PROGRAMMING GUIDE

LSI53C040 Enclosure Services Processor

Version 1.1

December 2000

LSI LOGIC®

S14021.A

Document DB15-000100-01, Second Edition (December 2000)
This document describes the LSI Logic LSI53C040 Enclosure Services Processor and will remain the official reference source for this product. This guide is intended for use with the SAF-TE Firmware C1 source code release and meets the criteria set within the LSI Logic Software Release Procedure.

To receive product literature, visit us at http://www.lsilogic.com.

Preface

This programming guide assists experienced firmware developers who wish to customize the LSI53C040 SAF-TE Firmware for specialized enclosure management applications, or to develop firmware using the LSI53C040 firmware architecture as a basis. It assumes a thorough knowledge of the "C" programming language and the SAF-TE specification, and understanding of the components and services that will be provided in the enclosure design. For more background material that may provide information on these subjects, refer to the Section "References," and Section "Related Publications," in this document.

This guide contains general considerations for developers who are designing or customizing firmware and includes:

- describing the SCSI portion of the LSI53C040 firmware, including the SCSI commands supported,
- describing the two-wire serial portion of the LSI53C040 firmware,
- describing the LSI53C040 implementation of the firmware for the binary inputs/outputs, and of the LED outputs that provide enclosure monitoring and notification services to the host,
- giving information on the LSI Logic implementation of the SAF-TE command set, and
- describing the firmware configuration utility and the data structures that hold the specific information on the components and services in the enclosure.

Audience

This document was prepared for firmware developers who plan on customizing the SAF-TE firmware for enclosure management applications.

Organization

This document has the following chapters and appendix:

- Chapter 1, **Using the Programming Guide**, provides details about the conventions used in this guide, describes the SAF-TE firmware, and lists the features.
- Chapter 2, **General Design Considerations**, provides design information for firmware developers.
- Chapter 3, **SAF-TE Source Code**, discusses the Boot Module, Configuration Module, and SAF-TE Module that contains the safte.c source code.
- Chapter 4, **SAF-TE Command Implementation**, lists the SAF-TE commands supported by the LSI53C040 chip.
- Chapter 5, **Configuration Data and the Configuration Utility**, provides the potential questions that may be asked while running the configuration utility.
- Appendix A, **LSI53C040 Board Utilities**, discusses the data transfer types.

Related Publications

LSI53C040 Enclosure Services Processor Technical Manual, Order Number S14042.A

LSI53C040 Enclosure Services Processor SAF-TE Firmware User's Guide, Order Number S14004.A

References

Yeralan, Sencer and Ashutosh Ahluwalia. 1997. *Programming and Interfacing the 8051 Microcontroller.*

Ayala, Kenneth J. 1997. *The 8051 Microcontroller: Architecture, Programming, & Applications.* Second Edition.

Scope of this Programming Guide

This programming guide only discusses the firmware implementation of the SAF-TE protocol, even though the LSI53C040 device could support the SES protocol. LSI Logic will not be pursuing a SES firmware implementation.

Conventions Used in This Manual

Hexadecimal numbers are indicated by the prefix “0x” —for example, 0x32CF. Binary numbers are indicated by the prefix “0b” —for example, 0b0011.0010.1100.1111.

This manual makes frequent references to the source code of the LSI53C040 SAF-TE firmware. We use routine names, typed in **bold**, *italics*, and regular text, to indicate the specific section of code being referenced.

Revision Record

Revision	Date	Remarks
1.0	5/99	First Edition.
1.1	12/00	Product names changed from SYM to LSI.

Contents

Figures

Tables

Chapter 1
Using the Programming Guide

This chapter provides a general overview of the LSI53C040 Enclosure Services Processor Firmware and includes these topics:

1.1 Introduction

The LSI53C040 is an enclosure services processor with 28 programmable, multipurpose I/O (MPIO) pins for enclosure monitoring and 24 programmable, multipurpose I/O pins for visual LED indicators. The LSI53C040 firmware includes configuration data tables that allow the user to map specific monitoring functions to each of these pins, so that the firmware can be adapted to any enclosure environment.

The LSI53C040 uses the SAF-TE or SES protocol to detect drive presence, condition a slot for drive insertion or removal, and monitor enclosure services. The fan, power supply, door lock, alarm, and slot drive power are examples of enclosure services.

1.2 General Description

The LSI53C040 SAF-TE firmware controls an 80C32 microcontroller core in the LSI53C040 device. This microcontroller is compatible with the Intel MCS51 family. It runs independently in interrupt mode.

The LSI53C040 SAF-TE firmware contains three major architectural components, each implementing a separate I/O interface to the chip. The SCSI block governs the SCSI interface and implementation of all the SCSI commands used to send data packets to the host. The Two-Wire Serial (TWS) interface bus is primarily used as an input bus. See Section 2.5, “TWS Bus,” page 2-9 for more detailed information.

1.3 Features

This section describes the LSI53C040 SAF-TE firmware capabilities and associated features, which include MPIO/MPLED Mapping, TWS Interface Peripheral Support, the SAF-TE Interface, and Programmable Enclosure Configuration monitoring.

1.3.1 MPIO/MPLED Mapping

- Ability to assign MPIO/MPLED pins to dedicated functions such as drive, power supply, and fan status monitoring
- Automatically configures MPIO/MPLED pins
- Generates listing of assigned pins
- Ability to modify blink patterns for LEDs

1.3.2 TWS Interface Peripheral Support

- National Semiconductor LM75 2-Wire Serial Digital Temperature Sensor and Thermal Watchdog
- Dallas Semiconductor DS1621 2-Wire Serial Digital Thermometer and Thermostat
- National Semiconductor LM78 fan, power supply, and temperature monitoring

1.3.3 SAF-TE Interface

The SAF-TE interface complies with the SAF-TE Specification R041497 and provides these features:

- Supports Read Buffer Commands
  - Read Enclosure Configuration
  - Read Enclosure Status
  - Read Device Slot Status
  - Read Global Flags
- Supports Write Buffer Commands
  - Write Device Slot Status
  - Performs Slot Operation
  - Send Global Commands
- Supports the Upload Firmware command
- Allows selection from one of 11 SCSI IDs (7–0, 15, 14, 13)
- Allows connection from any data line, bits 8 through 15, to any one of the SHID[2:0] (SCSI High ID) pins on the LSI53C040
- Supports Slot Power Control Option
- Supports Over Temperature LED Option

1.3.4 Programmable Enclosure Configuration Monitoring

The enclosure configuration monitoring allows:

- Up to 14 device slots
- Up to 6 fans and power supplies (single or dual input status)
- Up to 15 binary temperature sensors (single input status)
- Up to 4 temperature sensors (TWS)
- Optional Ready device for use (slot power control) and Prepare device for insertion/removal output signals
- Programmable Vendor, Product, and Enclosure ID
- Host Controllable Door Lock and Speaker Option
- Selection of one or two LED's per device slot
- Global LED's option for enclosure, drive, and array status

Chapter 2
General Design Considerations

This chapter provides design information for firmware developers and includes these topics:

2.1 8051/8032 Background

The user is encouraged to become familiar with the 8032 8-bit microcontroller prior to reading this guide. The primary differences between the 8051 and the 8032 are the internal RAM (256 bytes vs. 128 bytes for the 8051), also the 8032 has one additional timer. Several excellent resources are available to aid the user in becoming familiar with the 8032, including those listed in the Preface of this guide.

