMSA          ;set global frame count to 0h

       stm    #0000000000000000b,DMSDN
              ;00000000~~~~~~~~  Reserved
              ;~~~~~~~~00000000  Global Frame count

       stm    #0001000000010000b ,DMPREC
              ;0~~~~~~~~~~~~~~~ (FREE)    DMA stops on emulation stop
              ;~0~~~~~~~~~~~~~~            Reserved
              ;~~0~~~~~~~~~~~~~ (DPRC[5]) Channel 5 low  priority
              ;~~~1~~~~~~~~~~~~ (DPRC[4]) Channel 4 high priority
              ;~~~~0~~~~~~~~~~~ (DPRC[3]) Channel 3 low  priority
              ;~~~~~0~~~~~~~~~~ (DPRC[2]) Channel 2 low  priority
              ;~~~~~~0~~~~~~~~~ (DPRC[1]) Channel 1 low  priority
              ;~~~~~~~0~~~~~~~~ (DPRC[0]) Channel 0 low  priority
              ;~~~~~~~~00~~~~~~ (INTOSEL) N/A
              ;~~~~~~~~~~0~~~~~ (DE[5])   Channel 5 disabled
              ;~~~~~~~~~~~1~~~~ (DE[4])   Channel 4 enabled
              ;~~~~~~~~~~~~0~~~ (DE[3])   Channel 3 disabled
              ;~~~~~~~~~~~~~0~~ (DE[2])   Channel 2 disabled
              ;~~~~~~~~~~~~~~0~ (DE[1])   Channel 1 disabled
              ;~~~~~~~~~~~~~~~0 (DE[0])   Channel 0 disabled
************************************
```

Example 3–19. McBSP Data Transfer in ABU Mode

This example transfers 16-bit words received on McBSP0 to data space via the DMA controller. The destination is configured as a circular buffer. The address in the destination buffer is incremented by one after each transfer. ABU mode is used to implement the circular buffer. In ABU mode, the element count register represents the buffer size, and the frame count is not used. Only the DMA configuration is shown, which in this example, is to generate an interrupt to the CPU when the 100h buffer is full. The interrupt service routine for this interrupt is not shown in this example.

Transfer mode:	ABU (non-decrement) mode
Source address:	McBSP0 data receive register (DRR10)
Destination buffer:	03000h – 030FFh in data space
Buffer size:	100h single (16-bit) words
Sync event:	McBSP0 receive event
Channel use:	DMA channel #1

```
*********************************
       stm    DMSRC1,DMSA         ;set source address to DRR10
       stm    DRR1_0,DMSDN
       stm    DMDST1,DMSA         ;set destination address to 3000
       stm    #3000h,DMSDN
       stm    DMCTR1 ,DMSA        ;set buffer size to 100h words
       stm    #100h ,DMSDN
       stm    DMSFC1 ,DMSA
       stm    #0001000000000000b ,DMSDN
              ;0001~~~~~~~~~~~ (DSYN)         McBSP0 receive sync event
              ;~~~~0~~~~~~~~~~ (DBLW)         Single-word mode
              ;~~~~~000~~~~~~~~               Reserved
              ;~~~~~~~~00000000 (Frame Count)  Frame count is not
              ;                                relevant in ABU mode

       stm    DMMCR1 ,DMSA
       stm    #0101000001001101b ,DMSDN
              ;0~~~~~~~~~~~~~~ (AUTOINIT)  Autoinitialization disabled
              ;~1~~~~~~~~~~~~~ (DINM)   DMA Interrupts enabled
              ;~~0~~~~~~~~~~~~ (IMOD)   Interrupt at full buffer
              ;~~~1~~~~~~~~~~~ (CTMOD)  ABU (non-decrement) mode
              ;~~~~0~~~~~~~~~~          Reserved
              ;~~~~~000~~~~~~~~ (SIND)   No modify on source address (DRR10)
              ;~~~~~~~~01~~~~~~ (DMS)    Source in data space
              ;~~~~~~~~~~0~~~~~          Reserved
              ;~~~~~~~~~~~011~~ (DIND)   Post increment destination address
              ;                           with DMIDX0
              ;~~~~~~~~~~~~~~01 (DMD)    Destination in data space

       stm    DMIDX0,DMSA                ;set element address index to +1
       stm    #0001h,DMSDN

       stm    #0000001000000010b ,DMPREC
              ;0~~~~~~~~~~~~~~~ (FREE)    DMA stops on emulation stop
              ;~0~~~~~~~~~~~~~~ Reserved
              ;~~0~~~~~~~~~~~~~ (DPRC[5]) Channel 5 low  priority
              ;~~~0~~~~~~~~~~~~ (DPRC[4]) Channel 4 low  priority
              ;~~~~0~~~~~~~~~~~ (DPRC[3]) Channel 3 low  priority
              ;~~~~~0~~~~~~~~~~ (DPRC[2]) Channel 2 low  priority
              ;~~~~~~1~~~~~~~~~ (DPRC[1]) Channel 1 high priority
              ;~~~~~~~0~~~~~~~~ (DPRC[0]) Channel 0 low  priority
              ;~~~~~~~~00~~~~~~ (INTOSEL) N/A
              ;~~~~~~~~~~0~~~~~ (DE[5])   Channel 5 disabled
              ;~~~~~~~~~~~0~~~~ (DE[4])   Channel 4 disabled
              ;~~~~~~~~~~~~0~~~ (DE[3])   Channel 3 disabled
              ;~~~~~~~~~~~~~0~~ (DE[2])   Channel 2 disabled
              ;~~~~~~~~~~~~~~1~ (DE[1])   Channel 1 enabled
              ;~~~~~~~~~~~~~~~0 (DE[0])   Channel 0 disabled
*********************************
```

Example 3–20. McBSP Data Transfer in Double-Word Mode

This example transfers 32-bit words received on McBSP0 to data space via direct memory access in multi-frame mode. The DMA controller is configured to automatically read DRR20 and DRR10 and transfer the data to sequential locations starting at 03000h in data space.

Transfer mode:	Multi-frame mode
Source address:	McBSP0 data receive registers (DRR20/DRR10)
Destination address:	03000h in data space
Transfer size:	50h double (32-bit) words
Sync event:	McBSP0 receive event
Channel use:	DMA channel #1

```
*********************************
       stm    DMSRC1,DMSA          ;set source address to DRR20
       stm    DRR2_0,DMSDN
       stm    DMDST1,DMSA          ;set destination address to 3000
       stm    #3000h,DMSDN
       stm    DMCTR1 ,DMSA         ;set buffer size to 50h words
       stm    #050h-1,DMSDN

       stm    DMSFC1 ,DMSA
       stm    #0001100000000000b ,DMSDN
              ;0001~~~~~~~~~~~~ (DSYN)          McBSP0 receive sync event
              ;~~~~1~~~~~~~~~~~ (DBLW)          Double-word mode
              ;~~~~~000~~~~~~~~                 Reserved
              ;~~~~~~~~00000000 (Frame Count)   Single Frame

       stm    DMMCR1 ,DMSA
       stm    #0100000001001101b ,DMSDN
              ;0~~~~~~~~~~~~~~~ (AUTOINIT)  Autoinitialization disabled
              ;~1~~~~~~~~~~~~~~ (DINM)   DMA Interrupts enabled
              ;~~0~~~~~~~~~~~~~ (IMOD)   Interrupt at complete block transfer
              ;~~~0~~~~~~~~~~~~ (CTMOD)  Multi-frame mode
              ;~~~~0~~~~~~~~~~~           Reserved
              ;~~~~~000~~~~~~~~ (SIND)   No modify on source address (DRR20)
              ;~~~~~~~~01~~~~~~ (DMS)    Source in data space
              ;~~~~~~~~~~0~~~~~           Reserved
              ;~~~~~~~~~~~011~~ (DIND)   Post increment destination address
              ;                           with DMIDX0
              ;~~~~~~~~~~~~~~01 (DMD)    Destination in data space
       stm    DMIDX0,DMSA                ;set element address index to +1
       stm    #0001h,DMSDN

       stm    #0000001000000010b ,DMPREC
              ;0~~~~~~~~~~~~~~~ (FREE)    DMA stops on emulation stop
              ;~0~~~~~~~~~~~~~~ Reserved
              ;~~0~~~~~~~~~~~~~ (DPRC[5]) Channel 5 low  priority
              ;~~~0~~~~~~~~~~~~ (DPRC[4]) Channel 4 low  priority
              ;~~~~0~~~~~~~~~~~ (DPRC[3]) Channel 3 low  priority
              ;~~~~~0~~~~~~~~~~ (DPRC[2]) Channel 2 low  priority
              ;~~~~~~1~~~~~~~~~ (DPRC[1]) Channel 1 high priority
              ;~~~~~~~0~~~~~~~~ (DPRC[0]) Channel 0 low  priority
              ;~~~~~~~~00~~~~~~ (INTOSEL) N/A
              ;~~~~~~~~~~0~~~~~ (DE[5])   Channel 5 disabled
              ;~~~~~~~~~~~0~~~~ (DE[4])   Channel 4 disabled
              ;~~~~~~~~~~~~0~~~ (DE[3])   Channel 3 disabled
              ;~~~~~~~~~~~~~0~~ (DE[2])   Channel 2 disabled
              ;~~~~~~~~~~~~~~1~ (DE[1])   Channel 1 enabled
              ;~~~~~~~~~~~~~~~0 (DE[0])   Channel 0 disabled
*********************************
```

Example 3–21. McBSP to Data Memory Transfer With Data Sorting

This example implements the sorting example shown in Figure 3–5 on page 3-23. Single words received on the McBSP0 are transferred to data memory starting at address 03000h. The incoming data is structured as four frames of four elements each. The DMA sorts the incoming data by modifying the destination addresses with the element index register (DMIDX0) and the frame index register (DMFRI0). The resulting stored data is sorted by element number instead of frame number.

Transfer mode:	Multi-frame mode
Source address:	McBSP0 data receive register (DRR10)
Destination address:	03000h in data space
Transfer size:	10h single (16-bit) words
Sync event:	McBSP0 receive event
Channel use:	DMA channel #1

```
*********************************
       stm    DMSRC1,DMSA         ;set source address to DRR10
       stm    DRR1_0,DMSDN
       stm    DMDST1,DMSA         ;set destination address to 3000
       stm    #3000h,DMSDN
       stm    DMCTR1 ,DMSA        ;set elements per frame to 4h
       stm    #0004h–1 ,DMSDN
       stm    DMSFC1 ,DMSA
       stm    #0001000000000011b ,DMSDN
              ;0001~~~~~~~~~~~ (DSYN)         McBSP0 receive sync event
              ;~~~~0~~~~~~~~~~ (DBLW)         Single-word mode
              ;~~~~~000~~~~~~~~               Reserved
              ;~~~~~~~~00000011 (Frame Count)  Frames per block = 4h–1
       stm    DMMCR1 ,DMSA
       stm    #0100000001010101b ,DMSDN
              ;0~~~~~~~~~~~~~~~ (AUTOINIT)  Autoinitialization disabled
              ;~1~~~~~~~~~~~~~~ (DINM)   DMA Interrupts enabled
              ;~~0~~~~~~~~~~~~~ (IMOD)   Interrupt at complete block transfer
              ;~~~0~~~~~~~~~~~~ (CTMOD)  Multi-frame mode
              ;~~~~0~~~~~~~~~~            Reserved
              ;~~~~~000~~~~~~~~ (SIND)   No modify on source address (DRR10)
              ;~~~~~~~~01~~~~~~ (DMS)    Source in data space
              ;~~~~~~~~~~0~~~~~           Reserved
              ;~~~~~~~~~~~101~~ (DIND)   Post increment destination address
              ;                           with DMIDX0 and DMFRI0
              ;~~~~~~~~~~~~~~01 (DMD)    Destination in data space
       stm    DMIDX0,DMSA          ;set element address index to +4
       stm    #0004h,DMSDN

       stm    DMFRI0,DMSA         ;set frame address index to –11
       stm    #0FFF5h,DMSDN

       stm    #0000001000000010b ,DMPREC
              ;0~~~~~~~~~~~~~~~ (FREE)    DMA stops on emulation stop
              ;~0~~~~~~~~~~~~~~ Reserved
              ;~~0~~~~~~~~~~~~~ (DPRC[5]) Channel 5 low  priority
              ;~~~0~~~~~~~~~~~~ (DPRC[4]) Channel 4 low  priority
              ;~~~~0~~~~~~~~~~~ (DPRC[3]) Channel 3 low  priority
              ;~~~~~0~~~~~~~~~~ (DPRC[2]) Channel 2 low  priority
              ;~~~~~~1~~~~~~~~~ (DPRC[1]) Channel 1 high priority
              ;~~~~~~~0~~~~~~~~ (DPRC[0]) Channel 0 low  priority
              ;~~~~~~~~00~~~~~~ (INTOSEL) N/A
              ;~~~~~~~~~~0~~~~~ (DE[5])   Channel 5 disabled
              ;~~~~~~~~~~~0~~~~ (DE[4])   Channel 4 disabled
              ;~~~~~~~~~~~~0~~~ (DE[3])   Channel 3 disabled
              ;~~~~~~~~~~~~~0~~ (DE[2])   Channel 2 disabled
              ;~~~~~~~~~~~~~~1~ (DE[1])   Channel 1 enabled
              ;~~~~~~~~~~~~~~~0 (DE[0])   Channel 0 disabled

