Implementation of a TMS320-SCSI Target Controller Using the TMS320C25 Digital Signal Processor Application Report

Literature Number: SPRA025
Job Number 61061

IMPORTANT NOTICE

Texas Instruments Incorporated (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to current specifications 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.

Please be aware that TI products are not intended for use in life-support appliances, devices, or systems. Use of TI product in such applications requires the written approval of the appropriate TI officer. Certain applications using semiconductor devices may involve potential risks of personal injury, property damage, or loss of life. In order to minimize these risks, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards. Inclusion of TI products in such applications is understood to be fully at the risk of the customer using TI devices or systems.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI 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.

WARNING

This equipment is intended for use in a laboratory test environment only. It generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to subpart J of part 15 of FCC rules, which are designed to provide reasonable protection against radio frequency interference. Operation of this equipment in other environments may cause interference with radio communications, in which case the user at his own expense will be required to take whatever measures may be required to correct this interference.

Read This First

About This Manual

Today hard-disk-drive designers are challenged to develop higher performance, smaller form-factor drives. A typical high-performance disk-drive architecture is shown in Figure i–1. There is a microprocessor to handle the system interface and a digital signal processor (DSP) to implement the servo control. With system integration becoming a key factor to the design of future disk drives, the question becomes “Can the system interface and servo control both be accomplished with one processor without sacrificing performance?”

Because of the performance requirements of digital servo controllers today, general-purpose microprocessors cannot execute the algorithms in real time at the required sampling frequency. However, the TMS320 DSPs have an internal architecture that allow them to execute these complex servo-control algorithms at a fraction of their total processing bandwidth. This additional bandwidth may be used to implement target-controller functions, such as Small Computer System Interface (SCSI), along with servo control in software. Figure i–2 shows the architecture for a DSP-based servo controller and system interface uniprocessor solution.

This application note describes an example SCSI target controller that was designed with a TMS320C25 DSP controlling a specialized SCSI interface chip. Although a Texas Instruments SCSI interface chip was used in this system, any number of standard SCSI interface chips can be used in conjunction with the TMS320C25 to develop a SCSI target controller. The TMS320C25 SCSI target controller implements all SCSI bus phases and performs SCSI command interpretation.

Because the TMS320C25 DSP supports high level languages such as C, the target controller code can be written in a high level language to reduce the software development time. TMS320 debugging and development tools allow the software engineer to test and debug the code in the same language in

which it was written. This is an improvement over past tools that forced all software debugging to be done in assembly language. The software to implement this application was developed in the C language. It can be licensed from Texas Instruments. Contact your local TI field sales representative or authorized distributor for licensing information.

Figure i–1. Typical High-Performance Disk-Drive Architecture

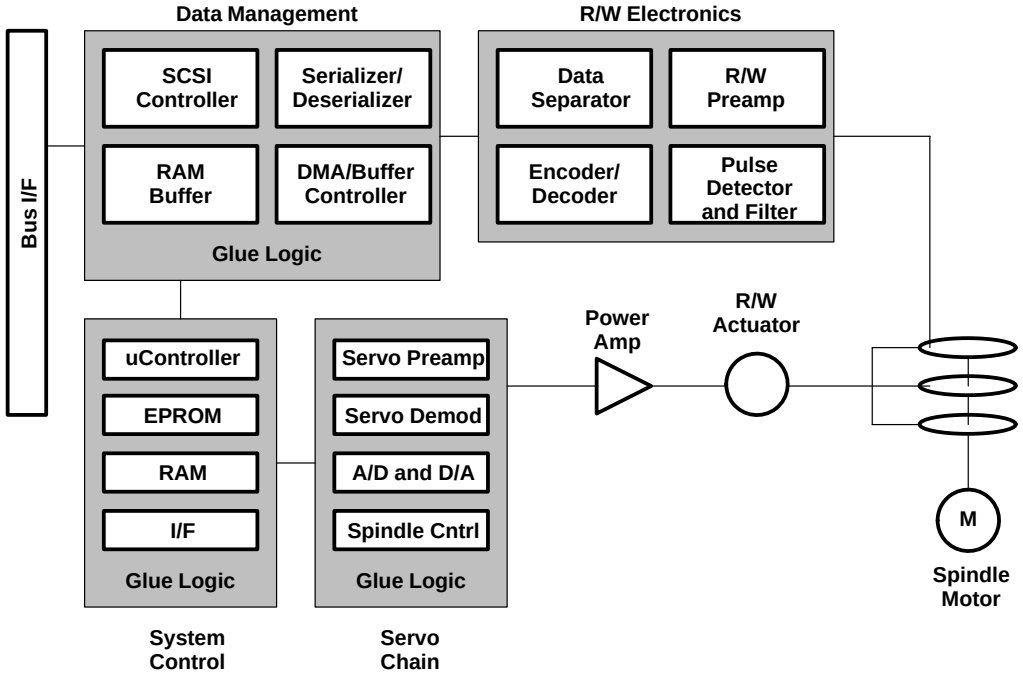
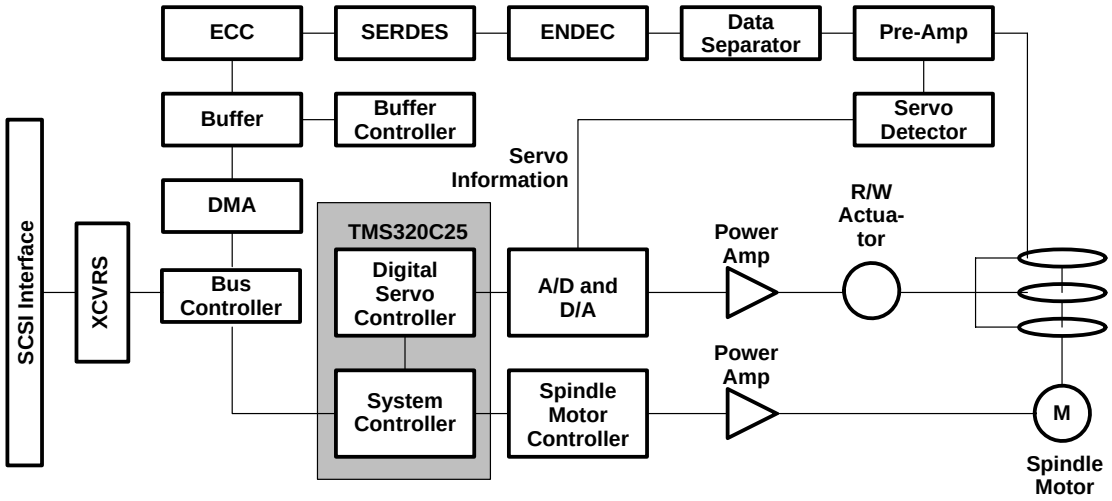


Figure i–2. DSP-Based Servo and System Interface Uniprocessor Architecture



This document is not intended to serve as a tutorial on the Small Computer Systems Interface bus; refer to ANSI X3.131–1986 for detailed information regarding the SCSI bus.

How to Use This Manual

This document contains the following chapters:

Chapter 1 TMS320 DSP Overview—gives an overview of the key features of the five generations of the TMS320 DSP family of products.

Chapter 2 SCSI Overview and Specifications—describes the addressing method, interface, data transfer, host memory/host adapter/SCSI controller relationship, communication protocol, a SCSI command example, bus signals, bus conditions, messages, status phase, and commands for the SCSI target controller.

Chapter 3 TMS320-SCSI Hardware Theory of Operations—provides descriptions of the TMS320-SCSI target controller, functional module, hardware, and SN75C091A.

Chapter 4 Firmware Theory of Operations—includes firmware design overview and operation, command execution routines, system initialization routines, SCSI information transfer routines, SCSI-sense data-buffer management routines, a target-controller command overhead example, and firmware drawings.

Chapter 5 TMS320-SCSI Physical Characteristics—shows the board layout and connectors and discusses installation.

Appendix A Signals and Schematics—includes the signal descriptions, schematics for the TMS320-SCSI target controller, and timing diagrams.

Appendix B Equations and Pin Assignments—lists the memory-I/O decoder logic equations and pin assignments for U39 and the FIFO-control logic equations and pin assignments for U46.

Glossary

Related Documentation From Texas Instruments

TMS320 Family Development Support Reference Guide (literature number SPRU011) describes the TMS320 family of digital signal processors and covers the various products that support this product line. This includes 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.

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.

If You Need Assistance. . .

If you want to. . .	Do this. . .
Request more information about Texas Instruments Digital Signal Processing (DSP) products	Call the CRC†: **(800) 336–5236** Or write to: Texas Instruments Incorporated Market Communications Manager, MS 736 P.O. Box 1443 Houston, Texas 77251–1443
Order Texas Instruments documentation	Call the CRC†: **(800) 336–5236**
Ask questions about product operation or report suspected problems	Call the DSP hotline: **(713) 274–2320**
Report mistakes in this document or any other TI documentation	Send your comments to: Texas Instruments Incorporated Technical Publications Manager, MS 702 P.O. Box 1443 Houston, Texas 77251–1443

† Texas Instruments Customer Response Center

Contents

1 TMS320 Digital Signal Processors Overview 1-1
1.1 TMS320C25 Digital Signal Processor 1-3
1.2 TMS320 Fixed-Point DSPs 1-6
1.2.1 TMS320C1x 1-6
1.2.2 TMS320C2x 1-6
1.2.3 TMS320C5x 1-6
1.3 TMS320 Floating-Point DSPs 1-7
1.3.1 TMS320C3x 1-7
1.3.2 TMS320C4x 1-8
1.4 TMS320 Development Support 1-9

2 SCSI Overview and Specifications 2-1
2.1 Addressing Method 2-3
2.2 Interface 2-3
2.3 Data Transfer 2-4
2.4 Host Memory/Host Adapter/SCSI Controller Relationship 2-5
2.5 Communication Protocol 2-7
2.6 SCSI Command Example 2-9
2.6.1 Single Command Example 2-9
2.6.2 Disconnect Example 2-9
2.7 SCSI Bus Signal Descriptions 2-11
2.8 SCSI Bus Conditions 2-13
2.8.1 ATTENTION Condition 2-13
2.8.2 RESET Condition 2-13
2.8.3 UNIT ATTENTION Condition 2-13
2.9 Messages 2-14
2.9.1 COMMAND COMPLETE (00h) 2-14
2.9.2 EXTENDED MESSAGE SYNCHRONOUS DATA TRANSFER REQUEST (01h) 2-14
2.9.3 SAVE DATA POINTER (02h) 2-14
2.9.4 RESTORE POINTERS (03h) 2-14
2.9.5 DISCONNECT (04h) 2-15
2.9.6 INITIATOR DETECTED ERROR (05h) 2-15
2.9.7 ABORT (06h) 2-15
2.9.8 MESSAGE REJECT (07h) 2-15
2.9.9 NO OPERATION (08h) 2-16

2.9.10 MESSAGE PARITY ERROR (09h) 2-16
2.9.11 BUS DEVICE RESET (0Ch) 2-16
2.9.12 IDENTIFY (C0h or 80h) 2-16
2.10 Status Phase 2-18
2.10.1 GOOD (00h) 2-18
2.10.2 CHECK CONDITION (02h) 2-18
2.10.3 BUSY (08h) 2-18
2.10.4 RESERVATION CONFLICT (18h) 2-18
2.11 SCSI Commands 2-19

3 TMS320-SCSI Hardware Theory of Operations 3-1
3.1 TMS320-SCSI Target Controller General Description 3-2
3.2 TMS320-SCSI Functional Modules Descriptions 3-3
3.3 TMS320-SCSI Hardware 3-5
3.3.1 Read/Write Strobe Decode Logic 3-5
3.3.2 Instruction Memory 3-6
3.3.3 Microprocessor Scratch RAM 3-7
3.3.4 Data Buffer 3-8
3.3.5 FIFO Memory 3-8
3.3.6 Test Condition Logic 3-10
3.3.7 SCSI Bus Interface Controller 3-10
3.3.8 Programmable Interrupt Generator 3-11
3.3.9 LED Register 3-11
3.4 SN75C091A Overview 3-12
3.4.1 SN75C091A Architecture 3-12
3.4.2 Features 3-13
3.4.3 Command Sequencer 3-13
3.4.4 SCSI $\overline{\text{REQ}}$/$\overline{\text{ACK}}$ Handshake Controller 3-13
3.4.5 Arbitration and Selection Controller 3-14
3.4.6 Register File 3-14
3.4.7 Microprocessor Interface 3-14
3.4.8 Receive and Transmit FIFOs 3-14
3.4.9 Interrupt Control 3-14
3.4.10 DMA Interface 3-15
3.4.11 Byte Stack Control 3-15
3.4.12 Parity Generators/Checkers 3-15

4 Firmware Theory of Operations 4-1
4.1 Firmware Design Overview 4-2
4.2 Firmware Operation 4-3
4.2.1 Command Reception 4-3
4.2.2 Command Process 4-8
4.2.3 Command End Status 4-17
4.3 Command Execution Routines 4-18

4.4 System Initialization Routines 4-19
4.5 SCSI Information Transfer Routines 4-20
4.6 SCSI-Sense Data-Buffer Management Routines 4-21
4.7 Target-Controller Command Overhead Example 4-22

5 TMS320-SCSI Physical Characteristics 5-1
5.1 Physical Layout 5-2
5.2 Connectors 5-3
5.2.1 J1 Connector 5-3
5.2.2 J2 Connector 5-3
5.2.3 J3 Connector 5-3
5.3 Configuration Switches and Jumpers 5-4
5.4 Indicators 5-5
5.5 Interfacing TMS320-SCSI With SCSI Host Adaptors 5-6

A Signals and Schematics A-1
A.1 Hardware Signal Glossary A-2
A.2 TMS320-SCSI Target Controller Board Schematics A-5
A.3 Timing Diagrams A-16

B Equations and Pin Assignments B-1
B.1 Memory/IO-Decoder Logic Equations and Pin Assignments (U39) B-2
B.2 FIFO-Control Logic Equations and Pin Assignments (U46) B-3

C Glossary C-1

Figures

i-1 Typical High-Performance Disk-Drive Architecture iv
ii-2 DSP-Based Servo and System Interface Uniprocessor Architecture v
1–1 TMS320 Family of Devices 1-2
1–2 TMS320C25 Block Diagram 1-5
2–1 Sample SCSI Configurations 2-2
2–2 Simplified SCSI System 2-4
2–3 Snapshot Prior to Initial Selection 2-6
2–4 Phase Sequences Using the Arbitration Phase (Multiple-Host and/or Multiple-Target SCSI Bus System) 2-8
2–5 Phase Sequences That Are Not Using the Arbitration Phase (Single-Host and Single-Target SCSI Bus System) 2-8
2–6 READ Command Example Without Disconnect 2-9
2–7 READ Command Example With DIsconnect 2-10
2–8 Mode-Sense Buffer for XMIT (64) and Mode-Select Buffer for RCVE (64) 2-59
2–9 SCSI Mode-Sense Data (All Initiators) 2-60
2–10 Page Codes (1) and Page Formats (0) for Current, Changeable, and Default 2-60
2–11 Page Codes (3) and Page Formats (1) for Current 2-61
2–12 Page Codes (3) and Page Formats (1) for Changeable 2-61
2–13 Page Codes (4) and Page Formats (1) for Default and Changeable 2-62
3–1 TMS320-SCSI Hardware Block Diagram 3-5
3–2 SN75C091A Functional Block Architecture 3-12
4–1 SCSI Command Buffers Structure for 8 Initiators 4-4
4–2 Command Buffer for Initiator 0 4-4
4–3 Command Buffer for Initiator 1 4-5
4–4 Command Buffer for Initiator 2 4-5
4–5 Command Buffer for Initiator 3 4-6
4–6 Command Buffer for Initiator 4 4-6
4–7 Command Buffer for Initiator 5 4-7
4–8 Command Buffer for Initiator 6 4-7
4–9 Command Buffer for Initiator 7 4-8
4–10 Request Sense Data (Initiators 0–7) 4-9
4–11 Request Sense Data for Initiator 0 4-9
4–12 Request Sense Data for Initiator 1 4-9
4–13 Request Sense Data for Initiator 2 4-10
4–14 Request Sense Data for Initiator 3 4-10
4–15 Request Sense Data for Initiator 4 4-10
4–16 Request Sense Data for Initiator 5 4-11

4–17 Request Sense Data for Initiator 6 4-11
4–18 Request Sense Data for Initiator 7 4-11
4–19 Command-Execution Pointers for Group 0 Commands (Local Unit in Ready State) . . . 4-13
4–20 Command-Execution Pointers for Group 0 Commands (Local Unit in Not-Ready State) 4-14
4–21 Command-Execution Pointers for Group 1 Commands (Local Unit in Ready State) . . . 4-15
4–22 Command-Execution Pointers for Group 1 Commands (Local Unit in Not-Ready State) 4-16
4–23 Command-Execution Pointers for Groups 2, 3, 4, 5, 6, and 7 4-17
4–24 READ Command Example Without Disconnect 4-22
4–25 READ Command Example With Disconnect 4-22
4–26 TMS320-SCSI Target Controller Time Delays 4-24
5–1 TMS320-SCSI Target-Controller Printed Circuit Board 5-2
A–1 Clock Reset and Power Schematics A-5
A–2 Microprocessor-Instruction Memory (TMS320C25 Program-Memory Interface) A-6
A–3 Dynamic RAM Control Logic A-7
A–4 Data Buffer A-8
A–5 Memory-Select Decoder and Scratch RAM Interface A-9
A–6 I/O-Port Decoder and Test-Condition Logic A-10
A–7 FIFO Controller A-11
A–8 SCSI DMA FIFO A-12
A–9 SCSI-Bus Controller Single-Ended Interface A-13
A–10 Zero-Wait-State Instruction Data RAM A-14
A–11 Programmable Interrupt Generator A-15
A–12 Program-Memory Interface Timing (TMS320C25 Timing With TMS27512-20) A-16
A–13 Data RAM-Interface Timing (TMS320C25 Read/Write Timing With CY7C199-25) A-17
A–14 Data Buffer Read Timing (Read Timing With 74ACT4503 and TM024EAD9–10) A-18
A–15 Data Buffer Write Timing (TMS320C25 Write Timing With 74ACT4503 and TM024EAD9–10) A-19
A–16 SCSI-Bus Read Timing (TMS320C25 Read Timing With SN75C091A Register File) . . A-20
A–17 SCSI-Bus Write Timing (TMS320C25 Write Timing With SN75C091A Register File) . . A-21

Tables

2–1 Messages Supported by Target Controller 2-14
2–2 IDENTIFY Message Functions 2-17
2–3 Status Codes 2-18
2–4 SCSI Commands 2-19
2–5 TEST UNIT READY Command 2-20
2–6 REZERO UNIT Command 2-21
2–7 REQUEST SENSE Command 2-22
2–8 Extended Sense Data Format 2-23
2–9 Sense Key Descriptions 2-24
2–10 Additional Sense Codes of the REQUEST SENSE Command 2-25
2–11 FORMAT UNIT Command 2-28
2–12 Defect List Header for the FORMAT UNIT Command 2-29
2–13 Defect Descriptor Block Format's CDB for the FORMAT UNIT Command 2-29
2–14 REASSIGN BLOCK Command 2-30
2–15 Defect List Header Format 2-31
2–16 Defect Descriptor CDB of the REASSIGN BLOCK Command 2-31
2–17 READ Command 2-32
2–18 WRITE Command 2-33
2–19 SEEK Command 2-34
2–20 INQUIRY Command 2-35
2–21 Data Format 2-36
2–22 Inquiry Data (All Initiators) for TMS320-SCSI 2-38
2–23 MODE SELECT Command 2-40
2–24 Parameter List Header 2-41
2–25 Block Descriptor 2-42
2–26 List of Page Codes 2-42
2–27 Error Recovery Parameter Option Format 2-43
2–28 Disconnect/Reconnect Parameters Option Format 2-45
2–29 Direct-Access Device Format Parameters Option Format 2-47
2–30 Rigid Disk Drive Geometry Parameters Option Format 2-50
2–31 RESERVE UNIT Command Format 2-53
2–32 RELEASE UNIT Command 2-55
2–33 MODE SENSE CDB Command 2-58
2–34 Parameter Values Sent by Target Controller Using Page Control Field 2-58
2–35 Parameter List Header Command 2-59
2–36 START/STOP UNIT Command 2-63

2–37 READ CAPACITY Command 2-64
2–38 READ CAPACITY Command During DATA IN Phase 2-65
2–39 EXTENDED READ Command 2-66
2–40 EXTENDED WRITE Command 2-67
2–41 EXTENDED SEEK Command 2-68
2–42 WRITE AND VERIFY Command 2-69
2–43 WRITE BUFFER Command 2-70
2–44 WRITE BUFFER Data Format 2-71
2–45 READ BUFFER Command 2-72
2–46 Read Buffer Header Format 2-73
2–47 READ LONG Command 2-74
2–48 WRITE LONG Command 2-75
4–1 Command Overhead for a 4K-Byte Write 4-23
5–1 SCSI IDs for Switch Bank Positions 5-6
A–1 Hardware Signal Glossary A-2
B–1 Memory/IO-Decoder Pin Assignments (U39) B-2
B–2 FIFO-Control Pin Assignments (U46) B-3

IMPORTANT NOTICE

Texas Instruments Incorporated (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to current specifications 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.

Please be aware that TI products are not intended for use in life-support appliances, devices, or systems. Use of TI product in such applications requires the written approval of the appropriate TI officer. Certain applications using semiconductor devices may involve potential risks of personal injury, property damage, or loss of life. In order to minimize these risks, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards. Inclusion of TI products in such applications is understood to be fully at the risk of the customer using TI devices or systems.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI 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.

WARNING

This equipment is intended for use in a laboratory test environment only. It generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to subpart J of part 15 of FCC rules, which are designed to provide reasonable protection against radio frequency interference. Operation of this equipment in other environments may cause interference with radio communications, in which case the user at his own expense will be required to take whatever measures may be required to correct this interference.

Read This First

About This Manual

How to Use This Manual

This document contains the following chapters:

Style and Symbol Conventions

This document uses the following conventions.

- Program listings, program examples, interactive displays, filenames, and symbol names are shown in a `special typeface` similar to a typewriter's. Examples use a **`bold version`** of the special typeface for emphasis; interactive displays use a **`bold version`** of the special typeface to distinguish commands that you enter from items that the system displays (such as prompts, command output, error messages, etc.).

  Here is a sample program listing:

```
0011  0005  0001         .field    1, 2
0012  0005  0003         .field    3, 4
0013  0005  0006         .field    6, 3
0014  0006               .even
```

  Here is an example of a system prompt and a command that you might enter:

  `C:` **`csr –a /user/ti/simuboard/utilities`**

- In syntax descriptions, the instruction, command, or directive is in a **bold typeface** font and parameters are in an *italic typeface.* Portions of a syntax

that are in **bold** should be entered as shown; portions of a syntax that are in *italics* describe the type of information that should be entered. Here is an example of a directive syntax:

.asect "*section name***",** *address*

.asect is the directive. This directive has two parameters, indicated by *section name* and *address*. When you use .asect, the first parameter must be an actual section name, enclosed in double quotes; the second parameter must be an address.

- Square brackets (**[** and **]**) identify an optional parameter. If you use an optional parameter, you specify the information within the brackets; you don't enter the brackets themselves. Here's an example of an instruction that has an optional parameter:

  LALK *16–bit constant [, shift]*

  The LALK instruction has two parameters. The first parameter, *16-bit constant*, is required. The second parameter, *shift*, is optional. As this syntax shows, if you use the optional second parameter, you must precede it with a comma.

  Square brackets are also used as part of the pathname specification for VMS pathnames; in this case, the brackets are actually part of the pathname (they are not optional).

- Braces ({ and }) indicate a list. The symbol **|** (read as *or*) separates items within the list. Here's an example of a list:

  ```
  { * | *+ | *– }
  ```

  This provides three choices: *, *+, or *–.

  Unless the list is enclosed in square brackets, you must choose one item from the list.

- Some directives can have a varying number of parameters. For example, the .byte directive can have up to 100 parameters. The syntax for this directive is:

  .byte $value_1$ *[, ... ,* $value_n$*]*

  This syntax shows that .byte must have at least one value parameter, but you have the option of supplying additional value parameters, separated by commas.

Information About Cautions and Warnings

This book may contain cautions and warnings.

This is an example of a caution statement.

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

This is an example of a warning statement.

A warning statement describes a situation that could potentially cause harm to you.

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

Related Documentation From Texas Instruments

<**Writer**: Pull the appropriate description from the RelatedDocumentation catalog by creating the component that has the same name as the book you need to describe.>

Trademarks

<**Writer**: Pull the appropriate trademarks from the Trademark catalog by creating the component that has the same name as the product for which you are listing a trademark notice.>

Trademarks

ADI and *AutoCAD* are trademarks of Autodesk, Inc.
Apollo and *Domain* are trademarks of Apollo Computer, Inc.
ATVista is a trademark of Truevision, Inc.

If You Need Assistance. . .

If you want to. . .	**Do this. . .**
Request more information about Texas Instruments Digital Signal Processing (DSP) products	Call the CRC† hotline: **(800) 336–5236** Or write to: Texas Instruments Incorporated Market Communications Manager, MS 736 P.O. Box 1443 Houston, Texas 77251–1443
Order Texas Instruments documentation	Call the CRC† hotline: **(800) 336–5236**
Ask questions about product operation or report suspected problems	Call the DSP hotline: **(713) 274–2320**
Report mistakes in this document or any other TI documentation	Fill out and return the reader response card at the end of this book, or send your comments to: Texas Instruments Incorporated Technical Publications Manager, MS 702 P.O. Box 1443 Houston, Texas 77251–1443

†
Texas Instruments Customer Response Center

Chapter 1

TMS320 Digital Signal Processors Overview

Since the first TMS320 digital signal processor (DSP) was introduced in 1982, Texas Instruments Incorporated (TI) has been dedicated to the advancement of digital signal processing technology and its applications. TI recognizes that fast time to market, increased productivity, and design ease are of primary importance in the development of DSP-based applications. Therefore, TI offers a comprehensive program of world-class development support for TMS320 DSPs that facilitates the design process from system concept to production and allows users to take advantage of rapidly evolving DSP technology.

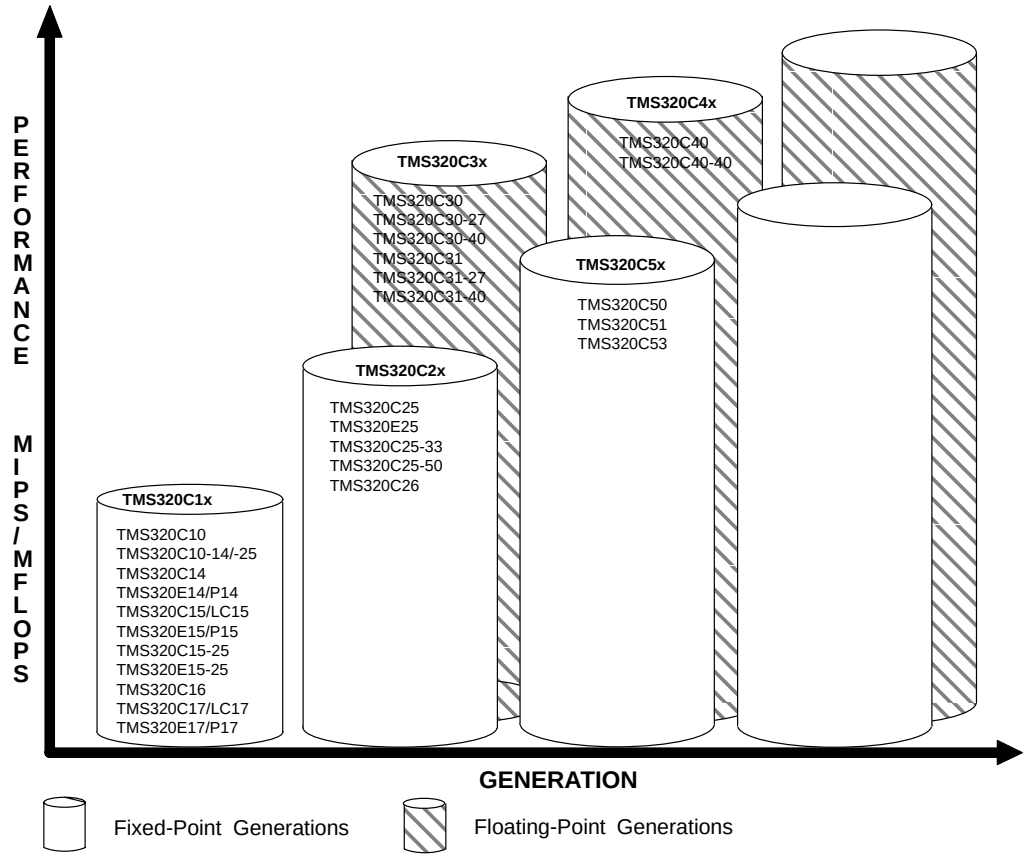
The TMS320 support program includes leading-edge hardware and software development systems: optimizing ANSI C compilers, a user-friendly Programmer's Interface consisting of C/assembly language source debuggers with code-profiling capabilities, low-cost evaluation tools, simulators, realtime emulators, realtime operating systems, and application software. More than ever, TMS320 DSP users enjoy a development environment that is comparable to the environment available for general-purpose microprocessor systems.

Support from third-party companies complement the TI product/service offerings. Please consult the *TMS320 Third-Party Support Reference Guide* (literature number SPRU052) for details.

The powerful instruction sets, inherent flexibility, high-speed number-crunching capabilities, and innovative architectural designs have made the high-performance cost-effective TMS320 digital signal processor family the ideal solution for many automotive, computer, consumer, industrial, military, and telecommunication applications. The TMS320C25 is one of many TMS320 DSPs offered by Texas Instruments. TMS320 DSPs include five generations of fixed-point and floating-point devices, as shown in Figure 1–1. Further expansion of this family is planned, creating even higher-performing, more versatile spin-offs and new generations. The TMS320 family offers different performance ranges between generations. Within each generation, members are object-code and in some cases pin compatible.