This section covers some of the key highlights of the 8032 architecture and some important Archimedes compiler features. The remainder of the chapter provides an overview of the LSI53C040 firmware and general design considerations necessary to use this firmware most effectively.

2.1.1 8032 Architecture Features

The architectural features of the 8032 microcontroller are key to the understanding and use of the LSI53C040 firmware. These features include: Section 2.1.1.1, “Register Banks,” page 2-2, Section 2.1.1.2, “Memory Areas,” page 2-2, and Section 2.1.1.3, “Special Function Registers,” page 2-4.

2.1.1.1 Register Banks

Four register banks that contain eight registers each in the 8032 reside in the lower 128 bytes of the internal RAM. See Figure 2.1 for an example of these register banks.

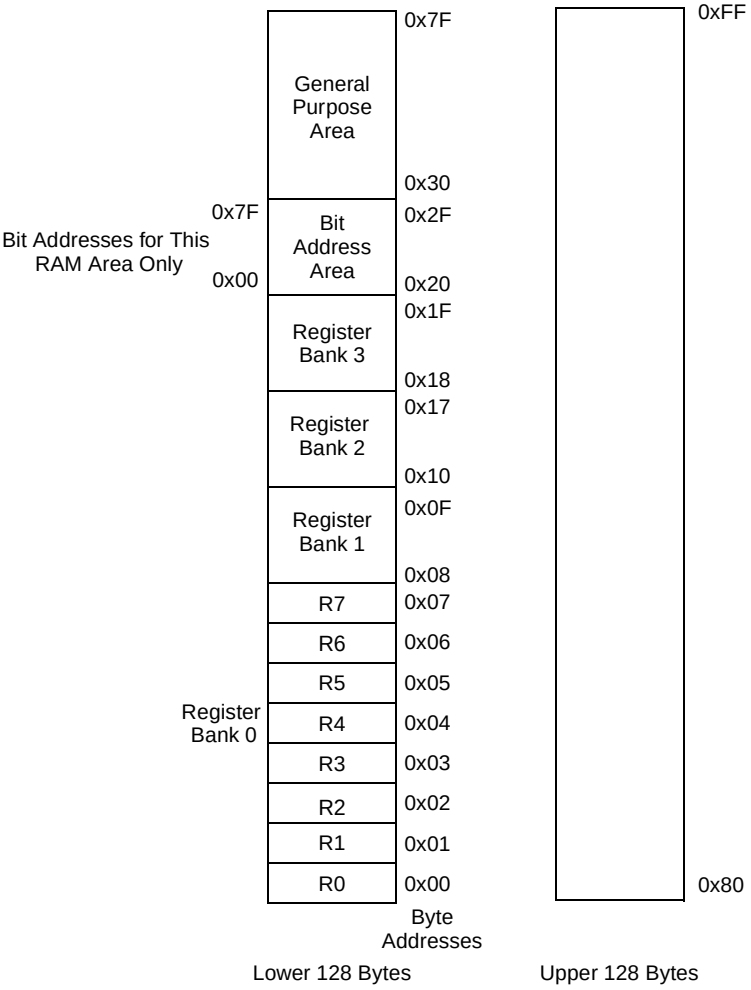
2.1.1.2 Memory Areas

The 8032 architecture supports a number of physically separate memory areas for program and data. Each memory area offers certain advantages and disadvantages. Refer to the *Intel 8-bit Embedded Controllers* databook or other 8051 reference material for more information about the 8032 memory architecture. The following sections briefly discuss program memory, the internal data memory, and the external data memory.

Program Memory – Since the 8032 is a ROMless variant of the 8051, an external 16 Kbytes memory is required to hold the program code. The LSI53C040 has the ability to automatically download this program code from a serial EEPROM over the TWS bus into the external 16 Kbytes memory space.

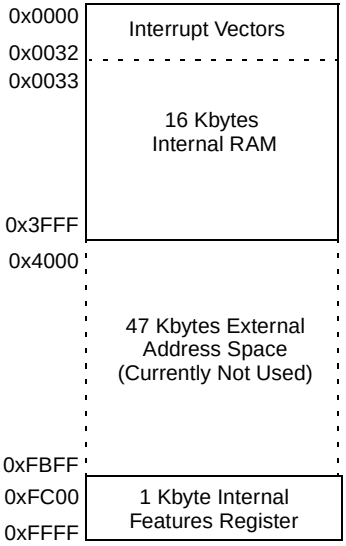
Internal Data Memory – The 8032 contains 256 bytes of internal data memory, which can be read and written. The first 128 bytes of internal data memory are both directly addressable and indirectly addressable. The upper 128 bytes of data memory (from 0x80 to 0xFF) can be addressed only indirectly. With indirect addressing, the referenced register contains the address of the register or cell that actually contains the data to be used. There is also a 16-byte area starting at 0x20 that is bit-addressable. See Figure 2.1 for more detailed information.

Figure 2.1 Internal RAM



External Data Memory Map – The address decode block in the LSI53C040 decodes addresses that are generated by the microcontroller. Additionally, it multiplexes memory space accesses between the different register and memory blocks according to the memory map. See Figure 2.2 for an example of this memory map.

Figure 2.2 LSI53C040 Memory Map



2.1.1.3 Special Function Registers

The 8032 provides a distinct memory area for accessing Special Function Registers (SFRs). SFRs are used in the program to control timers, counters, serial I/Os, port I/Os, and peripherals. SFRs reside from address 0x80 to 0xFF, and can be accessed by bits, bytes, and words. The special function registers are listed in Table 2.2.

2.1.2 Archimedes Compiler Features

The user should be familiar with several Archimedes compiler features that are related to the above 8032 features before reading the source code.