*********************************
```

Chapter 4

Enhanced 8-Bit Host Port Interface (HPI-8)

The enhanced 8-bit host port interface, also referred to as HPI-8, is an improved version of the standard 8-bit HPI designed to interface a host device or host processor to the ’54x.

The HPI-8 is available on selected devices in the ’54x family of TI DSPs.

Topic **Page**

4.1 Introduction to the Enhanced 8-Bit Host Port Interface (HPI-8)

The HPI-8 is an 8-bit parallel port that interfaces a host device or host processor to the ’54x. Information is exchanged between the ’54x and the host device through on-chip ’54x RAM.

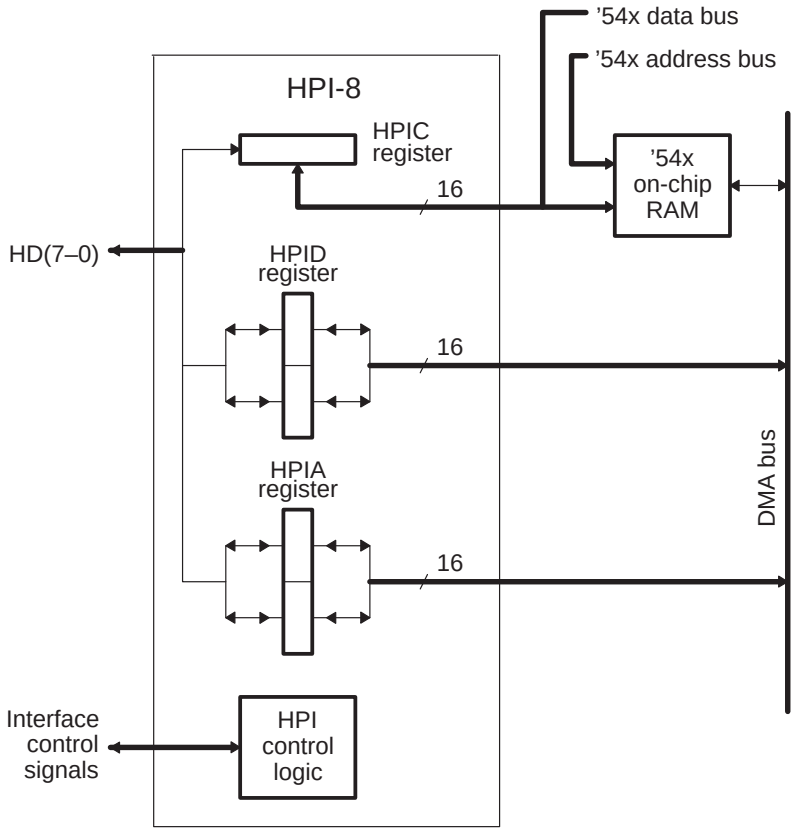
The main differences between the enhanced 8-bit HPI and the standard 8-bit HPI are listed in Table 4–1.

Table 4–1. Main Differences Between Enhanced 8-Bit HPI and Standard 8-Bit HPI

Enhanced 8-bit HPI (HPI-8)	Standard 8-bit HPI
❑ Access to all on-chip RAM locations	Access to only a fixed 2K portion of on-chip RAM
❑ Host accesses always synchronized to the’54x clock (no host-only mode)	Host-only mode allows asynchronous host accesses
❑ Both host and ’54x always have access to on-chip RAM (no host-only mode)	Host-only mode gives host exclusive access to RAM

The HPI-8 functions as a slave and enables the host processor to access the on-chip memory of the ’54x. The interface consists of an 8-bit, bidirectional data bus and various control signals. Sixteen bit transfers are accomplished in two parts with the HBIL input designating high or low byte. The host communicates with the HPI-8 through dedicated address and data registers, which the ’54x cannot directly access. The HPI control register, which is accessible by both the host and the ’54x, includes bits for configuring the protocol and for controlling communication (handshaking). A simple block diagram of the HPI-8 is shown in Figure 4–1.

Figure 4–1. Host Port Interface Block Diagram



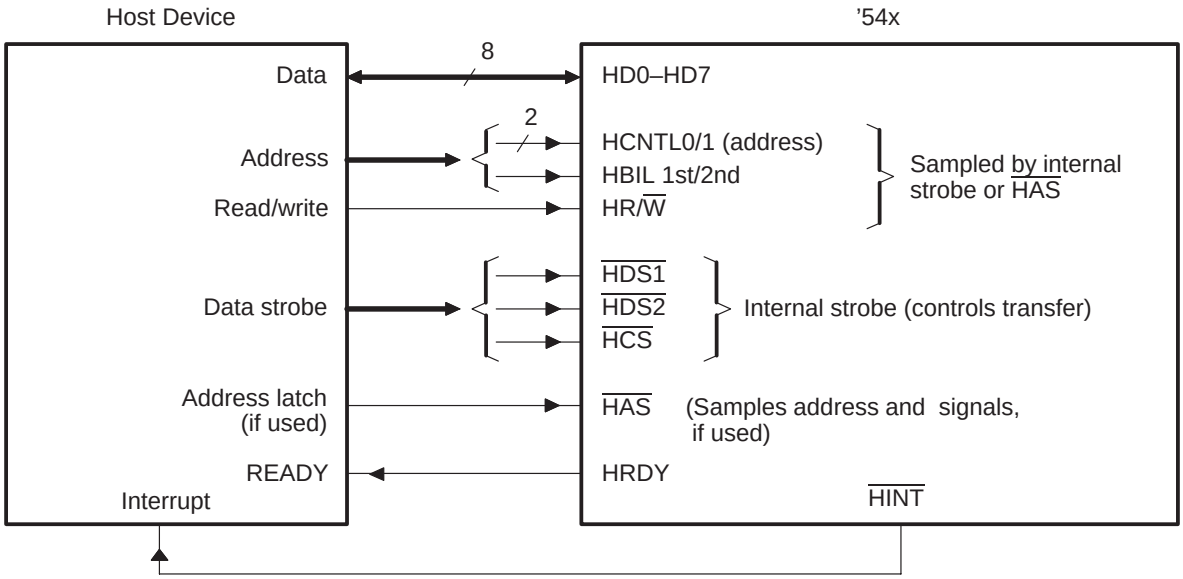
The HPI-8 provides 16-bit data to the ’54x while maintaining the economical 8-bit external interface. Successive bytes transferred are automatically combined into 16-bit words. When the host device performs a data transfer with HPI-8 registers, the HPI-8 control logic automatically performs an access to the internal ’54x RAM to complete the transaction. The ’54x can then access the data within its memory space.

Both the ’54x and the host can access the entire on-chip RAM of the ’54x. The ’54x clock must be active for host accesses to occur, and the HPI-8 is not functional while the ’54x device is in reset mode (except the ’5410 – see section 4.8.2, *Access to HPI-8 During Reset (’5410 Only)*, on page 4-28). The host accesses are synchronized to the ’54x clock internally to ensure proper arbitration of the on-chip RAM accesses. In the case of a conflict between a ’54x and a host cycle where both accesses involve the same memory location, the host has access priority and the ’54x CPU waits one ’54x clock cycle.

4.2 HPI-8 Basic Functional Description

The external HPI-8 interface can connect to a variety of host devices with little or no additional logic necessary. The 8-bit data bus (HD0–HD7) exchanges information with the host. The two control inputs (HCNTL0 and HCNTL1) indicate which internal HPI-8 register is accessed. These inputs, along with HBIL, are commonly driven by host address-bus bits or a function of these bits. Figure 4–2 shows a simplified diagram of a connection between the HPI-8 and a host device.

Figure 4–2. Generic System Block Diagram



Using the HCNTL0/1 inputs, the host can specify an access to the HPI control register (HPIC), the HPI address register (HPIA) (which serves as the pointer into '54x RAM), or the HPI data register (HPID). Because of the 16-bit word structure of the '54x, all HPI-8 transfers must consist of two consecutive bytes. The dedicated HBIL pin indicates whether the first or second byte is transferred. An internal control register bit determines whether the first or second byte is placed the most significant byte of a 16-bit word.

The HPID register can also be accessed with an optional automatic address increment feature. The autoincrement feature provides a convenient way of reading from, or writing to, consecutive word locations. In autoincrement mode, the HPIA register is automatically incremented during consecutive transfers. This feature is described further in section 4.3.3, *Address Autoincrement*, on page 4-11.

The HPI-8 includes interrupt logic to facilitate software handshaking. The host can interrupt the ’54x CPU by writing to a dedicated bit in the HPIC. Similarly, the ’54x can use the HINT output pin to interrupt the host. The HINT output is asserted when the ’54x writes to a dedicated bit in the HPIC. The host can also acknowledge and clear HINT pin by writing to the HINT bit in the HPIC register.

Table 4–2 lists and describes the three registers that the HPI-8 utilizes for communication between the host device and the ’54x CPU.

Table 4–2. HPI-8 Register Description

Name	’54x Address	Description
HPIA	—	HPI address register. Directly accessible only by the host. Contains the address in the ’54x on-chip RAM where the current access occurs.
HPIC	002Ch	HPI control register. Directly accessible by either the host or by the ’54x. Contains control and status bits for HPI-8 operations.
HPID	—	HPI data register. Directly accessible only by the host. Contains data that is read from the ’54x on-chip memory if the current access is a read, or data that is written to on-chip memory if the current access is a write.

The various strobe signals – data strobes ($\overline{\text{HDS1}}$ and $\overline{\text{HDS2}}$), read/write strobe (HR/W), and the address strobe ($\overline{\text{HAS}}$) – enable the HPI-8 to interface to a variety of host devices.

The HPI ready pin (HRDY) provides a convenient way to automatically adjust the host access rate to the HPI-8 access rate. The HRDY pin allows insertion of host wait states for hosts that support a ready input. This enables deferred completion of host accesses when the host bus cycle times are faster than the HPI-8 access rate.

All of these features combined provide a flexible and efficient interface to a wide variety of industry-standard host devices.

4.3 Details of HPI-8 Operation

This section includes a detailed description of each HPI-8 external interface pin function, as well as descriptions of the control register bit functions and the HPI-8 memory map. The logical interface timings, HPI-8 initialization, and read/write sequences are discussed in section 4.4, *Host Read/Write Access to HPI-8*, on page 4-14.

The external HPI-8 interface signals implement a flexible interface to a variety of host device types. Devices with single- or multiple-data strobes, with or without address latch enable (ALE) signals, can easily be connected to the HPI-8.

Table 4–3 gives a detailed description of each HPI-8 external interface pin.

Table 4–3. HPI-8 Signal Names and Functions

HPI Pin	Host Pin	State†	Signal Function
$\overline{\text{HAS}}$	Address latch enable (ALE) or address strobe or unused (tied high)	I	Address strobe input. Hosts with a multiplexed address and data bus connect $\overline{\text{HAS}}$ to their ALE pin or equivalent. HBIL, HCNTL0/1, and HR/$\overline{\text{W}}$ are then latched on the $\overline{\text{HAS}}$ falling edge. When used, $\overline{\text{HAS}}$ must precede the latter of $\overline{\text{HCS}}$, $\overline{\text{HDS1}}$, or $\overline{\text{HDS2}}$ (see '54x data sheet for detailed HPI-8 timing specifications). Hosts with separate address and data bus can connect $\overline{\text{HAS}}$ to a logic-1 level. In this case, HBIL, HCNTL0/1, and HR/$\overline{\text{W}}$ are latched by the latter of $\overline{\text{HDS1}}$, $\overline{\text{HDS2}}$, or $\overline{\text{HCS}}$ falling edge while $\overline{\text{HAS}}$ stays inactive-high.
HBIL	Address or control lines	I	Byte identification input. Identifies first or second byte of transfer (but not most significant or least significant — this is specified by the BOB bit in the HPIC register, described later in this section). HBIL is low for the first byte and high for the second byte.
HCNTL0, HCNTL1	Address or control lines	I	Host control inputs. These inputs select a host access to the HPIA register, the HPI data latches (with optional address increment), or the HPIC register.
$\overline{\text{HCS}}$	Address or control lines	I	Chip select. Serves as the enable input for the HPI-8 and must be low during an access but may stay low between accesses. $\overline{\text{HCS}}$ normally precedes $\overline{\text{HDS1}}$ and $\overline{\text{HDS2}}$, but this signal also samples HCNTL0/1, HR/$\overline{\text{W}}$, and HBIL if $\overline{\text{HAS}}$ is not used and $\overline{\text{HDS1}}$ or $\overline{\text{HDS2}}$ is already low (this is explained in further detail later in this section). Figure 4–3, *HPI Strobe and Select Logic*, on page 4-8 shows the equivalent circuit of the $\overline{\text{HCS}}$, $\overline{\text{HDS1}}$, and $\overline{\text{HDS2}}$ inputs.

† I = Input, O = Output, Z = High-impedance

Table 4–3. HPI-8 Signal Names and Functions (Continued)

HPI Pin	Host Pin	State†	Signal Function
HD0–HD7	Data bus	I/O/Z	Parallel bidirectional 3-state data bus. HD7 (MSB) through HD0 (LSB) are placed in the high-impedance state when the HPI is not active ($\overline{\text{HDS}}$x OR $\overline{\text{HCS}}$ = 1) or when EMU1/$\overline{\text{OFF}}$ is active-low. These pins can also be used for general purpose input/output when the HPI-8 is disabled. For more details on this feature, refer to section 4.9, *HPI-8 Data Pins as General Purpose I/O Pins*, on page 4-30.
$\overline{\text{HDS1}}$, $\overline{\text{HDS2}}$	Read strobe and write strobe or data strobe	I	Data strobe inputs. Control transfer of data during host access cycles. Also, when $\overline{\text{HAS}}$ is not used, these inputs sample HBIL, HCNTL0/1, and HR/$\overline{\text{W}}$ when $\overline{\text{HCS}}$ is already low (which is the case in normal operation). Hosts with separate read and write strobes connect those strobes to either $\overline{\text{HDS1}}$ or $\overline{\text{HDS2}}$. Hosts with a single data strobe connect it to either $\overline{\text{HDS1}}$ or $\overline{\text{HDS2}}$, connecting the unused pin high. Regardless of $\overline{\text{HDS}}$ connections, HR/$\overline{\text{W}}$ is still required to determine direction of transfer. Because $\overline{\text{HDS1}}$ and $\overline{\text{HDS2}}$ are internally exclusive-NORed, hosts with a high true data strobe can connect this to one of the HDS inputs with the other $\overline{\text{HDS}}$ input connected low. Figure 4–3 on page 4-8 shows the equivalent circuit of the $\overline{\text{HDS1}}$, $\overline{\text{HDS2}}$, and $\overline{\text{HCS}}$ inputs.
HINT	Host interrupt input	O/Z	Host interrupt output. Controlled by the HINT bit in the HPIC. Driven high when the ’54x is being reset. Placed in high-impedance when EMU1/$\overline{\text{OFF}}$ is active-low.
HRDY	Asynchronous ready	O/Z	HPI ready output. When high, indicates that the HPI-8 is ready for a transfer to be performed. When low, indicates that the HPI-8 is busy completing the internal portion of the previous transaction. Placed in high-impedance when EMU1/$\overline{\text{OFF}}$ is active-low. $\overline{\text{HCS}}$ enables HRDY; that is, HRDY is always high when $\overline{\text{HCS}}$ is high.
HR/$\overline{\text{W}}$	Read/write strobe, address line, or multiplexed address/data	I	Read/write input. Hosts must drive HR/$\overline{\text{W}}$ high to read HPI-8 and low to write HPI-8. Hosts without a read/write strobe can use an address line for this function.

† I = Input, O = Output, Z = High-impedance

The $\overline{\text{HCS}}$ input serves primarily as the enable input for the HPI-8, and the $\overline{\text{HDS1}}$ and $\overline{\text{HDS2}}$ signals control the HPI-8 data transfer; however, the logic with which these inputs are implemented allows their functions to be interchanged if desired. The equivalent circuit for these inputs is shown in Figure 4–3.

Figure 4–3. HPI Strobe and Select Logic

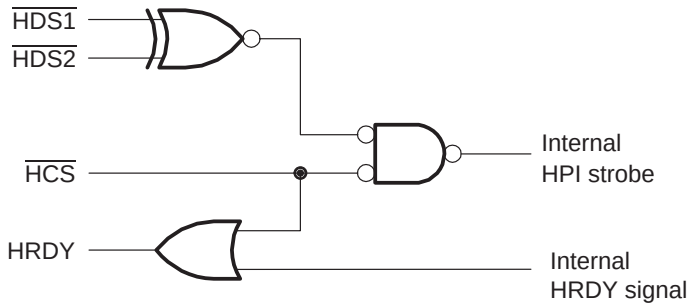


Figure 4–3 shows that the internal HPI strobe signal, which is used to control transfers, is derived from all three input signals — $\overline{\text{HCS}}$, $\overline{\text{HDS1}}$, and $\overline{\text{HDS2}}$. Note that if $\overline{\text{HCS}}$ is used in place of $\overline{\text{HDS1}}$ and $\overline{\text{HDS2}}$ to control HPI-8 access cycles, HRDY operation is affected (because $\overline{\text{HCS}}$ enables HRDY, and when $\overline{\text{HCS}}$ is high, HRDY is forced high). It is also important to note that because $\overline{\text{HDS1}}$ and $\overline{\text{HDS2}}$ are exclusive-NORed, driving both of these inputs low does not constitute an enabled condition.

The falling edge of the internal strobe is used to sample the HCNTL0/1, HBIL, and HR/$\overline{\text{W}}$ inputs whenever the $\overline{\text{HAS}}$ input is not used. Therefore, when $\overline{\text{HAS}}$ is not used, the latest of $\overline{\text{HCS}}$, $\overline{\text{HDS1}}$, or $\overline{\text{HDS2}}$ is the signal that actually controls sampling of the HCNTL0/1, HBIL, and HR/$\overline{\text{W}}$ inputs. In addition to sampling the control inputs, the internal strobe defines the boundaries of an HPI-8 cycle. This function of the internal strobe is described in section 4.4, *Host Read/Write Access to HPI-8,* on page 4-14.

When using the $\overline{\text{HAS}}$ input to sample HCNTL0/1, HBIL, and HR/$\overline{\text{W}}$, these signals can be removed earlier in an access cycle, thus allowing more time to switch bus states from address to data information. This additional time facilitates interface to hosts with multiplexed address and data busses. In this type of system, an address latch enable (ALE) signal is often provided and can be used to drive the $\overline{\text{HAS}}$ input.

The two control pins (HCNTL0 and HCNTL1) indicate which internal HPI-8 register is accessed. Table 4–4 describes the HCNTL0/1 pin functions.

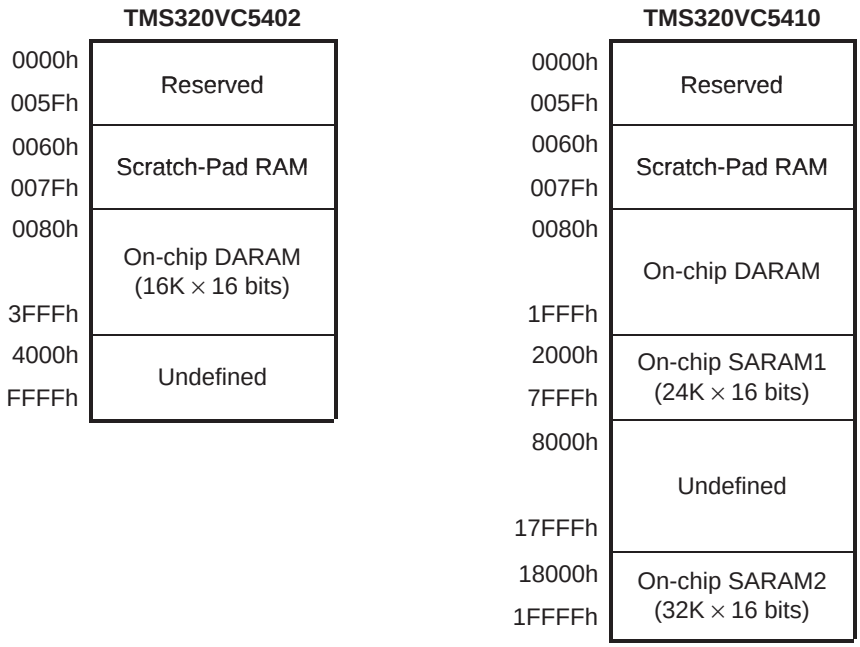
Table 4–4. HPI-8 Input Control Signals and Function Selection

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.

4.3.1 HPI-8 Address Register and Memory Map

The host uses the HPIA register as a pointer to ’54x on-chip memory, and all on-chip RAM locations are accessible through the HPI-8. Because the internal memory map of each ’54x device is unique, the address range that the HPI-8 can access varies from device to device. For example, the ’VC5410 includes much more on-chip RAM than the ’VC5402. The HPI-8 memory maps for the ’5402 and ’5410 devices are shown in Figure 4–4.

Figure 4–4. HPI-8 Memory Maps



All on-chip RAM blocks—program RAM and data RAM—are mapped to one contiguous address range within the HPI memory map. The addresses within this memory map cannot be dynamically re-mapped by the user (that is, the HPI-8 memory map is not affected by any programmable register bits).

4.3.2 Extended HPI-8 Addressing

For devices that include on-chip RAM mapped beyond the first 64K addresses, the HPI-8 includes an extended addressing feature. Seven extended address bits are internally available to allow HPI-8 access to extended pages of on-chip RAM. The host accesses this extended addressing feature using the extended HPI address (XHPIA) bit of the HPIC register. When the host sets the XHPIA bit, a 7-bit register representing the extended address bits (HPIA16:22) is accessible in place of the HPIA register. To initialize these extended address bits, the host must perform a write access to the HPIA register, with the seven LSBs of each byte designating the value of HPIA 16:22. Note that during this write access, the first and second byte values are both written to the same register. Therefore, if the same value is not written for both bytes, the second byte value is used to initialize the extended addresses, and the first byte value is discarded.

After initializing the extended address bits, the host must clear the XHPIA bit to regain access to the lower sixteen HPI address bits in the HPIA register. The XHPIA bit must also be cleared to zero for proper operation of the autoincrement feature. The autoincrement feature does not function properly when the XHPIA bit is set to one.

It is important to note that neither the XHPIA nor the seven extended address bits are initialized after reset, so the host should always initialize these bits after the ’54x is reset. The XHPIA bit is described in more detail in section 4.3.4, *HPI-8 Control Register Bits and Functions*, on page 4-12.

4.3.3 Address Autoincrement

The HPI-8 address autoincrement feature provides a convenient way of accessing consecutive word locations in on-chip RAM. When the autoincrement feature is enabled, the HPIA register is automatically incremented for each access. Although the access times are not changed, performance is increased because the host does not have to update the HPIA register between each memory access. The autoincrement feature is enabled when the HCNTL0 pin is set to one and the HCNTL1 pin is set to zero. Note that for devices with extended on-chip RAM, the XHPIA bit of the HPIC register must be set to one for proper autoincrement operation.

When autoincrement is enabled, a data read causes a postincrement of the HPIA, and a data write causes a preincrement of the HPIA. Therefore, to write a particular address using autoincrement, the HPIA register should be initialized to address-1. The increment function affects all 16 bits of the HPIA, and on devices with extended on-chip RAM (except the ’5410), the increment feature also affects the extended addresses. If the HPIA is set to FFFFh and autoincrement is enabled, the next access will change the HPI address to 010000h. Because the autoincrement circuitry of the ’5410 does not affect the extended HPI addresses, the above example will change the ’5410 HPI address to 000000h.

4.3.4 HPI-8 Control Register Bits and Functions

The bits of the HPIC register control and monitor HPI-8 operation. These bits are: BOB (selects first or second byte as most significant), DSPINT and HINT (can be used to generate ’54x and host interrupts, respectively), XHPIA (used by the host to access extended addresses), and HPIENA (indicates HPI-8 enabled or disabled status). Table 4–5 presents a detailed description of the HPIC bit functions.

Table 4–5. HPI Control Register (HPIC) Bit Descriptions

Bit	Reset Value	Function
BOB	0	Byte-order bit. This bit determines the placement for the two bytes of a transfer. If BOB = 1, the first byte of a transfer is least significant. If BOB = 0, the first byte is most significant. This bit can only be accessed (written or read) by the host, and it must be initialized before the first data or address register access.
DSPINT	0	Host-to-’54x interrupt. When the host writes a 1 to this bit, a ’54x interrupt is generated. The bit can only be written to by the host, and is always read as 0 by both the host and the ’54x. When the host writes to HPIC, both bytes must write the same value. For a detailed description of the DSPINT function, see section 4.5 on page 4-23.
HINT	0	’54x-to-host interrupt. This bit determines the state of the ’54x HINT output, which can be used to interrupt the host. When the HINT bit is set to 1, the HINT output is driven low, and when the bit is cleared to 0, the output is driven high. The HINT bit can only be set by the ’54x, and it can only be cleared by the host. The host clears the bit by writing a 1 to it. For a detailed description of the HINT function, see section 4.5 on page 4-23.