Chapter 1 provides an overview of the TMS320 digital signal processors.

Figure 1–1. TMS320 Family of Devices



Topic **Page**

TMS320C25 DSP **1-3**
TMS320 Fixed-Point DSPs **1-6**
TMS320 Floating-Point DSPs **1-7**
TMS320 Development Support **1-9**

1.1 TMS320C25 Digital Signal Processor

The TMS320C25 is a member of the TMS320 family of digital signal processors. Digital signal processors (DSPs) are special-purpose microprocessors designed for high-performance applications including real-time signal processing and other demanding systems. Some of the features that make the TMS320C25 very suitable for SCSI controller applications is a fast instruction-cycle time of 100/80 ns, a bus-transfer rate of 20 Mbytes/second, and an interrupt response time of 400 ns. The TMS320C25's general-purpose instruction set is supported by specialized instructions for signal processing.

The TMS320C25 is shown in Figure 1–2. As the block diagram indicates, all subsections including multiplier are implemented in hardwired logic. This is different from traditional microprocessors that use internal software or microcode to execute their instructions. This hardwired implementation gives the TMS320C25 a sustained performance of 10/12.5 MIPS and makes it well suited for high-performance designs.

All instructions except branches or I/O are executed in a single cycle. A repeat-instruction capability allows multicycle instructions to be executed in a single cycle. Burst-data-transfer rates from internal memory to external memory are 20 Mbytes/second. From external memory to external memory, burst-transfer rate is 10 Mbytes/second. The TMS320C25

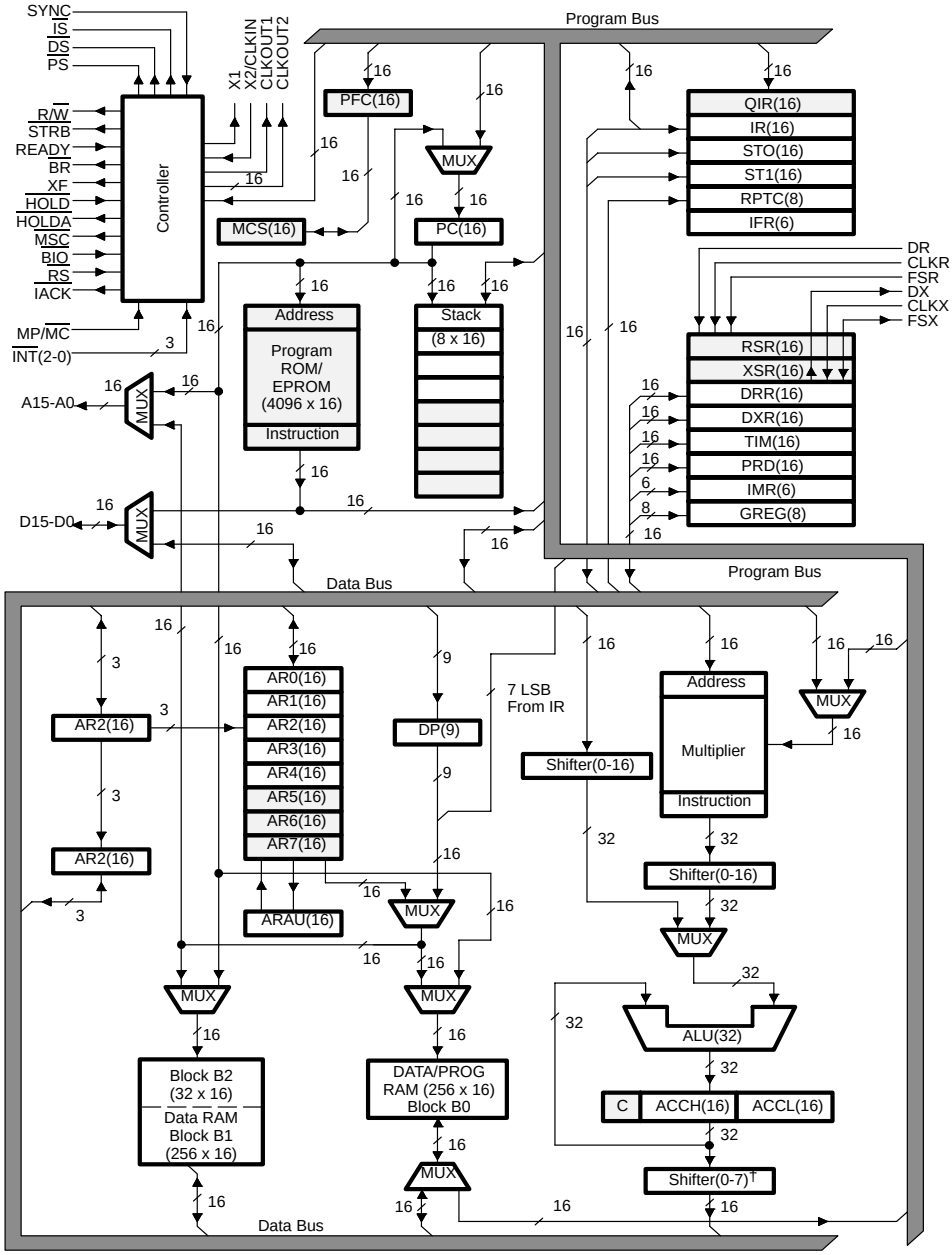
- Requires a single 5 V ± 5% power supply
- Operates on low-power CMOS technology
- Is packaged in a 68-pin PLCC
- Has an eight-deep hardware stack that allows an interrupt-response time of three cycles (Larger software stacks can be built in on-chip or off-chip memory.)
- Offers an on-chip timer and a CODEC-compatible, full-duplex serial port
- Contains 64K words of program-memory address space and 64K words of data-memory address space
- Includes 544 words of on-chip RAM and 4K words of on-chip ROM/EPROM

The high performance of the TMS320C25 allows it to be used for multiple functions. In a disk drive, it can be used for servo control of the read/write actuator, as well as SCSI controller functions. Similarly in other computer peripherals like tape drives and printers, it can be used for servo control and SCSI controller functions. In computer applications the TMS320C25 can be used for audio, speech, or image processing in addition to SCSI controller functions.

To help development of the TMS320C25-based systems, a large number of development systems are available. These include assembler/linkers, macro

assemblers, high-level-language (HLL) compilers including C and Pascal, HLL debuggers, and software simulators. Hardware-development systems include standalone and PC-plug-in development boards, in-circuit emulators, logic analyzers, and bus functional simulation models. These development tools are available from both Texas Instruments and third parties supporting the TMS320C25. The TMS320C25 is a member of the TMS320C2x generation of fixed-point DSPs.

Figure 1–2. TMS320C25 Block Diagram



NOTE: Lighter shaded areas are for TMS320C25 & TMS320E25. Darker shaded areas indicate a bus.

1.2 TMS320 Fixed-Point DSPs

There are three generations of fixed-point DSPs—TMS320C1x, TMS320C2x, and TMS320C5x. All generations of the TMS320 fixed-point DSPs have common features, such as a 16-bit architecture with 32-bit ALU and accumulator. They are based on a Harvard architecture with separate buses for program and data, allowing both instructions and operands to be fetched simultaneously. They also feature a 16×16=32-bit hardware multiplier for a single-cycle multiply and a hardware stack for fast context saves. An overflow saturation mode is included to prevent wrap around. All instructions (except branches) are executed in a single cycle. Performance ranges between the three generations from 5 MIPS (million of instructions per second) to 28.5 MIPS.

1.2.1 TMS320C1x

The TMS320C1x generation is based on the TMS320C10. It features 144 words of on-chip RAM and 4K words of address space with instruction-cycle time as fast as 160 ns. Other members of this generation include TMS320C15 and its EPROM version, TMS320E15, TMS320C16, TMS320C17/E17, and TMS320C14/E14. All these devices have expanded memory of 256 words of on-chip RAM and 4K words of on-chip ROM/EPROM. The TMS320C14/E14 has been optimized for digital control applications. The TMS320C16 has the same architecture as the other TMS320C1x devices except that the program address space has been expanded to 64K words.

1.2.2 TMS320C2x

The TMS320C2x generation is based on the TMS320C25. It provides 544 words of on-chip RAM, and 4K words of on-chip ROM. Total address space is expanded to 64K words for both data and program. Other members include an EPROM version TMS320E25 and TMS320C26.

1.2.3 TMS320C5x

The TMS320C5x generation is the highest-performance 16-bit fixed-point generation and includes the TMS320C50, TMS320C51, and TMS320C53. The ’C5x performance level is achieved through a faster cycle time, larger on-chip memory space, and systematic integration of more signal-processing functions. The TMS320C50 has 10K words of on-chip RAM and 2K words of on-chip ROM, the TMS320C51 has 2K words of on-chip RAM and 8K words of on-chip ROM, the TMS320C53 has 4K words of on-chip RAM, and 16K words of on-chip ROM. The instruction set has been considerably enhanced. New features include a separate parallel logic unit (PLU), shadow registers for fast context save, JTAG serial scan emulation, and software wait states. This generation is designed to execute instructions in 35 ns.

1.3 TMS320 Floating-Point DSPs

There are two generations of floating-point DSPs—the TMS320C3x and the TMS320C4x, all of which have a 32-bit architecture with 40-bit extended-precision registers. The floating-point devices are based on Von Neuman architecture. However, multiple buses have been included to provide even faster throughput than traditional Harvard architectures. Its features include a hardware floating-point multiplier and a floating-point ALU.

1.3.1 TMS320C3x

The TMS320C3x — the first 32-bit floating-point generation of the TMS320 family of DSPs — includes the TMS320C30 and TMS320C31. The TMS320C30 features 2K×32 words of on-chip RAM, 4K×32 of on-chip ROM, and 64-word instruction cache. Other features include a separate DMA, two serial ports, and two timers. The TMS320C30 features two external 32-bit data buses and a 16M-word address space. Instruction-cycle time is 60 ns and performance reaches up to 33 MFLOPS (million floating-point operations/second).

The TMS320C30 is a high-performance 32-bit device that is capable of executing up to 33 MFLOPS (million floating-point operations per second). The architecture of the TMS320C30 is specifically designed to be an efficient compiler platform. This feature makes it possible to program realtime DSPs, using the TI extended-precision, optimizing C compiler. Greater parallelism and general-purpose features facilitate high performance and ease of use. Applications are greatly enhanced by the large address space, multiprocessor interface, internally/externally generated wait states, two timers, two serial ports, multi-interrupt structure, and multitasking capabilities. The TMS320C30 supports a wide variety of system applications, ranging from host-processor to dedicated-coprocessor.

The TMS320C31 is a low-cost TMS320C3x, incorporating all of the functionality and speed of the TMS320C30's core CPU at a cost comparable to that of high-speed/fixed-point DSPs. This has been achieved by omitting the expansion bus, the 4K×32-bit internal ROM, and one serial port, adding a boot ROM, and by packaging the TMS320C31 in a 132-pin quad flat pack (QFP) package. It is designed for DSP applications requiring cost-effective floating-point performance.

1.3.2 TMS320C4x

The TMS320C40 has been specifically designed to meet the needs of parallel processing. Key features include two identical 32-bit external address and 32-bit data buses, six 8-bit communication ports, and a six-channel/self-programmable DMA coprocessor. It contains a 512-byte instruction cache and two blocks of 4K bytes of on-chip RAM. This generation is designed to execute an instruction in 40 ns, to perform up to 275 MOPS (million operations/second), and to provide a 320 Mbytes/second throughput. The TMS320C40 is source-code upward compatible with the TMS320C30.

1.4 TMS320 Development Support

Texas Instruments supports designers in the complete development of their application, from concept to production. This results in fast time to market, design ease, and increased productivity. TI development support includes assemblers, linkers, compilers, simulators, C/assembly source debuggers, standard evaluation modules, and software development systems.

Application books/reports describe theory and implementation of selected TMS320 applications, including algorithms, code, and block/schematic/logic diagrams. Currently, there are over 2,000 pages of application reports to support the TMS320 family.

Chapter 2

SCSI Overview and Specifications

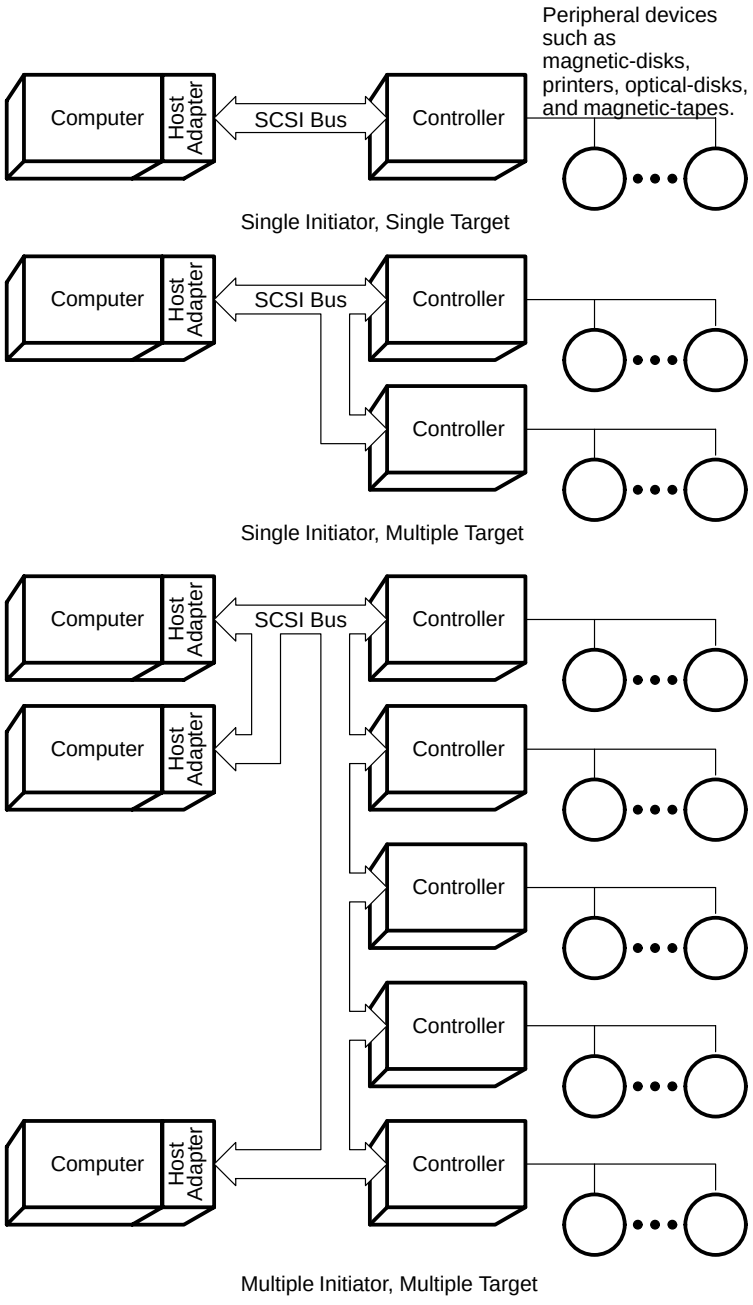
Chapter 2 provides a basic overview of the Small Systems Computer Interface (SCSI) and how the SCSI specification is implemented by the TMS320-SCSI target controller.

SCSI is an 8-bit parallel I/O bus that provides host computers with device independence within a class of peripherals. This means that different disk drives, tape drives, and printers can be added to a host computer without major modifications to the system hardware or software. Figure 2–1 illustrates sample SCSI configurations.

Topic **Page**

Addressing Method **2-3**
Interface **2-3**
Data Transfer **2-4**
Host Memory/Host Adapter/SCSI Controller Relationship **2-5**
Communication Protocol **2-7**
SCSI Command Example **2-9**
SCSI Bus Signal Descriptions **2-11**
SCSI Bus Conditions **2-13**
SCSI Commands **2-19**

Figure 2–1. Sample SCSI Configurations



2.1 Addressing Method

The SCSI bus uses logical as opposed to physical addressing for data blocks. Each logical unit can be interrogated to determine how many blocks it contains. A logical unit may coincide with all or part of a peripheral device.

2.2 Interface

The SCSI bus provides two electrical specifications—single-ended and differential. The single-ended interface uses TTL logic levels and is primarily intended for applications with shorter cable lengths. The single-ended cable length is 20 feet maximum.

The differential interface uses EIA-485 driver and receiver signals and is intended for applications requiring longer cable lengths. A cable length of up to 80 feet can be used.

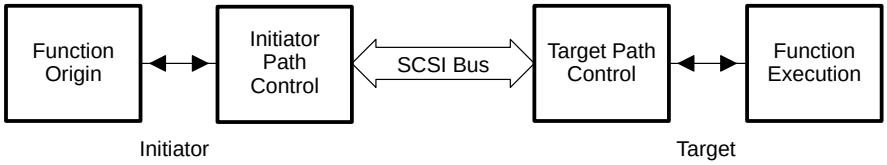
2.3 Data Transfer

The SCSI bus provides two methods of handshaking—asynchronous and synchronous. Asynchronous requires a handshake for every byte transferred. Synchronous transfers a series of bytes before any handshake occurs. This increases the data transfer rate.

The SCSI bus supports up to eight devices, including the host(s). The bus protocol includes provision for the connection of multiple hosts, referred to as initiators. Initiators are SCSI devices capable of initiating an operation. The interface protocol also includes provision for connection of multiple peripherals, referred to as target devices. SCSI target devices are capable of responding to a request from an initiator to perform an operation. All peripherals are targets and some targets can act as initiators by initiating operations such as the copy operation.

Figure 2–2 shows a system in which an initiator and target controller communicate on the SCSI bus in order to execute a command.

Figure 2–2. Simplified SCSI System

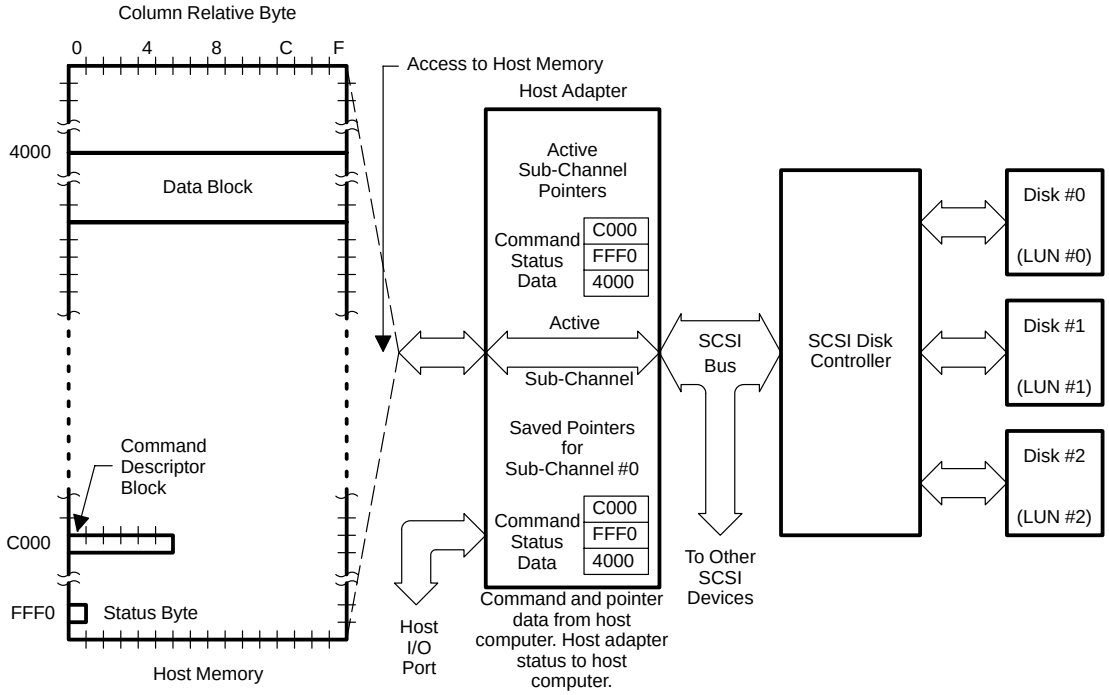


2.4 Host Memory/Host Adapter/SCSI Controller Relationship

Host memory blocks are used for command, data, and status interchange between the host system and the SCSI controller. The SCSI host adapter, which acts as the SCSI peripheral's gateway into host memory, is in the middle of this exchange. This host adapter provides an information path from the SCSI bus to the host bus, helps assure data integrity and proper performance of the I/O subsystem, and is an important part of the overall intelligence of SCSI.

The host memory contains three I/O blocks—command, data, and status blocks. The SCSI controller reads the command block and writes to the status block in order to perform the task specified by the host in the command block. Likewise, the controller must both read and write to the data block. The SCSI controller reaches into host memory via the SCSI host adapter. In order for it to reach into the right spot in memory, the host adapter must know the addresses of the command, data, and status blocks. In other words, the host must give the host adapter a pointer to the start of each block. After the SCSI controller receives information from the command block, the memory pointer for the command block advances to the next byte. The data and status pointers also function in this way.

Figure 2–3. Snapshot Prior to Initial Selection



Although the initiator host adapter and the target disk controller are disconnected, they are logically connected together in such a way that many I/O commands can be executed in the system simultaneously using a single physical bus. The connection runs from the host memory I/O block to the peripheral device performing the operation. An operation between the host and a target may become temporarily suspended in order to perform a higher priority task. In this case, the logical connection remains intact although the physical connection is broken. When the operation is resumed, the logical connection permits the host and target to finish their operation as if they had never been interrupted. Figure 2–3 illustrates this concept.

2.5 Communication Protocol

Communication over the SCSI bus is controlled by a sequence of bus phases. The SCSI bus has eight different phases. Four of these phases (Command, Data, Status, and Message) are referred to as information-transfer phases because they are each used to transfer data or information over the SCSI bus. The bus phases include:

- Bus-free phase—This phase indicates that no SCSI device is actively using the SCSI bus and that it is available for use.
- Arbitration phase—This phase permits only one SCSI device to gain control of the SCSI bus as either an initiator or a target. Arbitration is required in systems that have multiple hosts and/or multiple targets. Arbitration is not required within a single-host and single-target system. Refer to Figure 2–4 and Figure 2–5.
- Selection phase—This phase permits an initiator to select a target to perform an operation such as a read or write command.
- Reselection phase—This phase permits a target to reconnect to an initiator to continue an operation that was previously started but was suspended by the target device.
- Command phase—This phase allows the target to request command information from the initiator.
- Data phase—This phase refers to both the data-in and data-out phases.

  The data-in phase allows the target to request that data be sent from the target to the initiator. The data-out phase allows the target to request that data be sent from the initiator to the target.
- Status phase—This phase allows the target to request that status information be sent from the target to the initiator.
- Message phase—The message phase refers to both the message-in and message-out phases. Multiple messages may be sent during either phase. The first byte transferred in either of these phases is either a single-byte message or the first byte of a multiple-byte message.

  The message-in phase allows the target to request that message byte(s) be sent from the target to the initiator. The message-out phase allows the target to request that message byte(s) be sent from the initiator to the target.

Figure 2–4. Phase Sequences Using the Arbitration Phase (Multiple-Host and/or Multiple-Target SCSI Bus System)

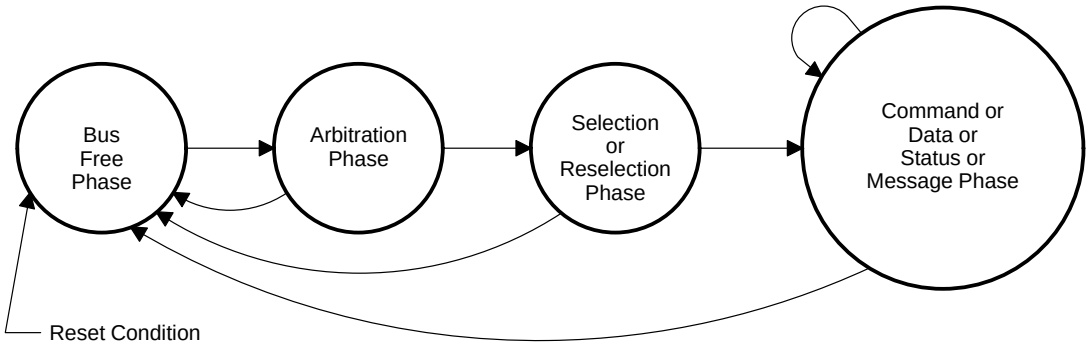
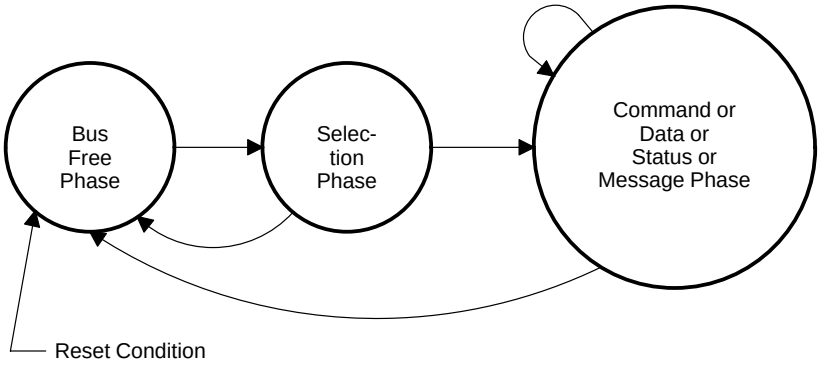


Figure 2–5. Phase Sequences That Are Not Using the Arbitration Phase (Single-Host and Single-Target SCSI Bus System)

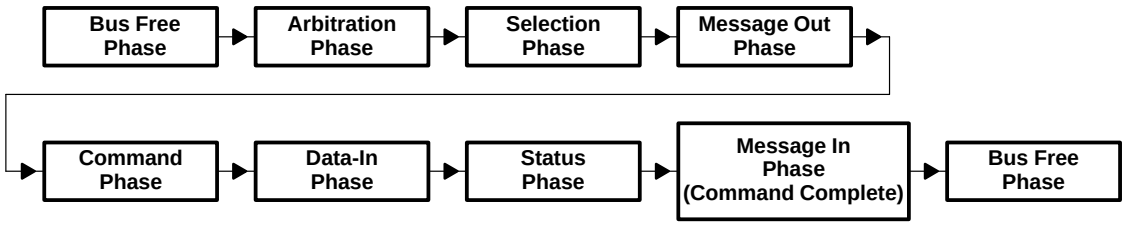


2.6 SCSI Command Example

2.6.1 Single Command Example

A typical operation on the SCSI bus is likely to include a single READ command to a peripheral device. This operation is described in detail starting with a request from the initiator. This example assumes that no malfunctions or errors occur and is illustrated in Figure 2–6.

Figure 2–6. READ Command Example Without Disconnect



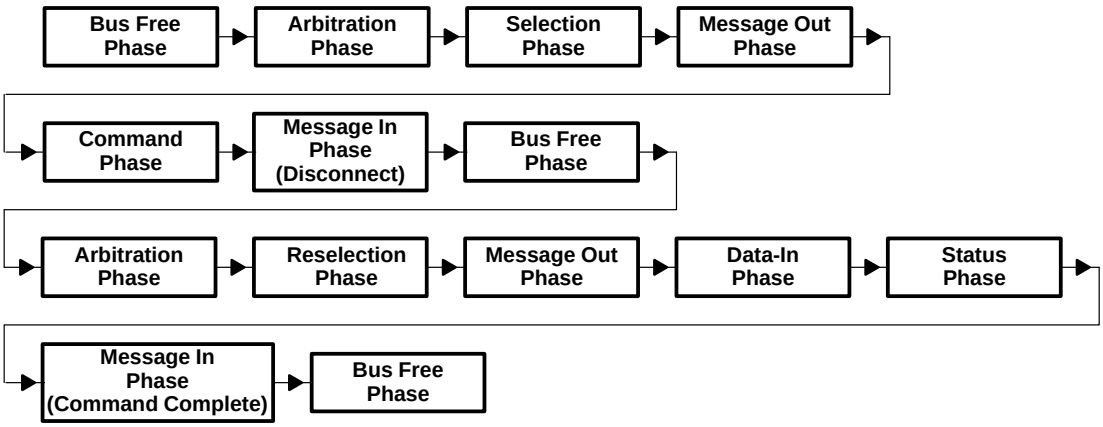
The initiator has active pointers and a set of stored pointers representing currently active, but physically disconnected SCSI devices (an initiator without disconnect capability does not require stored pointers). The initiator sets up the active pointers for the operation requested, arbitrates for the SCSI bus, and selects the target. Once this process is completed, the target assumes control of the operation.

The target obtains the command from the initiator (in this case, a READ command). The target interprets the command and executes it. In this case, the target gets the data from the peripheral device and sends it to the initiator. At the completion of the READ command, the target sends a status byte to the initiator. To end the operation, the target sends a COMMAND COMPLETE message to the initiator.

2.6.2 Disconnect Example

In the above single command example, the length of time necessary to obtain the data may require a time-consuming physical seek. In order to improve system throughput, the target may disconnect from the initiator, freeing the SCSI bus to allow other requests to be sent to other logical units. To do this, the initiator needs to be reselectable and capable of restoring the pointers upon reconnection. The target needs to be capable of arbitrating for the SCSI bus and reselecting the initiator. See Figure 2–7.

Figure 2–7. READ Command Example With DIsconnect



After the target has received the READ command (and has determined that there will be a delay), it disconnects by sending a DISCONNECT message and releasing BSY, to indicate that the bus is still in use.

When the data is ready to be transferred, the target reconnects to the initiator. As a result of this reconnection, the initiator restores the pointers to their most recently saved values (which, in this case, are the initial values) and the target continues (as in the single command example) to finish the operation. The initiator recognizes that the operation is complete when COMMAND COMPLETE message is received.

2.7 SCSI Bus Signal Descriptions

The SCSI bus signals and their descriptions are as follows:

Data Signals (D7–D0): The initiator and target use this bidirectional, 8-bit, parallel bus to transfer data. This bus not only transfers data, commands, status, and messages but also transfers SCSI device ID codes during selection and reselection phases. The SCSI device ID is the bit-significant representation of the SCSI address, referring to one of the signal lines DB7–DB0. DB7 has the highest priority.