2.1.2.1 Control Directives

The four control directives are: AREGS/NOAREGS, REGISTERBANK, and SMALL.

The **AREGS** control causes the compiler to use absolute addressing for registers R0 through R7. Absolute addressing improves the efficiency of the generated code. **PUSH** and **POP** instructions function only with direct or absolute addresses. By using the **AREGS** directive, functions can directly push and pop registers.

The **NOAREGS** control disables absolute addressing for registers R0 through R7. Functions that are compiled with **NOAREGS** are not dependent on the register bank and may use all 8051 register banks. This directive may be used for functions that are called from other functions using different register banks.

The **REGISTERBANK** control selects the specific register bank to use for subsequent functions declared in the source file. Resulting code may use the absolute form of register access when the absolute register number can be computed. The **using** function attribute supersedes the effects of the **REGISTERBANK** directive.

The **SMALL** memory model is used to compile the firmware for performance reasons. This results in all function variables and local data segments being placed in the internal data memory of the 8032 by default. This results in smaller code size and faster execution. Since the internal memory size is very limited (256 bytes), larger or infrequently used variables may be declared to be in any of the 8032 memory ranges. Since this is the compiler's default memory model, a **#pragma small** is not needed in the source file, and **SMALL** does not have to be on the C51 command line.

2.1.2.2 Compiler Keywords

A list of compiler keywords are:

_at_ variables may be located at absolute memory locations in the C program source modules using the **_at_** keyword.

sbit the **sbit** data type allows the user to access bit-addressable SFRs.

data directly accessible internal data memory; fastest access to variables (128 bytes).

bdata bit-addressable internal data memory; allows mixed bit and byte access (16 bytes).

idata indirectly addressable internal data memory; accessed across the full internal address space (256 bytes).

pdata paged (256 bytes) external data memory (16 Kbytes).

xdata external data memory (refers to any location in the full 64 Kbytes of external data memory).

code program memory (16 Kbytes).

sfr SFRs are declared in the same fashion as other variables. The only difference is that the data type specified is **sfr** rather than **char** or **int**.

interrupt an extension for standard C function declarations that indicates that the function is an interrupt function.

using specifies which register bank the function uses.

2.1.2.3 Generic Pointers vs. Memory-Specific Pointers

Due to the unique architecture of the 8051 and its derivatives, the compiler provides two different types of pointers: generic pointers and memory-specific pointers.

Generic pointers are always stored using three bytes. The first byte is for the memory type, the second is for the high-order byte of the offset, and the third is for the low-order byte of the offset. Generic pointers may be used to access any variable regardless of its location in 8051 memory space.

Memory-specific pointers always include a memory type specification in the pointer declaration and always refer to a specific memory area. Because the memory type is specified at compile-time, the memory type byte required by generic pointers is not needed by memory-specific pointers. Like generic pointers, the user may specify the memory area in which a memory-specific pointer is stored. Memory-specific pointers may be used to access variables in the declared 8051 memory area only. Memory-specific pointers provide the most efficient method of accessing data objects, but at the cost of reduced flexibility.

For more information, refer to your C compiler documentation, specifically the sections covering directives and language extensions.

Note: The user must be aware that this compiler does **not** generate re-entrant code, that function parameters are **not** passed on the stack, and that interrupt level routines and background routines **cannot** share subroutines. Virtually all of the source code is written in C. Assembly language is only needed for stack initialization before control is passed to the C code.

2.2 The LSI53C040 Firmware

The heart of the LSI53C040 architecture is its 52 input and output pins (28 MPIO pins and 24 MPLED pins). The user may specify (by using the configuration utility) an almost unlimited number of configurations. The SAF-TE firmware was designed to make these configurations simple to implement by using this utility. Additionally, the SAF-TE firmware is coupled with a software state machine that polls all of the binary inputs and drives all of the binary outputs. Simultaneously, the firmware maintains copies of all the necessary SAF-TE/SCSI command responses so that the SCSI interrupt software can return these responses to a host/initiator at any time.

This state machine design allows the user to specify all of the inputs and outputs without having to change the 8032 firmware. In addition, this design actually simplifies and shortens the size of the 8032 firmware, by moving much of the decision making process to the configuration utility. Consequently, more memory is available in the 8032 for user enhancements. Refer to Chapter 5 for more detailed information about configuration data and the configuration utility.

The program and data for this state machine design is passed from the configuration utility to the firmware by using a data structure called **config**. In addition, the **config** data structure also contains all the information gathered by the configuration utility, such as the SCSI ID, TWS devices to be supported, scanning intervals, etc.

The 8032 microcontroller is at the core of the LSI53C040 device and runs at 40 MHz. This microcontroller provides more than sufficient processing power to complete the work to be done by the LSI53C040. The environmental monitoring protocols, such as SAF-TE and SES, do

not require much SCSI traffic. Therefore, the LSI53C040 is an excellent choice for enclosure monitoring functions, although attention to certain software details is necessary to make effective use of its features.

2.3 SCSI and DMA

The most important of those details is the SCSI interface. The DMA engine in the LSI53C040 helps improve the overall speed of the SCSI interface, but that alone is not enough. Because this core is relatively slow, the software is designed to complete the SCSI work as quickly as possible.

The SCSI interface is done at interrupt level. The 8032 has 6 interrupt sources, but only two interrupt levels. SCSI is done at the lower interrupt level, while DMA is done at the higher interrupt level. This is necessary because the 8032 is placed in a low power mode during DMA, and the DMA interrupt is used to "wake up" the 8032 when DMA is completed.

Note: The 8032 must be placed in low power mode so that it is not executing instructions; that is, the 8032 cannot run while the DMA machine is running.

DMA must be done to complete the SCSI work. Since the SCSI code is running at interrupt level and since there are only two interrupt levels, the SCSI interrupt must be a low level interrupt, and the DMA must be a high level interrupt.

The SCSI code is designed to be able to supply command responses immediately after commands are received, so there is no delay between the time a command is received and the time the response is sent. This feature makes efficient use of the SCSI bus and of the slower LSI53C040 core. It also means that the SCSI code does not need to disconnect from the initiator/host after the command is received, and then reconnect after the data is ready to send, minimizing the overall time spent on the SCSI bus. This is necessary because the LSI53C040 sees only 3 bits of the high SCSI bus (not all 8 bits), so the LSI53C040 cannot arbitrate for the bus to perform the reconnection/reselection whenever the LSI53C040 is placed at a high SCSI ID [15:8]. The LSI53C040 could disconnect and reconnect if it were placed at a low ID [7:0], but that would place unwanted restrictions on the general use of this code.