XHPIA	X	Extended address enable. When XHPIA=1, host writes to the HPIA register are loaded into the most significant bits HPIA[***n***:16]. If XHPIA=0, host writes to HPIA are loaded into HPIA[15:0]. All ***n***+1 address bits are incremented in the autoincrement mode. Reading the HPIA register is performed in the same manner. Only the host has access to this bit.
HPIENA	X	HPI enable status bit. This bit latches the reset value of the HPIENA pin, and can be used by the ’54x to determine if the HPI-8 is enabled or disabled. This bit is not affected by writes, and is not available to the host. Note: This bit is not available on all devices. For more details, see the specific HPIC register diagrams that follow.

The HPIC is organized on the host side as a 16-bit register with the same high and low byte contents (although access to certain bits is limited, as described previously). The upper 8 bits of the HPIC are unused on the ’54x side. The host accesses the HPIC register using the appropriate selection of HCNTL0/1, and performs two consecutive byte accesses to the 8-bit HPI-8 data bus. When the host writes to HPIC, both the first and second byte written must be the same value. The ’54x accesses the HPIC as a memory-mapped register at address 002Ch in data memory space.

The layout of the HPIC bits is shown in Figure 4–5. In this figure, a zero is specified for a read operation to indicate that the bit is always read as zero; similarly, an X specifies that the read value for this bit is indeterminate. For write operations, a one is specified to indicate that the bit must only be written with a one; an X is specified for a write operation to indicate that the bit can be written with a zero or a one. Note that unused bits are reserved for future expansion and it is recommended that they be written as zero, unless specified otherwise.

Figure 4–5. HPIC Diagram — Host and ’54x Accesses

Host Reads From HPIC										
15–13	12	11	10–9	8	7–5	4	3	2	1	0
0	XHPIA†	HINT	0	BOB	0	XHPIA†	HINT	0	X	BOB

Host Writes to HPIC											
15–13	12	11	10	9	8	7–5	4	3	2	1	0
X	XHPIA†	HINT	DSPINT	X	BOB	X	XHPIA†	HINT	DSPINT	X	BOB

’54x Reads From HPIC								
15–8	7	6–4	3	2	1	0
0	HPIENA‡	0	HINT	0	X	0

’54x Writes to HPIC					
15–4	3	2	1	0
X	HINT	X	1	X

X denotes bits that are unaffected by writes, or bits that can be read as either 1 or 0.
† This bit is only available on ’54x devices with on-chip RAM mapped in extended addresses.
‡ This bit is not available on the ’5410.

Because the ’54x can write to the HINT bit, which is read twice on the host interface side, the first and second byte reads by the host may yield different data if the ’54x changes the state of this bit in between the two host read operations. The characteristics of host and ’54x HPIC read/write cycles are summarized in Table 4–6.

Table 4–6. HPIC Host and ’54x Read/Write Characteristics

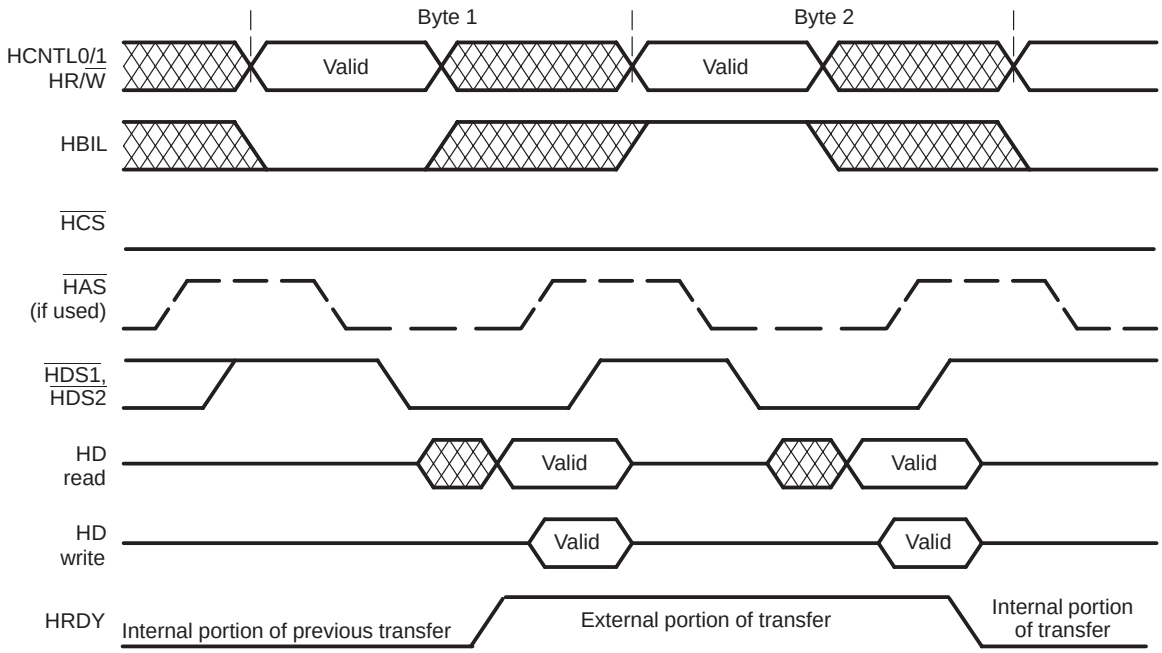
Device	Read	Write
Host	2 bytes	2 bytes (Both bytes must be equal)
’54x	16 bits	16 bits

4.4 Host Read/Write Access to HPI-8

An HPI-8 transfer is composed of an external portion in which the host exchanges data with the HPI-8 registers, and an internal portion in which the HPI-8 logic exchanges data between the registers and on-chip RAM. The external portion of the transfer always requires two bytes, regardless of the type of access — HPIA, HPIC, or data access. During the external portion of the transfer, the host drives the HBIL input low for the first byte, and then high for the second byte. If the host breaks this sequence of first byte/second byte (HBIL low/high) during an ongoing HPI-8 transfer, data may be lost and an unpredictable operation may result. To recover from such a condition, the host must repeat the access with the correct HBIL polarities for each byte.

Also for each byte, the host controls the duration of the access using the strobe and select inputs — $\overline{\text{HDS1}}$, $\overline{\text{HDS2}}$, and $\overline{\text{HCS}}$. The derivation of the internal HPI strobe from these inputs is described in section 4.3, *Details of HPI-8 Operation*, on page 4-6. The falling edge of the HPI strobe marks the beginning of the first or second byte of a transfer. This event usually occurs at the beginning of the host bus cycle. The rising edge of the HPI strobe marks the end of the first or second byte of an HPI transfer. This event usually occurs at the end of the host bus cycle. During the exchange of the second byte, the rising edge of the HPI strobe marks the end of the external portion of the cycle and initiates the internal portion of the cycle. Consequently, the HPID value read during the external portion of a memory read is the contents of the address specified in the previous access. A typical HPI-8 access is shown in Figure 4–6.

Figure 4–6. HPI-8 Timing Diagram



The cycle begins with the host driving HCNTL0/1, HR/$\overline{W}$, HBIL, and $\overline{HCS}$, indicating the type of transfer and whether the cycle is to be a read or write. Then the host asserts the $\overline{HAS}$ signal (if used) followed by one of the data strobe signals. If HRDY is not already high, it goes high when the previous internal cycle is complete, allowing data to be transferred. Following the external HPI-8 cycle, HRDY goes low (HPI-8 not ready) and stays low while the internal portion of the transfer is accomplished. HRDY then switches high again when the internal portion of the cycle is complete. You should note, however, that HRDY is always high when HCS is high.

4.4.1 Latency of HPI-8 Accesses

HPI-8 access time is composed of the time required for the external portion of a transfer and the time required for the internal portion. The time required for the external portion of an HPI-8 transfer is a fixed delay based on the setup times, hold times, and buffer delay times of the various signal pins. The internal delay of an HPI-8 transfer usually accounts for the majority of the access time and is related to the ’54x clock frequency. This internal delay varies depending on the type of access. An accurate analysis of the HPI-8 access time requires consideration of both the internal and external delays. Because the HPI strobe edges define the boundaries of an access, analysis of the timing requirements for these signals is sufficient. The timing requirements for HDS1,2 are given in the device-specific data sheets, and these timings incorporate both the external and internal delays of an HPI-8 access. The relationship between the HPI-8 internal access delay and the ’54x clock rate is explained in the paragraphs that follow.

The fastest accesses are:

- ❏ Read accesses to the HPIC and HPIA registers
- ❏ Write accesses to the HPIC register with the HINT and DSPINT bits written as zeros

These accesses only exchange data between the host and the internal registers, so no additional internal access time is required. The HRDY signal is never driven low for these accesses, since another HPI-8 access can begin as soon as the external portion of the access is complete.

Write accesses to the HPIC register with a one written to either the HINT or DSPINT bits have a longer latency. These accesses require some time after the external portion of the transfer completes for the internal interrupt logic of the HPI-8 to perform its function. The HRDY signal is held low for approximately three ’54x clock cycles during the internal portion of these accesses.

The slowest and most common HPI-8 accesses are memory accesses. A write access to the HPIA register, or a read/write access to the HPID register, initiates an internal memory transfer that exchanges the desired data between the HPID and the on-chip ’54x memory. This process requires several ’54x clock cycles each time an HPI-8 memory access is made. The minimum time required for the internal portion of an HPI-8 memory transfer is five ’54x clock cycles.

The maximum duration for the internal portion of an HPI-8 memory transfer depends on several factors. The HPI-8 and direct memory access (DMA) controller share the DMA bus for internal memory accesses, and only one of the modules can access the bus at a time. (For more details on the DMA controller, see Chapter 3.) When HPI-8 accesses and DMA controller accesses contend for bus usage, internal arbitration logic resolves the conflict. If the HPI-8 and DMA controller initiate an access at the same time, the HPI-8 has priority and the DMA controller transfer is postponed for one ’54x clock cycle. If the HPI-8 initiates an access while a DMA transfer is in progress, the HPI-8 access is delayed (HRDY stays low) until the current word of the DMA transfer is complete. This delay time, during which the host waits for the DMA controller, depends on the type of DMA transfer in progress. The internal delay for each of the HPI-8 access types are summarized in Table 4–7.

Table 4–7. HPI-8 Internal Delays by Access Type

Access Type		Internal Delay
Read access to HPIA or HPIC or write access to HPIC with HINT and DSPINT cleared to zero.		No internal delay
Write access to HPIC with either HINT or DSPINT set to one.		3 CPU clock cycles
Memory access when DMAC is inactive		5 CPU clock cycles
Memory access when DMAC is active	16 bit DMAC transfer to/from on-chip RAM†	5 CPU clock cycles + 4 CPU clock cycles
	32 bit DMAC transfer to/from on-chip RAM†	5 CPU clock cycles + 8 CPU clock cycles

† Note: additional latency can occur due to wait-states when the DMAC accesses off-chip memory.

Because the HPI-8 access times vary by access type, the host should always sample the HRDY output to adjust its bus cycle time for the varying HPI-8 rate. This sampling can be accomplished by connecting HRDY to the host READY input (if available) to generate host wait-states. Alternatively, the host bus cycle time can be set to the slowest possible HPI-8 access time. The function of HRDY for the various HPI-8 access types is summarized in Table 4–8.

Table 4–8. Wait-State Generation Conditions

Wait States Generated (HRDY driven low)		
Register	**Reads**	**Writes**
HPIC	No	1 to DSPINT/HINT – Yes All other cycles – No
HPIA	No	Yes
HPID	Yes	Yes

Due to the prefetch nature of internal HPI-8 accesses, special care must be taken under certain conditions. During random (non-sequential) transfers, or sequential accesses with a significant amount of time between them, it is possible for the ’54x to change the contents of the location being accessed during the time between a host read and the previous host access. If this occurs, the data read may be different from the actual memory contents being accessed. Where this is of concern in a system, two reads from the same address, or an address write prior to the read access, can be made to ensure that the most recent data is read.

4.4.2 Access Sequence Examples

Before accessing data, the host must first initialize HPIC, in particular BOB (bit 0 and bit 8), and then the HPIA register. The initialization must occur in this order because the state of BOB affects the HPIA register access.

On devices with extended on-chip RAM, the host should also initialize the XHPIA bit of the HPIC before accessing the HPIA register. The XHPIA bit can be initialized in the same write access to the HPIC that initializes BOB. By writing a one to the XHPIA bit, the host gains access to the seven extended HPI addresses. The host then writes the HPIA register with the seven LSBs designating the value of the extended addresses (HPIA 16:22). When initializing the extended HPI addresses, the same value should be written for the first and second bytes of the access. After initializing the extended addresses, the host must perform another access to the HPIC, writing a zero in the XHPIA bit to regain access to the lower sixteen address bits in the HPIA register. The extended address feature is discussed in more detail in section 4.3.2. on page 4-10.

After initializing BOB, the host can then write to the HPIA register with the correct byte alignment. When the host writes to the HPIA, the on-chip ’54x memory is automatically read and the contents at the given address are transferred to the two 8-bit data latches — the first and second bytes of the HPID register.

Table 4–9 illustrates the sequence involved in initializing the BOB and XHPIA bits of the HPIC and HPIA registers for an HPI-8 memory read. Note that the first six rows of the table show the initialization of the extended address feature, which is only required for devices with extended on-chip RAM. In this example, BOB is set to zero and a read is requested of the first on-chip memory location (in this case 0060h) which contains FFFEh.

Table 4–9. Initialization of BOB and HPIA

Event	HRDY	HD	HR/W	HCNTL1/0	HBIL	HPIC	HPIA	HPID	
								Latch1	Latch2
Host writes HPIC, 1st byte†	1	10	0	00	0	10xx	xxxx	xxxx	xxxx
Host writes HPIC, 2nd byte†	1	10	0	00	1	1010	xxxx	xxxx	xxxx
Host writes HPIA, 1st byte†	1	00	0	10	0	1010	xxxx	xxxx	xxxx
Host writes HPIA, 2nd byte†	1	00	0	10	1	1010	xx00	xxxx	xxxx
Internal delay†	0							xxxx	xxxx
Internal HPI RAM read complete†	1							xxxx	xxxx
Host writes HPIC, 1st byte	1	00	0	00	0	00xx	xxxx	xxxx	xxxx
Host writes HPIC, 2nd byte	1	00	0	00	1	0000	xxxx	xxxx	xxxx
Host writes HPIA, 1st byte	1	00	0	10	0	0000	00xx	xxxx	xxxx
Host writes HPIA, 2nd byte	1	60	0	10	1	0000	0060	xxxx	xxxx
Internal delay	0						0060	xxxx	xxxx
Internal RAM read complete	1						0060	FF	FE

† These accesses initialize the extended addressing feature and are only required for devices with extended on-chip RAM.

In the example shown in Table 4–9, the BOB and XHPIA bits of HPIC are initialized first; then the write access to the seven LSBs of the HPIA register initializes the extended HPI address bits. The first and second bytes of this write access must write the same values for proper initialization of the extended HPI address bits. Notice that the write access to the HPIA register automatically initiates an internal read access; however, the data read is indeterminate because the lower sixteen address bits are not yet initialized. These steps are only required for devices with extended on-chip RAM.

Next, the host initializes the lower 16 address bits by writing a zero to the XHPIA bit and writing the 16-bit address to the HPIA register. As before, writing to the HPIA register automatically initiates an internal HPI-8 memory access. This time, all address bits are initialized so the internal read retrieves the contents of the specified memory location. The last line of Table 4–9 shows the condition of the HPI-8 after the internal RAM read is complete. That is, after a delay following the end of the HPIA register write, the read is completed and the data is placed in the HPID register. The host must perform an additional read of HPID to actually retrieve this data. The sequence involved in this access is shown in Table 4–10.

Table 4–10. Read Access to HPI-8 With Autoincrement

								HPID	
Event	**HRDY**	**HD**	**HR/W**	**HCNTL1/0**	**HBIL**	**HPIC**	**HPIA**	**Latch1**	**Latch2**
Host reads data, 1st byte	1	FF	1	01	0	0000	0060	FF	FE
Host reads data, 2nd byte	1	FE	1	01	1	0000	0061	FF	FE
Internal delay	0						0061	FF	FE
Internal HPI RAM read complete	1						0061	6A	BC

In the access shown in Table 4–10, the data obtained from reading HPID is the data from the read initiated in the previous cycle (the one shown in Table 4–9). During this HPID read access, the contents of the first byte data latch are driven on the HD pins when HBIL is low, and the contents of the second byte data latch are driven on the HD pins when HBIL is high. The access performed in Table 4–10 also initiates another read of location 0061h (because autoincrement was specified in this access by setting HCNTL1/0 to 01). When autoincrement is selected, the increment occurs with each 16-bit word transferred (not with each byte); therefore, as shown in Table 4–10, the HPIA is incremented by one. The last line of Table 4–10 indicates that after some delay following the read of the second byte, the contents of location 0061h (6ABCh) are read and placed in the HPID register.

During a 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, and the second byte data latch is overwritten by the data coming from the host while the HBIL pin is high. At the end of this write access, the bytes in both data latches are transferred as a 16-bit word to the on-chip RAM at the address specified by the HPIA register. The address is incremented prior to the memory write if autoincrement is selected.

An HPI write access is illustrated in Table 4–11. In this example, autoincrement is enabled and the HPIA register is incremented before the write occurs (preincremented). Because the previous read access caused a postincrement of the HPIA register, address 0061h is skipped. After the internal portion of the write is completed, location 0062h of on-chip RAM contains 1234h. If a read of address 0062h follows this write, the same data (1234h) is read back.

Table 4–11. Write Access to HPI With Autoincrement

								HPID	
Event	**HRDY**	**HD**	**HR/W**	**HCNTL1/0**	**HBIL**	**HPIC**	**HPIA**	**Latch1**	**Latch2**
Host writes data, 1st byte	1	12	0	01	0	0000	0061	12	FE
Host writes data, 2nd byte	1	34	0	01	1	0000	0062	12	34
Internal delay	0						0062	12	34
Internal RAM write complete	1						0062	12	34

4.5 DSPINT and HINT Operation

The host and the ’54x can interrupt each other using bits in the HPIC register. The sections that follow explain this process.

4.5.1 Host Device Using DSPINT to Interrupt the ’54x

A ’54x interrupt is generated when the host writes a one to the DSPINT bit in HPIC. This interrupt can be used to wake up the ’54x from IDLE. The host and the ’54x always read this bit as zero. A ’54x write has no effect. After a one is written to DSPINT by the host, it is not necessary for a zero to be written before another interrupt can be generated. Writing a zero to this bit has no effect. The host should not write a one to the DSPINT bit while writing to BOB or HINT; otherwise, an unwanted ’54x interrupt is generated.

On the ’54x, the host-to-’54x interrupt vector address is xx64h. This interrupt is located in bit nine of the IMR/IFR registers. Because the ’54x interrupt vectors can be re-mapped into the on-chip RAM, the host can instruct the ’54x to execute pre-programmed functions by initializing the DSPINT interrupt vector. When the ’54x interrupts are re-mapped to on-chip RAM, the host can: 1) write the opcode for a branch instruction at address xx64h, and 2) write the start address of a function at address xx65h in the interrupt vector table prior to interrupting the ’54x. When using this technique, care must be taken to prevent the host from corrupting the other interrupt vector*s.*

4.5.2 ’54x Using HINT to Interrupt the Host Device

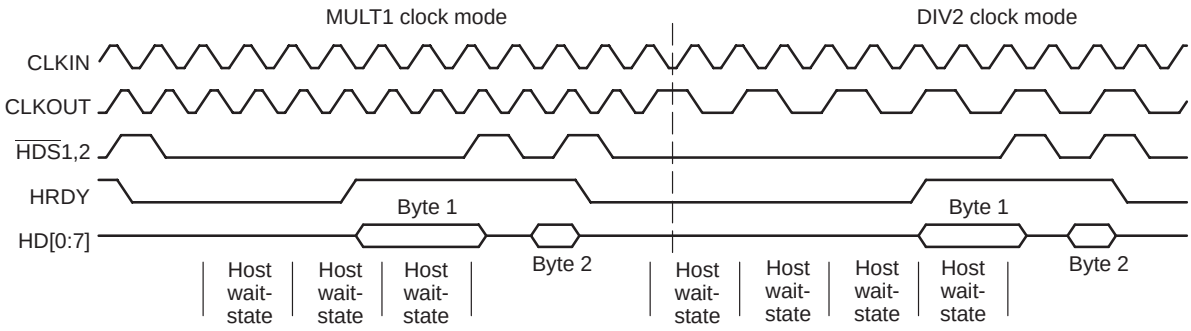
When the ’54x writes a one to the HINT bit in HPIC, the $\overline{\text{HINT}}$ output is driven low, and the HINT bit is read as a one. Consequently, the $\overline{\text{HINT}}$ signal can be used as an active-low interrupt source for a host device. The host can clear this interrupt and cause the $\overline{\text{HINT}}$ signal to be deasserted by writing a one to the HINT bit. The HINT bit is cleared by this write, and the $\overline{\text{HINT}}$ signal is driven high. If the ’54x or host writes a zero, the HINT bit remains unchanged.

4.6 Considerations for HPI-8 Transfers While Changing Clock Modes