Data Bus Parity (DP): Parity is a SCSI bus option. If a SCSI bus uses parity, the parity must be odd and must be supported by all devices on the bus. Parity is valid for all information transfer phases, the selection phase, and the reselection phase.

SEL (Select): The initiator uses this signal to select a target to perform a command. If the disconnect option is supported, the target uses SEL to reselect and initiator from which it disconnected. After reselection, command execution continues. I/O differentiates between selection and reselection.

BSY (Busy): This *wired or* signal indicates that the bus is in use (busy). The initiator uses the BSY signal during the arbitration phase. The selected target uses the BSY signal to acknowledge selection and to indicate that it has bus control. The target also uses BSY to participate in arbitration.

C/D (Control/Data): The target uses this signal to indicate if control or data information is being transferred. The asserted state (true) indicates command, status, or message information is being transferred. The negated state (false) indicates data is being transferred.

I/O (Input/Output): The target uses this signal to define the direction of information transferred between the initiator and the target. Transfer direction is defined with respect to the initiator. The asserted state (true) indicates information is being transferred to the initiator from the target. The negated state (false) indicates that information is being transferred from the initiator to the target.

MSG (Message): The target uses this signal to indicate to the initiator that the information being transferred is a message. The target sends a command completion message to the initiator after completion of each command.

REQ (Request): The target uses this signal to initiate a request for necessary data information transfer between the initiator and the target. In response to the assertion of REQ by the target, the initiator accepts data from the bus during an information-in phase or places information on the data bus during an information-out phase. The target continues to assert REQ until the initiator responds with ACK, except during synchronous data transfers.

ACK (Acknowledge): This initiator uses this signal to respond to the assertion of the REQ signal by the selected target. Assertion of the ACK signal by the initiator indicates that is has placed information on the data bus during an information-out phase, or accepted data from the bus during an information-in phase. The REQ/ACK handshake is used for the transfer of all information between the initiator and the target.

ATN (Attention): The initiator uses this signal to inform the selected target that a message is available. The target, at its convenience, requests the initiator's message, using the message-out phase. The initiator may assert the ATN signal during the selection phase or at any time after the target assumes control of the bus.

RST (Reset): This *wired or* signal is used by any device on the bus. Normally, RST is asserted only by the initiator during power-up. Targets may also assert RST during power-up or power-down.

Reference Appendix A for all signals.

2.8 SCSI Bus Conditions

2.8.1 ATTENTION Condition

The ATTENTION condition allows the initiator to inform the target controller that the initiator is ready to send a message. The target controller gets this message at its convenience by entering the MESSAGE OUT phase. The target controller detects the assertion of the $\overline{\text{ATN}}$ signal — signifying that the block of INFORMATION bytes has been received or transmitted. Until the target controller performs a MESSAGE OUT phase, the initiator sends or receives dummy information.

2.8.2 RESET Condition

The target controller is reset whenever power is turned on, the RST signal is asserted, or the BUS DEVICE RESET message is received. These three conditions have the same effect upon the target controller except that the self-diagnostic sequence is performed only at power up. After the target controller is reset, RESET clears all of the uncompleted commands, releases all reservations, and resets all SCSI device operating modes (MODE SELECT commands, etc.) to their default conditions.

2.8.3 UNIT ATTENTION Condition

The UNIT ATTENTION condition for the logical unit begins when either the medium for each initiator is loaded (inserted into the drive unit), the target controller is reset, or one of the MODE SELECT parameters is changed. The UNIT ATTENTION condition persists for each initiator until the initiator issues a command to the logical unit other than the REQUEST SENSE or INQUIRY; thereupon, the target controller returns the CHECK CONDITION status to that issuing initiator. If the next command (in response to the CHECK CONDITION status) from that initiator to the logical unit is REQUEST SENSE, the UNIT ATTENTION sense key is returned. If any command other than REQUEST SENSE is received, the UNIT ATTENTION condition is lost.

If the UNIT ATTENTION condition is pending and the initiator sends the INQUIRY command before the target controller reports its CHECK CONDITION status, the controller ignores UNIT ATTENTION and performs the INQUIRY command.

If the UNIT ATTENTION condition is pending and the initiator sends the REQUEST SENSE command before the target controller reports its CHECK CONDITION status, the controller discards all of the pending sense data, reports the UNIT ATTENTION sense key, and clears the UNIT ATTENTION condition for that initiator.

2.9 Messages

The target controller supports the messages listed in Table 2–1.

Table 2–1. Messages Supported by Target Controller

Code	Message Name	Direction
00h	COMMAND COMPLETE	In
01h	EXTENDED MESSAGE SYNCHRONOUS DATA TRANSFER REQUEST	In/Out
02h	SAVE DATA POINTER	In
03h	RESTORE POINTERS	In
04h	DISCONNECT	Out
05h	INITIATOR DETECTED ERROR	Out
06h	ABORT	– – –
07h	MESSAGE REJECT	In/Out
08h	NO OPERATION	Out
09h	MESSAGE PARITY ERROR	Out
0Ch	BUS DEVICE RESET	Out
X0h	IDENTIFY	In/Out

2.9.1 COMMAND COMPLETE (00h)

After the current command is executed and the VALID status is issued to the initiator, the target controller sends the COMMAND COMPLETE message to the initiator. The controller's SCSI bus is placed into BUS FREE unless the initiator sets the $\overline{\text{ATN}}$ line.

2.9.2 EXTENDED MESSAGE SYNCHRONOUS DATA TRANSFER REQUEST (01h)

SCSI devices initiate this message exchange whenever it is appropriate to negotiate a new data transfer agreement (either synchronous or asynchronous). SCSI devices not capable of synchronous transfer respond with a MESSAGE REJECT message.

2.9.3 SAVE DATA POINTER (02h)

Before the DISCONNECT message is issued, the target controller sends the SAVE DATA POINTER message to the initiator.

2.9.4 RESTORE POINTERS (03h)

After a successful RESELECTION phase, the target controller sends the RESTORE POINTERS message to the initiator. If the initiator rejects that mes-

sage with the MESSAGE REJECT message, the target controller terminates the present command with the CHECK CONDITION status and sets the sense key to HARDWARE ERROR (04h) for that initiator.

2.9.5 DISCONNECT (04h)

By sending the DISCONNECT message, the target controller informs the initiator that the controller's SCSI bus has been placed into the BUS FREE phase. The target controller does not disconnect if the MESSAGE REJECT message is received from the initiator.

2.9.6 INITIATOR DETECTED ERROR (05h)

By sending the INITIATOR DETECTED ERROR message, the initiator informs the target controller that an error occurred during the operation. Upon receiving this message, the controller places the SCSI bus into the BUS FREE phase.

2.9.7 ABORT (06h)

To clear the target controller's present operation, the initiator sends the ABORT message to the controller. The controller clears all of the pending data and status which pertained to the issuing initiator, and the controller places the SCSI bus into the BUS FREE phase. The controller will not clear any of the pending data and status which pertain to the other initiators; it will not send any status or ending message during this operation. If the controller is not performing an operation for the initiator, ABORT is not considered an error message.

2.9.8 MESSAGE REJECT (07h)

Both the initiator and the target controller send the MESSAGE REJECT message to indicate their last incoming message either is inappropriate or has not been implemented.

Before the target controller creates its MESSAGE REJECT message, the initiator asserts $\overline{\text{ATN}}$ prior to releasing $\overline{\text{ACK}}$; the handshake of the last message is then rejected. After the target controller sends MESSAGE REJECT, the controller places the SCSI bus into the MESSAGE IN phase. Prior to requesting additional message bytes from the initiator, the controller sends MESSAGE REJECT so that it can determine which message has been rejected.

Should the initiator issue MESSAGE REJECT, the target controller immediately terminates the present command with the CHECK CONDITION status and sets the sense key or additional sense code to HARDWARE ERROR (04h) or MESSAGE REJECT ERROR (43h) for that initiator.

2.9.9 NO OPERATION (08h)

The initiator sends the NO OPERATION message if it has no valid message for the target controller's request. The target controller receives but ignores this message.

2.9.10 MESSAGE PARITY ERROR (09h)

The initiator sends a MESSAGE PARITY ERROR message to indicate that a parity error occurred on one or more bytes of the last message sent from the controller. Prior to releasing $\overline{\text{ACK}}$, the initiator asserts $\overline{\text{ATN}}$ for the last byte of the incorrect message so that the target controller knows which one is erroneous. The target controller then resends that message. Should the MESSAGE PARITY ERROR be received again, the target controller aborts the current command for that initiator and places the SCSI bus into the BUS FREE phase. No further reconnection is attempted; neither STATUS nor COMMAND COMPLETE messages are returned for the command. Either the sense key or the error code is set to ABORTED COMMAND (0Bh) or PARITY ERROR (47h) for that initiator.

2.9.11 BUS DEVICE RESET (0Ch)

To clear all of the target controller's current commands, the initiator sends the BUS DEVICE RESET message to the controller. The controller eradicates all commands, steps through the initial power-up checks, performs its self-configuration procedure, and places the SCSI bus into the BUS FREE state.

2.9.12 IDENTIFY (C0h or 80h)

The initiator as well as the target controller can send the IDENTIFY message. The initiator sends the IDENTIFY message after it selects the target controller. The target controller sends the IDENTIFY message as its first message after the RESELECT operation.

In addition, this message specifies that the sender can support some or all of the optional messages. Table 2–2 shows that the only changeable bit is bit 6. Therefore, this message can be issued with one of two values: C0h and 80h. C0h will support the DISCONNECT/RESELECT feature; 80h will not.

Table 2–2. IDENTIFY Message Functions

Bit(s)	Identify Message Function
7	Always set to indicate `IDENTIFY`
6	Set to indicate the ability to disconnect
5 – 3	Reserved (must be 0)
2 – 0	Logical unit number 0

If the initiator responds to the controller's IDENTIFY message with the MESSAGE REJECT message, the target controller immediately terminates the present command with the CHECK CONDITION status and sets the sense key or additional sense code to HARDWARE ERROR (04h) or MESSAGE REJECT ERROR (43h) for that initiator.

Note:

The target controller will not disconnect if, during the SELECTION phase, the initiator does not set its initiator's SCSI DEVICE ID onto the bus or if the initiator does not send the appropriate IDENTIFY message (with C0h, bit 6, set) to the disk drive.

2.10 Status Phase

During the STATUS phase, the target controller sends the STATUS byte to the initiator upon termination of each command unless the command is cleared by the ABORT message, by a BUS DEVICE RESET message, or by a RESET condition. The target controller supports the status codes listed in Table 2–3.

Table 2–3. Status Codes

Code	Status
00h	GOOD
02h	CHECK CONDITION
08h	BUSY
18h	RESERVATION CONFLICT

2.10.1 GOOD (00h)

This status indicates that the target controller successfully completed the command.

2.10.2 CHECK CONDITION (02h)

Any abnormal condition, error, or exception that causes SENSE DATA to be set also causes the CHECK CONDITION status to be issued. The REQUEST SENSE command should be issued after the CHECK CONDITION status to determine the nature of that condition/error/exception.

2.10.3 BUSY (08h)

The target controller is busy. This status is returned when one command is being processed while another command is being executed for the same logical unit.

2.10.4 RESERVATION CONFLICT (18h)

This status is returned if the target controller attempted to access the logical unit that is reserved for another initiator.

2.11 SCSI Commands

A request to a target device is performed by sending a command descriptor block (CDB) to the target.

The first byte of the command descriptor block is the operation code. The operation code is divided into the opcode and the group code. The three-bit group code field provides eight group codes. The target controller supports group-0 and group-1 commands. The group-0 commands consist of six bytes; group-1 commands consist of ten bytes.

The last byte of the command descriptor block is the control byte. The control byte consists of the flag and the link bit for SCSI devices that support a LINKED COMMAND COMPLETE message. The target controller requires the control byte to be set to zero.

The relative address (RelAddr) bit of the group-1 command is set to one to indicate that the logical block address (LBA) portion of the command descriptor block is a twos complement displacement. The (RelAddr) bit is not supported by the target controller and must be set to zero.

The target controller supports the SCSI commands listed in Table 2–4.

Table 2–4. SCSI Commands

Command	Opcode	Command	Opcode
TEST UNIT READY	00h	MODE SENSE	1Ah
REZERO UNIT	01h	START/STOP UNIT	1Bh
REQUEST SENSE	03h	READ CAPACITY	25h
FORMAT UNIT	04h	READ EXTENDED	28h
REASSIGN BLOCK	07h	WRITE EXTENDED	2Ah
READ	08h	SEEK EXTENDED	2Bh
WRITE	0Ah	WRITE AND VERIFY	2Eh
SEEK	0Bh	WRITE BUFFER	3Bh
INQUIRY	12h	READ BUFFER	3Ch
MODE SELECT	15h	READ LONG	3Eh
RESERVE UNIT	16h	WRITE LONG	3Fh
RELEASE UNIT	17h		

Command **TEST UNIT READY – 00h**

The TEST UNIT READY command provides a means for the initiator to check the logical unit to determine whether the unit is ready.

Command Parameters

The command descriptor block (CDB) for the TEST UNIT READY command is formatted as shown in Table 2–5.

Table 2–5. TEST UNIT READY Command

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (00h)							
1	Logical Unit Number			Reserved				
2	Reserved							
3	Reserved							
4	Reserved							
5	Control Byte (00h)							

Error Conditions If the logical unit is not ready or if the UNIT ATTENTION condition is pending, the target controller returns the CHECK CONDITION status in response to the TEST UNIT READY command. The REQUEST SENSE command can be issued to obtain detailed information about the status of the logical unit.

If the logical unit is disconnected while performing a command, the target controller returns the BUSY status.

If the Logical Unit Number specified within CDB is not zero, the target controller returns the CHECK CONDITION status with the sense key set to ILLEGAL REQUEST (05h).

Command **REZERO UNIT – 01H**

The REZERO UNIT command requests that the target controller set the logical unit to logical block address (LBA) zero.

Command Parameters

The CDB for the REZERO UNIT command is formatted as shown in Table 2–6.

Table 2–6. REZERO UNIT Command

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (01h)							
1	Logical Unit Number			Reserved				
2	Reserved							
3	Reserved							
4	Reserved							
5	Control Byte (00h)							

Error Conditions If the logical unit is not ready or if the UNIT ATTENTION condition is pending, the target controller returns the CHECK CONDITION status in response to the REZERO UNIT command. The REQUEST SENSE command can be issued to obtain detailed information about the status of the logical unit.

If the logical unit is disconnected while performing a command, the target controller returns the BUSY status.

If the logical unit number as specified within CDB is not zero, the target controller returns the CHECK CONDITION status with the sense key set to ILLEGAL REQUEST (05h).

Command **REQUEST SENSE – 03H**

The REQUEST SENSE command is used to request that the target transfer sense data to the initiator and provides a means for the initiator to obtain detailed information after it executes a command. Typically, the REQUEST SENSE command is issued if, upon completion of the previous command, the CHECK CONDITION status is returned to the initiator. In order to obtain the sense data that the target controller saved, the initiator should issue the REQUEST SENSE command as soon as it receives the CHECK CONDITION status.

During the REQUEST SENSE command, the initiator can issue several REQUEST SENSE commands to obtain extended sense data. However, should the target controller receive a command other than REQUEST SENSE, the controller clears the sense data for the previous command.

Command Parameters

The CDB for the REQUEST SENSE command is formatted as shown in Table 2–7.

Table 2–7. REQUEST SENSE Command

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (03h)							
1	Logical Unit Number			Reserved				
2	Reserved							
3	Reserved							
4	Allocation Length							
5	Control Byte (00h)							

The Allocation Length field (byte 4) specifies the number of bytes of data that the initiator allocates for the sense information. The target controller transfers sense data until either the size of the allocation length is exhausted or all of the sense data is transferred, whichever is less.

Extended Sense Data Format

The target controller returns the extended sense data format, available for all commands, when the allocation length specified in the REQUEST SENSE command is greater than 0 bytes. The extended sense data format is shown in Table 2–8.

Table 2–8. Extended Sense Data Format

Byte \ Bit	7	6	5	4	3	2	1	0
0	Valid	1	1	1	0	0	0	0
1	Reserved							
2	0	0	ILI	0	Sense Key			
3	Information Byte (MSB)							
4	Information Byte							
5	Information Byte							
6	Information Byte (LSB)							
7	Additional Sense Length (0Ah)							
8	Reserved							
9	Reserved							
10	Reserved							
11	Reserved							
12	Additional Sense Code							
13	Reserved							
14	Field Replaceable Unit							
15	FPV	C/D	Reserved			BPV	Bit Pointer	
16	Field Pointer							
17	Field Pointer							

If set, Valid (bit 7 of byte 0) indicates that the Information Byte field contains valid information about the error condition.

If set, Incorrect Length Indicator (ILI — bit 5 of byte 2) indicates that the data within the target controller was larger than the requested transfer size during the READ DATA BUFFER command. This bit is not used by any other command and is always zero except when a size difference occurs at which time bit 5 is set to one.

The Sense Key field (bits 0–3 of byte 2) indicates status information about any error detected during the operation. The target controller supports the sense keys listed and described in Table 2–9.

Table 2–9. Sense Key Descriptions

Sense Keys	Description
0h	NO SENSE — Indicates that the target controller has no specific sense key information for the designated logical unit.
1h	RECOVERED ERROR — Indicates that after performing the necessary recovery action, the target controller successfully completed the last command.
2h	NOT READY — Indicates that the addressed logical unit cannot be accessed. Operator intervention may be required to correct this error condition.
3h	MEDIUM ERROR — Indicates that the target controller terminated the command due to a nonrecoverable error condition such as a flaw within the medium or an error within the recorded data.
4h	HARDWARE ERROR — Indicates that the target controller detected a nonrecoverable hardware failure (for example, controller failure, device failure, parity error, etc.) while performing the command or during a self test.
5h	ILLEGAL REQUEST — Indicates that there was an illegal parameter in the command descriptor block or in the additional parameters supplied as data for some commands. If the target controller detects an invalid parameter in the command descriptor block, then it terminates the command without altering the medium. If the target controller detects an invalid parameter in the additional parameters supplied as data, then the target may have already altered the medium.
6h	UNIT ATTENTION — Indicates that the removable medium has been changed or the target controller has been reset.
7h	DATA PROTECT — Indicates that a write operation was attempted on a write-protected drive.
9h	Not currently used; is vendor-specific.
Bh	ABORTED COMMAND — Indicates that the target controller aborted the command. The initiator may be able to recover the lost data by trying the command again.
Eh	MISCOMPARE — Indicates that the source data did not match the data read from the medium.

The Information Byte field (bytes 3–6) is command-specific and only valid when the Valid bit is set to one. Refer to the individual command for bit information about the bytes that were returned within this field.

The Additional Sense Length field (byte 7) specifies the number of bytes of additional sense data that follows the initial sense data. The additional sense data contains information that further defines the nature of the CHECK CONDITION status code. The Additional Sense Length byte is set to 10 for all commands. Should the allocation length specified in the REQUEST SENSE CDB be too small to transfer the additional sense data, the excess data is not transferred; the Additional Sense Length field will not be adjusted to reflect this truncation.

Bytes 8–11 are reserved and are always set to zero.

The Additional Sense Code field (byte 12) contains additional information about the event that has occurred. When this field is set to zero, the target controller no longer has additional sense information or does not have any appropriate additional sense to return for the CHECK CONDITION status that it created. Refer to Table 2–10 for a list of additional sense codes and their defi-

nitions. The target controller can apply these additional sense codes with the same code to both the extended and the nonextended sense formats.

The Field Replaceable Unit field (FRU — byte 14) is not supported by the target controller and is always set to zero.

If set to one, the Field Pointer Valid bit (FPV — bit 7 of byte 15) indicates that the information within the control/data (C/D) bit position, bit pointer valid (BPV) bit position, and Field Pointer field (bytes 16 and 17) is valid. Normally, this bit is only valid when the ILLEGAL REQUEST sense key is returned. In that case, the Field Pointer field points to the byte that caused the error. When this field is set to zero, the disk drive no longer has information about which CDB or location is in error.

The Control/Data bit (C/D — bit 6 of byte 15) is valid only if the FPV bit is set to one. C/D is set to zero if the Field Pointer field is pointing to a byte within the command parameters that were passed to the target controller during the DATA OUT phase. C/D is set to one if the Field Pointer field is pointing to a byte within the CDB.

The Bit Pointer Valid bit (BPV — bit 3 of byte 15) is valid only if the FPV bit is set to one. BPV is set to one if the information within the Bit Pointer field is valid.

The Bit Pointer field (bits 0–2 of byte 15) is valid only if the BPV bit is set to one. This field specifies the erroneous bit position within the Field Pointer field that caused the ILLEGAL REQUEST sense key to be returned.

The Field Pointer field (bytes 16–17) is valid only if the FPV bit is set to one. This field specifies the erroneous byte position that caused the ILLEGAL REQUEST sense key to be returned. Field Pointer can point to either the command parameters passed during the DATA OUT phase or the CDB, depending on the value of the C/D bit.

Table 2–10. Additional Sense Codes of the REQUEST SENSE Command

Sense Codes	Description
00h	NO ADDITIONAL SENSE INFORMATION — The target controller has no additional sense information available for the previous command.
02h	NO SEEK COMPLETE — The target controller could not complete a SEEK operation.
03h	WRITE FAULT — The target controller determined that a fault occurred during a WRITE operation.
04h	DRIVE NOT READY — The target controller is not ready.
05h	DRIVE NOT SELECTED — The target controller cannot be selected.
06h	NO TRACK ZERO — The target controller could not rezero the positioner.
10h	ID FIELD CRC ERROR — A CRC error was found when reading the sector ID field.
11h	UNRECOVERED DATA ERROR — A block could not be read after the number of retry attempts specified in the MODE SELECT command. UNRECOVERED READ ERROR OF DATA BLOCKS
12h	ID FIELD ADDRESS MARK NOT FOUND — The target controller could not locate the address mark for the sector header. NO ADDRESS MARK FOUND IN ID FIELD

Table 2–10. Additional Sense Codes of the REQUEST SENSE Command (Continued)

Sense Codes	Description
13h	DATA ADDRESS MARK NOT FOUND — The target controller could not locate the address mark for the sector data area. NO ADDRESS MARK FOUND IN DATA FIELD
14h	RECORD NOT FOUND — The block sequence is improper, a block is missing, or the block cannot be read.
15h	SEEK ERROR — The address that is specified within the CDB of the current command is not the same as the address that is specified within the cylinder of the data header. SEEK POSITIONING ERROR
17h	RECOVERED READ ERROR with retries — The target controller encountered an error which was recovered using retries, without ECC, while reading the media.
18h	RECOVERED READ ERROR with ECC — While reading the media, the target controller used the ECC correction to recover from the error.
19h	DEFECT LIST ERROR — While accessing one of the defect lists, the target controller encountered an error.
1Ah	PARAMETER OVERRUN — The Parameter List Length specified in the CDB by the initiator is too large for the target controller.
1Bh	SYNCHRONOUS TRANSFER ERROR — An error occurred during the synchronous data transfer.
1Ch	PRIMARY DEFECT LIST NOT FOUND — The target controller cannot locate the Primary Defect List (commonly called P LIST).
1Dh	COMPARE ERROR — One or more bytes of the command's CDB does not match the corresponding byte(s) of the issued VERIFY or WRITE AND VERIFY command.
1Eh	RECOVERED ID WITH TARGET'S ECC CORRECTION — The sector ID field could not be read without the ECC correction.
20h	INVALID COMMAND OPERATION CODE — The initiator issued a command that either cannot be executed or is not applicable.
21h	ILLEGAL LOGICAL BLOCK ADDRESS — The address of the desired block is greater than the LBA returned by the READ CAPACITY data with the PMI bit not set in CDB.
22h	ILLEGAL FUNCTION FOR DEVICE TYPE — The target controller cannot perform the requested function.
24h	ILLEGAL FIELD IN CDB — One of the fields within the CDB contains either a value that is incorrect or a value other than zero if the field is reserved.
25h	INVALID LUN — The Logical Unit Number specified in either the CDB or the SCSI IDENTIFY message is not zero.
26h	INVALID FIELD IN PARAMETER LIST — One of the fields within the PARAMETER LIST contains either a value that is incorrect or a value other than zero if the field is reserved.
27h	WRITE-PROTECTED — Being write-protected, the target controller aborted the outstanding WRITE command.
28h	MEDIUM CHANGED — The drive detected the NOT READY condition followed by the READY condition.
29h	POWER-ON or RESET or BUS DEVICE RESET OCCURRED — Either the SCSI BUS RESET condition, BUS DEVICE RESET message, or POWER-ON/RESET condition reset the target controller.
2Ah	MODE SELECT PARAMETERS CHANGED — Another initiator changed the MODE SELECT parameters of the target controller and may affect current operations.
31h	FORMAT FAILED — The FORMAT UNIT command cannot be completed. MEDIUM FORMAT CORRUPTED

Table 2–10. Additional Sense Codes of the REQUEST SENSE Command (Concluded)

Sense Codes	Description
32h	NO DEFECT SPARE LOCATION AVAILABLE — No alternate tracks remain on the addressed target controller. This error condition can occur if the controller is processing either the FORMAT UNIT or the REASSIGN BLOCK command.
40h	RAM FAILURE — The target controller detected a RAM error during the SEND DIAGNOSTIC test operation.
43h	MESSAGE REJECT ERROR — The initiator issued the MESSAGE REJECT message in response to the message that the target controller sent.
44h	SCSI HARDWARE/FIRMWARE ERROR — The SCSI firmware detected either an internal firmware or a hardware error and cannot complete the current command. INTERNAL CONTROLLER ERROR
45h	SELECT/RESELECT FAILED — While attempting the reselection operation, the SCSI firmware detected a time-out error.
47h	SCSI INTERFACE PARITY ERROR — A parity error occurred on the SCSI bus and the target controller cannot recover the data.
48h	INITIATOR DETECTED ERROR — The initiator sent the INITIATOR DETECTED ERROR message and the target controller cannot recover from the error.
49h	INAPPROPRIATE/ILLEGAL MESSAGE — The initiator sent either an inappropriate or an illegal SCSI message to the target controller.

Error Conditions

If the CHECK CONDITION status is received in response to the REQUEST SENSE command, all sense data that the target controller returns is invalid.

Command **FORMAT UNIT – 04h**

The FORMAT UNIT command ensures that all data blocks can be accessed by the target controller. The controller maintains the Defect List — located in the reserved area of the RAM that is accessible only by the controller. During the formatting process, the initiator can use the spare part of the data buffer to specify that a set of defective blocks be reassigned.

The Initiator Defect List lists those defect descriptors that the initiator supplies to the target controller during the DATA OUT phase. The initiator furnishes this list when both the FmtData bit of the FORMAT UNIT's CDB is set to one and bytes 2 and 3 of the Defect List Header are nonzero. If the Defect List Length of the Defect List Header is equal to zero, the initiator will not transfer the defect descriptors.

Any defect that the target controller receives during the REASSIGN BLOCK command is appended to the Defect List that is maintained by the target controller (not by the Initiator Defect List).

Command Parameters

The CDB for the FORMAT UNIT command is shown in Table 2–11.

Table 2–11. FORMAT UNIT Command

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (04h)							
1	Logical Unit Number			FmtData	CmpLst	Defect List Format		
2	Format Data Pattern							
3	Interleave Factor (MSB)							
4	Interleave Factor (LSB)							
5	Control Byte (00h)							

The target controller currently ignores the Defect List Format (bits 0–2 of byte 1). The target controller only supports those defect descriptors that are in block format. Also, the target controller currently ignores CmpList (bit 3 of byte 1), Format Data Pattern (byte 2) that is vendor unique, and Interleave Factor (bytes 3–4).

When set to zero, FmtData (bit 4 of byte 1) indicates that the DATA OUT phase does not occur (the initiator will not supply the Defect List Header or the defect descriptors). The DATA OUT phase contains the 4-byte Defect List Header. When set to one, FmtData indicates that the DATA OUT phase occurs during the command's execution.

Command **Format Modes**

The format mode is selected by combining the bits of the FORMAT UNIT's CDB with the bits of the initiator's Defect List Header. By controlling the appropriate fields, the initiator can determine which defect lists are to be used for the FORMAT UNIT command. Because it only uses the defect lists supplied by the initiator, the target controller ignores all other modes of the formatting process.

Defect List Header Format

The Defect List Header is 4 bytes long; defect descriptors may or may not follow it. This header specifies several parameters for the FORMAT mode and the total number of bytes in the defect list. The Defect List Header's CDB for the FORMAT UNIT command is formatted as shown in Table 2–12.

Table 2–12. Defect List Header for the FORMAT UNIT Command

Byte \ Bit	7	6	5	4	3	2	1	0
0	Reserved							
1	FOV	DPRY	STPF	Reserved				VU
2	Defect List Length (MSB)							
3	Defect List Length (LSB)							

The target controller currently ignores Format Options Valid (FOV — bit 7 of byte 1), Disable Primary (DPRY — bit 6 of byte 1), Stop Format (STPF — bit 5 of byte 1), Vendor Unique (VU — bit 0 of byte 1), and Disable Certification (DCRT).

The Defect List Length (bytes 2–3) specifies the number of bytes that the initiator is to transfer.

Defect Descriptor Block Format

The Defect Descriptor Block Format's CDB for the FORMAT UNIT command is formatted as shown in Table 2–13.

Table 2–13. Defect Descriptor Block Format's CDB for the FORMAT UNIT Command

Byte \ Bit	7	6	5	4	3	2	1	0
0	Defect Logical Block Address (MSB)							
1	Defect Logical Block Address							
2	Defect Logical Block Address							
3	Defect Logical Block Address (LSB)							

The Defect Logical Block Address (Defect LBA — bytes 0–3) specifies the address of the logical block that contains the defective sector(s).

Command **REASSIGN BLOCK – 07h**

The REASSIGN BLOCK command requests that the target controller relocate the logical block(s) from defective sector(s) to nondefective sector(s).