2.4 8032-Based Timer

An 8032-based timer is also operated at interrupt level. This makes the time keeping more accurate than polling, and since the routine is of minimal size it does not seriously affect performance. Currently this time keeping is only used to control the periodic scanning of the TWS input devices, so timing accuracy is not important. However, the time keeping feature may be used in the future for more time-critical functions.

2.5 TWS Bus

During normal operation, the TWS bus is only used as an input bus. The TWS bus is also used for booting (or autodownload from flash), and to update the contents of the flash by using the SCSI bus, but these are not part of normal operation. The devices currently supported on the TWS bus are the Dallas 1621 and the National LM75 and LM78. These devices do not need to be monitored constantly, and some of them (most notably the LM78) need to be “left alone” for a sufficient period of time so that they can update all of their collected data. This means that these parts should be polled periodically. The host/initiator will probably/realistically poll the SAF-TE software every few minutes, so the SAF-TE software need only poll the TWS parts every few seconds. The software can poll the TWS parts continuously or as infrequently as every 8.5 minutes. Typically, the software will be configured to poll every 2 seconds. This interval is specified by the user with the configuration utility. The user can request that the software poll continuously (that is, every 0 seconds), but LSI Logic does not recommend this.

The TWS polling routines (mentioned above) collect their data and place that data in the SCSI/SAF-TE command response packets (just as the state machine does) so that these packets always contain up-to-date information for immediate transmission to the host/initiator when it is requested.

2.6 Power-On/Start-up

The first job during power-on/start-up is to test the reliability of the **config** data structure. The structure is checked for correct length and checksum. If either of these is wrong, the SAF-TE software will not run. Nearly all of the information in the **config** data structure is user-specific, and virtually all of the firmware is **not** specific to any user, so the software cannot signal that the **config** structure is corrupt. It is expected that the users customizing this software will make changes to light a failure LED or otherwise signal the general failure of the SAF-TE system for this condition, as well as for other similar errors.

Once validated, the information in the **config** data structure can be used to initialize the SCSI hardware, the TWS hardware, and the 52 I/O lines. When initialization is completed, interrupts are enabled and normal (background) processing begins.

2.7 Normal Processing

The background process first checks the health of the SCSI interrupt level software. If unhealthy, the hardware is reinitialized and the SCSI interrupt is enabled.

The background process handles the TWS polling if it is time. The background process also makes one pass through the state machine program to update the status of all the I/O pins. This process is repeated indefinitely.

The SCSI interrupt level code attempts to handle all normal SCSI errors in a straight-forward manner. If any abnormal errors occur, the interrupt software takes the simplest approach - it goes bus free. This is always a valid approach to errors by a SCSI target, plus it gets the LSI53C040 core off the bus as quickly as possible. Since there are few commands and no variants, the SAF-TE protocol works well in a production environment (the assumption is that issues relating to the host and target software have been resolved). When the SCSI interrupt software goes bus free, it signals the background process that it has done so by disabling the SCSI interrupt.

When the background process starts the next pass of its infinite loop (above), it detects the unhealthy state (as mentioned above), reinitializes the hardware, and re-enables the interrupt.

2.8 Interrupts

This section provides information on the 8032 processor and LSI53C040 interrupts.

2.8.1 8032 Processor Interrupts

The 8032 processor has two external interrupts (INT0 and INT1), which may be set to be level or edge triggered depending upon the setting of the control bits in the TCON register.

IT0 = H -> interrupt on falling edge of INT0 (edge triggered)

IT0 = L -> interrupt if INT0 = LOW (level triggered)

IT1 = H -> interrupt on falling edge of INT1 (edge triggered)

IT1 = L -> interrupt if INT1 = LOW (level triggered)

Both of these external interrupts may be configured as inputs from any of the LSI53C040 core interrupts.

Other interrupt sources from the 8032:

SERIAL
RX
TX

TIMER0

TIMER1

TIMER2

2.8.2 LSI53C040 Interrupts

The LSI53C040 interrupts may be input on either INT0 or INT1 of the 8032 core external interrupts. Refer to Section 3.4.3, “Interrupts,” page 3-39 for interrupt details about the SCSI Core, DMA Core, TWS (2), Timers, and SFF-8067. The interrupt registers are:

- Interrupt Status Register (0xFE04): Find source of interrupt.
- Interrupt Mask Register (0xFE0D): H = enable L = disable
- Interrupt Destination Register (0xFE0E): routing to INT0 and INT1

2.9 Debugging

Debugging may be done using **printf()** calls with the output going to the serial port of the 8032. You may then use any terminal emulator to capture the output from the serial port. The serial port uses software flow control using XON/XOFF. The configuration for the serial port is:

- 19200 baud, N-8-1 with XON/XOFF flow control (19200 bits per second, no parity, 8 data bits, 1 stop bit)

2.10 Setting Up A Development Environment

Development may be done using a DOS shell under Windows NT 4.0 and running batch files to run the make process.

Before you start, verify that the Archimedes tools have been installed into the C:\C251 directory, the Borland C++ Version 4.5 has been installed into the C:\BC45 directory, and the SAF-TE source code has been placed in the C:\SAFTE directory.

To set up a development environment, follow these steps:

Step 1. Install Archimedes C-51

Step 2. Install Borland C++ Version 4.5

Step 3. Change the path statement in MAKEBOOT.BAT to include:

```
C:\BC45\BIN;C:\C251\BIN
```

Step 4. Set the following environment variables in MAKEBOOT.BAT:

```
C51INC = C:\C251\INC
C51LIB = C:\C251\LIB
C251INC= C:\C251\INC
C251LIB= C:\C251\LIB
```

Step 5. Your development environment should now be setup and you are ready to run MAKEALL.BAT at the command prompt from the C:\SAFTE directory to create the .HEX files to download into the EEPROM on your SAFTE board.

2.10.1 Development Tools

Company: Borland

Product: Borland C++

Version: 4.5

Note: Use this product to compile the configuration program that allows the user to assign a specific configuration for the LSI53C040 board.

Company: Archimedes Software

Product: 8051/251 Development Suite

Version: 5.50bA

Includes: Compiler, Assembler, SimCASE for Windows

WEB URL: http://www.archimedesinc.com

Phone: (425) 822-6300

Note: The 8051 tools are the only tools currently needed, and purchasing the 251 tools is not required. Future releases of the firmware may use updated versions of the 8051 compiler, (such as 8051-IDE-V6NT).