Unlike the standard 8-bit HPI, the HPI-8 does not include an asynchronous host-only mode (HOM). All HPI-8 transfers are synchronized to the ’54x clock. This requires special considerations for HPI-8 transfers while changing clock modes because a change in the ’54x clock frequency causes a change in the HPI-8 access rate. For more information on changing the ’54x clock generator modes, refer to *TMS320C54x DSP Reference Set, Volume 1: CPU and Peripherals* (literature number SPRU131). The relationship between the ’54x clock rate and the HPI-8 access rate is described in section 4.4.1 on page 4-16.

When the ’54x clock mode is changed from PLL mode to DIV mode, the resulting clock rate is slower, and the HPI-8 access rate is also slower. In this case, the host can use the HRDY output to generate wait-states and adjust to the change in access rate. An example of HPI-8 accesses while changing from multiply-by-1 (MULT1) clock mode to divide-by-2 (DIV2) clock mode is shown in Figure 4–7.

Figure 4–7. HPI-8 Transfers While Switching to DIV Clock Modes



In the example shown in Figure 4–7, the difference in the ’54x clock frequency before and after switching clock modes is only a factor of two, and the resulting effect on HPI-8 access time is small. In this case, the host can compensate for the change in access rate with one additional wait-state. When higher clock multiply ratios are used, switching to a divide-by mode causes a significant decrease in the HPI-8 access rate, and several additional host wait-states may be required.

When the ’54x clock generator is switched from a divide-by mode to a multiply mode, the resulting ’54x clock rate and HPI-8 access rate are increased. In this case, if host wait-states are generated using HRDY, then the host automatically reduces the number of wait-states in a similar manner to that shown in Figure 4–7. If the resulting HPI-8 access rate is faster then the host bus cycle, then no wait-states are required and the host bus-cycle time defines the new access rate.

4.7 Considerations in IDLE Use

The HPI-8 allows the host to access on-chip RAM while the ’54x is in an IDLE mode. The only requirement for HPI-8 access during these modes is that the ’54x input clock (CLKIN) must remain active. For more information on the IDLE modes of the ’54x, refer to the *TMS320C54x DSP Reference Set, Volume 1: CPU and Peripherals* (literature number SPRU131).

4.7.1 HPI-8 Accesses During IDLE1 and IDLE2

Because the IDLE1 and IDLE2 modes do not affect the ’54x clock, no special considerations are required for HPI-8 transfers during these modes. However, in some cases, you may wish to disable the PLL circuit prior to initiating these power-down modes to further reduce power consumption. In these cases, the HPI-8 access rate is affected and special considerations must be made. For more information on considerations for HPI-8 usage during clock mode changes, refer to section 4.6 on page 4-24.

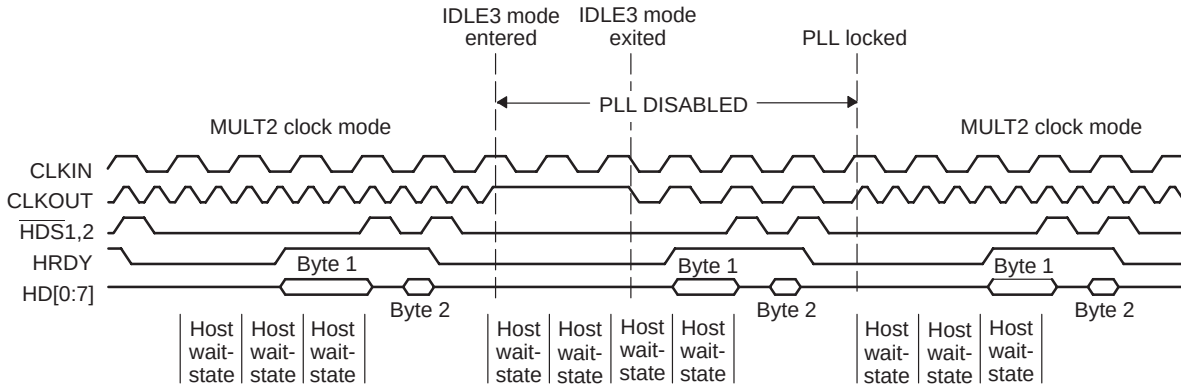
4.7.2 HPI-8 Accesses During IDLE3

The HPI-8 access rate is usually affected by the IDLE3 power-down mode because this mode affects the internal clock generation circuitry of the ’54x. When the ’54x enters the IDLE3 mode, the on-chip clock generator is disabled and HPI-8 accesses are automatically synchronized to the input clock source (CLKIN). The host can continue to access the HPI-8 while the ’54x is in the IDLE3 mode as long as an active clock is applied to the ’54x CLKIN pin. HPI-8 accesses remain synchronized to CLKIN until the ’54x is taken out of IDLE3 mode. After the ’54x is taken out of IDLE3 mode by an interrupt source, HPI-8 accesses are automatically synchronized to the output of the ’54x clock generator. If the clock generator is configured in PLL mode, HPI-8 accesses remain synchronized to the input source until the PLL lock timer counts down. For more information on the ’54x clock generator and programmable lock timer, refer to *TMS320C54x DSP Reference Set, Volume 1: CPU and Peripherals* (literature number SPRU131).

When the ’54x enters the IDLE3 mode while a clock multiply ratio of one is selected, the HPI-8 access rate remains the same (×1) during IDLE3. Also, if the clock generator is configured in DIV (divider) mode when the ’54x enters IDLE3, then the HPI-8 access rate is increased (×1) while the ’54x is in IDLE3. In both of these cases, no special considerations are required because the HPI-8 access rate is neither unaffected nor improved during IDLE3.

When the ’54x enters IDLE3 mode while a clock multiply ratio greater than one is selected, the HPI-8 access rate is reduced. In this case, the host can use the HRDY output to generate additional wait-states and adjust to the change in access rate. An example of this case is shown in Figure 4–8.

Figure 4–8. HPI-8 Transfers While the ’54x is in IDLE3 Mode



In Figure 4–8 the clock generator is configured in PLL mode with a multiply ratio of two when the ’54x enters IDLE3 mode. HPI-8 accesses are automatically synchronized to the input clock source, and the host must insert an additional wait-state for memory accesses based on the state of the HRDY pin. The figure also shows that HPI-8 accesses are automatically re-synchronized to the faster clock after the ’54x is taken out of IDLE3 mode and the clock generator stabilizes.

4.8 Effects of Reset on HPI-8 Operation

HPI-8 operation is affected when the ’54x device is reset. This section describes the effect of reset on the HPI-8 operation.

4.8.1 Accesses to HPI-8 After Reset

The HPI-8 does not allow a host access while the ’54x device is in reset (except ’5410, see section 4.8.2). After the ’54x device is released from reset, the host can access the HPI-8. After the ’54x device is released from the reset, the host can access the HPI-8. Because the HPI-8 access rate depends on the ’54x clock rate, and the ’54x clock rate is re-initialized by reset, care must be taken to match the speed of host accesses to the HPI-8 access rate selected after reset. The ’54x clock generator is initialized at reset to the mode selected by the clock mode pins (CLKMD1–3). If the host has control of the ’54x reset pin and clock mode pins, then it can configure the HPI-8 access rate after reset. For more information on the clock generator and clock mode pins, refer to the device-specific data sheets and to *TMS320C54x DSP Reference Set, Volume 1: CPU and Peripherals* (literature number SPRU131).

4.8.2 Access to HPI-8 During Reset (’5410 Only)

The HPI-8 of the ’5410 supports the host-only-mode (HOM) during reset for compatibility with previous versions of the 8-bit HPI. The ’5410 is not operational during reset, but the host can access the HPI-8, allowing program or data downloads to the on-chip memory. Accesses during reset are not internally synchronized to the ’54x clock, and the HRDY output is never de-asserted (except when writing a one to the DSPINT or HINT bits of the HPIC). The host should not write a one to either the DSPINT or HINT bits of the HPIC register while the ’5410 is in reset because the HPI-8 will not function properly, and the HRDY output will be de-asserted indefinitely. For the HPI-8 access time during reset, refer to the ’5410 data sheet (see *Related Documentation from Texas Instruments* in the Preface).

When HOM is used during reset, it is often convenient for the host to control the ’5410 reset input. The sequence of events for resetting the ’5410 and downloading a program to on-chip RAM while the ’5410 is in reset is summarized in Table 4–12.

Table 4–12. HPI-8 Operation During RESET ('5410 only)

Host	'5410	Mode	'5410 CLK
Waits 6 '5410 clock periods	Running	X	Running
Brings RESET low and waits 4 clocks	Goes into reset	HOM	Running
Can stop '5410 clock	In reset	HOM	Stopped or running
Writes program and/or data in on-chip RAM	In reset	HOM	Stopped or running
Turns on DSP clock if it was stopped†	In reset	HOM	Running
Brings RESET high	In reset	HOM	Running
Waits 20 '5410 clock periods	Comes out of reset	SAM	Running
Can access HPI-8	Running	SAM	Running

† Sufficient wake-up time must be ensured when the '5410 on-chip PLL is used.

Initially, the host stops accessing the HPI-8 at least six '5410 clock periods before driving the '5410 reset line low. The host then drives the '5410 reset line low and can start accessing the HPI-8 after a minimum of four '5410 clock periods. The HPI-8 mode is automatically set to HOM during reset, allowing a high-speed program download. The '5410 clock can even be stopped at this time; however, the clock must be running when the reset line falls and rises for proper reset operation of the '5410.

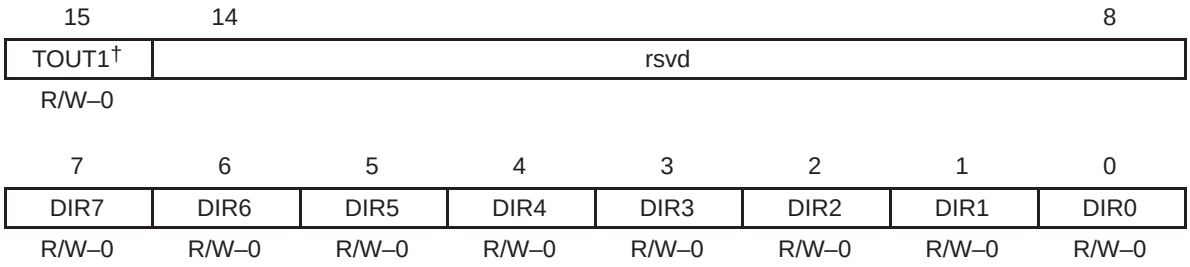
Once the host has finished downloading into on-chip RAM, the host stops accessing the HPI-8 and drives the '5410 reset line high. At least 20 '5410 periods after the reset line rising edge, the host can again begin accessing the HPI-8. This number of periods corresponds to the internal reset delay of the '5410. The HPI-8 mode is automatically set to SAM upon exiting reset. If the host writes a one to DSPINT while the '5410 is in reset, the interrupt is lost when the '5410 comes out of reset. The '5410 HPI-8 boot mode can then be used to start execution from address 02000h.

4.9 HPI-8 Data Pins as General Purpose I/O Pins (Not Available on ’5410)

The 8-bit bidirectional data bus of the HPI-8 can be used as general-purpose input/output (GPIO) pins. This feature is only available when the HPI-8 is disabled; that is, when the HPIENA pin is driven low during reset. Two memory-mapped registers are used to control the GPIO function of the HPI-8 data pins: the general-purpose I/O control register (GPIOCR), and the general-purpose I/O status register (GPIOSR). The GPIOCR is shown in Figure 4–9, and its bits are described in Table 4–13.

The direction bits (DIRx) of the GPIOCR are used to configure HD0-HD7 as inputs or outputs. The timer1 output bit (TOUT1) is available on devices that include two timers to enable the timer1 output on the HINT pin. When the HPI-8 is enabled, the TOUT1 bit and DIRx bits are forced to zero, and the general purpose I/O feature functions in input mode only.

Figure 4–9. General Purpose I/O Control Register (GPIOCR) MMR Address 003Ch



† Only available on devices with a second on-chip timer.

Note: R = Read, W= Write

Table 4–13. General Purpose I/O Control Register (GPIOCR) Bit Functions

No.	Name	Reset Value	Function
15	TOUT1	0	Timer1 output enable bit. The TOUT1 bit enables or disables the timer1 output on the HINT pin. The timer1 output is only available when the HPI-8 is disabled. Note, this bit is reserved on devices that have only one timer. TOUT1 = 0 The timer1 output is not available externally. TOUT1 = 1 The timer1 output is driven on the HINT pin.
14–8	rsvd	0	These pins are reserved and unaffected by writes.
7	DIR7	0	I/O pin 7 direction bit. DIR7 configures the HD7 pin as input or output. DIR7 = 0 The HD7 pin is configured as an input.

Table 4–13. General Purpose I/O Control Register (GPIOCR) Bit Functions (Continued)

No.	Name	Reset Value	Function	
			DIR7 = 1	The HD7 pin is configured as an output. When the HPI-8 is enabled, this bit is forced to 0, and is not affected by writes.
6	DIR6	0	I/O pin 6 direction bit. DIR6 configures the HD6 pin as input or output.	
			DIR6 = 0	The HD6 pin is configured as an input.
			DIR6 = 1	The HD6 pin is configured as an output. When the HPI-8 is enabled, this bit is forced to 0 and is not affected by writes.
5	DIR5	0	I/O pin 5 direction bit. DIR5 configures the HD5 pin as input or output.	
			DIR5 = 0	The HD5 pin is configured as an input.
			DIR5 = 1	The HD5 pin is configured as an output. When the HPI-8 is enabled, this bit is forced to 0 and is not affected by writes.
4	DIR4	0	I/O pin 4 direction bit. DIR4 configures the HD4 pin as input or output.	
			DIR4 = 0	The HD4 pin is configured as an input.
			DIR4 = 1	The HD4 pin is configured as an output. When the HPI-8 is enabled, this bit is forced to 0 and is not affected by writes.
3	DIR3	0	I/O pin 3 direction bit. DIR3 configures the HD3 pin as input or output.	
			DIR3 = 0	The HD3 pin is configured as an input.
			DIR3 = 1	The HD3 pin is configured as an output. When the HPI-8 is enabled, this bit is forced to 0 and is not affected by writes.
2	DIR2	0	I/O pin 2 direction bit. DIR2 configures the HD2 pin as input or output.	
			DIR2 = 0	The HD2 pin is configured as an input.
			DIR2 = 1	The HD2 pin is configured as an output. When the HPI-8 is enabled, this bit is forced to 0 and is not affected by writes.
1	DIR1	0	I/O pin 1 direction bit. DIR1 configures the HD1 pin as input or output.	
			DIR1 = 0	The HD1 pin is configured as an input.
			DIR1 = 1	The HD1 pin is configured as an output. When the HPI-8 is enabled, this bit is forced to 0 and is not affected by writes.
0	DIR0	0		I/O pin 0 direction bit. DIR0 configures the HD0 pin as input or output.

Table 4–13. General Purpose I/O Control Register (GPIOCR) Bit Functions (Continued)

No.	Name	Reset Value	Function	
			DIR0 = 0	The HD0 pin is configured as an input.
			DIR0 = 1	The HD0 pin is configured as an output. When the HPI-8 is enabled, this bit is forced to 0 and is not affected by writes.

The status of the GPIO pins (HDx, x=0:7) can be monitored using the bits of the general purpose I/O status register (GPIOSR). When an HDx pin is configured as an input (by writing a one to the DIRx bit of the GPIOCR), the corresponding bit in the GPIOSR can be read to determine the logic value sensed at the pin. Similarly, when an HDx pin is configured as an output, the logic value to be driven by the pin is written to the corresponding bit in the GPIOSR. The GPIOSR is shown in Figure 4–10, and its bits are described in Table 4–14.

Figure 4–10. General Purpose I/O Status Register (GPIOSR) MMR address 003Dh

15							8
rsvd							

7	6	5	4	3	2	1	0
IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0
R/W–x	R/W–x	R/W–x	R/W–x	R/W–x	R/W–x	R/W–x	R/W–x

Note: R = Read, W= Write
† Note, when the HD pins are configured as inputs, writes to the IOx bits have no effect.

Table 4–14. General Purpose I/O Status Register (GPIOSR) Bit Functions

No.	NAME	Reset Value	Function	
15–8	rsvd	0	These pins are reserved, and unaffected by writes.	
7	IO7	X	IO7 - I/O pin 7 status bit. This bit reflects the logic level on the HD7 pin. When the HD7 pin is configured as an input (DIR7 = 0 in the GPIOCR), the IO7 bit latches the logic value of the pin (1 or 0). Writes to the IO7 bit have no effect when the HD7 pin is configured as an input. When the HD7 pin is configured as an output (DIR7 = 1 in GPIOCR), the HD7 pin is driven to the logic level written in the IO7 bit (1 or 0).	
			IO7 = 0	The HD7 input is externally driven low, or the HD7 output is internally driven low.

Table 4–14. General Purpose I/O Status Register (GPIOSR) Bit Functions (Continued)

No.	NAME	Reset Value	Function	
			IO7 = 1	The HD7 input is externally driven high, or the HD7 output is Internally driven high.

Table 4–14. General Purpose I/O Status Register (GPIOSR) Bit Functions (Continued)

No.	NAME	Reset Value	Function
6	IO6	X	IO6 - I/O pin 6 status bit. This bit reflects the logic level on the HD6 pin. When the HD6 pin is configured as an input (DIR6 = 0 in the GPIOCR), the IO6 bit latches the logic value of the pin (1 or 0). Writes to the IO6 bit have no effect when the HD6 pin is configured as an input. When the HD6 pin is configured as an output (DIR6 = 1 in GPIOCR), the HD6 pin is driven to the logic level written in the IO6 bit (1 or 0). IO6 = 0 The HD6 input is externally driven low, or the HD6 output is internally driven low. IO6 = 1 The HD6 input is externally driven high, or the HD6 output is Internally driven high.
5	IO5	X	IO5 - I/O pin 5 status bit. This bit reflects the logic level on the HD5 pin. When the HD5 pin is configured as an input (DIR5 = 0 in the GPIOCR), the IO5 bit latches the logic value of the pin (1 or 0). Writes to the IO5 bit have no effect when the HD5 pin is configured as an input. When the HD5 pin is configured as an output (DIR5 = 1 in GPIOCR), the HD5 pin is driven to the logic level written in the IO5 bit (1 or 0). IO5 = 0 The HD5 input is externally driven low, or the HD5 output is internally driven low. IO5 = 1 The HD5 input is externally driven high, or the HD5 output is Internally driven high.
4	IO4	X	IO4 - I/O pin 4 status bit. This bit reflects the logic level on the HD4 pin. When the HD4 pin is configured as an input (DIR4 = 0 in the GPIOCR), the IO4 bit latches the logic value of the pin (1 or 0). Writes to the IO4 bit have no effect when the HD4 pin is configured as an input. When the HD4 pin is configured as an output (DIR4 = 1 in GPIOCR), the HD4 pin is driven to the logic level written in the IO4 bit (1 or 0). IO4 = 0 The HD4 input is externally driven low, or the HD4 output is internally driven low. IO4 = 1 The HD4 input is externally driven high, or the HD4 output is Internally driven high.

Table 4–14. General Purpose I/O Status Register (GPIOSR) Bit Functions (Continued)

No.	NAME	Reset Value	Function
3	IO3	X	IO3 - I/O pin 3 status bit. This bit reflects the logic level on the HD3 pin. When the HD3 pin is configured as an input (DIR3 = 0 in the GPIOCR), the IO3 bit latches the logic value of the pin (1 or 0). Writes to the IO3 bit have no effect when the HD3 pin is configured as an input. When the HD3 pin is configured as an output (DIR3 = 1 in GPIOCR), the HD3 pin is driven to the logic level written in the IO3 bit (1 or 0). IO3 = 0 The HD3 input is externally driven low, or the HD3 output is internally driven low. IO3 = 1 The HD3 input is externally driven high, or the HD3 output is Internally driven high.
2	IO2	X	IO2 - I/O pin 2 status bit. This bit reflects the logic level on the HD2 pin. When the HD2 pin is configured as an input (DIR2 = 0 in the GPIOCR), the IO2 bit latches the logic value of the pin (1 or 0). Writes to the IO2 bit have no effect when the HD2 pin is configured as an input. When the HD2 pin is configured as an output (DIR2 = 1 in GPIOCR), the HD2 pin is driven to the logic level written in the IO2 bit (1 or 0). IO2 = 0 The HD2 input is externally driven low, or the HD2 output is internally driven low. IO2 = 1 The HD2 input is externally driven high, or the HD2 output is Internally driven high.
1	IO1	X	IO1 - I/O pin 1 status bit. This bit reflects the logic level on the HD1 pin. When the HD1 pin is configured as an input (DIR1 = 0 in the GPIOCR), the IO1 bit latches the logic value of the pin (1 or 0). Writes to the IO1 bit have no effect when the HD1 pin is configured as an input. When the HD1 pin is configured as an output (DIR1 = 1 in GPIOCR), the HD1 pin is driven to the logic level written in the IO1 bit (1 or 0). IO1 = 0 The HD1 input is externally driven low, or the HD1 output is internally driven low. IO1 = 1 The HD1 input is externally driven high, or the HD1 output is Internally driven high.

Table 4–14. General Purpose I/O Status Register (GPIOSR) Bit Functions (Continued)

No.	NAME	Reset Value	Function	
0	IO0	X		IO0 - I/O pin 0 status bit. This bit reflects the logic level on the HD0 pin. When the HD0 pin is configured as an input (DIR0 = 0 in the GPIOCR), the IO0 bit latches the logic value of the pin (1 or 0). Writes to the IO0 bit have no effect when the HD0 pin is configured as an input. When the HD0 pin is configured as an output (DIR0 = 1 in GPIOCR), the HD0 pin is driven to the logic level written in the IO0 bit (1 or 0).
			IO0 = 0	The HD0 input is externally driven low, or the HD0 output is internally driven low.
			IO0 = 1	The HD0 input is externally driven high, or the HD0 output is Internally driven high.

4.9.1 Using the GPIO feature

To use the HPI data pins as general purpose inputs or outputs, the pins must first be configured appropriately. Afterwards, the pins can be monitored or manipulated by reading and writing the GPIOSR. Figure 4–11 shows a code segment example for using the GPIO feature of the HPI-8.

Figure 4–11. GPIO Code Example