During the DATA OUT phase of the REASSIGN BLOCK command, the initiator transfers the defect list that contains the logical block(s) to be reassigned. The target controller reassigns sectors to those logical blocks that are specified by the initiator. During the REASSIGN operation, the target controller moves the appropriate data blocks from the area of the data buffer that the initiator designated defective to the spare area of the data buffer that the controller allocated for relocation. Upon successful completion of the REASSIGN BLOCK command, the controller writes one or more entries into the replacement table.

Command Parameters

The CDB for REASSIGN BLOCK command is formatted as shown in Table 2–14.

Table 2–14. REASSIGN BLOCK Command

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (07h)							
1	Logical Unit Number			Reserved				
2	Reserved							
3	Reserved							
4	Reserved							
5	Control Byte (00h)							

Defect List Header Format

The Defect List Header, list header for REASSIGN BLOCK's parameters, is a 4-byte header and contains the Defect List Length that is followed by zeros or more defect descriptors. The length of each descriptor is 4 bytes.

During the DATA OUT phase of the REASSIGN BLOCK command, the initiator transfers information about the command's parameters to the target controller.

The CDB for Defect List Header (list header for REASSIGN BLOCK's parameters) is formatted as shown in Table 2–15.

Table 2–15. Defect List Header Format

Byte \ Bit	7	6	5	4	3	2	1	0
0	Reserved							
1	Reserved							
2	Defect List Length (MSB)							
3	Defect List Length (LSB)							

The Defect List Length field (bytes 2–3) specifies the total length, in bytes, of the defect descriptors that follow. This length is equal to four times the number of defect descriptors. A Defect List Length of zero is not considered by the target controller to be an error condition.

Defect Descriptor Block Format

Each defect descriptor contains a 4-byte Defect Logical Block Address (defect LBA — bytes 0–3) that specifies the location of the associated defect. These defect descriptors must be in ascending order. The CDB for the defect descriptor of the REASSIGN BLOCK command is formatted as shown in Table 2–16.

Table 2–16. Defect Descriptor CDB of the REASSIGN BLOCK Command

Byte \ Bit	7	6	5	4	3	2	1	0
0	Defect Logical Block Address (MSB)							
1	Defect Logical Block Address							
2	Defect Logical Block Address							
3	Defect Logical Block Address (LSB)							

Error Conditions

If the defect LBA is invalid, the target controller terminates the REASSIGN BLOCK command with a CHECK CONDITION status and sets the sense key or additional sense code to ILLEGAL REQUEST (05h) or ILLEGAL LOGICAL BLOCK ADDRESS (21h), respectively.

Command **READ – 08h**

The READ command requests that the target controller transfer data from the logical unit to the initiator.

Command Parameters

The CDB for the READ command is formatted as shown in Table 2–17.

Table 2–17. READ Command

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (08h)							
1	Logical Unit Number			Logical Block Address (MSB)				
2	Logical Block Address							
3	Logical Block Address (LSB)							
4	Transfer Length							
5	Control Byte (00h)							

The Logical Block Address field (LBA — bits 0–4 of byte 1 and all of bytes 2–3) specifies the address of the logical block where the READ operation begins.

The Transfer Length field (byte 4) specifies the number of contiguous logical blocks of data to be transferred. For example, a Transfer Length of zero indicates that 256 logical blocks are transferred.

Error Conditions If the LBA field is invalid and/or if the LBA plus the Transfer Length results in an invalid block address, the target controller will not transfer the pending data. Instead, the controller terminates the READ command with a CHECK CONDITION status and sets the sense key or additional sense code to ILLEGAL REQUEST (05h) or ILLEGAL LOGICAL BLOCK ADDRESS (21h), respectively.

Command **WRITE – 0Ah**

The WRITE command requests that the target controller write the data transferred from the initiator.

Command Parameters

The CDB for the WRITE command is formatted as shown in Table 2–18.

Table 2–18. WRITE Command

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (0Ah)							
1	Logical Unit Number			Logical Block Address (MSB)				
2	Logical Block Address							
3	Logical Block Address (LSB)							
4	Transfer Length							
5	Control Byte (00h)							

The Logical Block Address field (LBA — bits 0–4 of byte 1 and bits 0–7 of bytes 2 and 3) specifies the address of the logical block where the WRITE operation begins.

The Transfer Length field (byte 4) specifies the number of logical blocks of data to be transferred. For example, a Transfer Length of zero indicates that no logical blocks are transferred.

Error Conditions If the LBA field is invalid and/or if the LBA plus the Transfer Length results in an invalid block address, the target controller will not transfer the pending data. Instead, the controller terminates the WRITE command with a CHECK CONDITION status and sets the sense key or additional sense code to ILLEGAL REQUEST (05h) or ILLEGAL LOGICAL BLOCK ADDRESS (21h), respectively.

Command **SEEK – 0Bh**

The target controller utilizes the SEEK command as a no-operation process because access to the data block involves no mechanical movement from external sources.

Command Parameters

The CDB for the SEEK command is formatted as shown in Table 2–19.

Table 2–19. SEEK Command

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code (0Bh)							
1	Logical Unit Number			Logical Block Address (MSB)				
2	Logical Block Address							
3	Logical Block Address (LSB)							
4	Reserved							
5	Control Byte (00h)							

Error Conditions If the LBA field is invalid, the target controller terminates the SEEK command with a CHECK CONDITION status and sets the sense key or additional sense code to ILLEGAL REQUEST (05h) or ILLEGAL LOGICAL BLOCK ADDRESS (21h), respectively.

Command **INQUIRY – 12H**

The INQUIRY command provides a means for the initiator to request information from the disk target controller.

If the initiator transmits the INQUIRY command while the target controller's UNIT ATTENTION condition is pending, the controller executes the INQUIRY command, returns the GOOD status, and does not clear the UNIT ATTENTION condition.

If the initiator transmits the INQUIRY command directly to a nonexistent LUN (such as, any value other than zero), the target controller then transfers the INQUIRY response data back to the initiator and terminates the command with the GOOD status. Upon receipt, the initiator examines the Peripheral Device Type field to determine if the information is a valid LUN.

Command Parameters

The CDB for the initiator's INQUIRY command is formatted as shown in Table 2–20.

Table 2–20. INQUIRY Command

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (12h)							
1	Logical Unit Number			Reserved				
2	Reserved							
3	Reserved							
4	Allocation Length							
5	Control Byte (00h)							

The Allocation Length field (byte 4) specifies the number of bytes of data that the initiator allocates for the returned INQUIRY data. Note that an Allocation Length of zero indicates that no data is transferred to the initiator; this is not considered an error. Any other value indicates the maximum number of bytes that is transferred. The target controller terminates the data transmission when either the number of bytes specified in the Allocation Length field or all of the available INQUIRY data is transferred, whichever is less.

Data Format

The INQUIRY command returns 36 bytes of data to the initiator. Table 2–21 shows how this data is formatted.

Table 2–21. Data Format

Byte \ Bit	7	6	5	4	3	2	1	0
0	Peripheral Device Type							
1	RMB	Device-Type Modifier						
2	ISO Version		ECMA Version			ANSI-Approved Version		
3	Reserved				Response Data Format			
4	Additional Length (1Fh)							
5	Reserved							
6	Reserved							
7	Reserved							
8	Vendor Identification (MSB)							
9 – 14	Vendor Identification							
15	Vendor Identification (LSB)							
16	Product Identification (MSB)							
17 – 30	Product Identification							
31	Product Identification (LSB)							
32	Product Revision Level (MSB)							
33 – 34	Product Revision Level							
35	Product Revision Level (LSB)							

If the LUN field of the initiator's CDB is zero, the Peripheral Device Type field (byte 0) is also zero to indicate that the controller is a direct-access device (in this case, a disk drive). If the specified LUN is nonexistent (such as, any value other than zero), the target controller will not support the initiator; thus, the LUN field is set to 07Fh. The target controller only supports those initiators that are formatted with a LUN value of zero.

The Removable Media bit (RMB — bit 7 of byte 1) is always set to zero indicating that the target controller does not support a removable media.

The Device Type Qualifier field (bits 0–6 of byte 1) is always set to zero indicating that the target controller supports a direct-access device.

The ANSI-Approved Version field (byte 2) is always set to 01h because the target controller complies with the ANSI's SCSI American National Standard, X3.131–1986.

The Response Data Format field (bits 0–3 of byte 3) is always set to one because the target controller conforms to the SCSI's CCS standards (revision 4B) for direct-access devices.

Bits 4–7 of byte 3 are reserved and always set to zero.

Always set to 31 (1Fh), the Additional Length field (byte 4) defines the number of parameter bytes to follow the initiator's Allocation Length bytes.

The Vendor Identification field (bytes 8–15) is set to the ASCII string *TMS320-SCSI*.

The Product Identification field (bytes 16–31) uses a 16-character string that contains the Texas Instrument ASCII identifier followed by the product's name; characters are left-justified before the string is filled with ASCII spaces (20h).

The Revision Level field (bytes 32–35) uses a 4-character string to signify the target controller's firmware revision level in ASCII.

The Inquiry Data Structure is shown in Table 2–22.

Table 2–22. Inquiry Data (All Initiators) for TMS320-SCSI

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
RMB	Device Type Qualifier (0X00)							Peripheral Device Type (0X00)								Constant
Reserved (0X00)							Response Format (0X00)	ANSI Version (0X01)								Constant
Request Length (0X00)								Additional Length (0X1F)								Constant
Reserved (0X00)								Reserved (0X00)								Constant
Vendor Identification in ASCII “I”								Vendor Identification in ASCII “T”								Constant
Vendor Identification in ASCII “S”								Vendor Identification in ASCII “–”								Constant
Vendor Identification in ASCII “ S”								Vendor Identification in ASCII “ C”								Constant
Vendor Identification in ASCII “ ”								Vendor Identification in ASCII “ I”								Constant
Product Identification in ASCII “M”								Product Identification in ASCII “T”								Constant
Product Identification in ASCII “3”								Product Identification in ASCII “S”								Constant
Product Identification in ASCII “0”								Product Identification in ASCII “2”								Constant
Product Identification in ASCII “S”								Product Identification in ASCII “–”								Constant
Product Identification in ASCII “S”								Product Identification in ASCII “C”								Constant
Product Identification in ASCII “ ”								Product Identification in ASCII “ I”								Constant
Revision Level in ASCII “1”								Revision Level in ASCII “U”								Constant
Revision Level in ASCII “0”								Revision Level in ASCII “0”								Constant

Command

MODE SELECT – 15h

The MODE SELECT command allows the initiator to specify various parameters to the target controller. Any changes in the MODE SELECT parameters take effect immediately after the MODE SELECT command has terminated. The MODE SELECT command is complementary to the MODE SENSE command that allows the initiator to request that the target controller send values for the parameters to the initiator.

The initiator can send the target controller optional blocks of parameters that are separated into categories, or pages. The individual pages specify various options and features that the initiator may change. Pages 1, 2, 3, and 4 of the MODE SELECT data are saved in the scratch RAM and are used when the FORMAT UNIT command is executed.

The initiator sends to the target controller those pages for which it requests parameters to be changed. The initiator can send pages individually or as a group; the pages do not need to be sent in any particular order.

It is recommended that, prior to issuing a MODE SELECT command, the initiator first issue a MODE SENSE command (with the Page Code set for all pages and the Control field set for current values) to determine pages to be implemented, page lengths, and current values. This should be followed by another MODE SENSE command, with the Page Control Field set to changeable values, to determine which values may be altered. The initiator should analyze this information and should not issue a MODE SELECT command that attempts to change values in fields that are not implemented or are not changeable.

When a MODE SELECT command is issued that changes any parameters in pages 3 or 4, the target controller issues a CHECK CONDITION status with a sense key/additional sense code of UNIT ATTENTION/MODE SELECT CHANGED CONDITION (06h/2Ah), respectively, to the first command received from all initiators except the one that issued the MODE SELECT command. When a MODE SELECT command is issued that changes any parameters in pages 3 or 4, a FORMAT UNIT command must be issued prior to the next media access command.

The MODE SELECT command can affect the following types of target controller parameters:

- Saved values — The saved values are all the changeable MODE SELECT parameters saved by the target controller in the reserved part of the data buffer when performing the FORMAT UNIT command. The initiator may change the saved value of pages 1 and/or 2 by issuing a MODE SELECT command with the save parameters (SP) bit in the CDB set to one. This action does not change the saved value of pages 3 and 4.
- Current values — The current values are the MODE SELECT parameters used by the target controller during normal target controller operation. Any

MODE SELECT command issued to the target controller changes the current values.

Command Parameters

The CDB for the MODE SELECT command is formatted as shown in Table 2–23.

Table 2–23. MODE SELECT Command

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (15h)							
1	Logical Unit Number			PF	Reserved			SP
2	Reserved							
3	Reserved							
4	Parameter List Length							
5	Control Byte (00h)							

The Page Format (PF— bit 4 of byte 1) is set to one to indicate that the format of the data sent by the initiator (after the MODE SELECT header and the block descriptors, if any) complies with the Page Format as defined in the Common Command Set (CCS).

When set to one, the Save Parameters bit (SP — bit 0 of byte 1) specifies that the target controller should take the current values for pages 1 and/or 2 and write them to the reserved area of the data buffer as the saved values. Before the target controller saves the parameters, it makes any changes to these pages as specified in the current MODE SELECT command. If the target controller encounters an error during the MODE SELECT command, it terminates the command without writing the parameters to the reserved area of the data buffer as the saved values. If the Save Parameters (SP) bit is zero, the target controller updates the current values and does not modify the saved values.

The Parameter List Length field (byte 4) specifies the length, in bytes, of the parameters that are sent from the initiator to the target controller during the DATA OUT phase of the MODE SELECT command. A Parameter List Length of zero indicates that no data is transferred and is not considered an error by the target controller.

Parameter List Format

The initiator sends the MODE SELECT's parameter list to the target controller during the DATA OUT phase. This list consists of one Parameter List Header, zero or one Block Descriptors, and zero or many Page Descriptors. Note that the MODE SELECT's CDB specifies the entire length of the parameter list.

Parameter List Header Format

The 4-byte Parameter List Header is the first part of the parameter list and specifies the media type and the length of the block descriptor.

The CDB for the Parameter List Header of the MODE SELECT command is formatted as shown in Table 2–24.

Table 2–24. Parameter List Header

Byte \ Bit	7	6	5	4	3	2	1	0
0	Reserved							
1	Medium Type (00h)							
2	Reserved							
3	Block Descriptor Length (00h or 08h)							

The Medium Type field (byte 1) must be set to zero.

The Block Descriptor Length field (byte 3) specifies the length of the Block Descriptor (in bytes) that starts at byte four of the Parameter List's CDB. Because the target controller only supports zero or one Block Descriptors per MODE SELECT command, the only valid Block Descriptor Lengths are 0 and 8 bytes.

Block Descriptor Format

The 8-byte Block Descriptor is the second part of the Parameter List and immediately follows the Parameter List Header. It specifies the density of the media, the number of logical blocks, and one-block length.

The Block Descriptor Length is set to zero if the Block Descriptor is not located within the Parameter List; note that the controller does not consider this to be an error condition. The CDB for the Block Descriptor of the MODE SELECT command is formatted as shown in Table 2–25.

Table 2–25. Block Descriptor

Byte \ Bit	7	6	5	4	3	2	1	0
0	Density Code							
1	Number of Logical Blocks (MSB)							
2	Number of Logical Blocks							
3	Number of Logical Blocks (LSB)							
4	Reserved							
5	Logical Block Length (MSB)							
6	Logical Block Length							
7	Logical Block Length (LSB)							

The Density Code field (byte zero) must be set to zero.

The Number of Logical Blocks field (bytes 1–3) is ignored by the target controller.

The Logical Block Length field (bytes 5–7) specifies the length of one logical block (in bytes).

Page Descriptor Format With Page Header

The Page Descriptor is the last part of the Parameter List and specifies various parameters separated into pages. These parameters specify options and features that can be changed by the initiator. More than one page can be sent during the MODE SELECT command.

Each of the optional Page Descriptors is preceded by a Page Header, which is immediately followed by its corresponding page parameters. Each Page Header is 2-bytes long and identifies the page type length.

The Page Code field (bits 0–5 of byte 0) identifies the page type. Page Codes and their corresponding page descriptions are listed in Table 2–26.

Table 2–26. List of Page Codes

Page Code	Page Description
01h	Error Recovery Parameters
02h	Disconnect/reselect control parameters
03h	Direct access device format parameters
04h	Rigid disk drive geometry parameters
3Fh	Include pages one through four

The Page Length field (byte 1) specifies the number of bytes in the page, not including the Page Length byte. The initiator must send the entire page to the target controller.

Error Recovery Parameters Page

The target controller supports the Error Recovery Parameters options of the MODE SELECT command. When an initiator issues the FORMAT UNIT or MODE SELECT command (SP bit set to one), the controller saves the Error Recovery Parameters Page as well as a copy for that initiator. This allows any initiator to change its own parameters without affecting the other initiators' parameters. Having a Page Code of 01h, the Error Recovery Parameters Page is formatted as shown in Table 2–27.

Table 2–27. Error Recovery Parameter Option Format

Byte \ Bit	7	6	5	4	3	2	1	0
0	PS	Reserved	Page Code (01h)					
1	Page Length (0Ah)							
2	AWRE	ARRE	TB	RC	EEC	PER	DTE	DCR
3	Read Retry Count							
4	Correction Span							
5	Head Offset Count							
6	Data Strobe Offset Count							
7	Reserved							
8	Write Retry Count							
9	Reserved							
10	Recovery Time Limit (MSB)							
11	Recovery Time Limit (LSB)							

Note:

The target controller saves this page whenever the initiator issues a FORMAT UNIT or MODE SELECT command with the save parameters (SP) bit set to one.

The Page Code field (bits 0–5 of byte 0) identifies the page.

The Parameter Saveable (PS — bit 7 of byte 0) specifies that supported parameters can be saved.

The Page Length field (byte 1) specifies the number of bytes, excluding the Page Length byte, for that page. The initiator sends the entire page to the target controller.

The Disable Correction bit (DCR — bit 0 of byte 2) is currently ignored by the target controller. Originally, DCR was set to one so that the ECC correction could be disabled if the initiator attempted to read a sector from the controller.

The Disable Transfer on Error bit (DTE — bit 1 of byte 2) is currently ignored by the target controller.

The Post Error bit (PER — bit 2 of byte 2) is set to one so that the target controller will report all recoverable errors to the initiator.

The Enable Early Correction bit (EEC — bit 3 of byte 2) is always set to zero since the target controller does not currently support this option. Originally, EEC was set to one so that the target controller would not exhaust the Retry Count field before it attempted the ECC correction.

When set to one, the Read Continuous bit (RC — bit 4 of byte 2) overrides the EEC, DTE, PER, and DCR bits and disables all retry and data correction operations.

The Transfer Block bit (TB — bit 5 of byte 2) is only applicable when either the controller encounters a hardware error or, if DTE is set to one and a recoverable error is encountered. If TB is set to zero, the target controller does not transfer the block that contains the data error. If TB is set to one, the target controller transfers the block with the data error before the MODE SELECT command is terminated. In either case, the target controller reports the address of that block, instead of the preceding block, within the controller's Sense Data sector. Should an error other than DATA ERROR (such as, DATA NOT FOUND) terminate the transfer, the target controller does not transfer the block.

The Automatic Read Reallocation Enabled bit (ARRE — bit 6 of byte 2) is always set to zero because the target controller does not currently support this option.

The Automatic Write Reallocation Enabled bit (AWRE — bit 7 of byte 2) is always set to zero because the target controller does not currently support this option.

The Retry Count field (byte 3) specifies the maximum number of retry operations that the controller should attempt whenever it encounters an error. The target controller currently ignores this field.

The Correction Span field (byte 4) specifies the largest READ DATA error (in bits) that the controller should attempt to correct. The target controller currently ignores this field.

The Head Offset Count field (byte 5) specifies the forced-incremental offset from the center of the track for the controller to use whenever performing the Disk Read operation. The target controller currently ignores this field.

The Data Strobe Offset Count field (byte 6) is always set to zero because the target controller does not currently support this option.

Write Retry Count field (byte 8) specifies the maximum number of retry operations that the controller can attempt after it encounters a WRITE error. The target controller does not currently support this option.

The Recovery Time Limit field (byte 7) is always set to zero because the target controller does not currently support this option.

DIsconnect/Reconnect Parameters Page

The target controller supports the Disconnect/Reconnect Parameters options of the MODE SELECT command. When an initiator issues the FORMAT UNIT or MODE SELECT command (SP bit set to one), the controller saves the Disconnect/Reconnect Control Parameters page as well as a copy for each initiator. This allows any initiator to change its own parameters without affecting the other initiators' parameters. Having a Page Code of 02h, the Disconnect/Reconnect Control Parameters page is formatted as shown in Table 2–28.

Table 2–28. Disconnect/Reconnect Parameters Option Format

Byte \ Bit	7	6	5	4	3	2	1	0
0	PS	Reserved	Page Code (02h)					
1	Page Length (0Eh)							
2	Buffer Full Ratio							
3	Buffer Empty Ratio							
4	Bus Inactivity Limit (MSB)							
5	Bus Inactivity Limit (LSB)							
6	Disconnect Time Limit (MSB)							
7	Disconnect Time Limit (LSB)							
8	Connect Time Limit (MSB)							
9	Connect Time Limit (LSB)							
10	Maximum Burst Size (MSB)							
11	Maximum Burst Size (LSB)							
12	Reserved							DTDC
13	Reserved							
14	Reserved							
15	Reserved							

Note:

The target controller saves this page whenever the initiator issues a FORMAT UNIT or MODE SELECT command with the save parameters (SP) bit set to one.

The Page Code field (bits 0–5 of byte 0) identifies the page.

The Parameter Saveable (PS — bit 7 of byte 0) specifies that supported parameters can be saved.

The Page Length field (byte 1) specifies the number of bytes, excluding the Page Length byte, for that page. The initiator sends the entire page to the target controller.

The Buffer Full Ratio field (byte 2) specifies the number of maximum bits that the internal buffer must have before the target controller reconnects with and transmits data to the initiator. The target controller does not currently support this option.

The Buffer Empty Ratio field (byte 3) specifies the minimum number of bits that the internal buffer must have before the target controller reconnects with and receives data from the initiator. The target controller does not currently support this option.

The Bus Inactivity Limit field (bytes 4–5) specifies the length of time (in 100-microsecond increments, ranging between 1 for 100 microseconds and 650 for 65,000 microseconds) that the target controller can remain connected to the SCSI bus without any bus activity. If this field is set to zero, the target controller remains connected to the bus indefinitely.

The Disconnect Time Limit field (bytes 6–7) indicates the minimum time in 100-microsecond increments that the target waits after releasing the SCSI bus before attempting reselection.

The Connect Time Limit field (bytes 8–9) indicates the maximum time in 100-microsecond increments that the target is allowed to use the SCSI bus before disconnecting if the initiator has granted the disconnect privilege.

The Maximum Burst Size (bytes 10–11) indicates the maximum amount of data that the target transfers during a data phase before disconnecting if the initiator has granted the disconnect privilege. This value is expressed in increments of 512 bytes (for example, a value of one means 512 bytes, two means 1024 bytes, etc.). A value of zero indicates there is no limit on the amount of data transferred per connection.

The Data Transfer Disconnect Control (DTDC — bits 0–1 of byte 12) field defines further restrictions on when a disconnect is permitted.

Direct-Access Device Format Parameters Page

The target controller supports the Direct-Access Device Format Parameters options of the MODE SELECT command. When an initiator issues the FORMAT UNIT command, the controller saves the Direct-Access Device Format Parameters page. Having a Page Code of 03h, the Direct-Access Device Format Parameters page is formatted as shown in Table 2–29.

Table 2–29. Direct-Access Device Format Parameters Option Format

Byte \ Bit	7	6	5	4	3	2	1	0
0	PS	Reserved	Page Code (03h)					
1	Page Length (16h)							
2	Tracks Per Zone (MSB)							
3	Tracks Per Zone (LSB)							
4	Alternate Sectors Per Zone (MSB)							
5	Alternate Sectors Per Zone (LSB)							
6	Alternate Tracks Per Zone (MSB)							
7	Alternate Tracks Per Zone (LSB)							
8	Alternate Tracks Per Logical Unit (MSB)							
9	Alternate Tracks Per Logical Unit (LSB)							
10	Sectors Per Track (MSB)							
11	Sectors Per Track (LSB)							
12	Data Bytes Per Physical Sector (MSB)							
13	Data Bytes Per Physical Sector (LSB)							
14	Interleave (MSB)							
15	Interleave (LSB)							
16	Track Skew Factor (MSB)							
17	Track Skew Factor (LSB)							
18	Cylinder Skew Factor (MSB)							
19	Cylinder Skew Factor (LSB)							
20	SSEC	HSEC	RMB	SURF	INS	Reserved		
21 – 23	Reserved							

Note:

The target controller saves this page whenever the initiator issues a FORMAT UNIT command.

The Page Code field (bits 0–5 of byte 0) identifies the page.

The Parameter Saveable (PS — bit 7 of byte 0) specifies that supported parameters can be saved.

The Page Length field (byte 1) specifies the number of bytes, excluding the Page Length byte, for that page. The initiator sends the entire page to the target controller.

The Tracks Per Zone field (bytes 2–3) is always set to zero since the target controller does not currently support this option.

The Alternate Sectors Per Zone field (bytes 4–5) specifies the number of alternate sectors allocated to each zone during a format operation. The target controller does not currently support this option.

The Alternate Tracks Per Zone field (bytes 6–7) is always set to zero. The target controller does not currently support this option.

The Alternate Tracks Per Volume field (bytes 8–9) specifies the number of alternate tracks to deallocate for the entire target controller during a format operation. The target controller currently ignores this field.

The Sectors Per Track field (bytes 10–11) specifies the number of physical sectors that were allocated to each track. The target controller does not currently support this option.

The Data Bytes Per Physical Sector field (bytes 12–13) specifies the number of bytes that are allocated to each physical sector. The target controller supports between 256 and 4,096 bytes per sector.

The Interleave Value field (bytes 14–15) is currently ignored by the target controller and may be set to any value.

The Track Skew field (bytes 16–17) specifies the number of physical sectors between the last logical block of one track and the first logical block of the next sequential track of the same cylinder. The target controller currently ignores this field.

The Cylinder Skew field (bytes 18–19) specifies the number of physical sectors between the last logical block of one cylinder and the first logical block of the next sequential cylinder. The target controller currently ignores this field.

The Soft Sector format bit (SSEC — bit 7 of byte 20) cannot be set by the MODE SELECT command. The target controller currently ignores this field.

The Hard Sector format bit (HSEC — bit 6 of byte 20) cannot be set by the MODE SELECT command. The target controller currently ignores this field.

The Removable Media bit (RMB — bit 5 of byte 20) is not used by the target controller.

The Surface bit (SURF — bit 4 of byte 20) is not supported by the target controller.

The Inhibit Save bit (INS — bit 3 of byte 20) is not used by the target controller. The target controller always saves the parameters of pages 1–4 upon successful completion of the FORMAT UNIT command.

Rigid Disk Drive Geometry Parameters Page

The target controller supports the Rigid Disk Drive Geometry Parameters options of the MODE SELECT command. When an initiator issues the FORMAT UNIT command, the controller saves the Rigid Disk Drive Geometry Parameters Page. Having a Page Code of 04h, the Rigid Disk Drive Geometry Parameters page is formatted as shown in Table 2–30.

Table 2–30. Rigid Disk Drive Geometry Parameters Option Format

Byte \ Bit	7	6	5	4	3	2	1	0
0	PS	Reserved	Page Code (04h)					
1	Page Length (16h)							
2	Number of Cylinders (MSB)							
3	Number of Cylinders							
4	Number of Cylinders (LSB)							
5	Number of Heads							
6	Starting Cylinder — Write Precompensation (MSB)							
7	Starting Cylinder — Write Precompensation							
8	Starting Cylinder — Write Precompensation (LSB)							
9	Starting Cylinder — Reduced Write Current (MSB)							
10	Starting Cylinder — Reduced Write Current							
11	Starting Cylinder — Reduced Write Current (LSB)							
12	Drive Step Rate (MSB)							
13	Drive Step Rate (LSB)							
14	Landing Zone Cylinder (MSB)							
15	Landing Zone Cylinder							
16	Landing Zone Cylinder (LSB)							
17	Reserved							
18	Rotational Offset							
19	Reserved							
20	Medium Rotation Rate (MSB)							
21	Medium Rotation Rate (LSB)							
22	Reserved							
23	Reserved							

Note:

The target controller saves this page whenever the initiator issues a FORMAT UNIT command.

The Page Code field (bits 0–5 of byte 0) identifies the page.

The Parameter Saveable (PS — bit 7 of byte 0) specifies that supported parameters can be saved.

The Page Length field (byte 1) specifies the number of bytes, excluding the Page Length byte, for that page. The initiator sends the entire page to the target controller.

The Number of Cylinders field (bytes 2–4) specifies the maximum number of cylinders that are available on the target controller. The number of cylinders that are addressable by the user is equal to the Number of Cylinders field minus the sum of the Alternate Tracks per Volume field (converted into cylinders) and the Three Cylinders field (reserved by the target controller). The most-significant byte (byte 2) of this field must be set to zero.

The Number of Heads field (byte 5) specifies the maximum number of data heads on the target controller. The target controller supports one to fifteen heads.

The Starting Cylinder – Write Precompensation field (bytes 6–8) is not supported by the target controller.

The Starting Cylinder – Reduced Write Current field (bytes 9–11) is not supported by the target controller.

The Drive Step Rate field (bytes 12–13) is not supported by the target controller.

The Landing Zone Cylinder field (bytes 14–16) is not supported by the target controller.

Rotational Offset (byte 18) is not supported by the target controller.

Medium Rotation Rate (bytes 20 and 21) is not supported by the target controller.

Error Conditions

If a field that the target controller neither uses nor supports is not set to zero, the controller terminates the MODE SELECT command with the CHECK CONDITION status and sets the sense key or additional sense code to ILLEGAL REQUEST (05h) or INVALID FIELD IN PARAMETER LIST (26h), respectively.