This information was available from Archimedes Software as of October 13, 1998:

Part Number	Price
8051-IDE-V6NT	$1995
8051-CASM-V6NT	$1495

2.10.2 Source Code Composition

Table 2.1 shows the lines of code associated with each of the files that comprise the source code:

Table 2.1 Source Code Files

Filename	Lines of Code	Blanks + Comments	Code + Blanks + Comments
bootinit.asm	56	79	135
bootload.c	370	160	530
config.c	1931	421	2352
config.h	145	49	194
loader.h	28	17	45
reg040.h	107	108	215
reg51.h	63	18	81
safeinit.asm	13	15	28
safte.c	2371	1508	3879
Total	5084	2375	7459

As shown above, the majority of the code is written in “C” and very little is actually written in assembler.

2.10.3 Register Naming Translations

This section provides detailed information about the 8032 Registers, LSI53C040 Registers, TWS Registers, and miscellaneous registers.

2.10.3.1 8032 Registers

Standard 8051 register names are currently used for the 8032 core. Table 2.2 shows the register name, address, and description.

Table 2.2 Special Function Register Names

Special Function Register (SFR) Name	Address	Description
P0	0x80	Port 0
P1	0x90	Port 1
P2	0xA0	Port 2
P3	0xB0	Port 3
PSW	0xD0	Program Status Word
ACC	0xE0	Accumulator
B	0xF0	B Register
SP	0x81	Stack Pointer
DPL	0x82	Data Pointer High
DPH	0x83	Data Pointer Low
PCON	0x87	Power Control
TCON	0x88	Timer/Counter Control
TMOD	0x89	Timer/Counter Mode
TL0	0x8A	Timer/Counter 0 Low Byte
TL1	0x8B	Timer/Counter 1 Low Byte
TH0	0x8C	Timer/Counter 0 High Byte
TH1	0x8D	Timer/Counter 1 High Byte
IE	0xA8	Interrupt Enable
IP	0xB8	Interrupt Priority
SCON	0x98	Serial Control
SBUF	0x99	Serial Data Buffer

2.10.3.2 LSI53C040 Registers

The following registers have been named differently than in the LSI53C040 Technical Manual. These are memory mapped into external data space of the 8032 at the addresses shown, and are all 8-bit registers.

Table 2.3 SCSI Core/SFF-8067 Registers

Firmware Register Name	Address	LSI53C040 Register Name
SCSI_DATA_REG	0xFC00	CSD or ODR
INITIATOR_COMMAND_REG	0xFC01	ICR
MODE_REG	0xFC02	MR
TARGET_COMMAND_REG	0xFC03	TC
SCSI_BUS_STATUS_REG	0xFC04	CSB or SER
BUS_STATUS_REG	0xFC05	BSR or SDR
DMA_START_TARGET_RECEIVE_REG	0xFC06	SDTR
RESET_PARITY_INTERRUPT_REG	0xFC07	RPI or SDIR
SCSI_HIGH_DATA_REG	0xFC08	CSDHI
HIGH_SELECT_ENABLE_REG	0xFC0C	SENHI
DMA_STATUS_REG	0xFC10	DS
DMA_TRANSFER_LENGTH_REG	0xFC11	DTL
DMA_ADDR_LOW_REG	0xFC12	DSDL
DMA_ADDR_HIGH_REG	0xFC13	DSDH
DMA_INTERRUPT_REG	0xFC14	DMAI

Table 2.4 TWS Registers

Firmware Register Name	Address	LSI53C040 Register Name
TWS_0_DATA	0xFD00	none
TWS_0_CSR	0xFD01	none
TWS_1_DATA	0xFD02	none
TWS_1_CSR	0xFD03	none
LOAD_CHECKSUM	0xFD04	none
LOAD_ADDR_H	0xFD05	reserved
LOAD_ADDR_L	0xFD06	reserved

Table 2.5 Miscellaneous Registers

Firmware Register Name	Address	LSI53C040 Register Name
WDT_CON	0xFE00	WDTC
WDT_SECOND	0xFE01	WDSC
WDT_FINAL	0xFE02	WDFC
MISC_CON	0xFE03	MCR
INT_STATUS	0xFE04	ISR
TIM_1_CON	0xFE05	T1C
TIM_1_THRESHOLD	0xFE06	T1TH
TIM_1_SECOND	0xFE07	T1SC
TIM_1_FINAL	0xFE08	T1FC
TIM_2_CON	0xFE09	T2C
TIM_2_THRESHOLD	0xFE0A	T2T
TIM_2_SECOND	0xFE0B	T2SC

Table 2.5 Miscellaneous Registers (Cont.)

Firmware Register Name	Address	LSI53C040 Register Name
TIM_2_FINAL	0xFE0C	T2FC
INT_MASK	0xFE0D	IMR
INT_DESTINATION	0xFE0E	IDR

Table 2.6 System Registers

Firmware Register Name	Address	LSI53C040 Register Name
POW_CON_0_MASK	0xFF00	RES
POW_CON_0	0xFF01	POCO
POW_CON_1_MASK	0xFF02	RES
POW_CON_1	0xFF03	POCI
LED_BLINK	0xFF04	LBR
SYS_CON	0xFF05	SYSCTRL
IO_0_OUT	0xFF08	MPO0
IO_0_ENABLE	0xFF09	MPE0
IO_0_IN	0xFF0A	MPI0
IO_0_MASK	0xFF0B	MPLM0
IO_0_LATCH	0xFF0C	MPL0
IO_0_PULLDOWN	0xFF0D	MPPE0
IO_1_OUT	0xFF10	MPO1
IO_1_ENABLE	0xFF11	MPE1
IO_1_IN	0xFF12	MPI1

Table 2.6 System Registers (Cont.)

Firmware Register Name	Address	LSI53C040 Register Name
IO_1_MASK	0xFF13	MPLM1
IO_1_LATCH	0xFF14	MPL1
IO_1_PULLDOWN	0xFF15	MPPE1
IO_2_OUT	0xFF18	MPO2
IO_2_ENABLE	0xFF19	MPE2
IO_2_IN	0xFF1A	MPI2
IO_2_MASK	0xFF1B	MPLM2
IO_2_LATCH	0xFF1C	MPL2
IO_2_PULLDOWN	0xFF1D	MPPE2
IO_3_OUT	0xFF20	MPO3
IO_3_ENABLE	0xFF21	MPE3
IO_3_IN	0xFF22	MPI3
IO_3_MASK	0xFF23	MPLM3
IO_3_LATCH	0xFF24	MPL3
IO_3_PULLDOWN	0xFF25	MPPE3
LED_00_OUT	0xFF30	MLO0L
LED_01_OUT	0xFF31	MLO
LED_00_IN	0xFF32	MLI0L
LED_01_IN	0xFF33	MLI0H
LED_00_MASK	0xFF34	MLLM0L
LED_01_MASK	0xFF35	MLLM0H
LED_00_LATCH	0xFF36	MLL0L

Table 2.6 System Registers (Cont.)