```
GPIOCR     .set     3Ch    ;MMR address for GPIOCR is 3Ch
GPIOSR     .set     3Dh    ;MMR address for GPIOSR is 3Dh

           .text
STM        #0F0h, GPIOCR   ;Configure HD0-3 as in, and
                           ;HD4-7 as out.
           .   .   .

LDM        GPIOSR, A       ;Get GPIOSR value.
AND        #0Fh, A         ;Mask off MSBs.
STLM       A, AR3          ;Store value of HD0-3 in AR3.
STM        #050h, GPIOSR   ;Set HD4-7 to 0101b.

           .   .   .
```

In this example, the four LSBs of the HPI data bus (HD0–3) are configured as general purpose inputs, while the four MSBs (HD4–7) are configured as outputs. The status of HD0–3 is read and stored in AR3; then the HD4–7 MSBs are set to 0101b.

Chapter 5

Enhanced 16-Bit Host Port Interface (HPI-16)

The enhanced 16-bit host port interface, also referred to as HPI-16, is an improved version of the HPI designed to interface a variety of host processors to the ’54x.

Topic **Page**

5.1 HPI-16 Operational Overview

The HPI uses a parallel bus interface that provides a host processor with access to internal DSP memory. The HPI is used to transfer data between the host and DSP allowing data buffering, real-time data logging, and message processing.

There are several variations of HPIs designed by Texas Instruments. Currently, the 8-bit HPIs support 16-bit data, but interface to an 8-bit bi-directional bus. Two consecutive byte transfers with byte identification for the most and least significant bytes are required to make a 16-bit data word.

The HPI-16 is an enhanced 16-bit version of the ‘54x 8-bit host port interface that, like its predecessors, functions as a slave and allows the host processor to access internal memory without DSP CPU intervention. It provides a full 16-bit bi-directional data bus that does not require byte identification. In addition, only one host transfer is required to complete the access.The HPI-16 was designed to interface a wide variety of host processors to a TMS320 DSP.

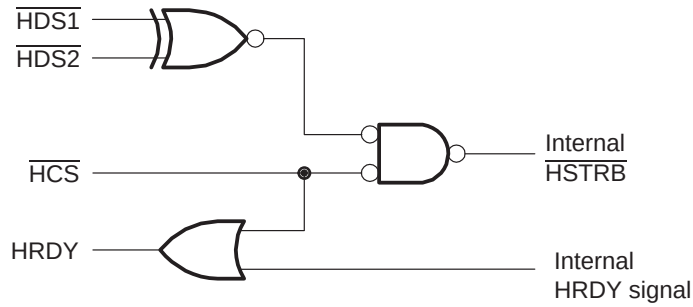
Key features of the HPI-16 include:

- ❏ 16-bit address bus for direct connection to the host address bus (18-bit address bus on the ’VC5420).
- ❏ 16-bit data bus that requires no byte sequencing.
- ❏ Flexible interface that includes several strobe and control signals suitable for a variety of 16-bit hosts.
- ❏ Both multiplexed and non-multiplexed operation for additional interfacing flexibility.
- ❏ Memory accesses that are synchronized with the direct memory access (DMA) controller providing access to the complete internal memory address range.
- ❏ Software polling of the HRDY pin.
- ❏ Software control of data fetching.
- ❏ Strobe and control signals.

There are two strobe inputs and an HPI select pin available for interfacing. These three input pins control the strobing of the HPI peripheral. The internal strobing logic, which is referred to as internal $\overline{\text{HSTRB}}$ throughout this chapter, functions as the actual strobe signal to the HPI peripheral.

As illustrated in Figure 5–1, the strobing logic is a function of three key strobe inputs — $\overline{\text{HCS}}$, $\overline{\text{HDS1}}$ and $\overline{\text{HDS2}}$. The $\overline{\text{HCS}}$, also known as the HPI chip select pin, must be low during strobe activity on the $\overline{\text{HDS}}$ pins. If $\overline{\text{HCS}}$ remains high, activity on the $\overline{\text{HDS}}$ pins is ignored and the HRDY pin goes high. The HRDY signal provides feedback to the host regardless of whether the current transfer is completed (HRDY goes high). Therefore, if the host is much slower than the HPI access and HRDY is not needed, $\overline{\text{HCS}}$ and one $\overline{\text{HDS}}$ strobe input can be tied together. This effectively chip-selects the HPI and provides the strobe simultaneously.

Figure 5–1. HPI Strobe and Select Logic



There are several ways to interface to a host, but all interfacing schemes fall into two general modes of operation: *multiplexed* and *non-multiplexed*. Multiplexed operation allows the host processor to control the HPI operation via two control signals called HCNTL0 and HCNTL1, while non-multiplexed mode performs data read/writes using only the HPI address and data buses. The HPI mode is hardware configurable using the HMODE pin. Multiplexed mode is selected when HMODE = 0.

Some devices may or may not support HMODE. You should refer to the device-specific data sheet to determine if both modes are supported. Table 5–1 contains a list of all possible HPI-16 pins and a description of their functions.

Table 5–1. HPI-16 Pin Descriptions

HPI-16 Signal	I/O/Z State†	Host Connection	Signal Description
HMODE	I	Usually static and tied high or low.	HPI mode select pin. The logic level of HMODE determines the operational mode of the HPI. A logic level 0 selects multiplexed mode (HCNTL0/1 are required). Logic level 1 selects non-multiplexed mode. This is a device-specific pin. To determine if HMODE is supported, see the device-specific data sheet.
$\overline{\text{HAS}}$	I	ALE – Address latch enable or address strobe	Address strobe. Used only in multiplexed mode. Host with multiplexed address/data buses connect this pin to their ALE. The falling edge of this input signal is used to latch the logic levels of the HR/$\overline{\text{W}}$, HCNTL0, and HCNTL1 pins. When used, the $\overline{\text{HAS}}$ signal must precede an active strobe ($\overline{\text{HCS}}$ or $\overline{\text{HDS}}$). Hosts with separate address/data buses must tie this signal high. As a result, the HR/$\overline{\text{W}}$, HCNTL0, and HCNTL1 inputs are latched on the falling edge of $\overline{\text{HDS}}$ when $\overline{\text{HCS}}$ = 0.
$\overline{\text{HCS}}$	I	Address or select line	HPI chip select. Serves as an HPI select input. $\overline{\text{HCS}}$ must be low for the HPI module to be selected. $\overline{\text{HCS}}$ may stay low between accesses. $\overline{\text{HCS}}$ normally precedes an active $\overline{\text{HDS}}$ strobe, but can be connected to an $\overline{\text{HDS}}$ for simultaneous select and strobe. For an illustration of the internal HPI strobing logic, see Figure 5–1.
$\overline{\text{HDS1}}$ $\overline{\text{HDS2}}$	I	Read strobe and write strobe or any data strobe.	HPI data strobe pins. Used for strobing data in/out of the HPI module. The direction depends on the logic level of HR/$\overline{\text{W}}$ signal. The $\overline{\text{HDS}}$ signals are also used to latch control information (if $\overline{\text{HAS}}$ is tied high) on the falling edge. During an HPID write access, data is latched into the HPID register on the rising edge of HDS. During read operations, these pins act as output-enables of the data bus.
HR/$\overline{\text{W}}$	I	R/$\overline{\text{W}}$ strobe	HPI read/write signal. Indicates to the HPI on the falling edge of $\overline{\text{HAS}}$ or $\overline{\text{HDS}}$ whether the current access is to be a read or write operation. A logic 1 indicates the transfer is a read-from-HPI operation, while a logic 0 is a write-to-HPI.

† I = Input, O = Output, Z = High-impedance

Table 5–1. HPI-16 Pin Descriptions (Continued)

HPI-16 Signal	I/O/Z State†	Host Connection	Signal Description
HCNTL0 HCNTL1	I	Address or control lines.	HPI access control inputs. The logic level of these pins is latched-in on the falling edge of $\overline{\text{HAS}}$ or $\overline{\text{HDS}}$. The four binary states of these pins determine the access mode of the current transfer (i.e. HPIC, HPIDinc, HPIA, and $\overline{\text{HPID}}$).
HA[*n*:0]	I	Host address bus	HPI address bus. The ***n***+1pins must be connected to the host address bus if the HPI is to operate in non-multiplexed mode. However, these pins can be tied to a logic level if the host has a multiplexed address/data bus and uses the HCNTL0/1 pins in multiplexed mode. The *n* value may vary depending on the DSP address range. For example, the ’VC5420 DSP has an 18-bit address range (A17:A0); therefore, *n* = 17.
HD[15:0]	I/O/Z	Host 16-bit data bus	HPI data bus. The HPI data bus is 16-bits wide and carries the data to/from the HPI module. These pins are Hi-Z when EMU1/$\overline{\text{OFF}}$ is active-low or when there are no read accesses occurring.
HRDY	O/Z	Asynchronous ready input.	HPI ready signal. Logic level 1 indicates the current transfer is complete. Logic level 0 indicates the HPI is not ready for the next access. The host must wait until HRDY goes high. The logic level can be software-polled by reading the HRDY bit in the HPIC register. The HRDY signal is forced high when $\overline{\text{HCS}}$ goes high. The pin is Hi-Z when EMU1/$\overline{\text{OFF}}$ is active-low.
$\overline{\text{HINT}}$	O/Z	Host interrupt pin.	Host Interrupt. The DSP can interrupt the host processor by writing a 1 to the HINT bit of the HPIC register. Before subsequent HINT interrupts can occur, the host must clear previous interrupts by writing a 1 to the HINT bit of the HPIC register. This pin is driven high when the DSP is in reset. This pin is active-low and inverted from the HINT bit value in the HPIC register. The pin is Hi-Z when EMU1/$\overline{\text{OFF}}$ is active-low.
$\overline{\text{HPIRS}}$	I	Control pin or tied high/low.	HPI reset pin. Places the HPI module in reset. No HPI accesses can occur, HD[15:0] are placed into the high-impedance state. This is a device-specific pin. To determine if $\overline{\text{HPIRS}}$ is supported, see the device-specific data sheet.

† I = Input, O = Output, Z = High-impedance

Table 5–1. HPI-16 Pin Descriptions (Continued)

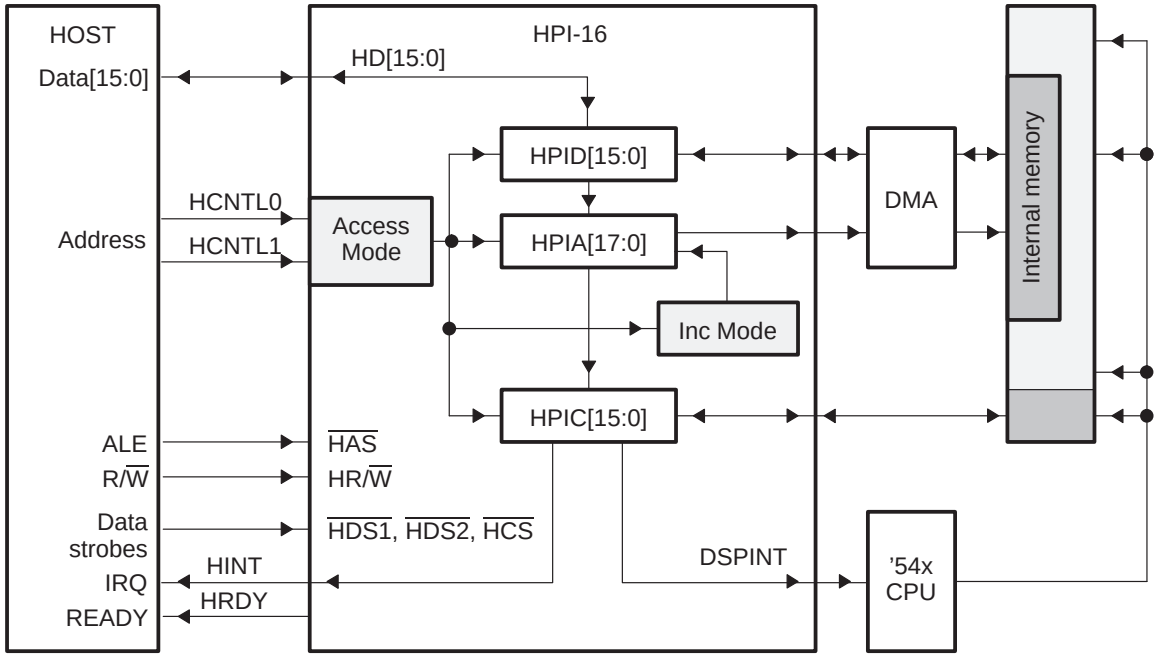
HPI-16 Signal	I/O/Z State†	Host Connection	Signal Description
HPIENA	I	Normally tied to V_{CC}	HPI enable. The HPIENA pin is used to completely deselect the HPI by shutting the module off. Connecting this pin to ground deactivates the HPI module and consumes no power. Otherwise, this pin should be tied to $+V_{CC}$. This is a device-specific pin. To determine if HPIENA is supported, see the device-specific data sheet.
HPI16	I	Usually static and pulled high or low	HPI16 select pin. This pin allows the HPI to support 8-bit host (0) or 16-bit hosts (1). For information on the 8-bit operation, see Chapter 4. This is a device specific pin. To determine if HPI-16 is supported, see the device-specific data sheet.

† I = Input, O = Output, Z = High-impedance

5.2 Multiplexed Mode

In multiplexed mode, the HPI operates with the HCNTL0/1 pins and is used for hosts that require the use of host/DSP interrupts, or when the address and data lines of the host are multiplexed. A host may control the HCNTL0/1 and HR/$\overline{W}$ pins in two ways. The first case is when the host drives the HCNTL0/1 and HR/$\overline{W}$ pins with dedicated output or address pins. In this case, the HCNTL0/1 and HR/$\overline{W}$ logic levels are latched on the falling edge of the host driven strobe ($\overline{HDS}$) signal. The second case is when a host has a multiplexed address and data bus. Since there is no dedicated address bus, the host uses the same bus to drive the control signals as it does for the data access. As a result, the HCNTL0/1 and HR/$\overline{W}$ signals are latched using the HPI $\overline{HAS}$ signal allowing time for the host to subsequently perform a write or read operation. Figure 5–2 shows a typical block diagram of a host controlling the HPI in multiplexed mode using $\overline{HAS}$.

Figure 5–2. Interfacing to the HPI-16 in Multiplexed Mode ('VC5420)



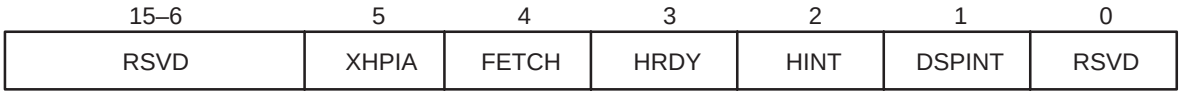
Notice that Figure 5–2 shows only one data bus and no address bus. The HCNTL and HR/$\overline{W}$ signals are driven by the host data bus and latched using the address latch enable pin connection to $\overline{HAS}$. Figure 5–2 and Table 5–2 illustrate how the HCNTL0/1 signals control the HPI. Higher data throughput can be achieved by initializing the HPIA register once, and then performing contiguous data memory accesses using the HPIA increment mode.

Table 5–2. HCNTL0/1 Modes

HCNTL1	HCNTL0	Host Access Mode
0	0	Host has R/$\overline{W}$ capability to the HPIC register.
0	1	Host has R/$\overline{W}$ capability to the HPID register. HPIA is automatically postincremented each time an HPI access is performed.
1	0	Host has R/$\overline{W}$ capability to the HPIA register.
1	1	Host has R/$\overline{W}$ capability to the HPID register. The HPIA register is not modified.

The host port interface configuration register (HPIC) is an internal configuration register accessible by the HPI and DSP CPU. This register provides the HPI and DSP with the capability to perform DSP-to-host and host-to-DSP interrupts and software polling of the HRDY pin, The DSP can access this register at location 0x2C, while the host can only access this register when in the specific HCNTL mode (00).

Figure 5–3. HPIC Register



The HPIC register is accessible by the host processor and the DSP CPU. Table 5–3 lists the bit-fields and a description of each.

Table 5–3. HPIC Bit Descriptions

Bit	Access By Host	Access By DSP	Description
DSPINT	R–0/W	R–0	Host-to-DSP interrupt. The host can interrupt the DSP by writing a 1 to DSPINT. This bit is always read as 0 by the host and DSP. DSP writes to this bit have no effect.
HINT	R/W–1	R/W–1	DSP-to-host interrupt. The DSP writes a 1 to the HINT bit to generate a host interrupt. The host interrupt HINT bit has an inverted logic level to the HINT pin. The host must write a 1 to HINT to clear the HINT pin. Writing a 0 to the HINT bit by the host or DSP has no effect.
HRDY	R	R	The logic level of the HRDY pin appears in this field. The host and DSP can read this bit for software polling of the HRDY pin. If HRDY = 0, the HPI-16 has not completed the current data access.

Table 5–3. HPIC Bit Descriptions (Continued)

Bit	Access By Host	Access By DSP	Description
FETCH	R–0/W	R–0	Host data fetch request. When a host writes a 1 to this bit, data located at the current HPIA address is fetched and loaded into the HPID register. This bit is always read as 0 by the host and DSP.
XHPIA	R/W	R	Extended address enable. When XHPIA=1, host writes to the HPIA register are loaded into the most significant bits, HPIA[*n*:16]. If XHPIA=0, host writes to HPIA are loaded into HPIA[15:0]. All *n*+1 address bits are incremented in the autoincrement mode. Reading the HPIA register is performed in the same manner. Only the host has access to this bit.

The HPIA register contains the address where the next data access occurs. The HPIA register is automatically incremented in the HCNTL0/1 = 10b mode. The FETCH = 1 bit issues a request to read data pointed to by the HPIA register. The FETCH bit can be used with the HRDY bit to perform software polling to determine when the internal access is complete. The FETCH and HRDY pins are useful when an update to the HPID register is desired. The FETCH bit is always read as 0. Performing a FETCH does not increment the HPIA register.

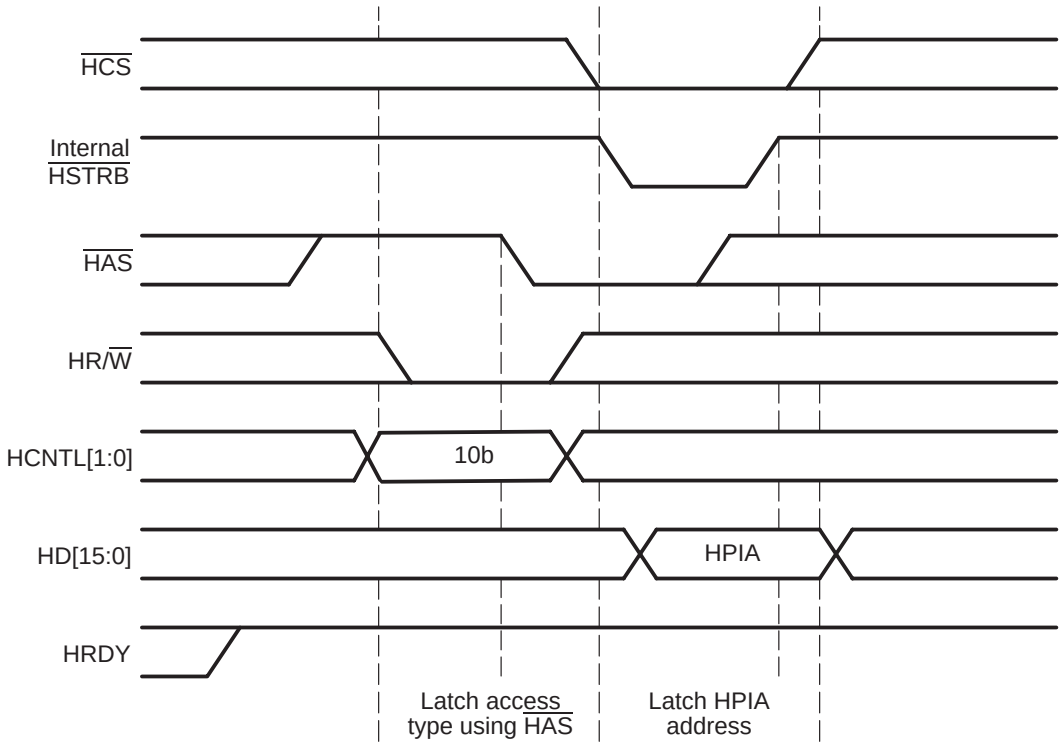
Interrupts between the host and DSP are possible by setting and polling DSPINT and HINT bits in the HPIC register. The DSPINT bit, when set to 1 by the host, posts an interrupt to the DSP CPU. The DSP can process this interrupt if desired by enabling the interrupt in the CPU's IMR register. The DSP can post a host interrupt by writing a one to the HINT bit. In this case, the host must also perform a write to the HINT bit to acknowledge the previous HINT request. The DSPINT bit is always read as zero. Writing zeros to the HINT and DSPINT bits has no effect.

The XHPIA bit logic level determines which address bits in the HPIA are initialized. Because the data bus is 16 bits and the address reach is greater than 16 bits (18 bits on the 'VC5420), the XHPIA bit loads the lower 16 bits of the HPIA when set to zero, and the extended address bits (>16) are loaded when XHPIA = 1. Initialization of all address bits is recommended before performing data accesses. XPIA must be changed in the same manner to read the contents on the HPIA register.

5.2.1 Host Accesses With $\overline{\text{HAS}}$