If the Medium Type field is not set to zero, the target controller terminates the MODE SELECT command with the CHECK CONDITION status and sets the sense key or additional sense code to ILLEGAL REQUEST (05h) or INVALID FIELD IN PARAMETER LIST (26h), respectively.

If the Block Descriptor Length field is set to a value other than 0 or 8, the target controller terminates the MODE SELECT command with the CHECK CONDITION status and sets the sense key or additional sense code to ILLEGAL REQUEST (05h) or INVALID FIELD IN PARAMETER LIST (26h), respectively,

If the Density Code field is not set to zero, the target controller terminates the MODE SELECT command with the CHECK CONDITION status and sets the sense key or additional sense code to ILLEGAL REQUEST (05h) or INVALID FIELD IN PARAMETER LIST (26h), respectively.

If the Data Bytes Per Physical Sector field is set to a value that is either less than 128 or greater than 4,096 bits, the target controller terminates the MODE SELECT command with the CHECK CONDITION status and sets the sense key or additional sense code to ILLEGAL REQUEST (05h) or INVALID FIELD IN PARAMETER LIST (26h), respectively.

If the value within the Number of Heads field of the Rigid Target Controller Geometry Parameters page exceeds the number of heads that are actually available to the target controller, it terminates the MODE SELECT command with the CHECK CONDITION status and sets the sense key or additional sense code to ILLEGAL REQUEST (05h) or INVALID FIELD IN PARAMETER LIST (26h), respectively.

If the value within the Page Length field of each page header does not equal the value within the Page Length field specified by this document and returned by the MODE SENSE command, the target controller terminates the MODE SELECT command with the CHECK CONDITION status and sets the sense key or additional sense code to ILLEGAL REQUEST (05h) or INVALID FIELD IN PARAMETER LIST (26h), respectively.

Command **RESERVE UNIT – 16h**

The RESERVE UNIT command reserves the specified logical unit to be used exclusively by the initiator or a designated third-party device. This reservation remains in effect until one of these conditions occurs:

- The target controller receives a RELEASE UNIT command from the same initiator.
- The target controller receives the BUS DEVICE RESET message from any initiator.
- A SCSI bus reset occurs.

If a logical unit that is reserved by another initiator receives a command (including RESERVE UNIT), the target controller returns the RESERVATION CONFLICT status.

The initiator that invoked RESERVE UNIT may also issue another RESERVE UNIT command in order to modify its parameters. If the target controller cannot grant the new reservation, the previous reservation is not modified and the controller returns the RESERVATION CONFLICT status. If the target controller grants the new reservation, the second command releases the first one.

Command Parameters

The CDB for the RESERVE UNIT command is formatted as shown in Table 2–31.

Table 2–31. RESERVE UNIT Command Format

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (16h)							
1	Logical Unit Number			3rdPty	Third-Party Device ID			Extent
2	Reservation ID							
3	Extent List Length (MSB)							
4	Extent List Length (LSB)							
5	Control Byte (00h)							

The Third-Party Device ID field (bits 1–3 of byte 1) is valid only when the 3rdPty bit is set to one. This field identifies the SCSI bus device that is to be reserved.

The Third-Party Reservation bit (3rd Pty — bit 4 of byte 1) is an option that is intended for multiple-initiator systems. Any device that uses the Third Party

Reservation option to reserve must also use the Third Party Reservation option to release before any other commands can be sent. If the bit is set to one, the initiator reserves the specified logical unit for the SCSI bus device that is identified in the Third-Party Device ID field. The ID is valid only when the Third Party Reservation bit is set to one.

The Reservation ID field (byte 2) is not supported and must be set to zero.

The Extent List Length field (bytes 3–4) is not supported and must be set to zero.

Error Conditions

If the Extent bit, Reservation ID field, or Extent List Length field is not set to zero, the target controller terminates the RESERVE UNIT command with the CHECK CONDITION status and sets the sense key or additional sense code to ILLEGAL REQUEST (05h) or INVALID FIELD IN CDB (24h), respectively.

Command **RELEASE UNIT – 17h**

The RELEASE UNIT command causes the initiator to release the logical unit that was previously reserved by the RESERVE UNIT command. Note that it is not considered an error to release a logical unit that is not currently reserved. Once the RELEASE UNIT command is issued, other initiators can then access the logical unit.

The initiator that invoked RELEASE UNIT may also issue another RELEASE UNIT command in order to modify its parameters. If the new reservation cannot be granted, the previous reservation is not modified and the controller returns the RESERVATION CONFLICT status. If the target controller grants the new reservation, the second command releases the first one.

Command Parameters

The CDB for the RELEASE UNIT command is formatted as shown in Table 2–32.

Table 2–32. RELEASE UNIT Command

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (17h)							
1	Logical Unit Number			3rdPty	Third-Party Device ID			0
2	Reserved							
3	Reserved							
4	Reserved							
5	Control Byte (00h)							

The Third-Party Device ID field (bits 1–3 of byte 1) identifies the SCSI bus device that is to be reserved for the disk drive.

The Third-Party Reservation bit (3rd Pty — bit 4 of byte 1) is set to one so that the reservation on the logical unit will be released if one of these conditions occurs:

- The unit was originally reserved using the Third Party option in the RESERVE UNIT command.
- The same initiator that issued the RESERVE UNIT command is requesting the release of the logical unit.
- The initiator specifies that it has the same SCSI bus device in the Third-Party Device ID field as was specified in that field by the initiator in the RESERVE UNIT command.

Error Conditions

If the Extent Reservation option is specified, the target controller terminates the command with a CHECK CONDITION status and sets the sense key or additional sense code to ILLEGAL REQUEST (05h) or INVALID FIELD IN CDB (24h), respectively.

If the Third-Party Release option is specified and the unit was not originally reserved with the Third-Party Reservation option, the target controller terminates the command with the CHECK CONDITION status and sets the sense key or additional sense code to ILLEGAL REQUEST (05h) or INVALID FIELD IN CDB (24h), respectively.

If the Third-Party Release option is specified and the Third-Party Device ID is not the same as the one specified in the original RESERVE UNIT command, the disk drive terminates the command with the CHECK CONDITION status and sets the sense key or additional sense code to ILLEGAL REQUEST (05h) or INVALID FIELD IN CDB (24h), respectively.

If the Third-Party Release option is specified and the Third-Party Reservation option identifies different SCSI initiators, the target controller terminates the command with the RESERVATION CONFLICT status.

Command **MODE SENSE – 1Ah**

The MODE SENSE command allows the initiator to receive various parameters from the target controller. Note that MODE SENSE is a complementary command to the MODE SELECT command.

The target controller sends blocks of parameters; these parameters are separated into categories, called pages. These pages specify various options and features that the initiator can change. Each page is preceded by its Page Code and its Page Length. The Page Code field (bits 0–5 of byte 0) of the MODE SELECT command specifies the hexadecimal value of the page(s) that the target controller is to return. Currently, the Page Codes are 01h for Page 1, 02h for Page 2, 03h for Page 3, 04h for Page 4, and 3Fh for all of the four pages. The value of the Page Length does not include bytes 0 or 1 of the page. Refer to the MODE SELECT command for detailed information about these pages.

The target controller utilizes four types of MODE SENSE data:

- Changeable value — any controller-supported parameter that the MODE SELECT command can change.
- Current value — any parameter of the MODE SENSE command that the target controller can use during its normal drive operation. Any MODE SELECT command that is issued to the target controller changes the current values.
- Default value — any parameter that the target controller can store into the reserved section of its data buffer.
- Saved value — any parameter of the MODE SELECT command that the target controller can save onto the media whenever it performs the FORMAT UNIT or MODE SELECT command with the SP bit in the CDB set to one.

At initialization time (that is, POWER-UP or RESET condition has occurred), the target controller reads the default values from its data buffer and uses them as current values.

When the target controller completes the FORMAT UNIT command, it writes all of the supported pages into its RAM. Note that the current and the default parameters are the same for an unformatted target controller. The initiator can then change the current values by executing the MODE SELECT command prior to issuing the FORMAT command. Upon completion of the FORMAT UNIT command, the target controller writes the current values (which may have been changed by a MODE SELECT command) as the saved values into its RAM.

Command Parameters

The CDB for the MODE SENSE command is formatted as shown in Table 2–33.

Table 2–33. MODE SENSE CDB Command

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (1Ah)							
1	Logical Unit Number			PF	Reserved			SP
2	Page Control		Reserved					
3	Reserved							
4	Allocation Length							
5	Control Byte (00h)							

The Page Format bit (PF — bit 4 of byte 1) indicates that the MODE SENSE data is to be transferred in the format specified by the Page Control field of the CDB.

The Page Control field (bits 6–7 of byte 2) defines the type of parameter values that the target controller is to send. These parameter values are listed in Table 2–34.

Table 2–34. Parameter Values Sent by Target Controller Using Page Control Field

Bit 7	Bit 6	Type of Parameter Value
0	0	Current values
0	1	Changeable values
1	0	Default values
1	1	Saved values

The Page Code field specifies which page or pages will be returned. Page codes 01h, 02h, 03h, and 04h are supported; Page Code 3Fh returns all four pages.

The Allocation Length field (byte 4) specifies the number of bytes that the initiator has allocated for MODE SENSE data from the controller.

Parameter List Header Format

The MODE SENSE data contains a four-byte Parameter List Header (followed by zero or more pages) that is formatted as shown in Table 2–35.

Table 2–35. Parameter List Header Command

Byte \ Bit	7	6	5	4	3	2	1	0
0	Sense Data Length							
1	Medium Type							
2	WP	Reserved						
3	Block Descriptor Length							

The Sense Data Length field (byte 0) specifies the maximum length of data that the initiator can receive. This length does not include the Data Length field. The value within the Sense Data Length field depends on the page(s) being requested.

The Medium Type field (byte 1) is always set to zero and indicates the type of media.

Refer to the MODE SELECT command for detailed information about each page.

Refer to Figure 2–8 through Figure 2–13 for the firmware data structures of the MODE SELECT command pages.

Figure 2–8. Mode-Sense Buffer for XMIT (64) and Mode-Select Buffer for RCVE (64)

Figure 2–9. SCSI Mode-Sense Data (All Initiators)

Figure 2–10. Page Codes (1) and Page Formats (0) for Current, Changeable, and Default

Figure 2–11. Page Codes (3) and Page Formats (1) for Current

Figure 2–12. Page Codes (3) and Page Formats (1) for Changeable

Figure 2–13. Page Codes (4) and Page Formats (1) for Default and Changeable

Command **START/STOP UNIT – 1Bh**

The START/STOP UNIT command permits the target controller to either spin-up or spin-down the disk drive.

Upon power-up or reset, the target controller places the disk drive into the SPIN-UP mode of operation. The disk drive remains in that state until the controller receives the START/STOP UNIT command with the Start bit set to zero.

Command Parameters

The CDB for the START/STOP UNIT command is formatted as shown in Table 2–36.

Table 2–36. START/STOP UNIT Command

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (1Bh)							
1	Logical Unit Number			Reserved				Immed
2	Reserved							
3	Reserved							
4	Reserved							Start
5	Control Byte (00h)							

When set to one, the Immediate bit (Immed — bit 0 of byte 0) indicates that the status is to be returned as soon as the operation is initiated. If the bit is set to zero, the target controller returns its status when the operations are completed.

The Start bit (bit 0 of byte 5) is set to zero if the target controller is to spin-down the disk drive. If the bit is set to one, the target controller is to prepare for operation.

Error Conditions If the target controller receives a command that accessed the media after the STOP portion of the START/STOP UNIT command (Start bit set to zero), the target controller terminates the command with the CHECK CONDITION status and sets the sense key or additional sense code to NOT READY (02h) or DRIVE NOT READY (04h), respectively.

Command **READ CAPACITY – 25h**

The READ CAPACITY command has these three functions:

- ❑ Determines the maximum number of logical blocks that can be accessed by the initiator.
- ❑ Returns the size of the logical block. This information is sent to the initiator during the DATA IN phase.
- ❑ Determines whether a file of a given size will fit within a physically contiguous space by requesting the number of blocks that appear after a specific block before it encounters a substantial delay (such as, a cylinder boundary).

Command Parameters

The CDB for the READ CAPACITY command is formatted as shown in Table 2–37.

Table 2–37. READ CAPACITY Command

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (25h)							
1	Logical Unit Number			Reserved				
2	Logical Block Address (MSB)							
3	Logical Block Address							
4	Logical Block Address							
5	Logical Block Address (LSB)							
6	Reserved (zeros)							
7	Reserved							
8	Reserved							PMI
9	Control Byte (00h)							

The Logical Block Address field (LBA — bytes 2–5) is used by the controller only if the Partial Medium Indicator (PMI) bit is set to one. This field specifies the block address to use to compute the last block before a substantial delay is encountered.

The Partial Medium Indicator bit (PMI — bit 0 of byte 8) set to zero indicates that the information returned is for the last block of the logical unit. If the bit is set to one, the information returned is for the last full logical block (address is specified within the LBA field) that the controller can transfer before it encounters a substantial delay (such as, a cylinder boundary).

Command Parameters

The target controller transmits the 8-byte READ CAPACITY data to the initiator during the DATA IN phase. The CDB for the READ CAPACITY data is formatted as shown in Table 2–38.

Table 2–38. READ CAPACITY Command During DATA IN Phase

Byte \ Bit	7	6	5	4	3	2	1	0
0	Logical Block Address (MSB)							
1	Logical Block Address							
2	Logical Block Address							
3	Logical Block Address (LSB)							
4	Block Length (MSB)							
5	Block Length							
6	Block Length							
7	Block Length (LSB)							

The Logical Block Address field (LBA — bytes 0–3) specifies the address of either the last logical block of the logical unit (if PMI bit is set to zero) or the last full logical block (if PMI bit is set to one) before the target controller encounters a substantial delay.

The Block Length field (bytes 4–7) specifies the size (in bytes) of the logical block.

Error Conditions

If LBA is not zero and PMI is set to zero, the target controller terminates the READ CAPACITY command with the CHECK CONDITION status and sets the sense key or additional sense code to ILLEGAL REQUEST (05h) or INVALID FIELD IN CDB (24h), respectively.

If LBA is invalid and PMI is set to one, the target controller terminates the READ CAPACITY command with the CHECK CONDITION status and sets the sense key or additional sense code to ILLEGAL REQUEST (05h) or ILLEGAL LOGICAL BLOCK ADDRESS (21h), respectively.

Command **READ (EXTENDED) – 28h**

The READ (EXTENDED) command requests that the target controller transfer data from the logical unit to the initiator.

Command Parameters

The CDB for the READ (EXTENDED) command is formatted as shown in Table 2–39.

Table 2–39. EXTENDED READ Command

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (28h)							
1	Logical Unit Number			Reserved				RelAddr
2	Logical Block Address (MSB)							
3	Logical Block Address							
4	Logical Block Address							
5	Logical Block Address (LSB)							
6	Reserved (zeros)							
7	Transfer Length (MSB)							
8	Transfer Length (LSB)							
9	Control Byte (00h)							

The Logical Block Address field (LBA — bytes 2–5) specifies the address of the first logical block that the target controller is to use during the READ operation.

The Transfer Length field (bytes 7–8) specifies the number of logical blocks of data that the controller is to transfer. A Transfer Length of zero indicates that no logical blocks are transferred.

Error Conditions If the LBA field is invalid, the target controller terminates the command with the CHECK CONDITION status and sets the sense key or additional sense code to ILLEGAL REQUEST (05h) or ILLEGAL LOGICAL BLOCK ADDRESS (21h), respectively.

If the value within the LBA field or the Transfer Length field is an invalid address, the target controller terminates the command with the CHECK CONDITION status and sets the sense key or additional sense code to ILLEGAL REQUEST (05h) or ILLEGAL LOGICAL BLOCK ADDRESS (21h), respectively.

Command **WRITE (EXTENDED) – 2Ah**

The WRITE (EXTENDED) command requests that the target controller transfer data from the initiator to the logical unit.

Command Parameters

The CDB for the WRITE (EXTENDED) command is formatted as shown in Table 2–40.

Table 2–40. EXTENDED WRITE Command

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (2Ah)							
1	Logical Unit Number			Reserved				RelAddr
2	Logical Block Address (MSB)							
3	Logical Block Address							
4	Logical Block Address							
5	Logical Block Address (LSB)							
6	Reserved (zeros)							
7	Transfer Length (MSB)							
8	Transfer Length (LSB)							
9	Control Byte (00h)							

The Logical Block Address field (LBA — bytes 2–5) specifies the address of the first logical block that the target controller is to use during the WRITE operation.

The Transfer Length field (bytes 7–8) is set to zero if the target controller is not to transfer data from the logical block. Otherwise, the field specifies the number of logical blocks of data that the controller is to transfer.

Error Conditions If the LBA field is invalid, the target controller terminates the command with the CHECK CONDITION status and sets the sense key or additional sense code to ILLEGAL REQUEST (05h) or ILLEGAL LOGICAL BLOCK ADDRESS (21h), respectively.

If the value within the LBA field or the Transfer Length field is an invalid address, the target controller terminates the command with the CHECK CONDITION status and sets the sense key or additional sense code to ILLEGAL REQUEST (05h) or ILLEGAL LOGICAL BLOCK ADDRESS (21h), respectively.

Command **SEEK (EXTENDED) – 2Bh**

The SEEK (EXTENDED) command requests that the target controller only access the data block. This is a no-operation command because the controller performs no mechanical movement.

Command Parameters

The CDB for the SEEK (EXTENDED) command is formatted as shown in Table 2–41.

Table 2–41. EXTENDED SEEK Command

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (2Bh)							
1	Logical Unit Number			Reserved				
2	Logical Block Address (MSB)							
3	Logical Block Address							
4	Logical Block Address							
5	Logical Block Address (LSB)							
6	Reserved (zeros)							
7	Reserved							
8	Reserved							
9	Control Byte (00h)							

The Logical Block Address field (LBA — bytes 2–5) specifies the address of the first logical block that the target controller seeks, yet performs no operation.

Error Conditions If the LBA field is invalid, the target controller terminates the command with the CHECK CONDITION status and sets the sense key or additional sense code to ILLEGAL REQUEST (05h) or ILLEGAL LOGICAL BLOCK ADDRESS (21h), respectively.

Command **WRITE AND VERIFY – 2Eh**

The WRITE AND VERIFY command requests that the target controller write data from the initiator as well as verify the accuracy of that data. The target controller supports byte-to-byte comparison; therefore, media verification with ECC standards is not supported.

Command Parameters

The CDB for the WRITE AND VERIFY command is formatted as shown in Table 2–42.

Table 2–42. WRITE AND VERIFY Command

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (2Eh)							
1	Logical Unit Number			Reserved			BytChk	RelAddr
2	Logical Block Address (MSB)							
3	Logical Block Address							
4	Logical Block Address							
5	Logical Block Address (LSB)							
6	Reserved (zeros)							
7	Transfer Length (MSB)							
8	Transfer Length (LSB)							
9	Control Byte (00h)							

The Byte Check bit (BytChk — bit 1 of byte 1) is set to zero if the DATA OUT phase does not occur between the initiator and the target controller. The bit is set to one if the specified blocks are to be read from the disk drive and then compared with data being transferred from the initiator to the controller during the DATA OUT phase — similar to the WRITE operation.

The Logical Block Address field (LBA — bytes 2–5) specifies the address of the first logical block where the write operation begins.

The Transfer Length field (bytes 7–8) of zero indicates that no logical blocks are transferred. Otherwise, the field specifies the number of logical blocks of data that the controller is to transfer.

Error Conditions If the LBA field is invalid, and/or if the LBA plus the Transfer Length results in an invalid block address, the target controller terminates the command with the CHECK CONDITION status and sets the sense key or additional sense code to ILLEGAL REQUEST (05h) or ILLEGAL LOGICAL BLOCK ADDRESS (21h), respectively. No data is transferred if this condition occurs.

Command **WRITE BUFFER – 3Bh**

The WRITE BUFFER command is used in conjunction with the READ BUFFER command as a diagnostic function for testing the target controller's data buffer memory and the SCSI bus integrity.

To determine the maximum amount of data that can be transferred with the READ BUFFER and WRITE BUFFER commands, the initiator can issue a READ BUFFER command with the Allocation Length field set to four. Bytes 2 and 3 returned by the target controller contain the maximum buffer size for the specified logical unit.

Command Parameters

The WRITE BUFFER CDB is formatted as shown in Table 2–43.

Table 2–43. WRITE BUFFER Command

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (3Bh)							
1	Logical Unit Number			Reserved				
2	Reserved							
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved							
7	Byte Transfer Length (MSB)							
8	Byte Transfer Length (LSB)							
9	Control Byte (00h)							

The Byte Transfer Length field (bytes 7 and 8) specifies the number of bytes the target controller transfers during the DATA OUT phase; TLD (bytes 7 and 8) specifies the number of bytes the target controller transfers during the DATA OUT phase to its internal buffer. The transfer length includes the 4 bytes of header information sent before the actual data. A transfer length of zero is not considered an error by the target controller, and no data is expected or read from the initiator.

Data Format The data sent by the initiator during the DATA OUT phase consists of a 4-byte header, immediately followed by the data bytes to be written to the data buffer. The WRITE BUFFER Data is formatted as shown in Table 2–44.

Table 2–44. WRITE BUFFER Data Format

Byte \ Bit	7	6	5	4	3	2	1	0
0	Reserved							
1	Reserved							
2	Reserved							
3	Reserved							
4	Data Byte 0							
5	Data Byte 1							
6	.							
	.							
	.							
n + 4	Data Byte *n*							

The data byte field (bytes 4 through n+4) contains the data to be written to the data buffer.

Error Conditions

If the byte transfer length exceeds the size of the buffer, the WRITE BUFFER command terminates with the CHECK CONDITION status and sets the sense key or additional sense code to ILLEGAL REQUEST (05h) or ILLEGAL LOGICAL BLOCK ADDRESS (21h), respectively.

Command **READ BUFFER – 3Ch**

The READ BUFFER command is used in conjunction with the WRITE BUFFER command as a diagnostic function for testing the data buffer memory and the SCSI bus integrity.

Command Parameters

The CDB for the READ BUFFER command is formatted as shown in Table 2–45.

Table 2–45. READ BUFFER Command

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code (3Ch)							
1	Logical Unit Number			Reserved				
2	Reserved							
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved							
7	Allocation Length (MSB)							
8	Allocation Length (LSB)							
9	Control Byte (00h)							

The Allocation Length field (bytes 7 and 8) specifies the number of bytes the initiator has allocated for the returned buffer data. An allocation length of zero is not considered an error by the target controller, and no data is sent to the initiator. The initiator may request up to 65,535 bytes to be transferred, including the 4-byte header. If the number of bytes requested exceeds the target controller's buffer size, the target controller transfers the entire buffer and terminates the command without an error. Under this condition, the initiator must check the value in the Available Length field in the Read Buffer Header to determine the number of bytes returned.

Data Format The data returned from the READ BUFFER command during the DATA IN phase consists of a 4-byte header immediately followed by the data bytes from the data buffer. This Read Buffer Header is formatted as shown in Table 2–46.

Table 2–46. Read Buffer Header Format

Byte \ Bit	7	6	5	4	3	2	1	0
0	Reserved							
1	Available Length (MSB)							
2	Available Length							
3	Available Length (LSB)							

The Available Length field (bytes 1–3) specifies the maximum amount of memory that the target controller has available in its data buffer. Depending on the Allocation Length specified in the CDB, the maximum amount of memory may or may not be the number of bytes actually transferred.

Error Conditions

If the data in the buffer has been modified since the last WRITE BUFFER command was issued, or if no WRITE BUFFER command has been issued since the last RESET condition, the READ BUFFER command is terminated with a CHECK CONDITION status and sets the sense key or additional sense code to MISCOMPARE (0Eh) or COMPARE ERROR (1Dh), respectively. If the Allocation Length is set to 4 or less, the target controller does not return this error.

Command **READ LONG – 3Eh**

The READ LONG command requests the target controller to perform a READ operation of one data block and the 6 ECC bytes associated with the block. The data and the ECC bytes are transferred to the initiator during the DATA IN phase.

Command Parameters

The CDB for the READ LONG command is formatted as shown in Table 2–47.

Table 2–47. READ LONG Command

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (3Eh)							
1	Logical Unit Number			Reserved				
2	Block Address (MSB)							
3	Block Address							
4	Block Address							
5	Block Address (LSB)							
6	Reserved (zeros)							
7	Transfer Length (MSB)							
8	Transfer Length (LSB)							
9	Control Byte (00h)							

The Block Address field (bytes 2–5) specifies the block at which the READ operation begins. The physical sector size is used to calculate the location.

The Transfer field (bytes 7 and 8) specifies the number of bytes to be transferred to the initiator.

Error Conditions If the block address is invalid, the target controller terminates the READ LONG command with a CHECK CONDITION status and sets the sense key or additional sense code to ILLEGAL REQUEST (05h) or ILLEGAL LOGICAL BLOCK ADDRESS (21h), respectively.

Command **WRITE LONG – 3Fh**

The WRITE LONG command requests the target controller to perform a write operation of one data block and the 6 ECC bytes associated with the block. The data and the ECC bytes are transferred from the initiator during the DATA OUT phase.

Command Parameters

The CDB for the WRITE LONG command is formatted as shown in Table 2–48.

Table 2–48. WRITE LONG Command

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (3Fh)							
1	Logical Unit Number			Reserved				
2	Block Address (MSB)							
3	Block Address							
4	Block Address							
5	Block Address (LSB)							
6	Reserved (zeros)							
7	Transfer Length (MSB)							
8	Transfer Length (LSB)							
9	Control Byte (00h)							

The Block Address field (bytes 2–5) specifies the block at which the WRITE operation begins. The physical sector size is used to calculate the location.

The Transfer field (bytes 7 and 8) specifies the number of bytes to be transferred from the initiator.

Error Conditions If the block address is invalid, the target controller terminates the READ LONG command with a CHECK CONDITION status and sets the sense key or additional sense code to ILLEGAL REQUEST (05h) or ILLEGAL LOGICAL BLOCK ADDRESS (21h), respectively.

Chapter 3

TMS320-SCSI Hardware Theory of Operations

The TMS320-SCSI is designed around the TMS320C25 microprocessor. The microprocessor controls access to different hardware modules via the address, data and control lines. Sixteen bits of address are used on the microprocessor address bus. These lines address the instruction PROM, microprocessor scratch RAM, and the expandable data buffer. The external I/O ports are addressed via the least significant four bits of these address lines. Sixteen bits of data lines are used on the microprocessor data bus to interconnect the microprocessor to the rest of the hardware.

This chapter provides the theory of operations for the TMS320-SCSI target controller. All hardware modules are described in detail. Refer to the hardware schematics diagrams and the signal glossary in Appendix A and the logic equations in Appendix B while reading this chapter.

Topic **Page**

TMS320-SCSI Target Controller General Description 3-2
TMS320-SCSI Functional Module Description 3-3
TMS320-SCSI Hardware 3-5
SN75C091A Overview 3-12

3.1 TMS320-SCSI Target Controller General Description

The TMS320-SCSI target controller is designed to provide a powerful and flexible hardware/firmware implementation of a general-purpose SCSI peripheral device controller.

The target controller interfaces to the single-ended SCSI bus with full implementation of SCSI phases, including DISCONNECTs and RESELECTs.

This single-board-printed-circuit assembly provides an evaluation model for applications involved with SCSI disk drives, where a single microprocessor is used for both SCSI and *servo control* functions.

An onboard 4-Mbyte data buffer is used to simulate the disk-drive media. This allows proper execution of the SCSI commands.

A programmable interrupt generator is incorporated with this design to simulate events that are normally involved with magnetic disk-drive applications.

The printed-circuit board is layed out with 26 spare locations of different sized integrated circuits, so the appropriate peripheral interfaces can be designed and prototyped. All components are socketed, thus allowing different speed components to be substituted and evaluated.

The TMS320-SCSI target firmware implements the SCSI common command set for a disk-drive controller. The firmware is written in C using the TMS320C25 C compiler; some time critical routines are written in TMS320C25 assembly language.

The target controller firmware interprets the SCSI request, converts the logical block address to the physical sector address contained within the data buffer (simulated media), performs data transfers, and posts the appropriate SCSI message and status.

The target controller can be configured in a single-host, multiple-target or in a multiple-host, multiple-target SCSI system.

The TMS320-SCSI uses the SN75C091A as the SCSI bus controller. It supports both synchronous and asynchronous data transfers.

3.2 TMS320-SCSI Functional Modules Descriptions

The TMS320-SCSI hardware is composed of the following modules that are configured as shown in Figure 3–1 on page 3-5:

- Microprocessor
- I/O port decode logic

  The I/O port decode logic selects one of the possible 16 I/O ports via the four least significant bits of the address bus and the I/O select signal. The read/write control signal decides if the port is being read/written to or read/written from the data bus.

- Instruction memory

  The 64K-word program memory holds the target firmware. The firmware program consists of the TMS320C25 instructions. These instructions are accessed via the address bus and the program-select signal. The selected instruction is then placed onto the data bus and stored into the TMS320C25 instruction register.

- Scratch RAM

  The 32K-word scratch RAM holds the firmware data structure and variables. These RAM locations are accessed via the address bus and the data-select signal. The read/write control signal controls reading or writing to or from the data bus.

- Data buffer

  The 4-megabyte data buffer is used to simulate the disk-drive media. This data buffer is organized into 128 pages of 32K bytes each. Accessing the data buffer locations is accomplished by first loading the page-select latch and then reading or writing the location addressed by the address bus.

- SCSI DMA FIFO

  The 64-byte FIFO is used for SCSI data transfers. For SCSI write operations, the FIFO is written to by the firmware as an outport and data bytes are taken out of the FIFO via the handshaking with the SCSI bus controller. For SCSI read operations, the FIFO is written to by the SCSI bus controller and data bytes are taken out of the FIFO by the firmware as an inport.

- External test condition

  The external test conditions are read as an inport. The firmware tests on these conditions to learn the states of user-selectable switches or the FIFO readiness.