Firmware Register Name	Address	LSI53C040 Register Name
LED_01_LATCH	0xFF37	MLL0H
LED_10_OUT	0xFF38	MLO1L
LED_11_OUT	0xFF39	MLO1H
LED_10_IN	0xFF3A	MLI1L
LED_11_IN	0xFF3B	MLI1H
LED_10_MASK	0xFF3C	MLLM1L
LED_11_MASK	0xFF3D	MLLM1H
LED_10_LATCH	0xFF3E	MLL1L
LED_11_LATCH	0xFF3F	MLL1H
LED_20_OUT	0xFF40	MLO2L
LED_21_OUT	0xFF41	MLO2H
LED_20_IN	0xFF42	MLI2L
LED_21_IN	0xFF43	MLI2H
LED_20_MASK	0xFF44	MLLM2L
LED_21_MASK	0xFF45	MLLM2H
LED_20_LATCH	0xFF46	MLL2L
LED_21_LATCH	0xFF47	MLL2H

2.10.4 Calling Trees

The three source files that have calling trees are:

- `Config.C`
- `Bootload.C`
- `Safte.C`

All names are function names with the names in all caps being macros.

Example:

```
a
    b
            c
            d
```

the function 'a' has one subfunction 'b'
    'b' has two subfunctions 'c' and 'd'
            'c' and 'd' have no subfunctions in this example.

2.10.4.1 Summary Calling Tree of Config.C

```
main
    abort
            ..printf
            ..fprintf
            ..exit
    init_pin_data
    ask_questions
            YorN
                    abort
                    ..fprintf
                    ..fgets
                    ..sscanf
                    ..toupper
            ask_text
                    abort
                    ..fprintf
                    ..fgets
                    ..strlen
            ask_scsi_id
                    ask_MtoN
                            abort
                            ..fprintf
                            ..fgets
                            ..sscanf
                    abort
                    zerotoN
                            ask_MtoN
                    ..fprintf
            zerotoN
            ask_MtoN
            ask_pins
                    abort
```

```
                ..fprintf
                ..fgets
                ..sscanf
                ..strcmp
        abort
        ask_map
                abort
                ..fprintf
                ..fgets
                ..sscanf
        ask_LM78
                ask_MtoN
                YorN
                ..printf
                ..sprintf
        ..fprintf
        ..sprintf
        ..printf
                ..fgets
        ..sscanf
assign_DM_addrs
        abort
        ..memset
        ..printf
        ..fprintf
fill_config_structure
        swap_2_bytes
        fill_program
                fill_instruction
                        swap_2_bytes
                        ..fprintf
                fill_output_pin_instruction
                        abort
                        fill_instruction
                        ..printf
                ..min
                ..printf
        checksum
        ..memset
        ..fprintf
create_hex_file
        abort
        ..fopen
        ..printf
        ..fprintf
        ..fclose
fill_loader_structure
        swap_2_bytes
..printf
```

2.10.4.2 Summary Calling Tree of Bootload.C

```
main
    tws_setup
    tws_init
    tws_read
           tws_poll_bb
           tws_poll_lrb
           tws_poll_pin
    delay
    ..moveit
moveit2
```

2.10.4.3 Summary Calling Tree of Safte.C

```
main
    validate_config
            checksum
    init_stuff
            tws_setup
            tws_init
    background_process
            init_time_keeping
            reset_scsi_hw
            execute_program_once
            background_code_load
                    tws_setup
                    tws_memory_write
                            tws_write
                                    tws_poll_bb
                                    tws_poll_lrb
            gather_TWS_input
                    gather_LM78_input
                            tws_setup
                            tws_write
                            tws_read
                                    tws_poll_bb
                                    tws_poll_lrb
                                    tws_poll_pin
                    tws_setup
                    tws_write
                    tws_read
    ..memset
            check_condition
ir_external0
ir_external1
    req_ack
    command_and_data_phases
            check_condition
            do_inquiry
                    invalid_LUN_test
                    min
                    send_data_bytes
                            req_ack
            do_request_sense
                    min
                    send_data_bytes
            invalid_LUN_test
            do_read_buffer
                    check_condition
                    do_read_enclosure_configuration
```

```
                        min
                        send_data_bytes
                do_read_enclosure_status
                        min
                        send_data_bytes
                do_read_device_slot_status
                        min
                        send_data_bytes
                do_read_global_flags
                        min
                        send_data_bytes
        do_send_diagnostic
        do_test_unit_ready
        do_write_buffer
                check_condition
                get_data_bytes
                        req_ack
                do_write_device_slot_status
                do_perform_slot_operation
                        check_condition
                do_send_global_flags
                do_code_load
                        check_condition
                        get_data_bytes
ir_serial
ir_timer0
ir_timer1
ir_timer2
max
```

2.11 Configuration Examples

The following examples show specific monitoring capabilities of the LSI53C040. Numerous configurations are possible subject to the 28 MPIO and 24 MPLED I/O pins.

2.11.1 Example 1

This example (Figure 2.3) shows an implementation supporting the maximum number of SCSI Devices (14) on an LVD bus, six power supplies, and six fans without any external logic. Note that the devices are on the LVD bus due to the initiator and the LSI53C040 each consuming a SCSI ID.