The host address strobe ($\overline{\text{HAS}}$) is only valid in multiplexed mode. The $\overline{\text{HAS}}$ signal enables glueless interfacing for host processors with multiplexed address and data buses. The falling edge of $\overline{\text{HAS}}$ is used to latch the HCNTL0/1 and HR/$\overline{\text{W}}$ states into the HPI. First, the host latches the transfer mode in the HPI by driving the HCNTL0/1 and HR/$\overline{\text{W}}$ pins to the desired access mode. $\overline{\text{HAS}}$ is not gated by $\overline{\text{HCS}}$; and therefore, allows time for the host to perform the subsequent access. The $\overline{\text{HAS}}$ signal may be brought high after the $\overline{\text{HDS}}$ is driven low, indicating the data access is about to occur. $\overline{\text{HAS}}$ is not required to be high, but must eventually transition high when there is a change in access type. Figure 5–4 illustrates a write-access using $\overline{\text{HAS}}$.

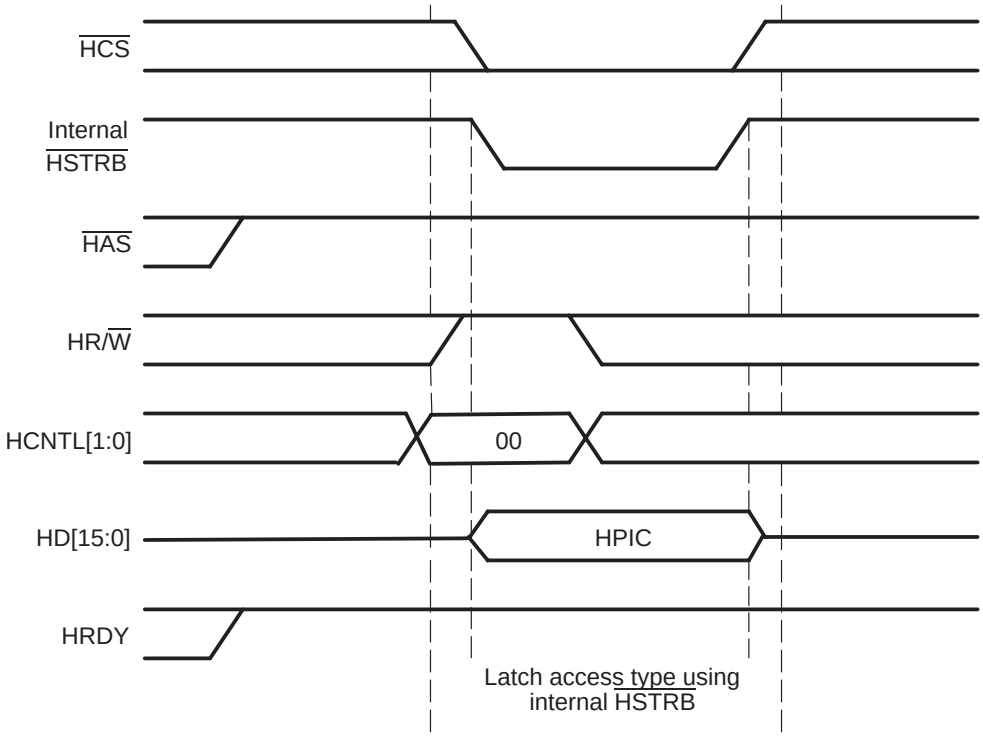
Figure 5–4. HPIA Write Using $\overline{\text{HAS}}$



5.2.2 Host Accesses Without $\overline{\text{HAS}}$

In cases where the host processor has dedicated signals (or bit I/O) capable of driving the HCNTL0/1 pins, the $\overline{\text{HAS}}$ signal is not required. These dedicated signals can be directly connected to the HCNTL0/1 and HR/$\overline{\text{W}}$ pins. This simplifies the access considerably because there is no setup and strobing of the HCNTL0/1 data relative to $\overline{\text{HAS}}$. For example, an HPIC read without using $\overline{\text{HAS}}$ is performed by driving the HCNTL0/1 value (00) and then strobing the HPI. After the falling edge of the internal $\overline{\text{HSTRB}}$ signal, the HPI drives the HPIC register data onto the data bus. Figure 5–5 shows the read-access sequence without using the $\overline{\text{HAS}}$ signal.

Figure 5–5. HPIC Read Without Using $\overline{\text{HAS}}$



5.2.3 Autoincrement Operation

All of the HPI peripherals include a feature designed to increase the data throughput of the HPI. This feature, called *autoincrement*, is used to pre-fetch data and point to the next higher data location (post-modified) after the current access is complete. This feature automatically modifies the HPIA register so that the host can perform consecutive HPI read/write transfers to a contiguous memory space without having to modify the HPIA register contents. This feature is enabled when the HCNTL0/1 mode = 01b.

Figure 5–6. HPID Read Using Autoincrement

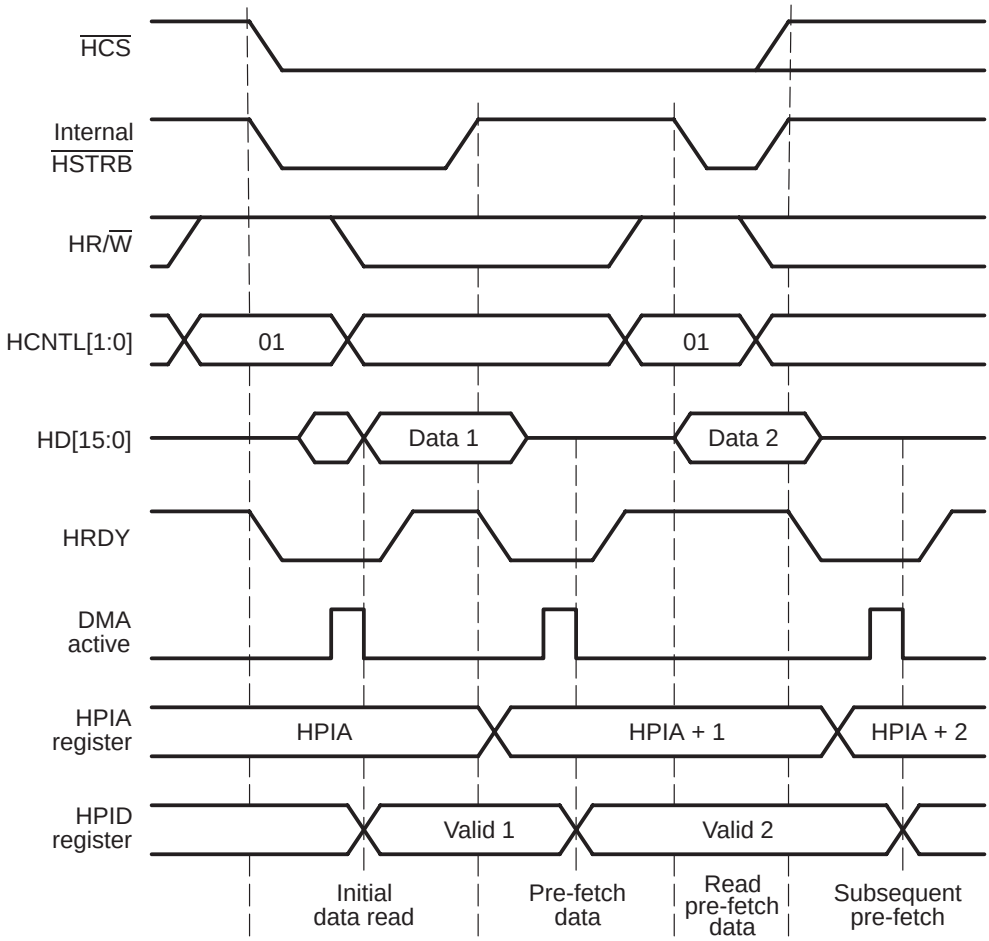
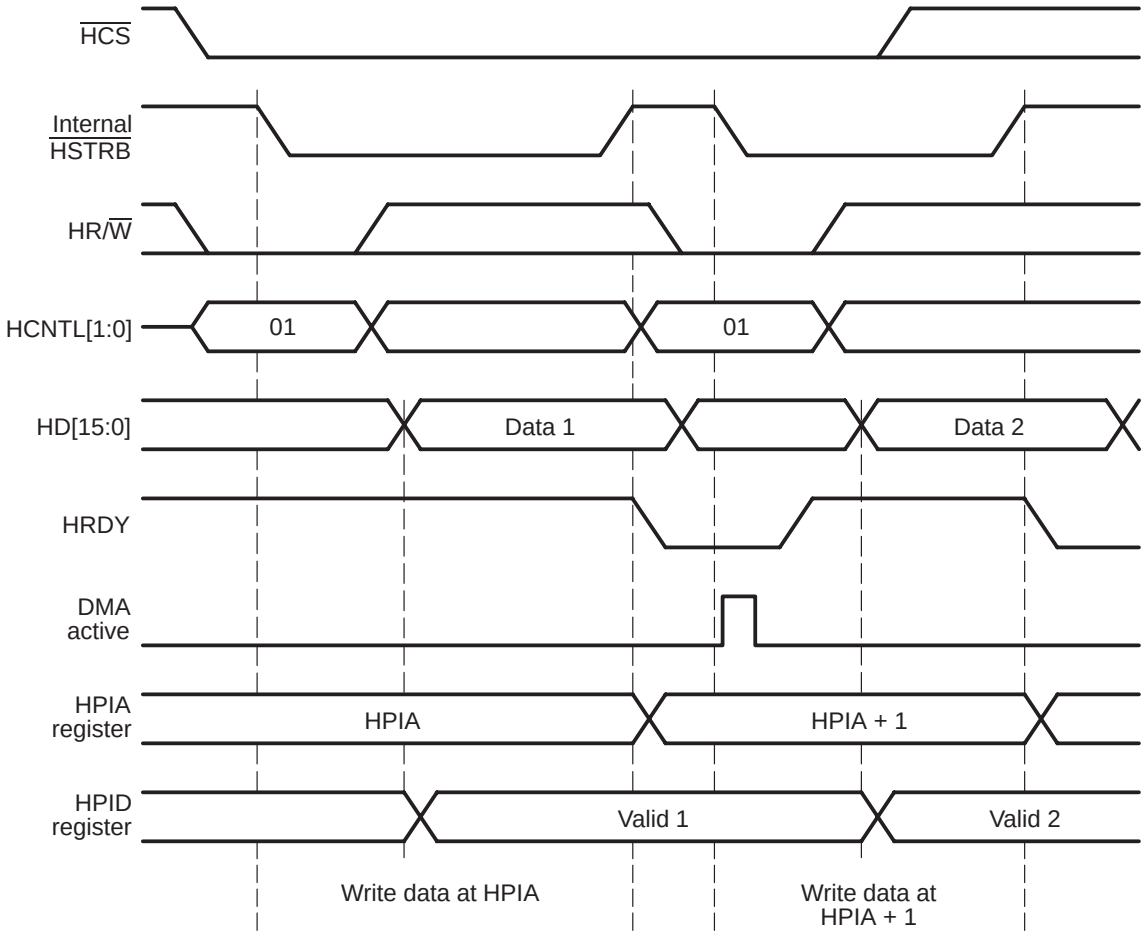


Figure 5–7. HPID Write Using Autoincrement



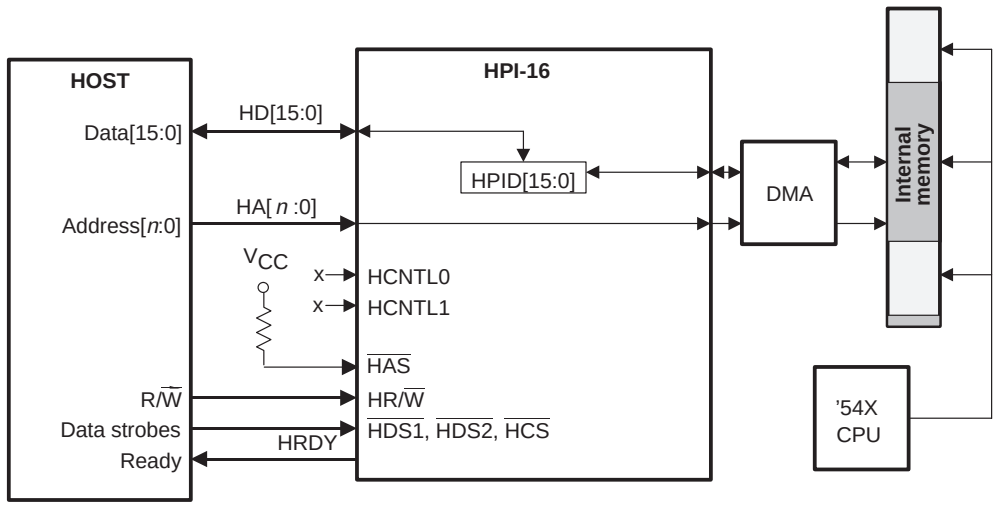
Due to the nature of pre-fetching, post-modified reads can cause invalid (old) data to be read by the host CPU. This occurs when the DSP has updated the next higher data location after the host has performed a read access. As a result of the post-modify and pre-fetch mechanism, the next data read by the host will not have the updated data value. If both DSP and HOST are writing to the same data locations, it is suggested that FETCH be performed before reading the data. FETCH is performed by setting the FETCH bit in the HPIC register. Subsequently, the HPIC register can be read to poll the HRDY bit before reading the data from the HPI. Figure 5–6 and Figure 5–7 show read and write operations with the autoincrement feature. For the initial read (with autoincrement) and for all reads without the autoincrement feature, the HRDY signal goes not-ready when the access begins (falling edge of internal $\overline{\text{HSTRB}}$). Subsequent reads in increment mode perform data pre-fetching, which increases the HPI throughput.

Write operations are always performed on the rising edge of the internal $\overline{\text{HSTRB}}$ signal. The increased throughput on write operations is realized when making consecutive writes and using the not-ready state of HRDY to initiate the next write operation. Since the host must transition and hold $\overline{\text{HSTRB}}$ high for at least 10ns before making the next write, the hold time can be satisfied while HRDY is not-ready. This is illustrated in Figure 5–7.

5.3 Non-Multiplexed Mode

In *non-multiplexed* mode, hosts with separate address and data buses can access the HPI data (HPID) register by using the 16-bit bi-directional data bus and the HPIA address register via the *n*-bit address bus. There is no HPIC register in the non-multiplexed mode; therefore, no interrupts (DSPINT and HINT) can be passed between the host and DSP. Due to the fact that a host address bus is available to drive the HPIA, there is no need for the autoincrement feature, which precludes the necessity of the FETCH and XHPIA bits. Software polling of the HRDY is not available, but the HRDY pin is fully functional and capable of holding off a subsequent host access before the access is completed internally to the DSP. The HRDY pin functions as previously stated in the multiplexed mode. Figure 5–8 illustrates how to interface to the HPI in non-multiplexed mode.

Figure 5–8. Interfacing to the HPI-16 in Non-Multiplexed Mode ('VC5420)



The HPI select pin ($\overline{\text{HCS}}$) and strobe signals ($\overline{\text{HDS1}}$ and $\overline{\text{HDS2}}$) function as stated in the multiplexed mode. The $\overline{\text{HAS}}$, HCNTL0, and HCNTL1 pins are not included because they have no function in the non-multiplexed mode.

A typical access in non-multiplexed mode is shown in Figure 5–9 and Figure 5–10. Data throughput is the same as in the multiplexed mode when not using the $\overline{\text{HAS}}$ signal or autoincrement feature. Bus arbitration with the DMA controller is handled in the same way.

Figure 5–9. HPID Read in Non-Multiplexed Mode

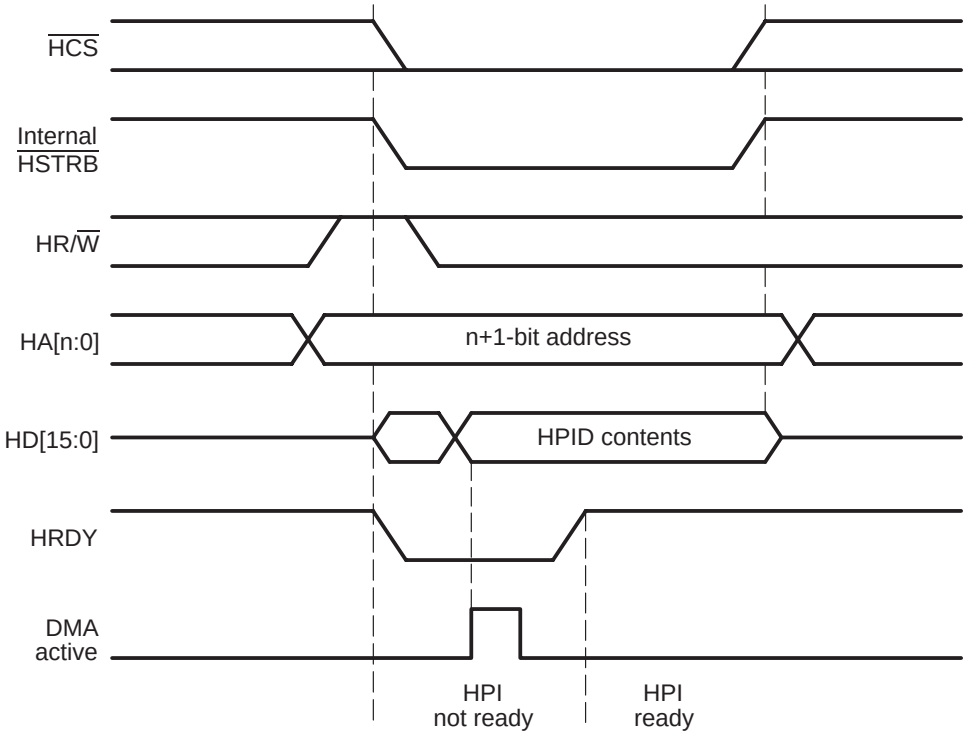
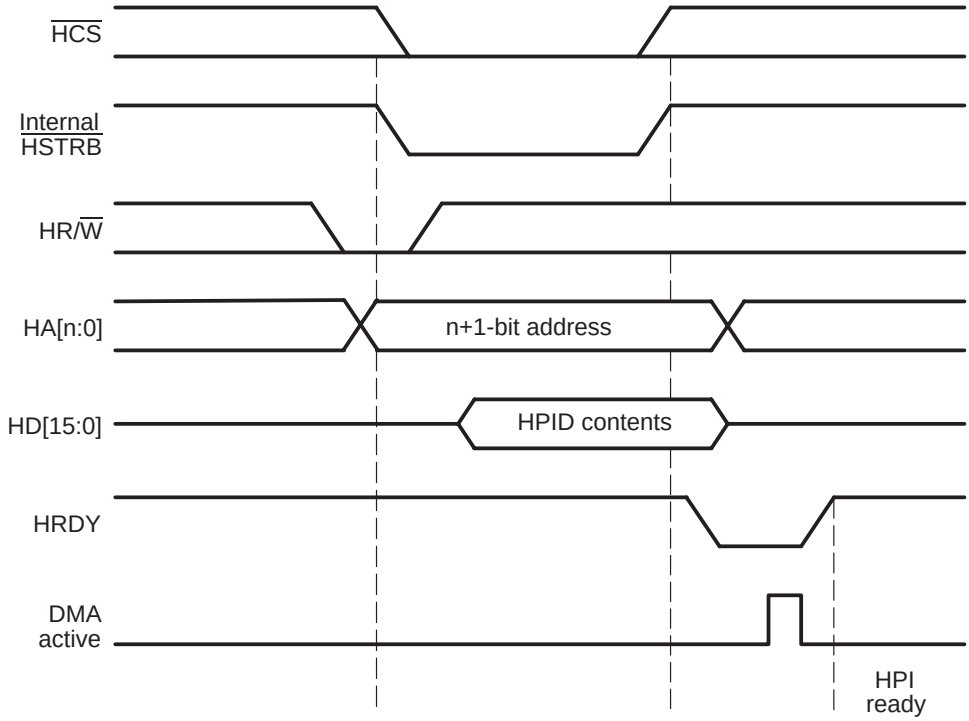


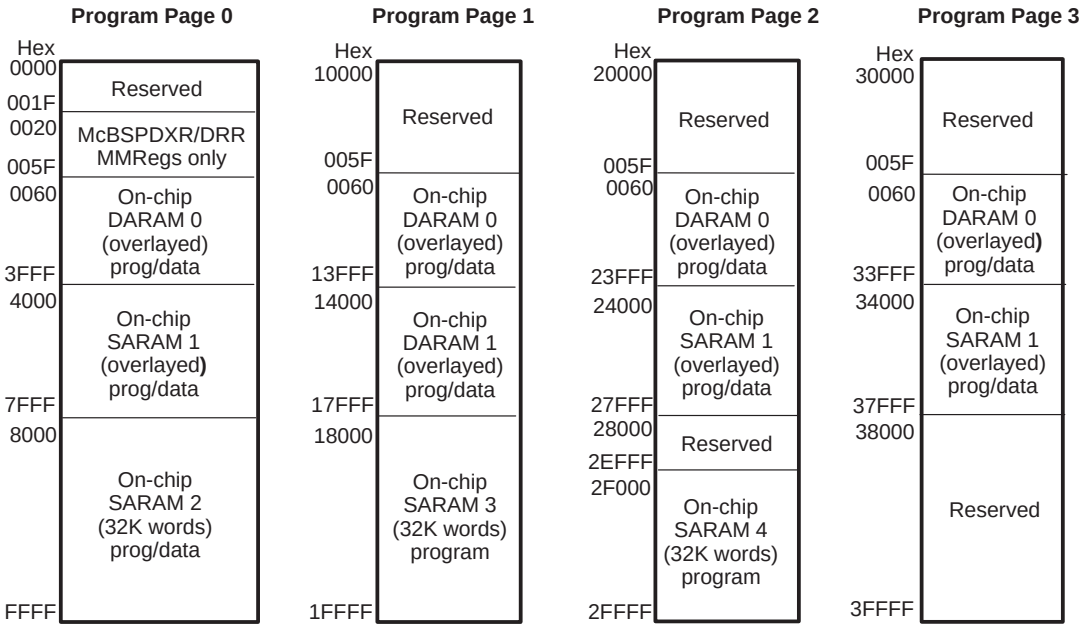
Figure 5–10. HPID Write in Non-Multiplexed Mode



5.4 HPI-16 Memory Map

Depending on the logic level of the HMODE pin, the HPI-16 peripheral can perform accesses to DSP memory (internal memory only on the 'VC5420) in multiplexed or non-multiplexed modes. As previously noted, multiplexed mode uses the HPIA register for access to the memory map. Non-multiplexed mode uses the *n*-bit HPI address bus to directly access the memory map. Both of these modes are capable of accessing all DSP memory locations that are accessible via the DMA controller. Figure 5–11 shows the HPI memory map for the 'VC5420, a device having an 18-bit address bus.

Figure 5–11.'VC5420 Memory Map Relative to the HPI



Note: I/O Space is not accessible.
All internal memory is divided into 8K blocks with the exception of the 4K word block on P2 (0x2F000–0x2FFFF)
DROM and OVLY bits do not affect the memory map.

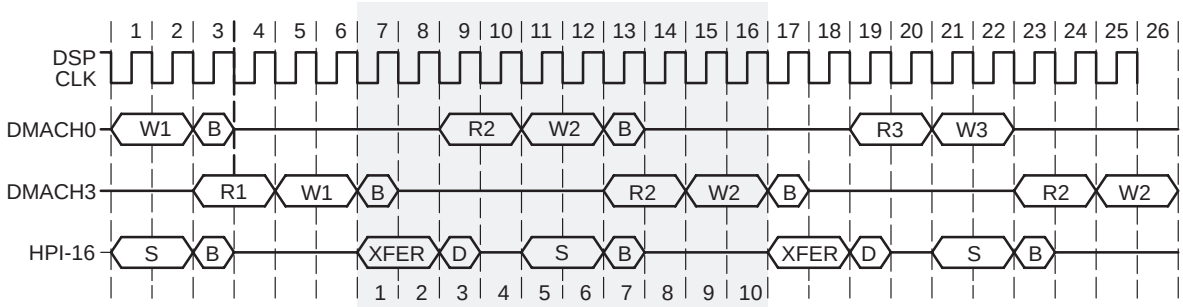
The HPI peripheral has no knowledge of any memory configuration bits controlled by the CPU. For this reason, the values of the OVLY and DROM bits do not affect the data location relative to the HPI. In the case of the 'VC5420, accesses to the internal DSP memory are always considered overlayed program and data. There is no way to distinguish between program and data spaces. I/O space is not accessible by the HPI.

You should refer to the device-specific data sheet for information on other HPI memory maps and configurations.

5.5 HPI-16 and DMA Interaction

The HPI-16 operates with the DMA controller granting the host access to any memory location (internal memory only on the ’VC5420). Each time the host CPU request an access, the DMA finishes any current transfers and releases the DMA bus to the HPI module. As a result, there are latencies associated with HPI accesses due to other DMA activity. The host CPU can access the DSP memory (writes) at the maximum rate of once every six DSP cycles, if no other DMA channels are active. However, HPI throughput decreases to one transfer every 10 cycles when there are two or more active DMA channels. The HPI has the highest priority, including the CPU. As a result, the HPI is granted access even if the CPU or other DMA channels are competing for the same memory location. Figure 5–12 illustrates the HPI access sequence relative to the DMA controller.

Figure 5–12. HPI and DMA Interaction



Note: S = Sync to DSP clock
B = Bid for DMA bus
D = Delay for propagation
Rn = Read cycle on the n^{th} access
Wn = Write cycle of the n^{th} access

The HPI transfer latency increases if one or more DMA channels are active; however, the HPI latency never increases beyond the latency of one active DMA channel. The HPI bus grants are interlaced between the revolving DMA channels. For this reason, data throughput on other DMA channels is sacrificed if high HPI data rates are required. It should be noted that even without HPI intervention, it is possible to lock out low priority DMA channels if all high priority DMA channels are continuously serviced. You should refer to the device-specific data sheet for HPI read and write transfer latencies for the various ’54x devices.

The HPIA register is not susceptible to latencies due to DMA controller arbitration. Accesses to the HPIA register do not use the DMA bus because this register is part of the HPI peripheral. HPIC register accesses have no latencies when writing values that have the DSPINT and HINT bits set to zero. HRDY is never driven low for these accesses because subsequent accesses can begin immediately. However, writing a one to the DSPINT or HINT bits introduces a latency due to the required access to the DSP CPU module. Consequently, the HRDY signal is held low for three DSP clock cycles.

5.6 HPI-16 Operation During Reset

The HPI-16 peripheral can operate while the DSP CPU is in reset. On the ’VC5420 device, the RESET signals are separated between core CPUs and the HPI. The HPI is placed in reset when the dedicated HPI reset pin, $\overline{\text{HPIRST}}$, is equal to zero. When the HPIRST signal is one, the HPI has full access to the internal memory space of the DSP. Accesses still require six DSP clock cycles, allowing the maximum data throughput of 33MB/s @ 100MIPS.

The host processor can place the DSP in reset by controlling the $\overline{\text{RS}}$ pin. In this case, the host must complete any current memory access and wait six DSP clock cycles before driving the $\overline{\text{RS}}$ pin low. During reset, the internal memory is available to the host. To release the DSP from reset, the host can drive the $\overline{\text{RS}}$ pin high, but is required to wait at least 20 DSP clock cycles before making subsequent memory accesses.

An additional method for releasing the DSP from reset is to leave each core’s $\overline{\text{RS}}$ pin high and cycle $\overline{\text{HPIRST}}$ low, then high. When this occurs, the host can write to location 0x2F (any value), which will internally reset that particular core.

5.7 HPI-16 Operation During IDLEn

The HPI-16 can continue to operate during the IDLE 1 and IDLE 2 states by using special clock management circuitry. This circuitry turns on relevant clocks to perform synchronous memory accesses and then turns these clocks off to conserve power. The host processor can wake the DSP from the IDLE states when in multiplexed mode via the DSPINT bit in the HPIC register. In non-multiplexed mode, external interrupts (ex. $\overline{\text{INT0}}$, $\overline{\text{INT1}}$) can be used to wake the DSP.

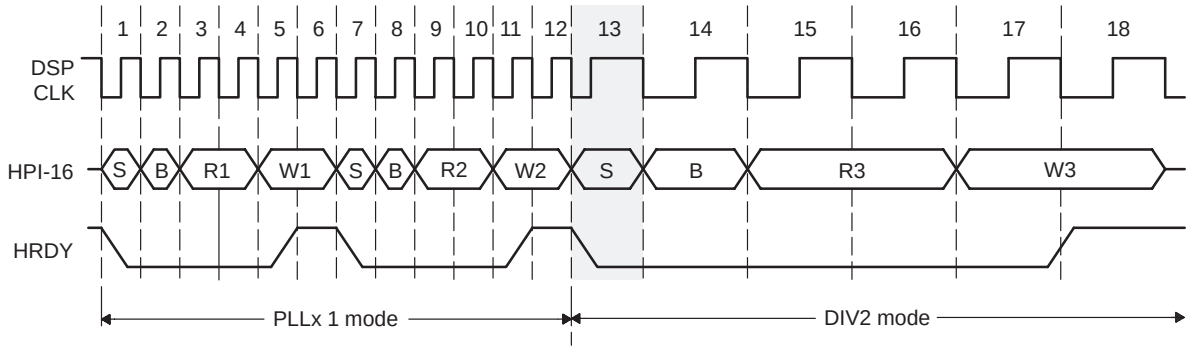
The host processor can wake the DSP from IDLE 3 by writing a one to the DSPINT bit of the HPIC register. Memory accesses during IDLE 3 are not supported.

5.8 Changes in DSP Clock Modes That Affect the HPI-16

As stated earlier, the HPI is synchronized to the DSP clock. The HPI can access internal DSP memory every six DSP clock cycles. Therefore, changes to the DSP clock mode affect the time required to complete memory accesses. If the DSP clock is slowed, the HPI access rate is slowed proportionally.

For example, if the clock mode is changed from a PLL mode to a DIV mode, the host can use the HRDY pin to effectively insert host wait-states. In Figure 5–13, the relative timing for a change from multiply-by-1 to DIV mode is shown. Notice that by using the HRDY pin, wait-states are effectively inserted to hold off the host until the current access is complete.

Figure 5–13. HPI-16 Operation During the PLL to DIV Clock Mode Change



Note: S = Sync to DSP clock
B = Bid for DMA bus
Rn = Read cycle on the n^{th} access
Wn = Write cycle of the n^{th} access

The effective HPI transfer rate is increased by changing the DSP clock to a PLLxn multiply mode. In this case, fewer host wait-states are required. In fact, as the multiplier mode is increased, it is possible that the HPI transfer rate will exceed the latency of the host access rate. When this occurs, no host wait-state is required.

Chapter 6

Interprocessor Communications

This chapter describes interprocessor communications associated with multi-core DSPs, and includes core-to-core FIFO communications and external memory interface-to-host port interface (EMIF-to-HPI) communications.

Topic **Page**

6.1 Communication Within Multi-Core DSPs

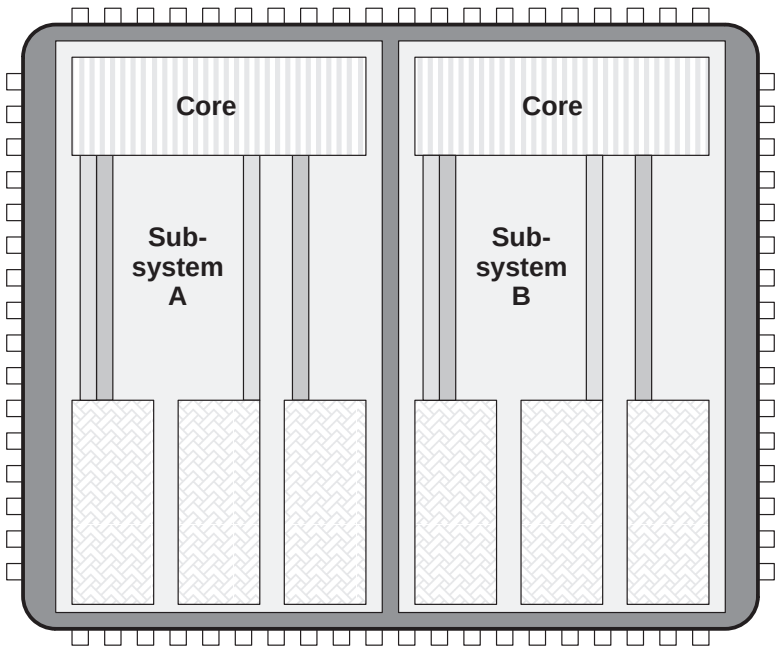
Multi-core (multi-CPU), multi-subsystem DSPs were introduced by Texas Instruments in 1999. These devices are capable of doubling performance either by executing duplicate application code on multiple cores or by running one application that divides tasks between the cores. The subsystems, each containing its own core, are independent of each other.

Dividing tasks between multiple cores requires data transfers and message processing between the subsystems. The methods for implementing interprocessor communications include:

- ❏ Use of FIFOs
- ❏ Host port interfaces (HPIs)
- ❏ Multichannel buffered serial ports (McBSPs)
- ❏ Mutual memory spaces

This chapter explains the implementation of these task management methods using the ’VC5420 as an example. For device specific information, you should refer to the appropriate data sheet.

Figure 6–1. ’VC5420 — The 2-Subsystem DSP

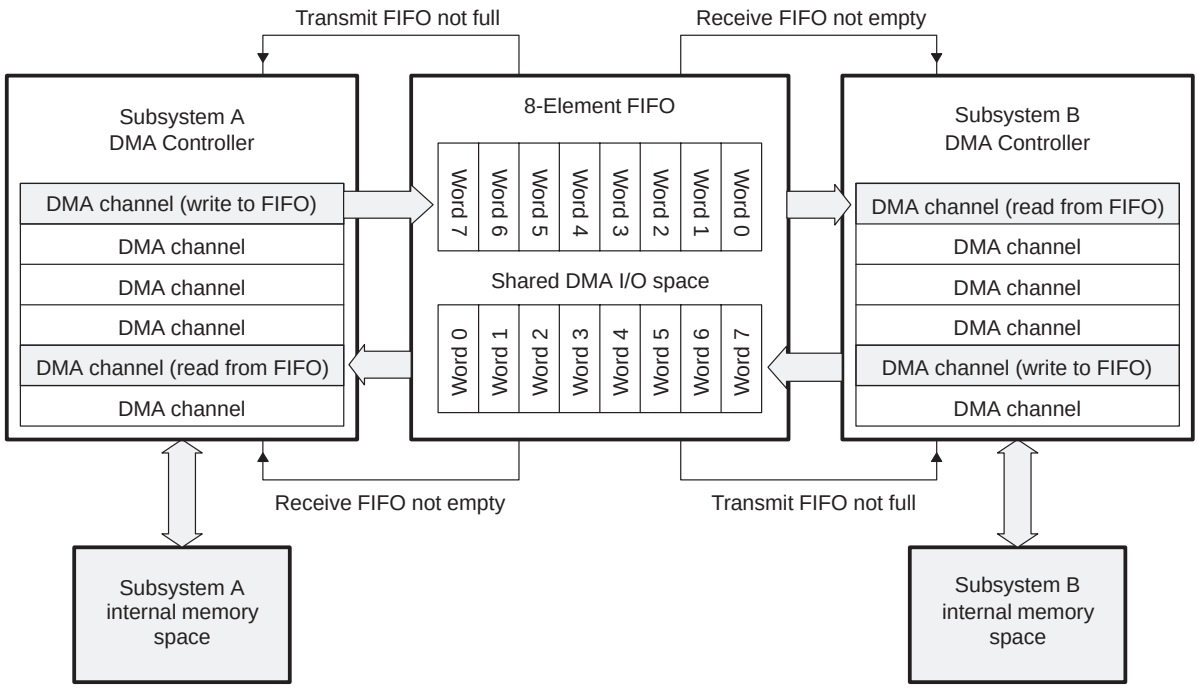


6.2 The Bi-Directional FIFO

Communications between subsystems can be accomplished using the first-in, first-out (FIFO) method described in this section. A second method is discussed in section 6.4 on page 6-10.

The FIFO peripheral consists of two uni-directional, 8-element-deep channels. Each channel supports 16-bit data (element) transfers and is dedicated to either transmit or receive operation. In the case of the 'VC5420, the FIFO is shared by both subsystems. One FIFO channel is dedicated for transmission of data from A to B, and the other channel is dedicated for data transmission from B to A. The subsystem DMA controller that is transmitting data can write eight words to the FIFO before the FIFO becomes full; however, if the receiving subsystem reads from its receive channel simultaneously, the transmitting subsystem can continue indefinitely.

Figure 6–2. 'VC5420 FIFO Configuration



Special internal handshaking is provided for handling overrun and underrun conditions of the 8-element FIFO buffer; and for this reason, it is impossible to lose data by either condition. This implies that data throughput is dependent upon both transmit and receive subsystem activity.

On the 'VC5420, handshaking is performed between the FIFO and DMA peripherals, and no intervention by the CPU is required once the DMA receive

and transmit channels are configured and enabled. Example 6–1 shows DMA channel information that is used to configure DMA channels 0 and 1 for FIFO transmit and receive, respectively.

Example 6–1. DMA Channels 0 and 1 Configured for FIFO Transmit and Receive

Transmit – DMA channel 0
DMSRC0 = #200h –source address
DMDST0 = #xxxh –this is a don't care
DMCTR0 = #0Fh –element count (buffer size)
DMSFC0 = #8000h –DMA sync mode
DMMCR0 = #0142h –DMA increment mode, memory space information.

Receive – DMA channel 1
DMSRC1 = #xxxh –this is a don't care
DMDST1 = #400h –destination address
DMCTR1 = #0Fh –element count (buffer size)
DMSFC1 = #7000h –DMA sync mode
DMMCR1 = #085h –DMA increment mode, memory space information.

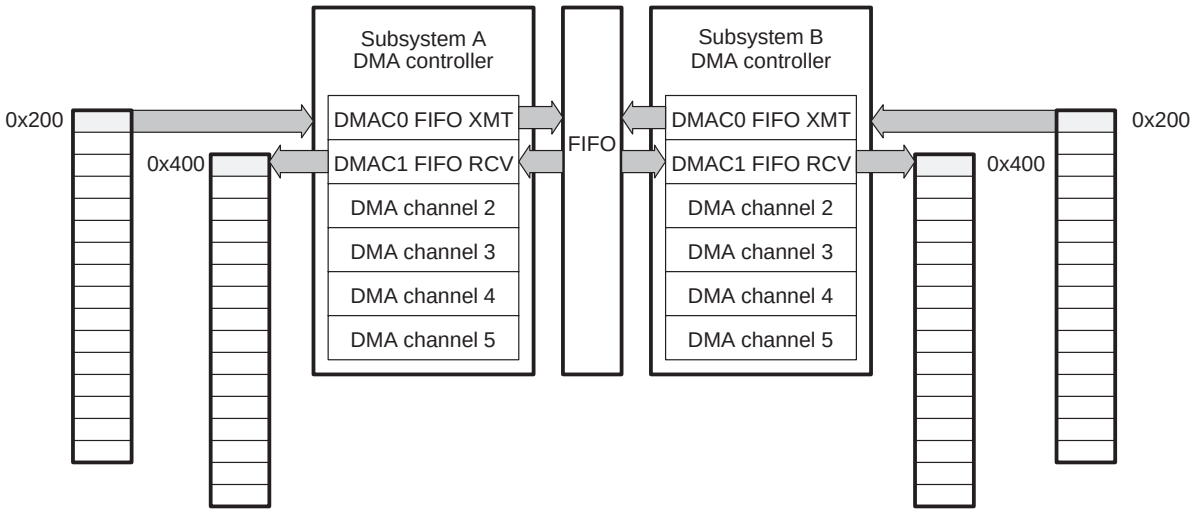