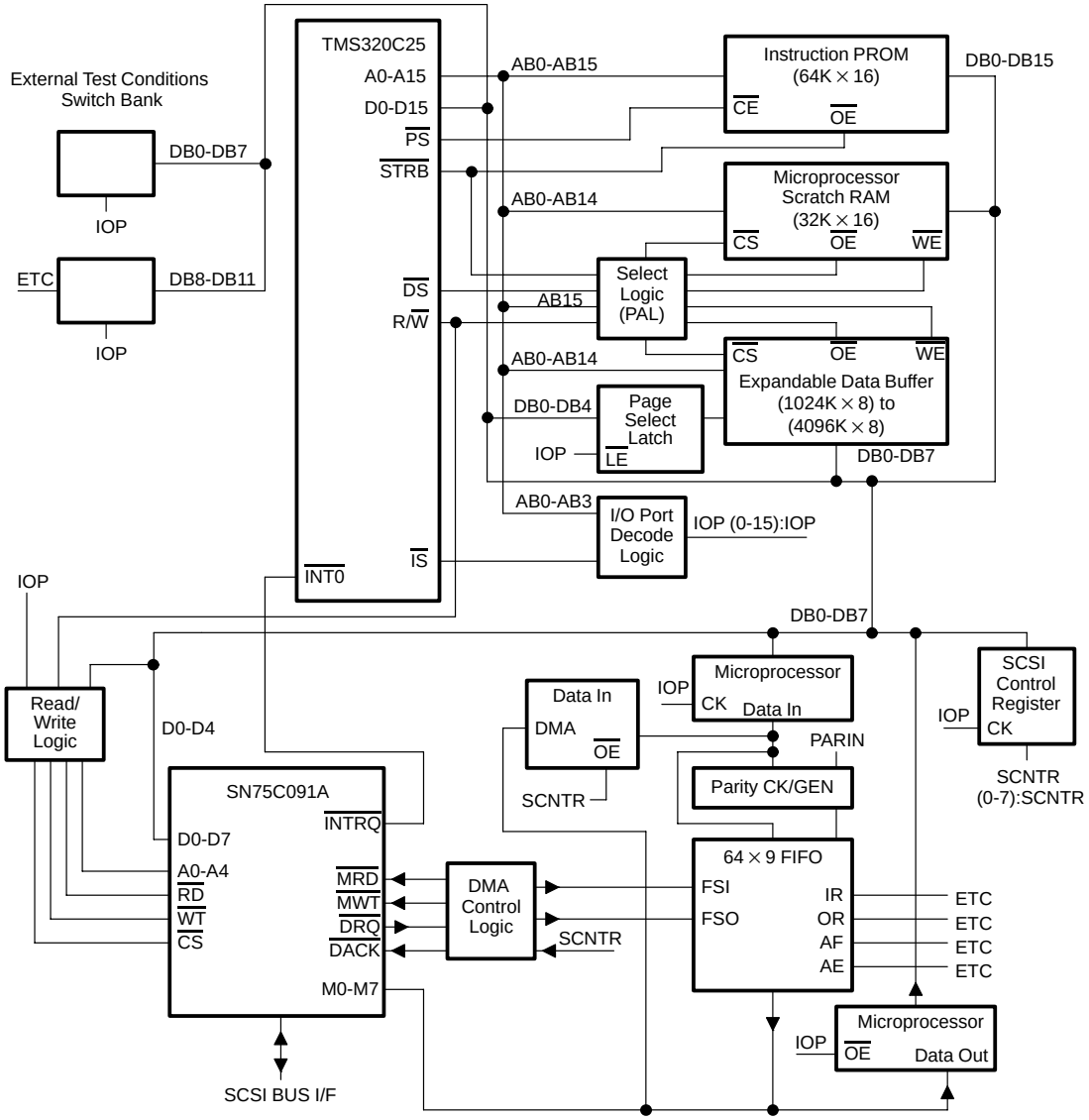
- SCSI bus interface controller

  The SCSI bus controller interfaces to the single-ended SCSI bus. The read/write logic handles accessing the registers within the SCSI bus controller for read or write operations. The SCSI data transfers are performed via the DMA logic to and from the FIFO.

3.3 TMS320-SCSI Hardware

Figure 3–1 is a block diagram of the TMS320-SCSI hardware. The following subsections describe the various operations.

Figure 3–1. TMS320-SCSI Hardware Block Diagram



3.3.1 Read/Write Strobe Decode Logic

The microprocessor access to the I/O space is accomplished via the two decoders designated by (U40 and U41), as shown in Figure A–6. The least-four-

significant bits of the address bus (AB0–AB4) are used in conjunction with the read strobe $\overline{\text{RSTRB}}$ to generate the strobe signal to place the selected I/O port onto the microprocessor data bus. The strobe signals generated by the decoder at (U40) are listed below. Refer to the appropriate section for a more detailed description.

$\overline{\text{RSBCD}}$ Places the data from the selected register of the SCSI bus interface controller onto the lower byte of the data bus.

$\overline{\text{RFIFO}}$ Places one byte of data from the FIFO onto the lower and upper byte of the data bus.

$\overline{\text{RCND1}}$ Places the external test condition onto the data bus.

$\overline{\text{CLRINT1}}$ Clears the interrupt generated by the programmable interrupt generator.

The least-four-significant bits of the address bus (AB0–AB4) are used in conjunction with the write strobe $\overline{\text{WRSTRB}}$ to generate the strobe signal to update the selected I/O port from the microprocessor data bus. The strobe signals generated by the decoder at (U41) are listed below. Refer to the appropriate section for a more detailed description.

$\overline{\text{WSBCD}}$ Updates the selected register of the SCSI bus interface controller from the lower byte of the data bus.

$\overline{\text{WFL}}$ Writes the FIFO from the lower byte of the data bus.

$\overline{\text{WFU}}$ Writes the FIFO from the upper byte of the data bus.

$\overline{\text{WFIFC}}$ Updates the FIFO control register from the lower byte of the data bus.

$\overline{\text{WSBCA}}$ Updates the external address register for the SCSI bus interface controller.

$\overline{\text{WAH}}$ Updates the external address register for the data buffer memory.

$\overline{\text{WINTCNT}}$ Updates the 16-bit interrupt generator from the data bus.

$\overline{\text{WLED}}$ Updates the LED register from the lower byte of the data bus.

3.3.2 Instruction Memory

The 64K words of instruction memory contain the code executed by the TMS320C25 microprocessor. This code controls all the devices on the target

controller and determines what I/O ports are activated for control of data transfers.

The TMS320C25 instructions could be fetched from two different sources, instruction PROMS or zero-wait-state SRAMs. The jumper pair, designated by (W12) when removed, selects the PROMs while the instruction source installing this jumper selects zero-wait-state SRAMs.

The instruction PROMs, designated by (U19 and U34), are shown in Figure A–2. Two TMS27C512s are used to form the 64K words. The PROM at (U19) contains the lower byte of the instructions while the PROM at (U34) contains the upper bytes. Two three-state buffers (U18 and U33) are used to minimize the turn-off time (time from enable to disable). These PROMs are selected via the $\overline{\text{PSPRM}}$ signal; this signal is generated by the PAL at (U39) as shown in Figure A–5. This line is activated when the jumper at (W12) is removed and $\overline{\text{MPPS}}$ is asserted by the microprocessor. The instruction is enabled onto the data bus via the $\overline{\text{MPSTRB}}$ signal. The writable control store, shown in Figure A–11, allows the program to be downloaded and then executed with no wait states. These high-speed RAMs are organized in 64K words (four CY7C199s are used). Two RAMs at (U22 and U37) contain the lower 32K words and the RAMs at (U21 and U36) contain the upper 32K words of program. The lower 32K words are selected via the $\overline{\text{PSRML}}$ signal. This signal is generated by the PAL at (U39) as shown in Figure A–5. The $\overline{\text{PSRML}}$ signal is activated when the jumper (W12) is installed and the $\overline{\text{MPPS}}$ line is asserted along with (AB15 = 0). The upper 32K words are selected via the $\overline{\text{PSRMU}}$ signal; this signal is generated by the PAL at (U39) as shown in Figure A–5. The $\overline{\text{PSRMU}}$ signal is activated when the jumper (W12) is installed and the $\overline{\text{MPPS}}$ line is asserted along with (AB15 = 1). Instructions are enabled onto the data bus via the $\overline{\text{RSTRB}}$ signal.

3.3.3 Microprocessor Scratch RAM

The 32K words of scratch RAM holds firmware variables, data structures, and replacement and look-up tables. Two CY7C199 RAM static chips are used to form 32K words as shown in Figure A–5.

These high-speed RAMs allow microprocessor access with no wait states. The RAM chip at (U20) holds the lower byte of the 16-bit word while the RAM chip at (U35) holds the upper byte.

These RAMs are selected by the $\overline{\text{DSSRM}}$ signal; this signal is generated by the PAL at (U39) as shown in Figure A–5. The $\overline{\text{DSSRM}}$ signal is activated when $\overline{\text{MPDS}}$ signal is asserted along with (AB15=0).

Writing these RAMs is controlled by the $\overline{\text{WSTRB}}$ signal. This signal activates when the $\overline{\text{MPRW}}$ signal is asserted by the microprocessor to indicate a write cycle along with $\overline{\text{MPSTRB}}$ signal.

Reading of these RAMs is controlled by the $\overline{\text{RSTRB}}$ signal. The $\overline{\text{RSTRB}}$ signal activates when the $\overline{\text{MPRW}}$ signal is not asserted by the microprocessor to indicate a read cycle along with $\overline{\text{MPSTRB}}$ signal.

3.3.4 Data Buffer

The data buffer is 8-bits wide and is expandable to four megabytes in size. It is used to simulate the disk drive media. The media geometry consists of one head and 128 cylinders with each track containing 64 sectors of 512 bytes each.

Four TM024EAD9 (one megabyte dynamic RAM modules) are used to form the four Mbytes, designated by (U4, U5, U6, and U7), as shown in Figure A–4.

The operation of these dynamic RAMs is controlled by the 74ACT4503 (dynamic RAM controller), designated by (U24), as shown in Figure A–3.

The access to these dynamic RAMs is activated by the $\overline{\text{DSDRM}}$ signal. This signal is generated by the PAL at (U39) as shown in Figure A–5. The $\overline{\text{DSDRM}}$ signal is activated when $\overline{\text{MPDS}}$ signal is asserted along with (AB15=1).

The lower 15 bits of the 22-bit address that is required for reading and writing these RAM locations are supplied by (AB0–AB14), whereas the upper 7 bits are supplied via the extended address register designated by (U23). The content of the extended address is loaded by the firmware once and stays unchanged for the whole track transfer of 32K bytes. The content is updated upon crossing the track boundary.

The read operation takes place when $\overline{\text{DSDRM}}$ signal is activated along with $\overline{\text{MPRW}}$ inactive to indicate a read cycle. The $\overline{\text{MPSTRB}}$ signal is used to place the content of the addressed location onto (DB0–DB7).

The write operation takes place when $\overline{\text{DSDRM}}$ signal is activated along with $\overline{\text{MPRW}}$ active to indicate a write cycle. The $\overline{\text{MPSTRB}}$ signal is used to update the content of the addressed location via the (DB0–DB7). The decode logic designated by (U38) selects one of the four RAM modules for the write operation.

The refresh operation of the dynamic RAMs is controlled internally by the dynamic RAM controller.

3.3.5 FIFO Memory

The FIFO memory allows efficient data transfers between the SCSI bus interface controller (75C091A) and the rest of the system. It allows the TMS320C25

firmware to perform the DMA task. This 64-byte deep FIFO, designated by (U56), is shown in Figure A–8.

The FIFO input (FI0–FI7) is a three-state bus. These are the possible sources for this bus:

- The latch designated by (U54) latches the lower byte of the data bus (DB0–DB7) to be written to the FIFO during the SCSI write operation.
- The latch designated by (U55) latches the upper byte of the data bus (DB0–DB7) to be written to the FIFO during the SCSI write operation.
- The three-state buffer designated by (U53) as shown in Figure A–9 allows the SCSI DMA data to be written to the FIFO during the SCSI read operation.

The possible destinations for the FIFO output lines (FO0–FO7) are:

- The three-state buffer designated by (U43) allowing the FIFO output to be placed on the lower byte of the data bus (DB0–DB7). This is used during SCSI read operations by the microprocessor.
- The three-state buffer designated by (U44) allowing the FIFO output to be placed on the upper byte of the data bus (DB8–DB15). This is used during SCSI read operations by the microprocessor.
- The three-state buffer designated by (U52) allowing the FIFO output to be placed on the DMA channel of the (75C091A) SCSI bus interface controller during the SCSI write operations.

The SCSI control register designated by (U32), as shown in Figure A–7, holds control information for SCSI data transfers. This register is loaded by the microprocessor via the data bus lines. The content of this register controls the direction of the data transfer, selection of either lower or the upper byte transfer, and enabling of Request/Acknowledge data transfer handshake.

The FIFO data transfer to and from the SN74C091A are controlled by DREQ (SBCRQ) and DACK ($\overline{\text{SBCACK}}$) signals of the SN75C091A. These lines cause the FIFO shift in (FSI) and FIFO shift out (FSO) to write or read the FIFO. The FIFO transfer signals (FSI) and (FSO) are generated by the logic contained inside the PAL designated by (U46), as shown in Figure A–7.

The data transfers to and from the microprocessor RAM or the data buffer are accomplished by the microprocessor executing the IN or the OUT instructions.

During the write operations, the microprocessor services the FIFO by monitoring the HF (half full line), and then executing the IN instruction in conjunction

with the RPT (repeat instruction) to read all 32 available bytes with a single RPT instruction.

During the read operations, the microprocessor services the FIFO by monitoring the $\overline{HF}$ (half empty line) and then executing the OUT instruction in conjunction with RPT (repeat instruction) to write all 32 available bytes with a single RPT instruction.

3.3.6 Test Condition Logic

The test condition logic allows the TMS320C25 to test external conditions as shown in Figure A–6. One of the 16 conditions is selected via the data bus lines that are loaded by the microprocessor as an I/O port.

These test conditions are enabled via the $\overline{RCND1}$ signal. This signal is generated by the Read/Write strobe decoders as shown in Figure A–6.

The three-state buffer designated by (U14) enables the user-selectable, 8-position switch to be placed onto (DB0–DB7) lines.

The three-state buffer designated by (U17) enables the FIFO status signals onto (DB8–DB11) lines. Four of the test condition inputs are left open and are not used.

3.3.7 SCSI Bus Interface Controller

The SN75C091A SCSI bus controller provides single-ended SCSI interface. The SN75C091A interfaces to the rest of the hardware via the microprocessor and the DMA ports as described below.

3.3.7.1 Microprocessor Port

The microprocessor port allows the TMS320C25 to setup and read the SN75C091A registers with command and status. The SN75C091A is selected via the $\overline{ISSBC}$ signal as shown in Figure A–9. Reading and writing registers are controlled by the $\overline{RSBCD}$ and $\overline{WSBCD}$ signals. The addresses to these registers are supplied via the address register as shown in Figure A–7. This address register is loaded by the microprocessor via the $\overline{WSBCA}$ signal. The data bus lines (DB0–DB7) are used to update or read the addressed registers. The INTRQ pin on the SN75C091A is used to generate an interrupt via the $\overline{MPINT2}$ line to indicate the completion of the command. The $\overline{MPINT2}$ line can also be read via the status register so the microprocessor can poll for completion in place of the interrupt.

3.3.7.2 DMA Port

The DMA port of the SN75C091A is interfaced to the 64-byte FIFO. Refer to the FIFO description mentioned earlier in this document. The DMA control

lines are generated by the PAL at (U46) as shown in Figure A–7. The DMA read cycle is controlled by the $\overline{\text{SBCMRD}}$ signal, whereas $\overline{\text{SBCMWT}}$ indicates a DMA write cycle. The $\overline{\text{SBCWAIT}}$ signal is generated by the SN75C091A to hold off the transfer handshake until the SN75C091A internal FIFO is ready to proceed with data transfers.

3.3.8 Programmable Interrupt Generator

These 8-bit counters are loaded via the microprocessor data lines designated by (U30 and U31), as shown in Figure A–11. The value of the counter increments with each clock cycle and causes the output of the flip-flop designated by (U57) to set. Once set, it generates a microprocessor interrupt level one $\overline{\text{MPINT}}$. The signal $\overline{\text{CLRINT}}$ clears the output of the flip-flop.

3.3.9 LED Register

The LED register is designated by (U28), as shown in Figure 5–1, *TMS320-SCSI Target-Controller Printed Circuit Board*. This 8-bit register is loaded by the microprocessor via the lower byte of the data bus. A binary value of zero turns on the LED, and a binary value of one turns the LED off. The register is cleared by the $\overline{\text{MPRS}}$ microprocessor reset line. The description of each bit position is listed in Section 5.4, *Indicators*.

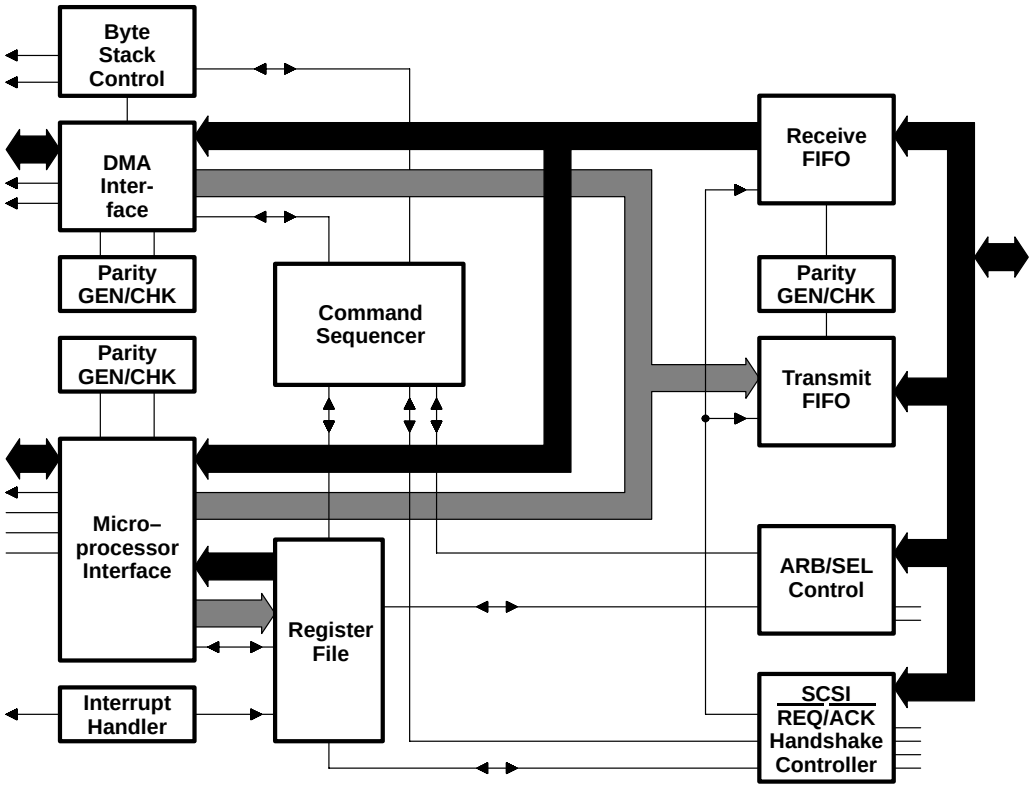
3.4 SN75C091A Overview

The SN75C091A interfaces a microprocessor subsystem to the ANSI small computer systems interface (SCSI). This single-ended SCSI implementation acts as one of up to eight nodes communicating over a maximum six-meter bus as detailed in the SCSI specification (ANSI X3.131–1986). Its microprocessor and DMA interfaces allow the SN75C091A SCSI bus controller (SBC) to be used in a variety of host or peripheral applications.

3.4.1 SN75C091A Architecture

The functional block architecture of the SN75C091A is shown in Figure 3–2.

Figure 3–2. SN75C091A Functional Block Architecture



3.4.2 Features

The SCSI bus interface

- Complies with ANSI X3.131–1986 SCSI standard
- Performs initiator and target functions
- Supports arbitration, selection, and reselection
- Performs asynchronous data transfers of up to 5 Mbytes/second
- Performs synchronous data transfers of up to 5 Mbytes/second with programmable offset up to 15
- Has on-chip 48-mA transceivers
- Provides optional parity generation, checking, and pass-through
- Reduces overhead associated with initiator multithreading by automatically handling save-data-pointer messages, disconnects, and reconnects
- Performs automatic message and command-length decoding
- Has two 32-byte FIFOs for command and message preloading

The microprocessor interface

- Provides chip control via directly-addressable registers
- Has optional address latch line for multiplexed address/data buses
- Allows DMA- or programmed-I/O data transfers
- Is interrupt-driven to minimize host polling
- Can execute multiphase commands to minimize interrupts
- Has 24-bit transfer counter
- Provides byte-stacking control to accommodate 8-, 16-, and 32-bit systems
- Offers optional parity generation and checking
- Is equipped with separate ports for DMA and microprocessor interfacing

3.4.3 Command Sequencer

The heart of the SN75C091A is a large state machine called the command sequencer. Microprocessor commands to the SN75C091A are interpreted by the command sequencer, which then activates subordinate state machines (for example, the arbitration controller, $\overline{\text{REQ}}$/$\overline{\text{ACK}}$ handshake controller, and DMA interface controller) to perform the functions necessary to carry out each command. The command sequencer enables the SN75C091A to execute powerful multiphase SCSI sequences with very few interrupts.

3.4.4 SCSI $\overline{\text{REQ}}$/$\overline{\text{ACK}}$ Handshake Controller

The $\overline{\text{REQ}}$/$\overline{\text{ACK}}$ handshake controller handles requests from the command sequencer to perform single SCSI phase transfers such as Message, Command,

Status, and Data. $\overline{\text{REQ}}$, $\overline{\text{ACK}}$, $\overline{\text{ATN}}$, $\overline{\text{MSG}}$, $\overline{\text{C}}/\overline{\text{D}}$, and $\overline{\text{I}}/\overline{\text{O}}$ are controlled and/or monitored (depending on the mode—target or initiator) to transfer information between the transmit and receive FIFOs and the SCSI bus. The $\overline{\text{REQ}}/\overline{\text{ACK}}$ handshake controller supports both synchronous and asynchronous data transfers.

3.4.5 Arbitration and Selection Controller

The arbitration and selection controller handles requests from the command sequencer to establish a connection with another device on the SCSI bus. SCSI bus signals $\overline{\text{BSY}}$, $\overline{\text{SEL}}$, $\overline{\text{I}}/\overline{\text{O}}$, and $\overline{\text{ATN}}$ are controlled and/or monitored to affect automatic completion of SCSI arbitration, selection, and reselection phases. This state machine also concurrently monitors the SCSI bus and alerts the command sequencer and interrupt logic if the SN75C091A has been selected or reselected by another device on the SCSI bus.

3.4.6 Register File

The register file consists of 32 registers that allow the local microprocessor to initiate, control, and monitor SCSI transfer operations performed by the SN75C091A. The 32-byte transfer and receive FIFOs can also be accessed through the register file.

3.4.7 Microprocessor Interface

The microprocessor interface provides the logic necessary for a microprocessor or other host computer system to access and store information in the register file. Both multiplexed and nonmultiplexed address/data buses are supported through this interface.

3.4.8 Receive and Transmit FIFOs

These 32-byte by 9-bit FIFOs provide a buffer between the SCSI bus and memory to improve transfer efficiency and minimize microprocessor overhead. These FIFOs are accessed in the same manner as a register in the register file—through the microprocessor interface for programmed I/O or through the DMA interface for SCSI data transfers performed via an intermediate DMA controller. The 32-byte FIFOs allow SCSI commands and messages to be preloaded or fully received, thus minimizing microprocessor intervention. The ninth bit allows parity pass-through mode for high-reliability systems.

3.4.9 Interrupt Control

Interrupt control logic monitors the various state machines to determine when microprocessor intervention is required. Interrupt status information is main-

tained in the register file and the interrupt is reported either by microprocessor polling or via the INTRQ signal (if external interrupts are enabled).

3.4.10 DMA Interface

The DMA interface provides the control logic necessary to interface the SN75C091A with an external DMA controller. DMA handshake signals DREQ and DACK and a separate 9-bit data port form the data path used to transfer SCSI data between external memory and the SN75C091A transmit and receive FIFOs without microprocessor intervention.

3.4.11 Byte Stack Control

The byte stack control is used in conjunction with the DMA interface to allow easy interfacing of the 8-bit SCSI bus to 16-, 24-, and 32-bit systems. This control logic facilitates loading and unloading of external bidirectional registers (byte stack registers) so that 16-, 24-, and 32-bit words can be broken down into their constituent bytes. No external logic is required to interface the SN75C091A to a 16-bit byte stack register; a decoder is necessary for interfacing to 24- and 32-bit systems.

3.4.12 Parity Generators/Checkers

Parity generation and checking is provided for all three SN75C091A ports (SCSI, DMA, and microprocessor). Versatile parity control via the register file allows the SN75C091A to adapt easily to any system. Parity generation control allows memory and SCSI parity information to be passed through the FIFOs. User-selectable parity sense (odd or even) provides error generation capabilities to assist in system checkout.

Chapter 4

Firmware Theory of Operations

This chapter discusses the theory of operations for the TMS320-SCSI firmware. The firmware data structures and variables are described in each appropriate subsection. References are made to the data structure drawings and the firmware source-code routines. The source code routines and program variables are referred to with their name enclosed in parentheses; for example, (Main) refers to the source code routine called Main.

Assembly source listings for all software are available upon request from Texas Instruments.

Topics discussed in this chapter include

Topic **Page**

Firmware Design Overview **4-2**
Firmware Operation **4-3**
Command Execution Routines **4-18**
System Initialization Routines **4-19**
SCSI Information Transfer Routines **4-20**
SCSI Sense-Data Buffer-Management Routines **4-21**
Target-Controller Command Overhead Example **4-22**

4.1 Firmware Design Overview

The TMS320-SCSI firmware's outermost layer (Main), is the controller's idle routine. This routine is entered when the controller is powered up or the SCSI reset condition has been detected.

The (Main) routine first initializes data structures and variables associated with the target controller firmware. While in (Main), the incoming SCSI command is queued and then executed. After executing the SCSI command the target controller concludes the SCSI bus operation by sending the status and the command complete messages.

While disconnected from the SCSI bus, the target controller may receive commands from other SCSI initiators. In this case, commands are queued and executed after the completion of the current command. If a command is received from the same initiator while disconnected from the SCSI bus, the target controller immediately concludes the command by sending busy status.

4.2 Firmware Operation

The execution of the SCSI commands by the target controller firmware involves three steps:

- ❑ Receive and queue command descriptor block (command reception)
- ❑ Decode and execute command (command process)
- ❑ Generate and return status (command end status)

4.2.1 Command Reception

The target controller firmware instructs the SCSI bus controller (SN75C091A) to respond to selection from SCSI initiator(s) via the (WaitSlctRcv) routine when it is disconnected from the SCSI bus.

The (SN75C091A) notifies the target controller firmware upon receipt of the command by posting the function-complete bit inside its status register.

The target controller firmware queues the SCSI commands via the (QSCSICmd routine. Commands are queued based upon the initiator's ID. Space is allocated to store one command per initiator. Buffer pointers are maintained to assure that commands are handled on a first-in-first-out basis. Figure 4–1 illustrates these command buffers. The two pointers (CMDBUFFIRST) and (CMDBUFLAST) point to the first and the last commands queued—a value of zero would indicate an empty command buffer.

The format of the queues are illustrated in Figure 4–2 through Figure 4–9. The field descriptions for each of the command buffers illustrated are as follows:

(CMDBUFNEXT)	Pointer to the next command buffer. When zero, it indicates no other commands are currently queued.
(CMDDISST)	Indicates if this command currently has disconnected from the SCSI bus.
(CMDPEND)	Indicates if the execution of this command is pending or has been completed.
(CMDIDPR)	Indicates whether the command received from the initiator included the initiator ID.
(CMDDISEN)	Indicates whether the identify message sent by the initiator allows disconnection.
(CMDLUNR)	Contains the logical unit number sent by the initiator during the identify message.
CMDCDB[0]:CMDCDB[9]	Contains the SCSI command bytes.

Figure 4–1. SCSI Command Buffers Structure for 8 Initiators

Figure 4–2. Command Buffer for Initiator 0

Figure 4–3. Command Buffer for Initiator 1

Figure 4–4. Command Buffer for Initiator 2

Figure 4–5. Command Buffer for Initiator 3

Figure 4–6. Command Buffer for Initiator 4

Figure 4–7. Command Buffer for Initiator 5

Figure 4–8. Command Buffer for Initiator 6

Figure 4–9. Command Buffer for Initiator 7

4.2.2 Command Process

The processing of the SCSI command starts upon detection of a nonzero value in (CMDBUFLAST), meaning a command has arrived and is pending further execution. The target controller firmware uses the (SCSICmdPrcss) routine to process the SCSI command.

The (SCSICmdPrcss) routine first checks for invalid fields such as an invalid logical unit number. If such conditions are detected, the target controller sets up the request sense data indicating the error via the (SnsDataSet) routine. Figure 4–10 illustrates the static buffers that are allocated for the request sense data. A separate sense buffer is maintained for each initiator. The request sense data buffers are formatted as shown in Figure 4–11 through Figure 4–18.

Figure 4–10. Request Sense Data (Initiators 0–7)

Figure 4–11. Request Sense Data for Initiator 0

Figure 4–12. Request Sense Data for Initiator 1

Figure 4–13. Request Sense Data for Initiator 2

Figure 4–14. Request Sense Data for Initiator 3

Figure 4–15. Request Sense Data for Initiator 4

Figure 4–16. Request Sense Data for Initiator 5

Figure 4–17. Request Sense Data for Initiator 6

Figure 4–18. Request Sense Data for Initiator 7

If the logical unit is reserved by another initiator, the target controller concludes the command by sending reservation conflict status and a command complete message to the initiator.

If none of the above mentioned conditions exists, the (SCSICmdPrcss) routine proceeds with further execution.

The decoding of the command code is accomplished by producing a pointer using the command code. This pointer points to an entry inside the command

tables. The entry itself contains the pointer to the routine that executes the command.