Figure 2.3 System Configuration

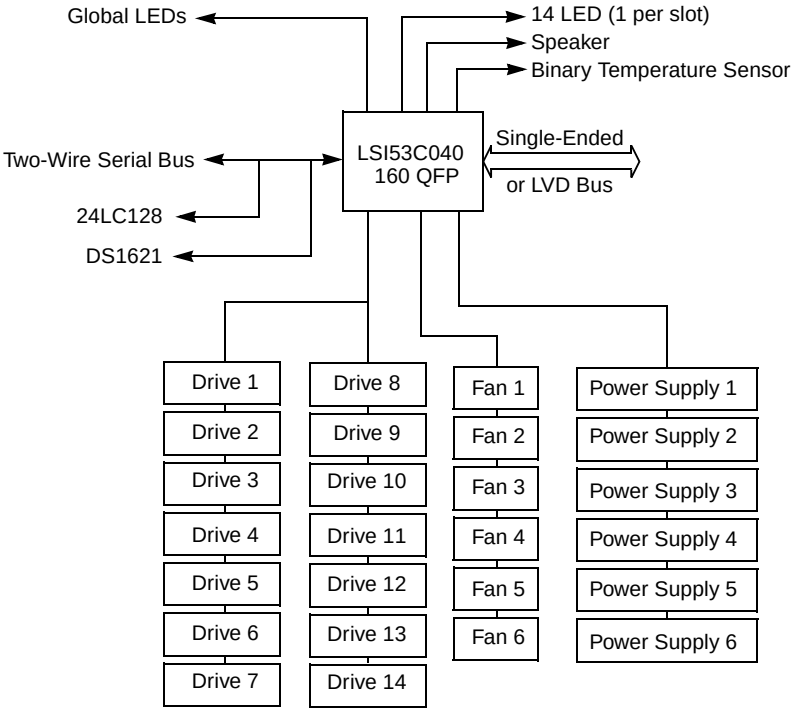


Table 2.7 below summarizes the MPIO/MPLED pin count required for this implementation. There are seven other available MPLEDs for LED functions.

Table 2.7 MPLED/MPIO Pin Usage for Example 1

SAF-TE Feature	MPIO Pins	MPLED Pins
Device Present signal	14	
Device Slot LEDS		14
Drive Ready for Use signal		
Drive Ready for Insertion/Removal signal		
Fan Monitoring (single input)	6	
Power Supply (single input)	6	
Fan Monitoring (dual input)		

Table 2.7 MPLED/MPIO Pin Usage for Example 1 (Cont.)

SAF-TE Feature	MPIO Pins	MPLED Pins
Power Supply (dual input)		
Binary Temperature Sensor	1	
Speaker	1	
Door Lock		
Global Enclosure LED		1
Global Array LED		1
Global Drive LED		1
TOTAL	28	17

2.11.2 Example 2

This example (Figure 2.4) shows a configuration using the optional drive slot power feature. Up to 10 drives with two LEDs per drive slot, 3 single input fans, three single input power supplies, one speaker, and one door lock are supported.

Figure 2.4 Optional Drive Slot Power Configuration

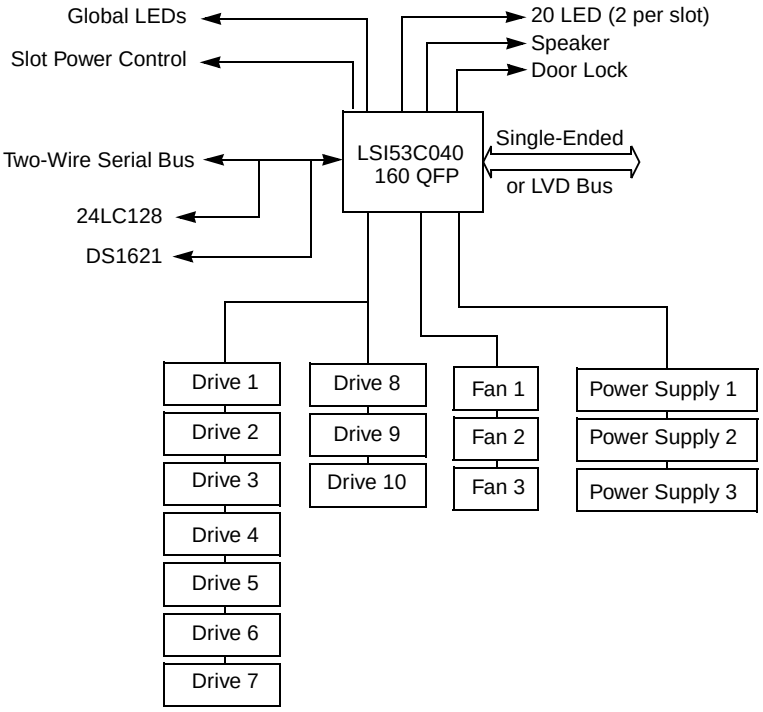


Table 2.8 below summarizes the MPIO/MPLED pin count required for this implementation. There is one additional available MPLED for other LED functions.

Table 2.8 MPLED/MPIO Pin Usage for Example 2

SAF-TE Feature	MPIO Pins	MPLED Pins
Device Present signal	10	
Device Slot LEDS		20
Drive Ready for Use signal	10	
Drive Ready for Insertion/Removal signal		
Fan Monitoring (single input)	3	
Power Supply (single input)	3	
Fan Monitoring (dual input)		
Power Supply (dual input)		
Binary Temperature Sensor		
Speaker	1	
Door Lock	1	
Global Enclosure LED		1
Global Array LED		1
Global Drive LED		1
TOTAL	28	23

2.11.3 Example 3

This example (Figure 2.5) shows a configuration with dual input fans and power supplies. Up to 14 drives with one LED per drive slot, three dual input fans, three dual input power supplies, one speaker, and one door lock are supported.

Figure 2.5 Dual Fans and Power Supplies Configuration

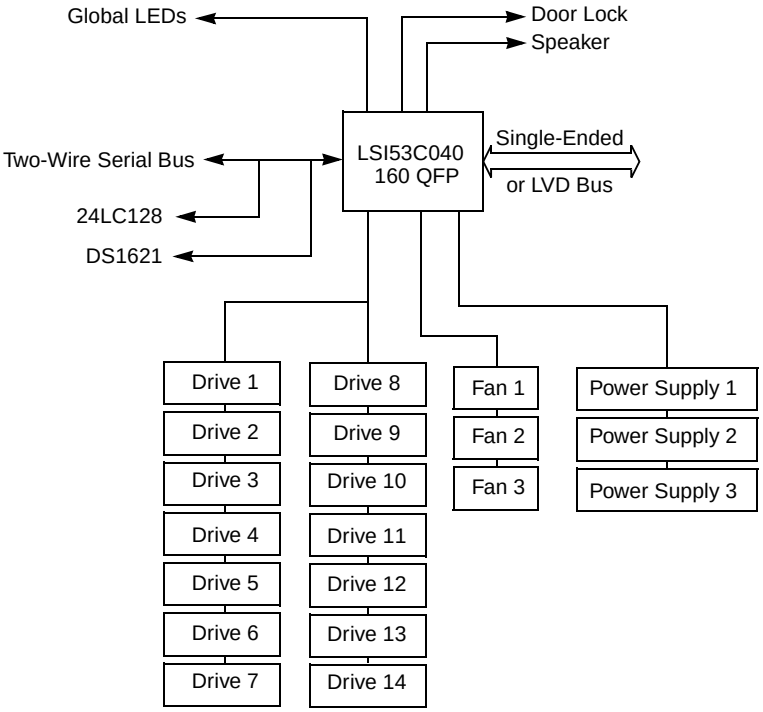


Table 2.9 below summarizes the MPIO/MPLED pin count required for this implementation. One additional MPLED is available for other LED functions.