Table 6–1. DMA Configuration to Support FIFO Transfers

Transmit Channel (DMA Channel 0)		
	Source	Destination
Location of source and destination	internal memory	FIFO
Memory space	data/prog	I/O
Memory address	200h	don't care
DMA index mode (buffer pointer increment mode)	post-increment	no increment
Internal DMA sync event	transmit FIFO not full	
Element count (buffer size)	0Fh	
Interrupt selection	DMA CH0 = IMR/IFR bit 6	

Receive Channel (DMA Channel 1)		
	Source	Destination
Location of source and destination	FIFO	internal memory
Memory space	I/O	data/prog
Memory address	don't care	400h
DMA index mode (buffer pointer increment mode)	no increment	post-increment
Internal DMA sync event	receive FIFO not empty	
Element count (buffer size)	0Fh	
Interrupt selection	DMA CH1 = IMR/IFR bit 7	

Figure 6–3 illustrates the physical setup for the DMA configuration shown in Table 6–1. Transfer buffers (located in either internal data or program space) must be initialized so that the DMA can process the data stream without CPU intervention. The data buffers have a total of 32 words (a 16-word receive buffer and a 16-word transmit buffer) and are located in internal memory at 200h and 400h.

Figure 6–3. DMA Configuration for FIFO Operation



The FIFO is located in a shared DMA I/O space; that is, both subsystem's DMA share a common I/O space. As a result, the DMA memory space selection must be configured as I/O space to access the FIFO. The memory space selection is made in the DMMCR register.

The source/destination address that points to the FIFO is a *don't care*, since all writes and reads are from one FIFO memory location. Consequently, the indexing mode for the FIFO side of the DMA should be configured for no-increment. The buffer size, also known as the element count in the DMCTR register, is always programmed as buffer size-1. The element count is used for both source and destination buffers.

Interrupts for DMA channels 0 and 1 are mapped to IMR bits 6 and 7. However, in Example 6–1 and Table 6–1, no interrupts are programmed or enabled. To enable interrupts, the DMA channels must be mapped to the IMR bits as shown. In addition, the DMA must be programmed to generate interrupts at a specific time while filling the FIFO.

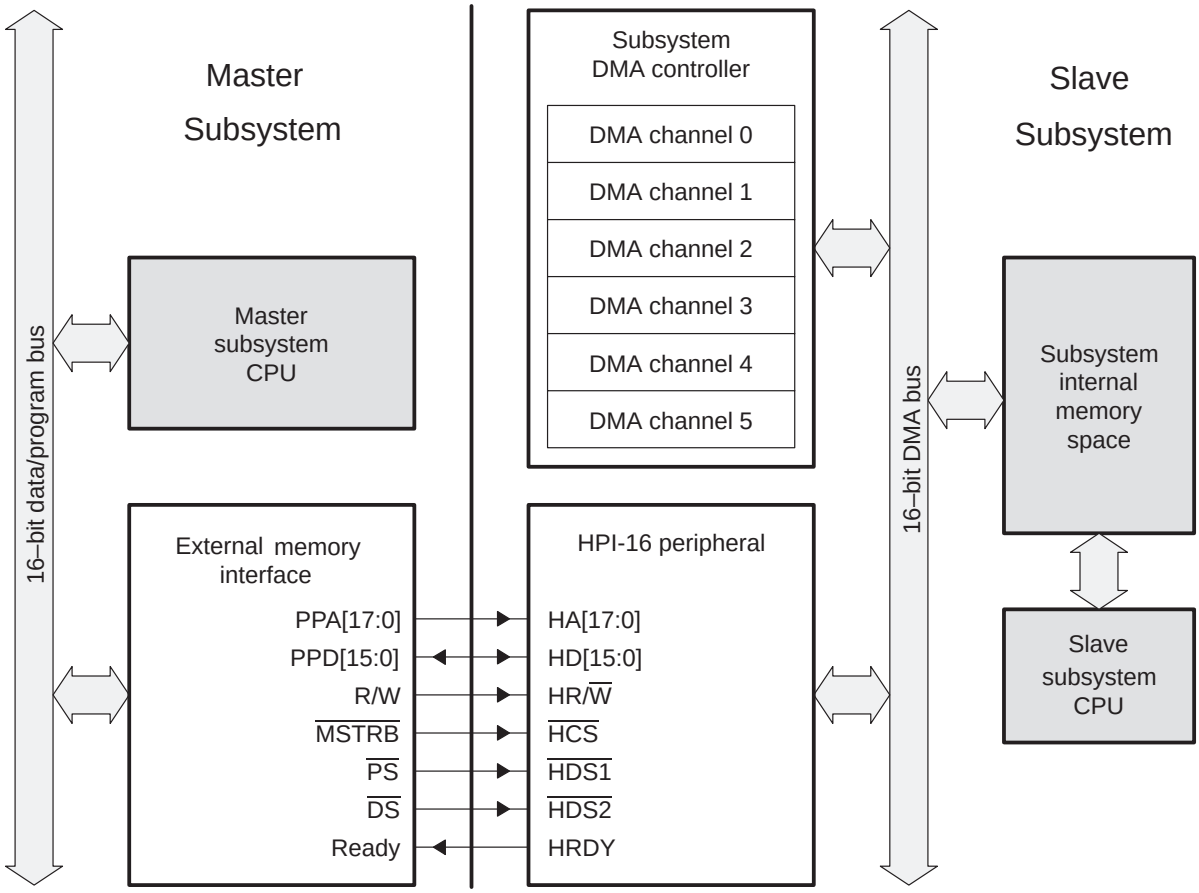
Interrupts can be generated when the FIFO buffers become 50% or 100% full. These interrupts are turned on in the DMA's DMMCR register and enabled in the CPU's interrupt mask register (IMR). If an interrupt is generated on a half-full buffer, the interrupt is generated when the receipt of the fourth word is completed. A full-buffer interrupt occurs after receipt of the eighth word is completed.

The latency to write a word to the FIFO is four CPU cycles. If the receive subsystem immediately reads the word, it requires four additional CPU cycles to read the data from the FIFO. Therefore, the maximum throughput is one 16-bit word every eight CPU cycles. Additional latency can be incurred when the DMA controller is processing other DMA channels or HPI transfers.

6.3 Accessing the HPI-16 From External Memory Space

The standard HPI, HPI-8, and HPI-16 peripherals are all designed to connect to another DSP's external memory space. Since the HPI is always a slave device, the master DSP must perform writes and reads to the external memory space to strobe the HPI peripheral. On multi-core DSPs, this connection is made internally. Figure 6–4 illustrates the internal connection between the external memory interface and the HPI-16 peripheral on the 'VC5420.

Figure 6–4. HPI-16 and External Memory Interface Connection ('VC5420)



The HPI-16 is a 16-bit device capable of performing data accesses from internal memory space. Due to the 18-bit address bus, the HPI-16 can address the full address range. The HPI-16 operates in non-multiplexed mode and is activated by strobing the $\overline{\text{HDS}}$ lines. The master device must perform an access to the external memory. The external program/data memory interface signals are mapped to the HPI-16 control pins.

The subsystems of the multi-core DSP share the HPI-16 peripheral and external memory interface. Because of this, external arbitration logic is required if direction of the communication (determined by the SELA/B pin) must be changed. The HPI-16 must operate in the non-multiplexed mode requiring that the HMODE pin be pulled high. The XIO pin must also be pulled high to enter a special state where the EMIF pins are connected internally to the HPI-16 pins. Table 6–2 summarizes the operation for each pin configuration.

Table 6–2. EMIF/HPI-16 Modes

HMODE	SELA/B	EMIF Modes (XIO=1)
1	0	EMIF-to-HPI master is subsystem A EMIF-to-HPI slave is subsystem B
1	1	EMIF-to-HPI master is subsystem B EMIF-to-HPI slave is subsystem A

An additional requirement when changing the direction of communication is the value of the MP/$\overline{\text{MC}}$ bit in the PMST register. The handshaking between subsystems must not only request/acknowledge a state change of the SELA/B pin, but also change the value of the MP/$\overline{\text{MC}}$ bit in both subsystems. The only time when initialization is not required is after reset. Table 6–3 lists the MP/$\overline{\text{MC}}$ bit levels at reset as a function of the XIO, HMODE, and SELA/B pins.

Table 6–3. MP/$\overline{\text{MC}}$ Bit Levels at Reset

'5420 Pins			MP/$\overline{\text{MC}}$ Bit	
XIO	HMODE	SELA/B	Subsystem A	Subsystem B
0	X	X	0	0
1	0	X	1	1
1	1	0	1	0
1	1	1	0	1

After reset, it is up to the handshaking logic and subsystems to correctly configure the SELA/B pin and MP/$\overline{\text{MC}}$ bit levels. The MP/$\overline{\text{MC}}$ value of subsystem A must always be set to the inverted logic level of the SELA/B pin. Furthermore, subsystem B must always have the MP/$\overline{\text{MC}}$ bit set to same logic level as the SELA/B pin.

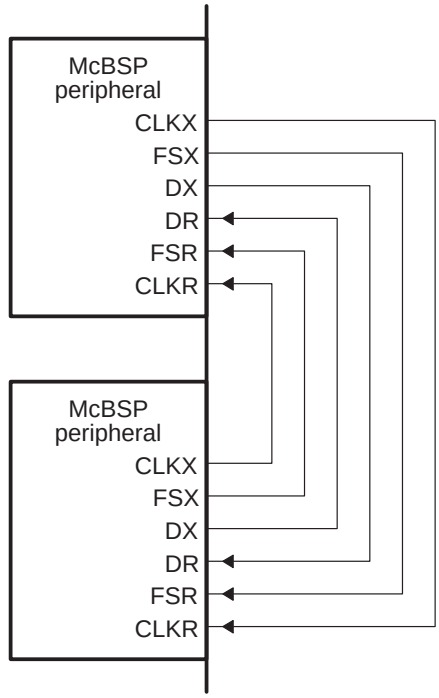
Due to the state of the XIO and HMODE pins, the master device must execute from external memory. The slave subsystem is configured to operate in HPI mode; therefore, all internal memory is enabled. As a result, the external memory location that the master subsystem accesses is the location that is accessed on the slave subsystem.

The maximum throughput of the EMIF/HPI-16 channel is one transfer every six DSP clock cycles, assuming there is no additional DMA activity. Transfer rates can decrease to one word every 14 DSP clock cycles if there is additional DMA activity. However, the master device does not need to program wait-states because the HPI ready pin (HRDY) is connected to the DSP CPU READY pin.

6.4 Subsystem Communication Using McBSP

Even though the 'VC5420 multi-core DSP does not support McBSP-to-McBSP communication by design, simply connecting a McBSP from each subsystem externally provides a very robust interprocessor communication mechanism. Figure 6–5 illustrates this external connection. There is only a 6-line connection required to implement the bi-directional/full-duplex communication channel. No master/slave configuration is required.

Figure 6–5. McBSP-to-McBSP Connection



The transmit section of the McBSPs must provide a shift-clock and frame-sync to the receive section of the other subsystem's McBSP. Therefore, the McBSP CLKX and FSX pins are configured as output signals. The CLKR and FSR pins are configured as input signals.

The McBSP can be programmed to support many different data transfer modes. The subsystem's DMA controller can be used for autobuffering transmit and receive data buffers. In addition, the McBSP peripheral can be programmed to generate CPU interrupts to indicate when certain conditions exist.

When used for interprocessor communications, the McBSP serial port is one of the simplest forms of subsystem communication. The McBSP-to-McBSP connection is not supported internally — it is the responsibility of the system designer to implement the connections at the system board level.

6.5 Interprocessor Interrupts

If DMA, HPI, and McBSP interrupts do not provide enough flexibility, an additional interrupt called the interprocessor Interrupt is provided. This interrupt is generated internally to notify the other subsystem of a certain event. To generate the interrupt, the CPU must write a one to the interprocessor interrupt request bit (IPIRQ) in the BSCR register. Each subsystem's BSCR register contains the IPIRQ bit. Furthermore, each subsystem can respond to the respective interrupt named IPINT.

The interrupting subsystem must write a one to the IPIRQ bit followed by a zero. Consequently, the interrupted subsystem's IFR register (IPINT bit) is updated and the CPU processes the IPINT interrupt, if enabled in the IMR. The interrupting subsystem may leave the IPIRQ bit set high indefinitely. This does not generate multiple interrupts on the interrupted subsystem. The IFR flag representing the IPINT interrupt is cleared automatically when the CPU processes the associated interrupt service routine. The IPIRQ bit must be set to zero before subsequent interrupts can be generated.

The interprocessor interrupt mechanism is not supported on all multi-core DSPs. You should refer to the device-specific data sheet to determine if this feature is supported on a particular device.

Chapter 7

Enhanced External Bus Interface (EnhXIO)

This chapter describes the improved functionality of the enhanced external parallel interface (XIO2) available on the ’5410 device.

Topic **Page**

7.1 Introduction to the Enhanced External Parallel Interface (XIO2)

The enhanced external parallel interface is available only on select ’54x devices. Improvements to the ’VC5410’s external interface (XIO2) include:

- ❏ Simplification of bus sequences
- ❏ More immunity to bus contention when transitioning between read and write operations
- ❏ External memory access to the DMA controller
- ❏ Optimization of the power-down modes

The bus sequence on the ’VC5410 still maintains all of the same interface signals as on previous ’54x devices, but the signal sequence has been simplified. Most external accesses now require three cycles composed of a leading cycle, an active (read or write) cycle, and a trailing cycle.

The leading and trailing cycles provide additional immunity against bus contention when switching between read and write operations. To maintain high-speed read access, a consecutive read mode that performs single-cycle reads, as on previous ’54x devices, is available.

7.1.1 Additional Features

The XIO2 also provides the ability for DMA transfers to extend to external memory. For more information on DMA capability, see Chapter 3.

The XIO2 improves low-power performance, already present on ’54x devices, by switching off the internal clocks to the interface when it is not being used. This power-saving feature is automatic, requires no software setup, and causes no latency in the operation of the interface.

Additional features integrated in the XIO2 include:

- ❏ The ability to automatically insert bank-switching cycles when crossing 32K memory boundaries
- ❏ The ability to program up to 14 wait states through software
- ❏ The ability to divide down CLKOUT by a factor of 1, 2, 3, or 4. Dividing down CLKOUT provides an alternative to wait states when interfacing to slower external memory or peripheral devices.

Although inserting wait states extends the bus sequence during read or write accesses, it does not slow down the bus signal sequences at the beginning and end of the access. Dividing down CLKOUT provides a method of slowing the entire bus sequence when necessary. The CLKOUT divide-down factor is controlled through the DIVFCT field in the bank-switching control register (BSCR).

7.2 Bus Sequences

Figure 7–1 shows the bus sequence for three cases: all I/O reads, memory reads in non-consecutive mode, and single memory reads in consecutive mode. The accesses shown always require three CLKOUT cycles to complete.

Figure 7–1. Non-Consecutive Memory Read and I/O Read Bus Sequence

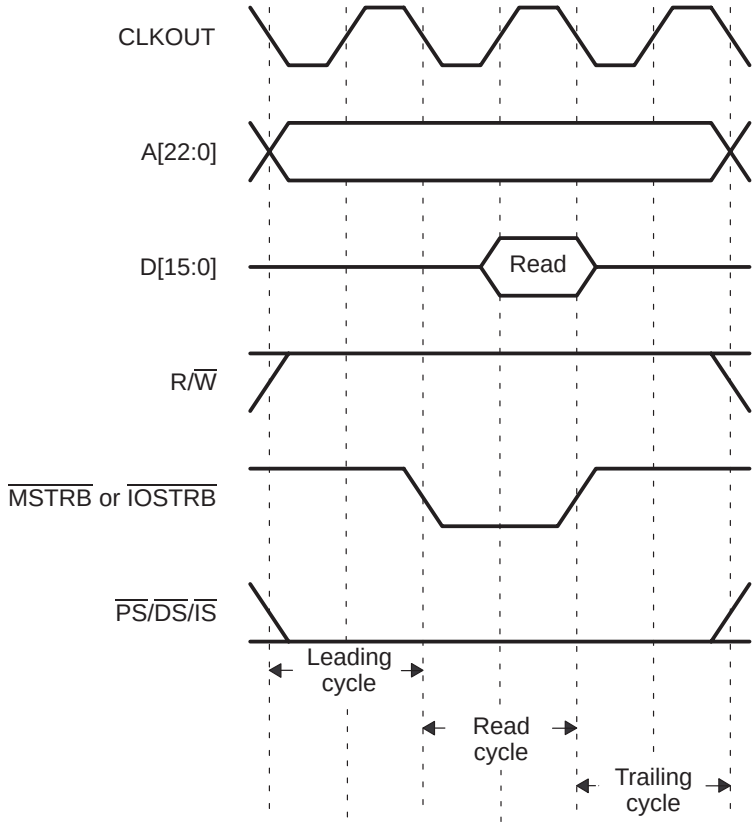


Figure 7–2 shows the bus sequence for repeated memory reads in consecutive mode. The accesses shown require (2+n) CLKOUT cycles to complete, where n is the number of consecutive reads performed.

Figure 7–2. Consecutive Memory Read Bus Sequence (n = 3 reads)

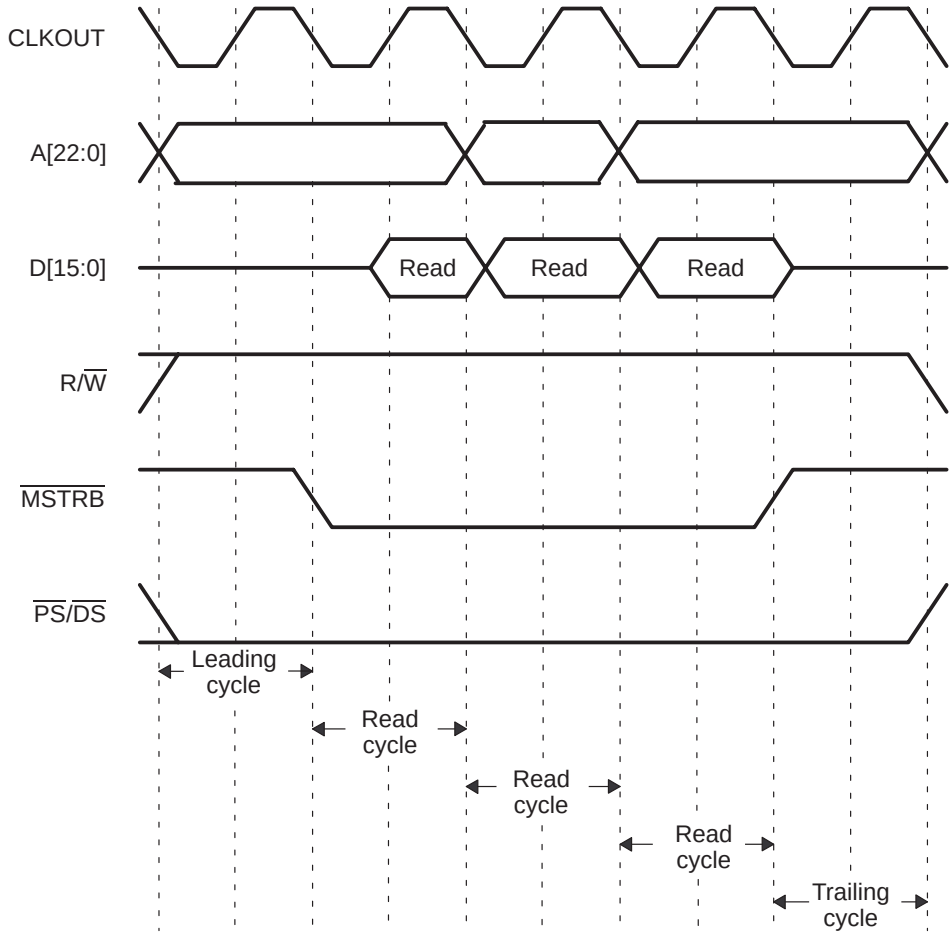
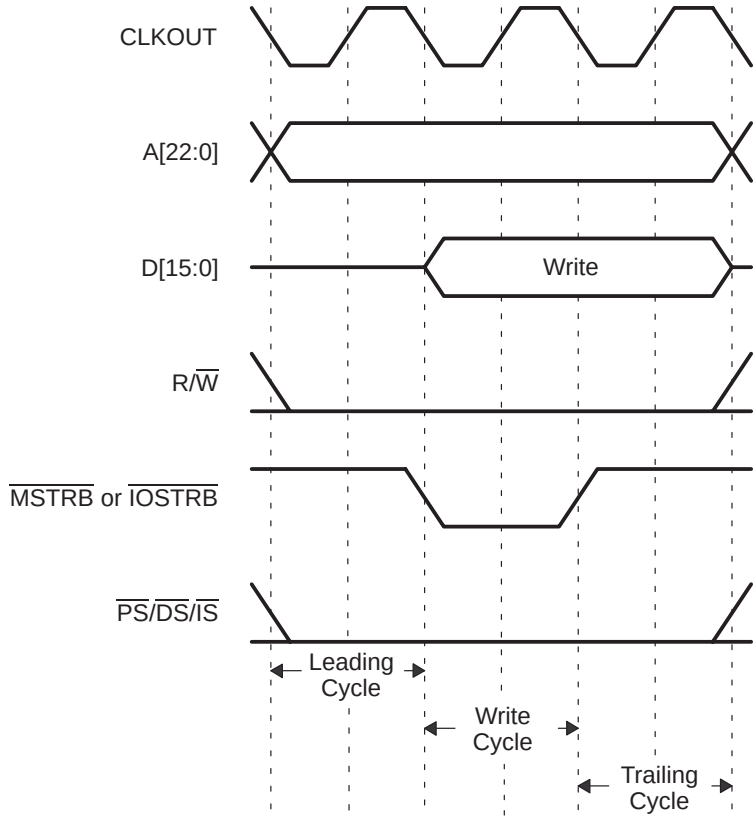


Figure 7–3 shows the bus sequence for all memory and I/O writes. The accesses shown always require three CLKOUT cycles to complete.

Figure 7–3. Memory Write and I/O Write Bus Sequence



Appendix A

Glossary

A

A-bis mode: In the A-bis mode (ABIS = 1), the McBSP can receive and transmit up to 1024 bits on a PCM link.

AC97: *Audio Codec '97*. A standard which uses a dual-phase frame feature with the first phase consisting of a single 16-bit word, and the second phase consisting of twelve 20-bit words.

Access mode: HPI access mode is determined by the logic levels of the HCNTL0/1 pins. Access mode can be an HPIC access, HPIA access, or HPID access with autoincrement.

ALE: *address latch enable*. Hosts that use a multiplexed address/data bus may use this to latch address/control data. Usually connected to $\overline{\text{HAS}}$.

AUTOINIT: *DMA auto-initialization bit*. This bit configures the channel to automatically initialize after a frame by loading from the global DMA registers. Located in the DMMCR register.

B

Bit Ordering: A feature that allows the LSB to be transferred to the serial port first when companded data is not used.

BOB: *HPI byte-order bit*. Selects which byte is transferred first. Only on HPI-8 HPI versions. Located in the HPIC register.

C

CLKG: *programmable data clocks*. Internal McBSP signals that can be programmed to drive receive and/or transmit clocking and framing.

CLKGDV: A programmable value that can be used to divide down the input clock source to the sample rate generator.

CLKR: *receive clock.* A McBSP interface signal.

CLKRM: *receive clock mode*. Located in the PCR register.

CLKRP: *receive frame clock polarity bit*. Located in the PCR.

CLK(R/X): *data clocks(receive/transmit)*.

CLKS: *external clock input.* A McBSP interface signal.

CLKSM: *clock source mode*. The CLKSM bit in the SRGR2 selects either the CPU clock (CLKSM = 1) or the external clock input (CLKSM = 0) CLKS, as the source for the sample rate generator input clock.

CLKSP: *clock polarity*. The rising edge of CLKS when CLKSP = 0, or the falling edge of CLKS when CLKSP = 1, causes the transition on the data bit-rate clock (CLKG) and frame sync (FSG).

CLKSRG: *clock sample rate generator*. The rising edge of CLKSRG generates clocking (CLKG) and framing (FSG).

CLKS_STAT: *CLKS pin status bit*. Indicates the logic level of the CLKS pin when CLKS is configured as a general-purpose input.

CLKSTP: *clock stop mode bit*. This bit is located in the SPCR1 register and is used to stop the serial port clock.

CLKX: *transmit clock.* A McBSP interface signal.

CLKXM: *transmit clock mode*. Located in the PCR register.

CLKXP: *transmit clock polarity bit*. Located in the PCR.

Companding: *compressing and expanding.* A quantization scheme for audio signals in which the input signal is compressed, and after processing, is reconstructed at the output by expansion. There are two distinct companding schemes: A-law, used in Europe, and μ-law, used in the United States.

CTMOD: *DMA transfer counter mode control bit*. Selects multiframe or ABU mode. Located in the DMMCR register.

D

DATDLY: *data delay (R/X)*. A delay at the beginning of actual data reception or transmission with respect to the start of the frame. The range of programmable data delay is 0 to 2 bit-clocks.

DBLW: *double-word mode*. Double-word mode is used when DMA data is 32-bits wide. Located in the DMSFC register.

DE: *DMA channel enable bit*. Enables/disables the DMA channel operation. Located in the DMPREC register.

DIND: *DMA destination address transfer index mode bit*. Offers several different indexing modes. Located in the DMMCR register.

DINM: *DMA interrupt generation mask bit*. Enables/disables DMA transfer interrupts. Located in the DMMCR register.