The command execution routines post the appropriate request sense data, if any, sets the SCSI command status, and returns control back to the (SCSICmdPrcss) routine.

The SCSI command tables, as mentioned above, consist of a set of pointers. The value for these pointers are precalculated and stored in command tables as part of the system initialization. Each table corresponds to a group code with a total of 32 entries, one for each command code.

Although there is the possibility for eight group code commands, the target controller only supports group 0 and group 1 commands. The data structures in Figure 4–19 through Figure 4–22 illustrate the command tables for group code 0 and group code 1 commands in the ready and not-ready conditions.

For all the supported command codes within this group there is an associated pointer that points to the start of the routine that executes the command. For example, command code 0 corresponds to (TESTRDYEP), which is the pointer for the test unit ready command execution. The command codes that are not supported by the target controller are set to point to the common routine that posts an illegal request because the command code is invalid. For example, command codes 5 and 6 are not supported and the pointer associated with these command codes are set to point to (ILLCMDEP), illegal command execution pointer.

Figure 4–19. Command-Execution Pointers for Group 0 Commands (Local Unit in Ready State)

Figure 4–20. Command-Execution Pointers for Group 0 Commands (Local Unit in Not-Ready State)

Figure 4–21 illustrates the command table for group 1 commands.

Figure 4–21. Command-Execution Pointers for Group 1 Commands (Local Unit in Ready State)

Figure 4–22. Command-Execution Pointers for Group 1 Commands (Local Unit in Not-Ready State)

All unsupported group codes—group codes 2, 3, 4, 5, and 6—are directed to the common table, as shown in Figure 4–23. All 32 command code entries are preinitialized to point to the (ILLCMDEP), illegal command execution pointers.

Figure 4–23. Command-Execution Pointers for Groups 2, 3, 4, 5, 6, and 7

4.2.3 Command End Status

The (SCSICmdPrcss) routine concludes the SCSI bus operation via the (SendStMsg) routine. The (SensStMsg) routine instructs the SCSI bus controller (SN75C091A) to send the SCSI status that was set previously by the command execution routine and the command complete message. At this time, the command operation is completed and control is returned back to the (Main) routine.

4.3 Command Execution Routines

The following is the list of the command routines ordered by the command operation codes.

(TESTRDYEP)	Test unit ready command routine
(REZEROEP)	Rezero unit command routine
(RQSNSEP)	Request sense command routine
(FORMATEP)	Format unit command routine
(REASSIGNEP)	Reassign block command routine
(READEP)	Read command routine (for group 0 six byte CDB)
(WRITEEP)	Write command routine (for group 0 six byte CDB)
(SEEKEP)	Seek command Routine (for group 0 six byte CDB)
(INQUIRYEP)	Inquiry command routine
(MODESLCTEP)	Mode select command routine
(RESERVEEP)	Reserve unit command routine
(RELEASEEP)	Release unit command routine
(MODESNSEP)	Mode sense command routine
(STRSTPEP)	Start/Stop unit command routine
(READCEP)	Read capacity command routine
(READEEP)	Read extended command routine (for group 1 ten byte CDB)
(WRITEEEP)	Write extended command routine (for group 1 ten byte CDB)
(SEEKEEP)	Seek extended command Routine (for group 1 ten byte CDB)
(WRITEVEP)	Write and verify command routine
(WRITEBEP)	Write buffer command routine
(READBEP)	Read buffer command routine
(READLEP)	Read long command routine
(WRITELEP)	Write long command routine

4.4 System Initialization Routines

The following firmware routines are executed during the power-up and reset conditions:

(CmdExTblInit)	This routine initializes the SCSI command tables.
(SCSIInit)	This routine initializes all variables associated with SCSI bus management.
(QCmdInit)	This routine initializes all SCSI command buffers to an empty state.

4.5 SCSI Information Transfer Routines

The following firmware routines are used to transfer data over the SCSI bus:

(SendSCR) This routine is used to transfer data from the scratch RAM to the SCSI bus. This transfer takes place via the microprocessor port of the (SN75C091A) SCSI bus controller. The calling routine specifies the phase to be established for the transfer, the starting transfer address, and the size of the transfer.

(RcvSCR) This routine is used to transfer data from the SCSI bus to the scratch RAM locations. This transfer takes place via the microprocessor port of the (SN75C091A) SCSI bus controller. The calling routine specifies the phase to be established for the transfer, the starting transfer address, and the size of the transfer.

4.6 SCSI Sense-Data-Buffer Management Routines

These firmware routines are used to initialize the data buffer.

(SnsDataSet)	This routine sets the sense data buffer for the specified initiator. The calling routine specifies the initiator ID, sense key value, and the information bytes value.
(SnsDataClr)	This routine clears the sense data for the specified initiator.

4.7 Target-Controller Command Overhead Example

The time involved to execute a SCSI command is referred to as the command overhead. This is the time from the SELECTION phase of the command initiated by the SCSI initiator until the time COMMAND COMPLETE message is posted by the target controller and the BUS-FREE phase is detected. See Figure 4–24 and Figure 4–25.

Figure 4–24. READ Command Example Without Disconnect

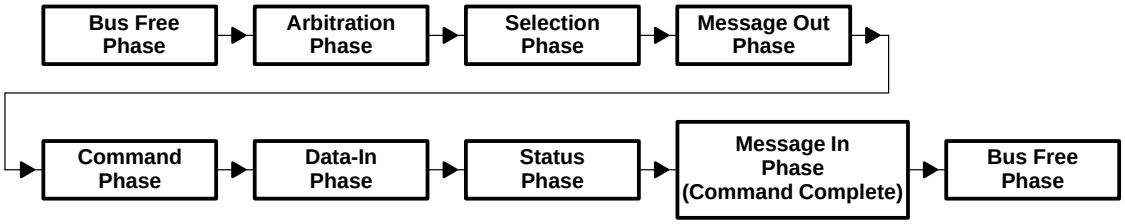
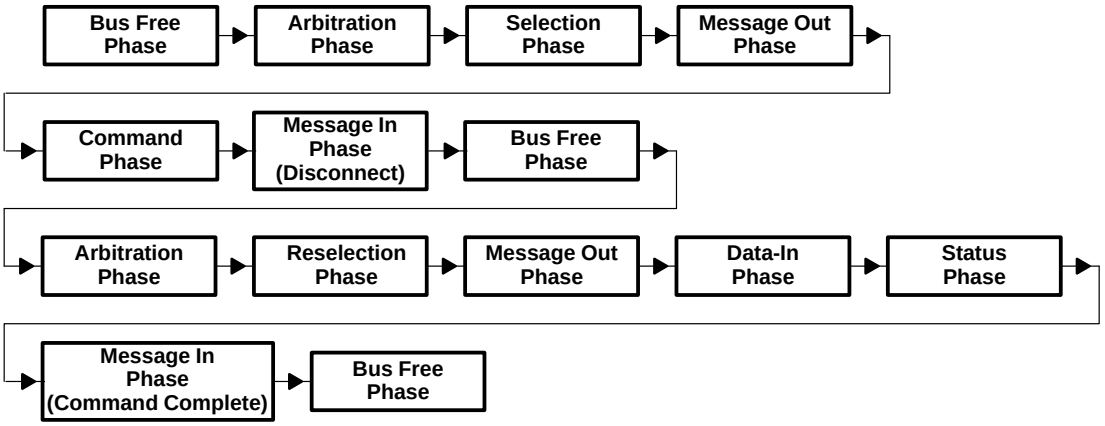


Figure 4–25. READ Command Example With Disconnect



The command overhead is the total delay involved by both the SCSI initiator and the SCSI target devices. The target controller command overhead is the total delay that is involved only by the target controller. Consider again the target lead command example in Figure 4–24 and Figure 4–25. A SCSI bus analyzer can be used to display the delays of each SCSI bus phase in order to benchmark the overhead for a READ or WRITE command. The timing for each SCSI bus phase during a READ command is tabulated in Table 4–1. Figure 4–26 displays the SCSI bus analyzer output for this READ command example. Using this technique, you can observe the performance of the TMS320C25-based SCSI system to evaluate the overhead for command processing.

Table 4–1. Command Overhead for a 4K-Byte Write

Event	Time (μS)
Selection to first REQ of message out	18.20
Last ACK of message out to first REQ of command out	1.00
Last ACK of command out to first REQ of data out	234.30
Last ACK of data out to first REQ of status in	138.70
Last ACK of status in to first REQ of message in	.90
Last ACK message in to bus free	.40
Total time	393.50

Figure 4–26. TMS320-SCSI Target Controller Time Delays

Figure 4–26. TMS320-SCSI Target Controller Time Delays (Concluded)

Chapter 5

TMS320-SCSI Physical Characteristics

This chapter discusses the physical characteristics, connectors, configuration switches and jumpers, indicators and method for interfacing with the SCSI host adaptors.

Topic **Page**

Physical Layout 5-2
Connectors 5-3
Configuration Switches and Jumpers 5-4
Indicators 5-5
Interfacing TMS320-SCSI With SCSI Host Adaptors 5-6

5.1 Physical Layout

Figure 5–1 illustrates the physical layout of the TMS320-SCSI printed circuit board. The connectors, configuration switches and indicators are described in Sections 5.2 through 5.4. Section 5.5 shows how the TMS320-SCSI interfaces with the SCSI host adaptors.

Figure 5–1. TMS320-SCSI Target-Controller Printed Circuit Board

5.2 Connectors

5.2.1 J1 Connector

The 60-pin connector, designated by (J1), provides the microprocessor signals. A logic analyzer can be used to monitor these signals.

5.2.2 J2 Connector

The 50-pin connector, designated by (J2), interfaces with the SCSI bus.

5.2.3 J3 Connector

The 4-pin connector, designated by (J3), connects to an external power supply.

5.3 Configuration Switches and Jumpers

The 8-position switch bank is designated by (SW1). The switch in the off position represents a binary value of one, and the on position represents a binary value of zero.

- Switch position 1 is used to specify the least-significant bit of the SCSI ID.
- Switch position 2 is used to specify the middle bit of the SCSI ID.
- Switch position 3 is used to specify the most-significant bit of the SCSI ID.
- Switch positions 4 and 5 are currently unused.
- Switch positions 5–7 are used to specify the frequency of the interrupt generated by the microprocessor internal timer to simulate the servo interrupt.

These are the hardware-selectable jumper definitions:

- Jumper post (W2), when removed, disables the clock to the microprocessor.
- Jumper post (W12), when removed, enables the PROMS output onto the microprocessor data bus. When installed, the writable-control-store output is placed onto the microprocessor data bus.
- Jumper post (W13), when installed, supplies terminator power to the SCSI bus.

5.4 Indicators

The 8-position LED designated as LED1 is described below:

- Bit position 0 (self diagnostics), when on, indicates a failure with the internal self test.
- Bit position 1 (DS1) is not used.
- Bit position 2 (DS2) is not used.
- Bit position 3 (power) stays on while power is applied to the board.
- Bit position 4 (servo) turns on upon entering the firmware servo interrupt routine and turns off upon exiting the firmware routine.
- Bit position 5 (SCSI) turns on upon the selection phase of the SCSI bus and turns off upon exiting the firmware end-message phase.
- Bit position 6 (WRITE) stays turned on while executing the SCSI bus data-in phase.
- Bit position 7 (READ) stays turned on while executing the SCSI bus data-out phase.

5.5 Interfacing TMS320-SCSI With SCSI Host Adaptors

Use a standard 50-pin SCSI cable to connect the SCSI connector on the TMS320-SCSI labeled as J2 to the 50-pin SCSI connector located on your SCSI host adaptor board. Make sure that pin 1 of either of the connectors matches pin 1 marked on the cable.

The jumper labeled as W13 located next to the SCSI connector on the TMS320-SCSI board supplies the SCSI bus terminator power. This jumper needs to be installed if your host adaptor does not supply terminator power; otherwise, it could be removed.

The switch bank positions 1, 2, and 3 labeled as SW1 and located on the TMS320-SCSI board allow selection of the SCSI IDs as listed in Table 5–1.

Table 5–1. SCSI IDs for Switch Bank Positions

SCSI ID	SW1–3	SW1–2	SW1–1
7	Off	Off	Off
6	Off	Off	On
5	Off	On	Off
4	Off	On	On
3	On	Off	Off
2	On	Off	On
1	On	On	Off
0	On	On	On

Appendix A

Signals and Schematics

This appendix lists and describes the hardware signals, supplies the corresponding schematics, and includes the timings diagrams.

Topic **Page**

Hardware Signal Glossary **A-2**

TMS320-SCSI Target Controller Board Schematics **A-5**

Timing Diagrams **A-16**

A.1 Hardware Signal Glossary

Table A–1 describes the signals included in Figures A–1 through A–11.

Table A–1. Hardware Signal Glossary

Mnemonic	Description
Refer to Figure A–1.	
AB15–AB0	16 address lines generated by the TMS320C25 to address external data/program memory or I/O. AB15 (MSB), AB0 (LSB).
CLKOUT1	Clock signal generated by the TMS320C25 with frequency of CLKIN/4 used to clock external devices.
CLKOUT2	Clock signal generated by the TMS320C25 with frequency of CLKIN/4 used to clock external devices.
CLK20	Clock signal for the SCSI bus controller chip (SN75C091A).
CLK40	Clock signal for the TMS320C25 microprocessor.
DB15–DB0	16 data lines used to transfer data between the TMS320C25 and external data/program memory or I/O. DB15 (MSB), DB0 (LSB).
$\overline{\text{MPBIO}}$	Test condition input to the TMS320C25. Used by BIOZ instruction.
$\overline{\text{MPBR}}$	Bus request signal generated by the TMS320C25 for accessing global memory.
MPCLKR	Receive clock input. Used for serial port transfers.
MPCLKX	Transmit clock input. Used for serial port transfers.
MPDR	Serial data receive input. Used for serial port transfer.
$\overline{\text{MPDS}}$	Data select signal. Generated by the TMS320C25 for accessing the external data space.
MPDX	Serial data transmit output. Used for serial port transfer.
MPFSR	Frame synchronization pulse for receive input. Used for serial port transfers.
$\overline{\text{MPFSX}}$	Frame synchronization pulse for transmit input/output. Used for serial port transfers.
$\overline{\text{MPHOLD}}$	Hold input. Causes the TMS320C25 to place address, data, and control lines in the high impedance state.
$\overline{\text{MPHOLDA}}$	Hold Acknowledge. Acknowledgement of hold by the TMS320C25.
$\overline{\text{MPIACK}}$	Interrupt Acknowledge. Indicates receipt of an interrupt by the TMS320C25.
$\overline{\text{MPINT0}}$	External interrupt level 0. Input to the TMS320C25 indicating an external interrupt.
$\overline{\text{MPINT1}}$	External interrupt level 1. Input to the TMS320C25 indicating an external interrupt.
$\overline{\text{MPINT2}}$	External interrupt level 2. Input to the TMS320C25 indicating an external interrupt.
$\overline{\text{MPIS}}$	I/O select signal. Generated by the TMS320C25 for accessing I/O space.
$\overline{\text{MPMSC}}$	Microstate complete. Generated by the TMS320C25 to indicate memory access completion.
$\overline{\text{MPPS}}$	Program select signal. Generated by the TMS320C25 for accessing the external program space.
$\overline{\text{MPRS}}$	Reset signal. Input to the TMS320C25 to force program execution to location zero.
$\overline{\text{MPRW}}$	Read/write signal. Generated by the TMS320C25 to indicate the direction of data when communicating with external devices.
$\overline{\text{MPSTRB}}$	Strobe signal. Generated by the TMS320C25 to indicate an external access.
$\overline{\text{MPSYNC}}$	Synchronization signal. Input to the TMS320C25 allows synchronization of multiple TMS320C25.
$\overline{\text{RRST}}$	Receive reset. SCSI reset signal.

Table A–1. Hardware Signal Glossary (Continued)

Mnemonic	Description
Refer to Figure A–2.	
MPREADY	Ready signal. Input to the TMS320C25 indicating that an external device is ready for the I/O transaction to complete.
PSPRM	Program selecting the PROM. Indicates that the TMS320C25 instruction is being fetched from the PROMs.
Refer to Figure A–3.	
CAS1:–CAS0	Column address select. Generated by the 74ACT4503 to select column bank-1 or column bank-0 of the dynamic RAMs.
DMRDY	Dynamic RAM ready. Indicates the dynamic RAM access will be completed.
DRA9–DRA0	10 dynamic RAM address lines. Multiplexed row and column address lines.
DSDRM	Data select dynamic RAMs. Indicates that dynamic RAMs are selected for I/O transaction.
RAS3:–RAS0	Row address select. Generated by the 74ACT4503 to select one of the four banks.
WAH	Write address high. Provides clocking of the extended address lines for the dynamic RAMs.
W3:–W0	Write pulse. Select one of the four banks of the dynamic RAMS for the write operation.
For Figure A–4, refer to previous pages for signal descriptions.	
Refer to Figure A–5.	
DSSRM	Data select scratch RAM. Selects scratch RAM for the I/O transaction.
ISSBC	I/O select SCSI bus controller. Selects the 75C091A for the I/O transaction.
PSRML	Program select RAM lower. Selects lower 32K of downloadable RAM for the TMS320C25 instruction fetch.
PSRMU	Program select RAM upper. Selects the upper 32K of downloadable RAM for the TMS320C25 instruction fetch.
RSTRB	Read strobe. Indicates that the TMS320C25 is reading from an external device.
WSTRB	Write strobe. Indicates that the TMS320C25 is writing to an external device.
XDSRM	XDS RAM. Indicates the XDS emulator is the instruction source for the TMS320C25.
Refer to Figure A–6.	
CLRINT1	Clear interrupt level 1. Clears the interrupt flip-flop for external interrupt level 1. This signal is generated by the firmware routine that services this interrupt.
FAF	FIFO almost full/empty. Indicates there are 8 or more bytes available inside the FIFO to be read or 8 or more empty locations are available inside the FIFO to be written.
FHF	FIFO half full/empty. Indicates there are 32 or more bytes available inside the FIFO to be read or 32 or more bytes can be written to the FIFO.
FIR	FIFO input ready. Indicates FIFO is ready to receive a byte.
FOR	FIFO output ready. Indicates FIFO has a byte ready to be read.
RCND1	Read condition 1. Enables the external test conditions to be read as an in port.
RFIFO	Read FIFO. Generated by firmware to read a byte from SCSI FIFO.
RSBCD	Read SCSI bus controller data. Generated by firmware to read the SCSI bus controller register addressed by (SBA4–SBA0).
WFIFC	Write FIFO control. Generated by firmware to write the FIFO control register.

Table A–1. Hardware Signal Glossary (Concluded)

Mnemonic	Description
WFL	Write FIFO lower. Writes the SCSI FIFO from the lower byte of the data bus.
WFU	Write FIFO upper. Writes the SCSI FIFO from the upper byte of the data bus.
WINTCNT	Write interrupt counter. Writes the 16-bit external interrupt counter from the data bus.
WSBCA	Write SCSI bus controller address. Writes SCSI bus controller register address.
WSBCD	Write SCSI bus controller data. Writes the SCSI bus controller register that is addressed by (SBCA4–SBCA0).
Refer to Figure A–7.	
CLRF	Clear FIFO. Firmware clears the FIFO by first writing a 0 and then a 1 in this bit position of the FIFO control register.
DRS	Drive SCSI data. Enables the FIFO onto the DMA channel of the SCSI bus controller.
ENRQ	Enable request. Enables receiving of the request for data transfers from the SCSI bus controller.
FSI	FIFO shift in. Causes a byte to be written into the FIFO.
FSO	FIFO shift out. Causes a byte to be read from the FIFO.
SBCA4–SBCA0	SCSI bus controller address (4–0). Addresses the SCSI bus controller registers.
SBCACK	SCSI bus controller acknowledge. Acknowledgement of request from the SCSI bus controller.
SBCRQ	SCSI bus controller request. Request from the SCSI bus controller to handshake for data.
SBCWAIT	SCSI bus controller wait. Causes the data transfer to wait when in demand mode.
SBCMRD	SCSI bus controller memory read. Indicates a DMA read operation from the SCSI bus controller.
SBCMWT	SCSI bus controller memory write. Indicates a DMA write operation to the SCSI bus controller.
SRD	SCSI read. Enables the SCSI bus controller DMA channel into the FIFO input bus.
SWTL	SCSI write lower. Enables the lower byte of the data bus into the FIFO input bus.
SWTU	SCSI write upper. Enables the upper byte of the data bus into the FIFO input bus.
Refer to Figure A–8.	
FI7–FI0	FIFO input bus lines. Used during SCSI data transfers.
FO7–FO0	FIFO output bus lines. Used during SCSI data transfers.
Refer to Figure A–9.	
ACK	Acknowledge. SCSI bus signal.
ATN	Attention. SCSI bus signal.
BSY	Busy. SCSI bus signal.
C/D	Control/Data. SCSI bus signal.
I/O	Input/Output. SCSI bus signal.
MSG	Message. SCSI bus signal.
SB7:–SB0	SCSI bus data lines.
SBP	SCSI data bus parity. SCSI bus signal.
SEL	Select. SCSI bus signal.
RST	Reset. SCSI bus signal.
For Figure A–10 and Figure A–11, refer to previous pages for signal descriptions.	

A.2 TMS320-SCSI Target Controller Board Schematics

Figure A–1. Clock Reset and Power Schematics

Figure A–2. Microprocessor-Instruction Memory (TMS320C25 Program-Memory Interface)

For timing diagram, reference Figure A–12.

Figure A–3. Dynamic RAM Control Logic

Figure A–4. Data Buffer

Figure A–5. Memory-Select Decoder and Scratch RAM Interface

For timing diagram, reference Figure A–13.

Figure A–6. I/O-Port Decoder and Test-Condition Logic

For timing diagram, reference Figure A–14.

Figure A–7. FIFO Controller

For timing diagram, reference Figure A–14.

Figure A–8. SCSI DMA FIFO

For timing diagram, reference Figure A–14.

Figure A–9. SCSI-Bus Controller Single-Ended Interface

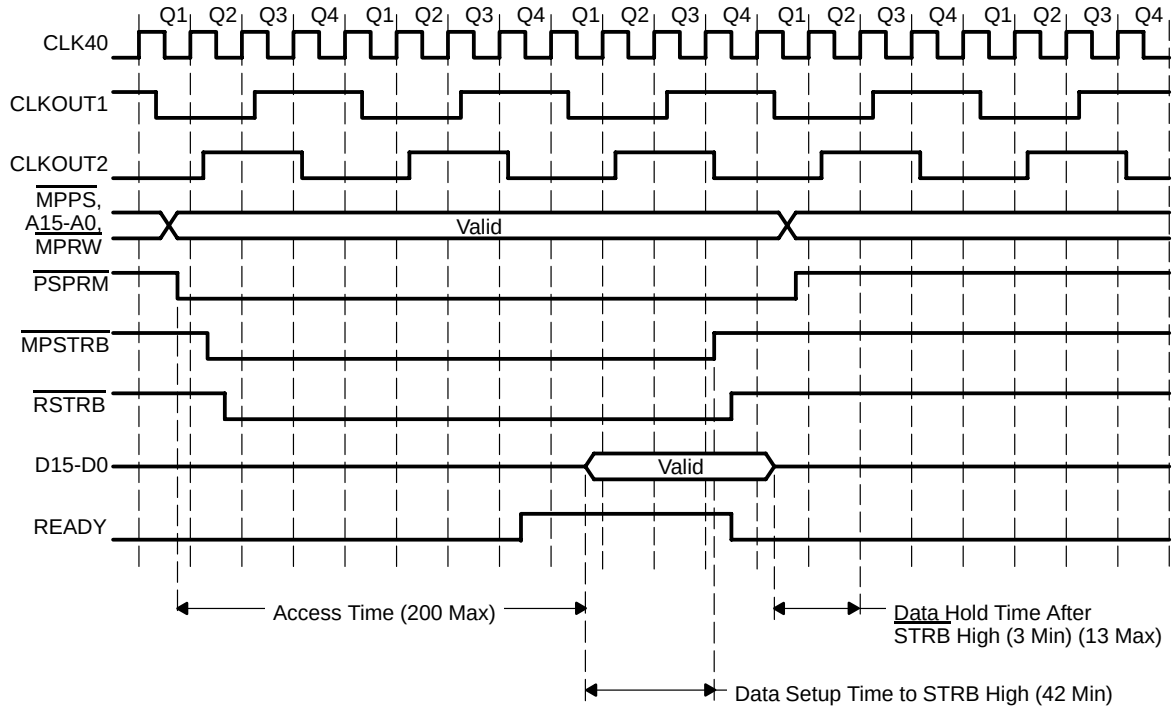
For timing diagram, reference Figure A–16 and Figure A–17.

Figure A–10. Zero-Wait-State Instruction Data RAM

Figure A–11. Programmable Interrupt Generator

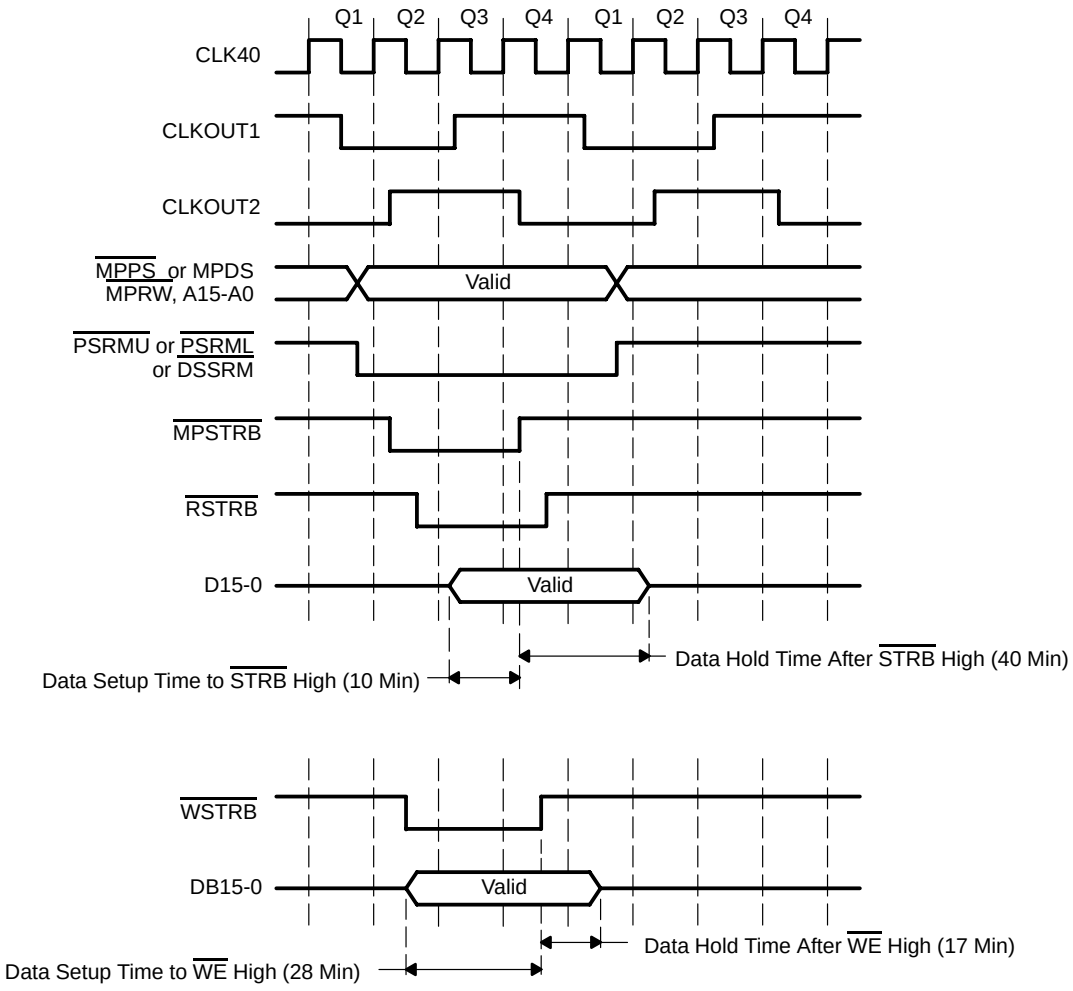
A.3 Timing Diagrams

Figure A–12. Program-Memory Interface Timing (TMS320C25 Timing With TMS27512-20)



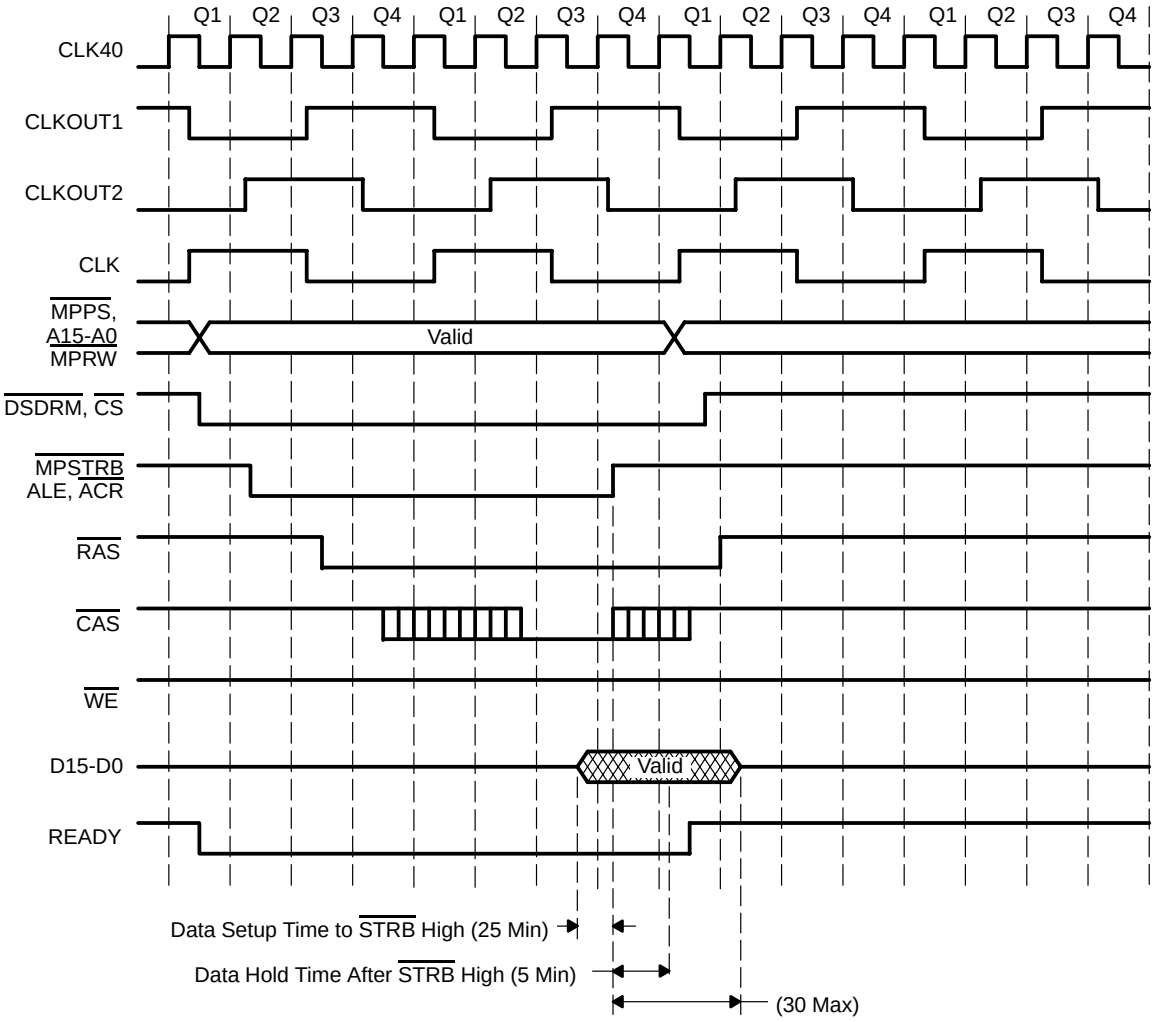
For schematic, reference Figure A–2.