DIR: *direction bit*. Selects the direction of a general-purpose I/O pin. Located in the GPIOCR register.

Direct Memory Access (DMA) Controller: The direct memory access (DMA) controller transfers data between regions in the memory map without intervention by the CPU. The DMA allows movement to and from internal memory, internal peripherals, or external devices to occur in the background of CPU operation.

DLB: *digital loop-back mode.* DLB mode allows testing of serial port code with a single DSP device by internally connecting DR, FSR, and CLKR through multiplexers to DX, FSX, and CLKX.

DMA: *direct memory access (controller)*.

DMCTR: *DMA channel element count register*.

DMD: *DMA destination address space select bit*. Located in the DMMCR register.

DMDST: *DMA channel destination address register*.

DMDSTP: *DMA destination program page address* (all channels).

DMFRI0: *DMA frame address index register 0*.

DMFRI1: *DMA frame address index register 1*.

DMGCR: *DMA global element count reload register*.

DMGDA: *DMA global destination address reload register*.

DMGFR: *DMA global frame count reload register*.

DMGSA: *DMA global source address reload register*.

DMIDX0: *DMA element address index register 0*.

DMIDX1: *DMA element address index register 1*.

DMMCR: *DMA channel transfer mode control register*.

DMPREC: *DMA priority and enable control register*.

DMS: DMA source address space select bit. Located in the DMMCR register.

DMSA: The address register for DMA sub-addressed registers.

DMSDI: *DMA sub-address data register with autoincrement*.

DMSDN: *DMA sub-address data register without autoincrement*.

DMSFC: *DMA channel sync select and frame count register*.

DMSRC: *DMA channel source address register*.

DMSRCP: *DMA source program page address* (all channels).

DPRC: Configures the DMA channel priority level. Located in the DMPREC register.

DR: *data receive.* Receives data from devices interfacing the McBSP.

DRR[1,2]: *data receive register.* Two 16-bit registers used to receive data through the synchronous serial ports (McBSPs).

DSPINT: *host port interface's DSP CPU interrupt*. The host can set this to interrupt the DSP. Located in the HPIC register.

DSYN: *DMA sync event control bits*. Configures the DMA to synchronize to a particular event. Located in the DMSFC register.

DX: *data transmit.* Communicates data to devices interfacing the McBSP.

DXENA: *data transmit delay bit*. When this bit is set, the transmitted data is delayed.

DXR[1,2]: *data transmit registers 1 and 2.* Two 16-bit registers used to transmit data through the synchronous serial ports (McBSPs).

DX_STAT: *DX pin status bit*. Indicates the logic level of the DX pin when DX is configured as a general-purpose input.

E

Extended Software-Programmable Wait-State Generator: Extends external bus cycles up to 14 machine cycles to interface with slower off-chip memory and I/O devices.

F

FETCH: A bit located in the HPIC register that is used to fetch the data at the current HPIA address.

FIFO: *first-in-first-out*. A hardware mechanism that allows several elements to be stored in order. The first to be written to the FIFO is the first to be read out.

FPER: *frame period register.* The FPER determines when the next frame-sync signal becomes active.

Frame Count: *DMA frame count*. Specifies the number of frames to be transferred. Located in the DMSFC register.

FREE: *free running mode.* Disabled when FREE = 0, and enabled when FREE = 1.

FRLEN (R/X)[1,2]: *frame length*. The number of serial words (8-, 12-, 16-, 20-, 24-, or 32-bit) transferred per frame. The length corresponds to the number of words, or logical time slots, or channels per frame-synchronization signal.

$\overline{\text{FRST}}$: *frame-sync generator reset.* Frame-sync logic is reset when $\overline{\text{FRST}}$ = 0. Frame-sync signal FSG is generated after (FPER + 1) number of CLKG clocks when $\overline{\text{FRST}}$ = 1.

FSG: *frame-sync signal.* Generated by the frame-sync generation logic activated by the $\overline{\text{FRST}}$ bit.

FSGM: *sample rate generator transmit frame-synchronization mode bit*. Located in the SRGR2 register.

FSR: *receive frame synchronization.* A McBSP interface signal.

FSRM: *receive frame-synchronization mode bit*. Located in the PCR register.

FSRP: *receive frame-synchronization polarity bit*. Located in the PCR.

FSX: *transmit frame synchronization.* A McBSP interface signal.

FSXM: *transmit frame-synchronization mode bit.* Located in the PCR register.

FSXP: *transmit frame-synchronization polarity bit.* Located in the PCR.

FWID: *frame width.* An 8-bit down counter that controls the active width of the frame-sync pulse.

G

general-purpose input/output pin (GPIO): Pins that can be used to supply input signals from an external device or output signals to an external device. These pins are not linked to specific uses; rather, they provide input or output signals for a variety of purposes, and include the general-purpose $\overline{\text{BIO}}$ input pin and XF output pin.

$\overline{\text{GRST}}$: *sample rate generator reset.* The sample rate generator is reset by the $\overline{\text{GRST}}$ bit in SPCR2 when $\overline{\text{GRST}}$ = 0, and pulled out of reset when $\overline{\text{GRST}}$ = 1.

GSYNC: *sample rate generator clock synchronization.*

H

HINT: *DSP-to-host interrupt.* The DSP can interrupt the host by writing to this bit. Located in the HPIC register.

HMODE: *HPI mode.* Valid for the 16-bit HPI only. Allows the HPI to operate in multiplexed/non-multiplexed modes.

Host: External device that drives the HPI peripherals.

Host Port Interface (HPI-8): An enhanced 8-bit parallel port that interfaces a host device or host processor to the ’54x. Information is exchanged between the ’54x and the host device through all on-chip ’54x RAM.

Host Port Interface (HPI-16): An enhanced 16-bit version of the ’54x 8-bit host port interface that provides a full 16-bit bi-directional data bus, which does not require byte identification. Memory accesses are synchronized with the direct memory access (DMA) controller providing access to the complete internal memory address range.

HPIA: *host port address register.* A register that is loaded with the address that points to the data location to access. Accessible only by the host.

HPIC: *host port control register*. A register used by the host and DSP. Location of the HINT, XHPIA, DSPINT, BOB, and HPIENA bits.

HPID: *host port data register*. Only accessible by the host. Data passes through this register when using the HPI.

HPIENA: *HPI enable status bit*. Allows the host to determine the logic level of the HPIENA pin. Located in the HPIC register.

HPI increment mode: The HPI address register is automatically modified after the transfer completes.

HRDY: A bit located in the HPIC register that is used for software polling of the HRDY pin.

$\overline{\text{HSTRB}}$: Internal strobe that controls the HPI peripheral. The internal HSTRB signal is a function of external pins $\overline{\text{HDS1}}$, $\overline{\text{HDS2}}$, and $\overline{\text{HCS}}$.

I

IMOD: *DMA interrupt generation mode bit*. Configures the interrupt to occur on full/half-buffer boundaries. Located in the DMMCR register.

INTOSEL: *interrupt multiplexed control bits*. Determines which IFR bits will be associated with the DMA interrupts. Located in the DMPREC register.

IPINT: *interprocessor interrupt*. Located in the IFR and IMR registers.

IPIRQ: *interprocessor interrupt request*. Used to interrupt between subsystems on multi-core DSPs.

L

latency: The delay between when a condition occurs and when the device reacts to the condition. Also, in a pipeline, the necessary delay between the execution of two instructions to ensure that the values used by the second instruction are correct.

LSB: *least significant bit.* The lowest order bit in a word.

M

Master: A device that takes control of a peripheral, bus, etc.

MSB: *most significant bit.* The highest order bit in a word.

Multichannel Buffered Serial Port (McBSP): McBSPs are high-speed, full-duplex, multichannel buffered serial ports that allow direct interface to other ’54x devices, codecs, and other devices in a system. In addition, the McBSP offer double-buffered registers, which allow a continuous data stream, and independent framing and clocking for receive and transmit.

Multiplexed mode: The HPI operates in a multiplexed address/data bus fashion.

N

Non-multiplexed mode: The HPI operates with separate address and data buses.

P

Pin Control Register (PCR): Used to configure the McBSP pins as inputs or outputs during normal serial port operation. Also used to configure the serial port pins as general purpose inputs or outputs during receiver and/or transmitter reset.

Programmable Bank-Switching Module: Allows the ’54x to switch between external memory banks without requiring external wait states for memories that need several cycles to turn off. Bank-switching logic automatically inserts one cycle when accesses cross a 32K word memory-bank boundary inside program or data space.

R

RBR[1,2]: *receive buffer registers 1 and 2.* Receives copied data from the receive shift register (RSR).

RCBLK: *receive current block.*

RCER[A,B]: *McBSP receive channel enable register for partitions A and B.*

RCOMPAND: *receive companding mode*. Configure the McBSP to decompress receive data.

RDATDLY: *receive data delay bit.* Delays reception of data for 0, 1, or 2 bits of delay.

Receive and Transmit Control Registers (RCR[1,2] and XCR[1,2]): Configure various parameters of the receive and transmit operations.

Receive Buffer Register (RBR[1,2]): *See RBR[1,2].*

$\overline{\text{RESET}}$: A means of bringing the CPU to a known state by setting the registers and control bits to predetermined values and signaling execution to start at a specified address.

REVT: *receive synchronization event to DMA.*

REVTA: *receive synchronization event to DMA in A-bis mode.*

RFIG: *receive frame ignore bit*. Ignore subsequent FSRs after first.

RFRLEN1: *receive frame length 1*. Sets the receive frame length for first phase.

RFRLEN2: *receive frame length 2*. Sets the receive frame length for second phase.

RFULL: *reception with overrun.* Indicates that the receiver has experienced overrun and is in an error condition.

RINT: *receive interrupt to CPU.*

RIOEN: *receive section general-purpose I/O mode*. The DX, FSX, and CLKX pins are general-purpose I/Os.

RJUST: *receive data justification and sign-extension.* RJUST in the SPCR1 selects whether data in the RBR(1,2) is right or left justified (with respect to the MSB) in the DRR(1,2). If right-justification is selected, RJUST further selects whether the data is sign-extended or zero-filled.

RMCM: *receive multichannel selection enable*.

RPABLK: *receive partition A block*. Selects which block of channels to receive. Located in the MCR1 register.

RPBBLK: *receive partition B block*. Selects which block of channels to receive. Located in the MCR1 register.

RPHASE: *receive phase bit*. McBSP can be configured for one or two phases.

RRDY: *receive ready bit.* Indicates the ready status of the McBSP receiver.

$\overline{\text{RRST}}$: *receive reset bit.*

RSR[1,2]: *receive shift registers 1 and 2.* Receives shifted data from the DR pin after the appropriate data delay.

rsvd: *reserved.*

RSYNCERR: *unexpected receive frame synchronization.* A sync pulse which occurs RDATDLY minus 1 bit-clock earlier than the first bit of the next associated word. This causes the current data reception to abort and restart.

RWDLEN1: *receive word length 1*. Sets the receive word length for first phase.

RWDLEN2: *receive word length 2*. Sets the receive word length for second phase.

S

Sample Rate Generator: The sample rate generator is composed of a three-stage clock divider that allows programmable data clocks (CLKG) and framing signals (FSG), which are McBSP internal signals that can be programmed to drive receive and/or transmit clocking (CLKR/X) and framing (FSR/X). The sample rate generator can be programmed to be driven by an internal clock source or an internal clock derived from an external clock source.

Serial Port Control Register(SPCR[1,2]): Memory-mapped registers that contains status and control bits for the serial-port interface.

SIND: *DMA source address transfer index mode bit*. Offers several different indexing modes. Located in the DMMCR register.

Slave: A device that is controlled by another device.

SOFT: *soft bit.* SOFT mode is disabled when SOFT = 0, and enabled when SOFT = 1.

Software-Programmable Wait-State Generator: See Extended Software-Programmable Wait-State Generator.

SPCR[1,2]: *serial port control registers 1 and 2.*

SPSA: The address register for the McBSP sub-address registers.

SRGR[1,2]: *sample rate generator registers 1 and 2.*

Subsystem: *Multi-core DSPs*. Referred to as an independent system consisting of a CPU, memory, and peripherals.

T

TDM Data Stream: Multiple channels can be independently selected for the transmitter and receiver by configuring the McBSP with a single phase frame. Each frame represents a time-division multiplexed (TDM) data stream.

W

WDLEN(R/X)[1,2]: *word length.* 3-bit fields in the receive/transmit control register that determine the word length in bits-per-word for the receiver and transmitter for each phase of a frame.

X

XCBLK: *transmit current block*.

XCER[A,B]: *McBSP transmit channel enable register for partitions A and B*.

XCOMPAND: *transmit companding mode*. Configure the McBSP to compress transmit data.

XDATDLY: *transmit data delay bit.* Delays transmission of data for 0, 1, or 2 bits of delay.

$\overline{\text{XEMPTY}}$: *transmit shift register (XSR[1,2]) empty.* Indicates when the transmitter has experienced underflow. When XSR[1,2] is empty, $\overline{\text{XEMPTY}}$ = 0. When XSR[1,2] is not empty, $\overline{\text{XEMPTY}}$ = 1.

XEVT: *transmit synchronization event to DMA.*

XEVTA: *transmit synchronization event to DMA in A-bis mode.*

XFIG: *transmit frame ignore bit*. Ignore subsequent FSXs after first.

XFRLEN1: *transmit frame length 1*. Sets the transmit frame length for first phase.

XFRLEN2: *transmit frame length 2*. Sets the transmit frame length for second phase.

XHPIA: *extended HPIA*. Select either the lower 16 bits or the most significant bits to be accessed from the HPIA register. Located in the HPIC register.

XINT: *transmit interrupt to CPU.*

XINTM: *receive/transmit interrupt mode.*

XIOEN: *transmit section general-purpose I/O mode*. The DX, FSX, and CLKX pins are general-purpose I/Os.

XMCM: *transmit multichannel selection enable*.

XPABLK: *transmit partition A block*. Selects which block of channels to transmit. Located in the MCR1 register.

XPBBLK: *transmit partition B block*. Selects which block of channels to transmit. Located in the MCR1 register.

XPHASE: *transmit phase bit*. McBSP can be configured for one or two phases.

XRDY: *transmit ready.* Indicates the ready state of the McBSP transmitter. When XRDY = 0, the transmitter is not ready. When XRDY = 1, the transmitter is ready with data in DXR[1,2].

$\overline{\text{XRST}}$: *transmitter reset.* This resets and enables the serial port transmitter. Disables the transmitter and in reset state when $\overline{\text{XRST}}$ = 0, and enables the transmitter when $\overline{\text{XRST}}$ = 1.

XSR[1,2]: *transmit shift registers 1 and 2.* Data written to the DXR[1,2] is shifted out to DX via the transmit shift registers (XSR[1,2]).

XSYNCERR: *unexpected transmit frame synchronization – transmit synchronization error.* A sync pulse which occurs earlier than XDATDLY bitclocks before the last transmit bit of the previous frame. When XSYNCERR = 0, no synchronization error is detected. When XSYNCERR = 1, an error is detected by the McBSP.

XWDLEN1: *transmit word length 1*. Sets the transmit word length for first phase.

XWDLEN2: *transmit word length 2*. Sets the transmit word length for second phase.

Index

A

B

C

D

E

F

G

H

I

L

M

N

P

R

S

T

U

W

X