Figure A–13. Data RAM-Interface Timing (TMS320C25 Read/Write Timing With CY7C199-25)



For schematic, reference Figure A–5.

Figure A–14. Data Buffer Read Timing (Read Timing With 74ACT4503 and TM024EAD9–10)



For schematic, reference Figure A–7 and Figure A–8.

Figure A–15. Data Buffer Write Timing (TMS320C25 Write Timing With 74ACT4503 and TM024EAD9–10)

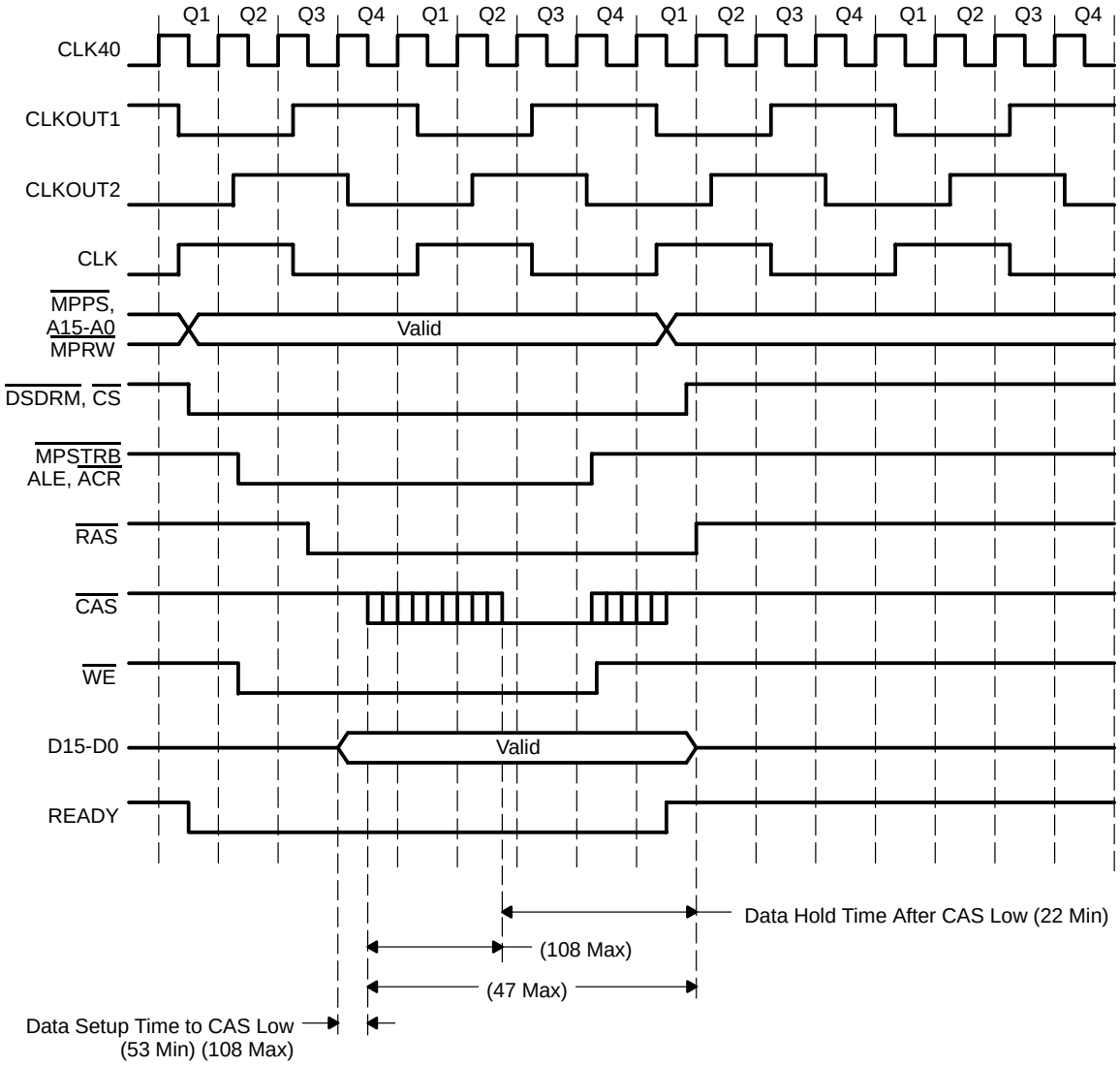
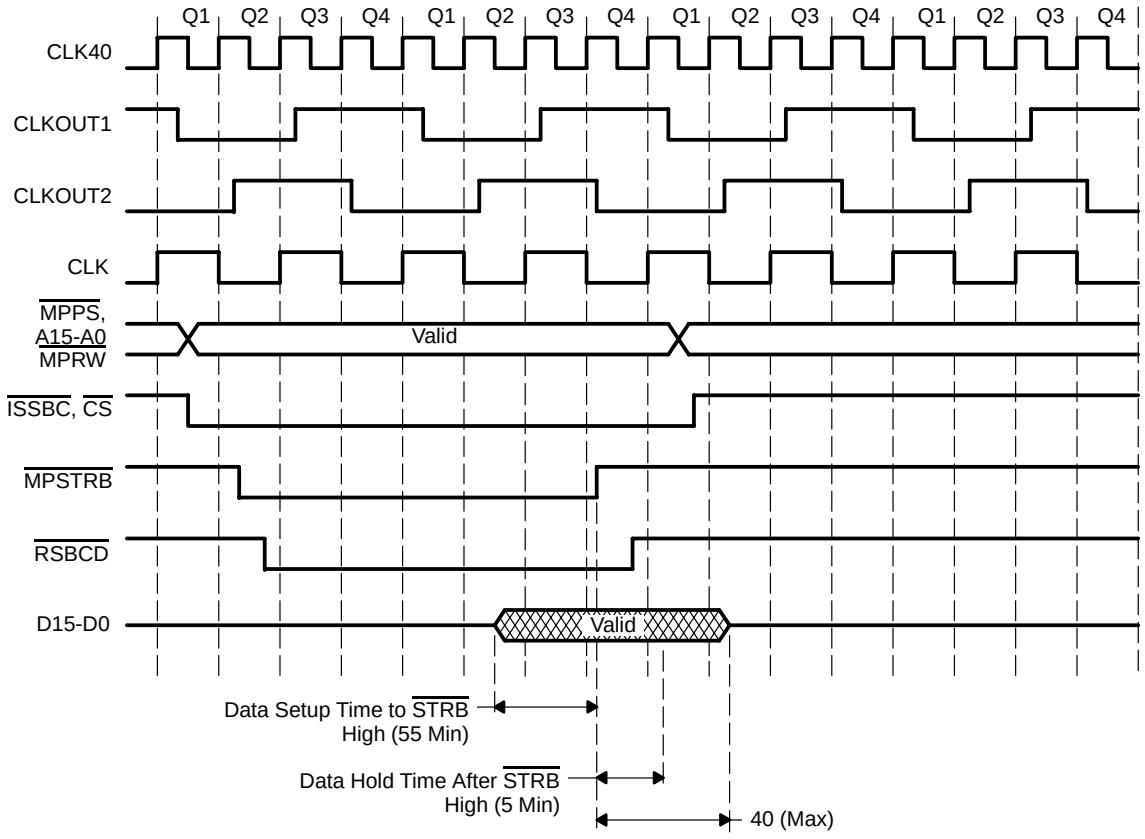
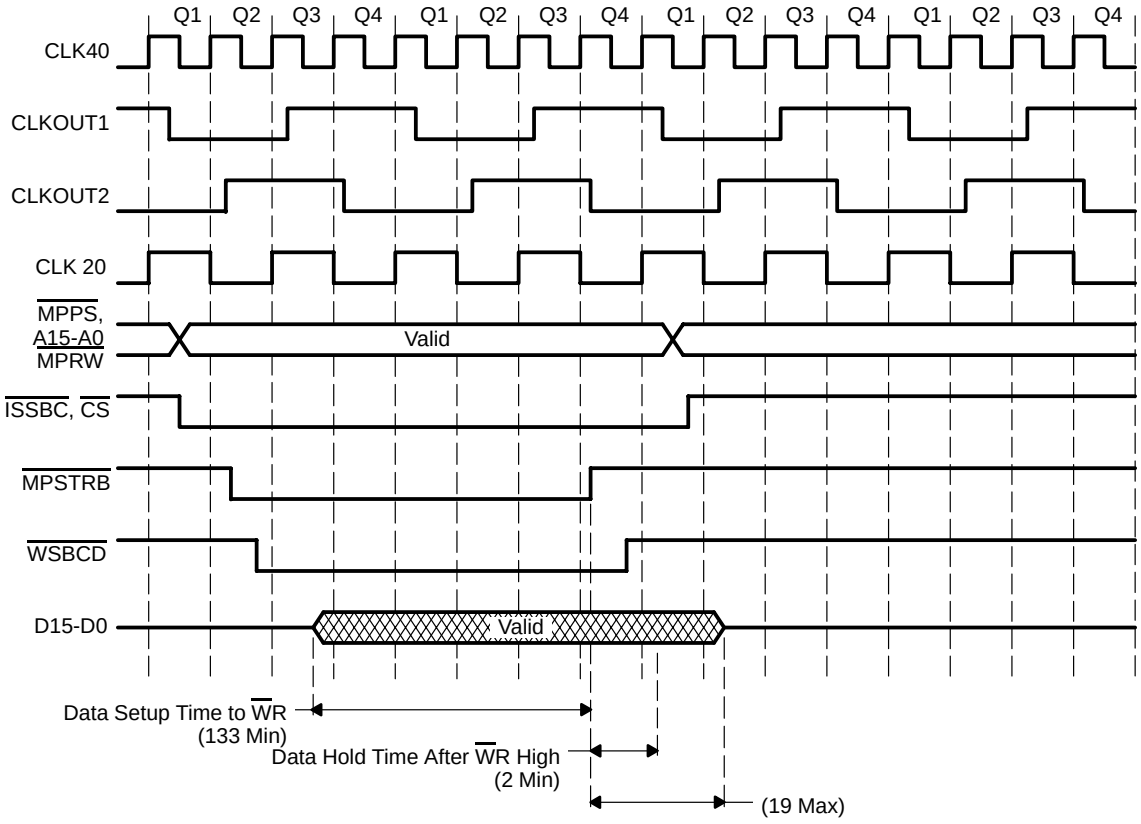


Figure A–16. SCSI-Bus Read Timing (TMS320C25 Read Timing With SN75C091A Register File)



For schematic, reference Figure A–9.

Figure A–17. SCSI-Bus Write Timing (TMS320C25 Write Timing With SN75C091A Register File)



For schematic, reference Figure A–9.

Appendix B

Equations and Pin Assignments

This chapter lists the memory-I/O decoder logic equations and pin assignments for U39 and the FIFO-control logic equations and pin assignments for U46 shown in the TMS320-SCSI Target Controller Board Schematics in Appendix A.

Topic **Page**

Memory/IO-Decoder Logic Equations and Pin Assignments (U39) B-2

FIFO-Control Logic Equations and Pin Assignments (U46) B-3

B.1 Memory/IO-Decoder Logic Equations and Pin Assignments (U39)

These are the select Pal equations:

$\overline{WSTRB} = \overline{MPSTRB}*\overline{MPRW}$

$\overline{RSTRB} = \overline{MPSTRB}*MPRW$

$\overline{PSRML} = \overline{MPPS}*\overline{XDSRM}*/A15$

$\overline{PSRMU} = \overline{MPPS}*\overline{XDSRM}*A15$

$\overline{DSDRM} = \overline{MPDS}*A15$

$\overline{ISSBC} = \overline{MPIS}*\overline{AB01}*\overline{AB2}*AB3$

$\overline{DSSRM} = \overline{MPDS}*\overline{A15}$

$\overline{PSPRM} = \overline{MPPS}*XDSRM$

Table B–1 lists the pin assignments for the TMS320 SCSI.

Table B–1. Memory/IO-Decoder Pin Assignments (U39)

Pin	Signal	Pin	Signal
1	MPDS	11	XDSRM
2	MPIS	12	PSPRM
3	AB01	13	DSSRM
4	MPPS	14	ISSBC
5	AB2	15	DSDRM
6	AB3	16	PSRMU
7	A15	17	PSRML
8	MPSTRB	18	RSTRB
9	MPRW	19	WSTRB
10	GND	20	VCC

B.2 FIFO-Control Logic Equations and Pin Assignments (U46)

$\overline{SBCACK}$ = SBCRQ*$\overline{SRD}$*SFIR*SBCMRD+
$\overline{SBCACK}$*SBCRQ+
$\overline{SBCACK}$*$\overline{SBCMRD}$+
SBCRQ*$\overline{SWTL}$*SFOR*SBCMWT+
SBCRQ*$\overline{SWTU}$*SFOR*SBCMWT+
$\overline{SBCACK}$*$\overline{SBCMWT}$

$\overline{SBCMRD}$ = SBCRQ*$\overline{SBCACK}$*$\overline{SRD}$+
$\overline{SBCMRD}$*SBCRQ

$\overline{SBCMWT}$ = SBCRQ*$\overline{SBCACK}$*$\overline{SWTL}$+
SBCRQ*$\overline{SBCACK}$*$\overline{SWTU}$+
$\overline{SBCMRD}$*SBCRQ

$\overline{FSI}$ = WFU*WFL*SRD+
WFU*WFL*$\overline{SBCRQ}$+
WFU*WFL*$\overline{ENRQ}$+
WFU*WFL*SBCACK

$\overline{SOCK}$ = $\overline{SWTL}$*$\overline{SBCACK}$*ENRQ+
$\overline{SWTU}$*$\overline{SBCACK}$*ENRQ+
$\overline{RFIFO}$

Table B–2 lists the pin assignments for the FIFO control logic.

Table B–2. FIFO-Control Pin Assignments (U46)

Pin	Signal	Pin	Signal
1	SFIR	11	WFU
2	SFOR	12	SOCK
3	ENRQ	13	FSI
4	NC	14	SBCACK
5	RFIFO	15	SBCMWT
6	SRD	16	SBCMRD
7	SBCWAIT	17	SWTU
8	SBCRQ	18	SWTL
9	WFL	19	NC
10	GND	20	VCC

Appendix C

Glossary

C

command descriptor block (CDB): A command block that is used to pass requests from an initiator to a target.

connect: A function used by an initiator to select a target to initiate an action.

D

disconnect: A function to cause a target to release the SCSI bus control (the SCSI bus is placed into the BUS FREE phase).

I

initiator: A SCSI device (usually a host computer) that requests another SCSI device to take an action.

L

logical unit: A physical or virtual device that is addressable through a target.

logical unit number (LUN): A 3-bit code for a logical unit.

R

reconnect: A function used by a target to select an initiator to resume processing after it has been disconnected.

reserved (R): A term used for a bit-, byte-, field-, or coded-value that is set aside for future SCSI standardization.

SCSI address: A unique address (0–7) assigned to a SCSI device.

SCSI device: A host computer, peripheral controller, or intelligent peripheral unit that is connected to the SCSI bus.

SCSI ID: The bit-significant representation of a SCSI address (this bit address is associated with a bit number of the data bus).

status: One byte of information sent from a target to an initiator on completion of a command.

T

target: A SCSI device (usually a controller) that takes the actions requested by an initiator.

V

vendor unique (VU): A bit-, field-, or coded-value that can be uniquely defined for each vendor or device.

Index

A

ABORT, messages, 2-15
ACK signal, 2-12
acknowledge signal, 2-12
addressing method, 2-3
ANSI X3.131–1986, vi
arbitration and selection controller, 3-14
arbitration phase, 2-7
architecture
  disk-drive, iv
  DSP-based servo, v
  SN75C091A, 3-12
  system interface, v
asynchronous, 2-4
ATN signal, 2-12
ATTENTION condition, 2-13
attention signal, 2-12

B

benefits of TMS320 DSPs, 1-1
block diagram, TMS320C25, 1-5
BSY signal, 2-11
bus
  phases, 2-7–2-8
  protocol, 2-4
BUS DEVICE RESET message, 2-16
bus-free phase, 2-7
busy signal, 2-11
BUSY status phase, 2-18
byte stack control, 3-15

C

C/D signal, 2-11
CHECK CONDITION status phase, 2-18
CMOS, 1-3, 1-6
CODEC-compatible serial port, 1-3
command
  buffer initiator, 4-4, 4-5, 4-6, 4-7, 4-8
  buffers field descriptions, 4-3
  end status, 4-17
  example, SCSI, 2-9–2-10
  execution routines, 4-18
  phase, 2-7
  process, 4-8–4-17
  READ, 2-10
  reception, 4-3
COMMAND COMPLETE, 2-14
command-execution pointers, 4-13, 4-14, 4-15, 4-16, 4-17
commands
  FORMAT UNIT, B-28
  INQUIRY, B-35
  MODE SELECT, B-39, B-40, B-41, B-42, B-43, B-44, B-45, B-46, B-47, B-48, B-49, B-50, B-51, B-52
  MODE SENSE, B-57, B-58, B-59
  READ, B-32
  READ (EXTENDED), B-66
  READ BUFFER, B-72
  READ CAPACITY, B-64, B-65
  READ LONG, B-74
  REASSIGN BLOCK, B-30
  RELEASE UNIT, B-55
  REQUEST SENSE, B-22
  RESERVE UNIT, B-53
  REZERO UNIT, B-21
  SEEK, B-34
  SEEK (EXTENDED), B-68
  START/STOP UNIT, B-63

TEST UNIT READY, B-20
WRITE, B-33
WRITE (EXTENDED), B-67
WRITE AND VERIFY, B-69
WRITE BUFFER, B-70
WRITE LONG, B-75
communication protocol, 2-7
condition
ATTENTION, 2-13
RESET, 2-13
configuration
jumpers, 5-4
switches, 5-4
connectors, 5-3
control/data signal, 2-11
CRC, telephone number, vii
Customer Response Center, vii

D

data
buffer, 3-8
buffer module, 3-3
bus parity signal, 2-11
phase, 2-7
signals, 2-11
transfer, 2-4
development environment, 1-1
development support, 1-1
TMS320, 1-9
device independence, 2-1
differential interface, 2-3
disconnect example, 2-9
DISCONNECT messages, 2-15
disk-drive architecture, iv
DMA interface, 3-15
DMA port, 3-10
DSP processors, overview, 1-1
DSP-based servo architecture, v

E

EPROM, 1-3
equations, B-1–B-4
EXTENDED MESSAGE SYNCHRONOUS DATA TRANSFER REQUEST, 2-14
extended sense data format, B-23
external test condition module, 3-3

F

field descriptions, command buffers, 4-3
FIFO memory, 3-8
FIFO-control logic equations, B-3
FIFO-control pin assignments, B-3
firmware
design overview, 4-2
operation, 4-3–4-17
routines, initialize the data buffer, 4-21
theory of operations, 4-1–4-26
fixed-point DSPs, 1-2, 1-6
floating-point DSPs, 1-2, 1-7–1-8
FORMAT UNIT, B-28

G

GOOD status phase, 2-18

H

handshaking methods, 2-4
hardware
development system, 1-1, 1-4
signal glossary, A-2, A-3, A-4
theory of operations, 3-1–3-16
TMS320-SCSI, 3-5
host adapter, 2-5–2-6
host memory, 2-5–2-6
hotline, vii
how to use this manual, vi

I

I/O port decode logic module, 3-3
I/O signal, 2-11
IDENTIFY
message functions, 2-17
messages, 2-16
if you need assistance, vii
indicators, 5-5
bit positions, 5-5
information transfer routines, 4-20

initialize data buffer, firmware routines, 4-21
INITIATOR DETECTED ERROR messages, 2-15
initiators, 2-4
  request-sense data, 4-9, 4-10, 4-11
  SCSI command buffers, 4-4, 4-5, 4-6, 4-7, 4-8
input/output signal, 2-11
INQUIRY, B-35
inquiry data, B-38
instruction memory, 3-6
  module, 3-3
interface, 2-3
  differential, 2-3
  single-ended, 2-3
interfacing TMS320-SCSI with host adaptors, 5-6
interrupt control, 3-14

J

J1 connector, 5-3
J2 connector, 5-3
J3 connector, 5-3
jumpers, 5-4

L

LED register, 3-11
logical, connection, 2-6

M

memory/IO-decoder
  logic equations, B-2
  pin assignments, B-2
message, COMMAND COMPLETE, 2-10
MESSAGE PARITY ERROR messages, 2-16
message phase, 2-7
MESSAGE REJECT message, 2-15
message signal, 2-11
messages, 2-14–2-17
  BUS DEVICE RESET, 2-16
  COMMAND COMPLETE, 2-14
  DISCONNECT, 2-15
  EXTENDED MESSAGE SYNCHRONOUS DATA TRANSFER REQUEST, 2-14
  IDENTIFY, 2-16
  INITIATOR DETECTED ERROR, 2-15
  MESSAGE PARITY ERROR, 2-16
  MESSAGE REJECT, 2-15
  NO OPERATION, 2-16
  RESTORE POINTERS, 2-14
  SAVE DATA POINTER, 2-14
  supported by target controller, 2-14
microprocessor
  interface, 3-14
  module, 3-3
  port, 3-10
  scratch RAM, 3-7
MODE SELECT, B-39, B-40, B-41, B-42, B-43, B-44, B-45, B-46, B-47, B-48, B-49, B-50, B-51, B-52
MODE SENSE, B-57, B-58, B-59
mode-sense
  buffer for XMIT, B-59
  data, B-60
MSG signal, 2-11
multiple hosts, 2-4

N

NO OPERATION message, 2-16

O

overview
  firmware design, 4-2
  TMS320 DSP processors, 1-1

P

page codes, B-60, B-61, B-62
page formats, B-60, B-61, B-62
parity generators/checkers, 3-15
peripherals, 2-4
phase
  arbitration, 2-7
  bus-free, 2-7
  command, 2-7
  data, 2-7
  message, 2-7
  reselection, 2-7
  selection, 2-7
  status, 2-7

phase sequences, 2-8
  not using the arbitration phase, 2-8
  using the arbitration phase, 2-8
phases, status, 2-18
physical
  characteristics, 5-1
  connection, 2-6
  layout, 5-2
pin assignments, B-1–B-4
prior to initial selection, 2-6
programmable interrupt generator, 3-11
protocol, communication, 2-7

R

RAM, 1-3
READ, B-32
  command example with disconnect, 4-22
  command example without disconnect, 4-22
  command with disconnect, 2-10
  command without disconnect, 2-9
READ (EXTENDED), B-66
READ BUFFER, B-72
READ CAPACITY, B-64, B-65
READ LONG, B-74
Read/Write strobe decode logic, 3-5
REASSIGN BLOCK, B-30
receive FIFOs, 3-14
register file, 3-14
related documentation, vii
  TMS320 Family Development Support Reference Guide, vii
  TMS320 Third-Party Support Reference Guide, vii
RELEASE UNIT, B-55
REQ signal, 2-12
REQ/ACK handshake controller, 3-13–3-16
REQUEST SENSE command, B-22
request signal, 2-12
request-sense data for initiator, 4-9, 4-10, 4-11
reselection phase, 2-7
RESERVATION CONFLICT status phase, 2-18
RESERVE UNIT, B-53
RESET condition, 2-13
reset signal, 2-12
REZERO UNIT, B-21
ROM, 1-3
RST signal, 2-12

S

SAVE DATA POINTER, 2-14
schematics, A-1–A-4, A1-5–A1-15
  clock reset and power, A1-5
  data buffer, A1-8
  dynamic RAM control logic, A1-7
  FIFO controller, A1-11
  I/O-port decoder and test-condition logic, A1-10
  memory-select decoder and scratch RAM interface, A1-9
  microprocessor-instruction memory, A1-6
  programmable interrupt generator, A1-15
  SCSI DMA FIFO, A1-12
  SCSI-bus controller single-ended interface, A1-13
  zero-wait-state instruction data RAM, A1-14
scratch RAM, 3-7
scratch-RAM module, 3-3
SCSI
  addressing method, 2-3
  bus, 2-4
  bus conditions, 2-13
  bus interface controller, 3-10
  bus interface controller module, 3-4
  bus phases, iii
  bus signal descriptions, 2-11–2-12
  command, iii
  command example, 2-9–2-10
  command execution, 2-4
  commands, 2-19
  controller, 2-5–2-6
  definition, iii
  DMA FIFO module, 3-3
  host adapter, 2-5
  IDs for switch bank positions, 5-6
  information transfer routines, 4-20
  interface, 2-3
  overview, 2-1–2-19
  sample configurations, 2-2
  specifications, 2-1–2-19
  system diagram, 2-4
  target controller, iii, 2-1
SCSI-Sense Data-Buffer management routines, 4-21
SEEK, B-34

SEEK (EXTENDED), B-68
SEL signal, 2-11
select signal, 2-11
selection phase, 2-7
sense codes of the REQUEST SENSE command, B-25
sense key descriptions, B-24
servo controller, iii
  TMS320C25, 1-3
signal
  acknowledge, 2-12
  attention, 2-12
  request, 2-12
  reset, 2-12
signals, A-1–A-4
single command example, 2-9
single-ended interface, 2-3
SN75C091A
  architecture, 3-12–3-16
  command sequencer, 3-13–3-16
  features, 3-13–3-16
  functional block architecture, 3-12–3-16
  overview, 3-12–3-16
software development system, 1-1, 1-4
START/STOP UNIT, B-63
status, phase, 2-7
status codes, 2-18
status phase, 2-18
  BUSY, 2-18
  CHECK CONDITION, 2-18
  GOOD, 2-18
  RESERVATION CONFLICT, 2-18
switches, 5-4
synchronous, 2-4
system initialization routines, 4-19

T

target controller, 2-4
  TMS320-SCSI, 2-1
target devices, 2-4
target-controller command overhead example, 4-22–4-26
target-controller, printed circuit board, 5-2
technology, CMOS, 1-6
test condition logic, 3-10
TEST UNIT READY command, B-20
theory of operations, firmware, 4-1–4-26
third-party companies, 1-1
TI standards, 1-1
time delays, 4-24
timing diagrams, A1-16–A1-22
TMS320
  development support, 1-9
  family of devices, 1-2
TMS320 Third-Party Support Reference Guide, 1-1
TMS320C1x, 1-6
TMS320C25, iii, 1-1
  block diagram, 1-5
  burst-transfer rate, 1-3
  development systems, 1-3
  DSPs, 1-3–1-5
  hardware development system, 1-4
  hardware stack, 1-3
  performance, 1-3
  servo controller, 1-3
  software development systems, 1-4
TMS320C2x, 1-6
TMS320C3x, 1-7
TMS320C4x, 1-8
TMS320C5x, 1-6
TMS320-SCSI
  functional modules, description, 3-3
  hardware, 3-5–3-11
  hardware block diagram, 3-5
  physical characteristics, 5-1–5-6
  target controller, description, 3-2
  target controller board schematics, A1-5
  target controller time delays, 4-24
  target-controller printed circuit board, 5-2
transmit FIFOs, 3-14

U

UNIT ATTENTION condition, 2-13

W

WRITE, B-33
WRITE (EXTENDED), B-67
WRITE AND VERIFY, B-69
WRITE BUFFER, B-70
WRITE LONG, B-75

Appendix D

Title

Topic	Page

D.1 Installation

Use a standard 50-pin SCSI cable to connect the SCSI connector on the TMS320-HP labeled as J2 to the 50-pin SCSI connector located on your SCSI host adaptor board. Make sure that pin 1 of either of the connectors matches pin 1 marked on the cable.

The jumper labeled as W13 located next to the SCSI connector on the TMS320-HP board supplies the SCSI bus terminator power. This jumper needs to be installed if your host adaptor does not supply terminator power; otherwise, it could be removed.

The switch bank positions 1, 2, and 3 labeled as SW1 and located on the TMS320-HP board allow selection of the SCSI IDs as listed in NO TAG.

Table D–1. SCSI IDs for Switch Bank Positions

SCSI ID	SW1–3	SW1–2	SW1–1
7	Off	Off	Off
6	Off	Off	On
5	Off	On	Off
4	Off	On	On
3	On	Off	Off
2	On	Off	On
1	On	On	Off
0	On	On	On

D.2

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software 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, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

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

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI 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.