Table 2.9 MPLED/MPIO Pin Usage for Example 3

SAF-TE Feature	MPIO Pins	MPLED Pins
Device Present signal	14	
Device Slot LEDS		14
Drive Ready for Use signal	10	
Drive Ready for Insertion/Removal signal		
Fan Monitoring (single input)		
Power Supply (single input)		
Fan Monitoring (dual input)	2	4 (Used as MPIOs)

Table 2.9 MPLED/MPIO Pin Usage for Example 3 (Cont.)

SAF-TE Feature	MPIO Pins	MPLED Pins
Power Supply (dual input)		6 (Used as MPIOs)
Binary Temperature Sensor		
Speaker	1	
Door Lock	1	
Global Enclosure LED		1
Global Array LED		1
Global Drive LED		1
TOTAL	28	23

Chapter 3
SAF-TE Source Code

This chapter discusses the SAF-TE source code and includes these topics:

For additional information about supporting serial, ISA, SCSI, and 8067 data transfers, refer to Appendix A, “LSI53C040 Board Utilities.”

3.1 SAF-TE Source Code Overview

The SAF-TE source code is comprised of the executable `config.exe`, a partial boot file, and the C-source file **`safte.c`.** The boot and configuration module sections provide a brief overview regarding their structures. The SAF-TE Module section includes compilation instructions, and description of the **`safte.c`** source code main program. The **`safte.c`** program is designed to run as firmware on the LSI53C040 chip. Additionally, this module discusses Interrupt Service Routines and Error Reporting. The FAQ section provides the firmware developer with answers to potential questions that may arise from using the **`safte.c`** source code.

3.2 Boot Module

The boot module consists of the concatenation of the user configurable component, `loader.hex`, and the generic boot component, `bootload.hex`. In Chapter 5, "Configuration Data and the Configuration Utility," the combined file that is referenced below

```
boot.hex = loader.hex + bootload.hex
```

is downloaded to the boot EEPROM on new boards either by placing the chip in a TWS programmer or the code can be downloaded over the ISA bus to the serial EEPROM (demonstration boards only). Once firmware is present on the LSI53C040 board, the boot code can be updated in exactly the same manner as the firmware. See Section 3.4.2.3, "Firmware Update," page 3-33 for more information.

On a power up, the values of the pull-up resistors A8–A11 determine the TWS bus and the chip address on that bus from which the boot code is automatically downloaded. Execution of the boot code results in the contents of the file `loader.hex` being saved into address HMA_LOADER_OPTIONS (3FE0h) of external data memory. In addition, the SAF-TE firmware is being downloaded from the serial EEPROM into external data memory. The serial EEPROM that contains the SAF-TE firmware is specified by the contents of `loader.hex`.

> Note: The automatic firmware downloads can be disabled by modifying the jumper settings on the LSI53C040 board. See the section entitled "Power On Configuration Options," in the *LSI53C040 Enclosure Services Processor Technical Manual* for specific information. The maximum size of the firmware is limited to 12912 or 0x2FA0 bytes.

The generic boot component, `bootload.hex`, is created by compiling the C-source file **`bootload.c`**, the assembly source code **`bootload.asm`**, linking the results, and creating the hex output file. The directory 8051 contains a makefile that will execute using the Borland 4.5 make function or the Watcom 11.0 Microsoft make clone, *nmake*. To recompile the `bootload.hex` component, type on the command line:

```
c:\8051> nmake clean
c:\8051> nmake all
```

The first component will delete the object, hex, and other intermediate files, while the second option will generate new files. Note that individual targets may be recompiled or linked by following the *make* command with the target specification.

Finally, the `bootload.c` source file contains four switches that can be set to change the characteristics of the resulting boot code. Each of the switches is defined in Table 3.1.

Table 3.1 Source file - bootload.c - Switches

Define	Action when TRUE
`LOADER_DEBUG`	Enables Timer 2 to be the clock on the serial output line. Enables error and status messages and display of external memory values during boot.
`LOADER_DELAY_LEDS`	Turns on then off each LED in sequence during boot.
`LOADER_MEMORY_TEST`	Enables the test byte 0x55 to be written to external memory for test purposes. (When FALSE, the memory test does nothing.)
`HALF_SPEED`	Sets the overflow rate for Timer 2. Requires `LOADER_DEBUG` to be TRUE.

Currently, these switches cannot be altered through compile time options. The source code #Define statements must be modified. Default settings have all switches set to FALSE.

3.3 Configuration Module

The utility `config.exe` is designed to allow users to configure the SAF-TE software to meet their unique requirements. Upon execution, this utility populates the **loader_options** and **config** data structures, then saves these structures in the Intel formatted files `loader.hex` and `config.hex`. The data format for these files is:

```
:numbytes address fill data sum
```

where:

numbytes	is the number of bytes of data (stored as 1 byte)
address	is a 2 byte address where data is to be written
fill	has the value 00
data	is the data to be written (a maximum of 16 bytes)
sum	is a 1 byte checksum over address and data in that line

Generation of the executable file `config.exe` can be accomplished by executing one of the two makefiles in the *config* directory. If a Watcom 11.0 compiler is resident, use the makefile entitled *makefile.watcom* and run with the command *nmake*. If a Borland 4.5 compiler is resident, use the makefile entitled *makefile.bc45* and run with the command *make*. The result of the execution of `config.exe` is independent of which compiler is used. To recompile the `config.exe` file, type on the command line:

```
c:\config> nmake clean
c:\config> nmake config.exe  (or just nmake)
```

The first component will delete the object and executable files, while the second will generate new files.

Note: See `makefile.txt` for more information. However, to use the *makefile.watcom*, type on the command line:

```
c:\config>nmake\Fmakefile.watcom clean
name\Fmakefile.watcom config.exe
```

3.3.1 Loader_Options Data Structure

The **loader_options** data structure, defined in `loader.h`, stores the bus number, bus speed, chip identification, beginning address, and the maximum image length for the “first” and “second” images. Additionally, this data structure stores the bus and address of the EEPROM that contains the boot code. By default, upon power up, the “first” image is defined as active and the “second” image is defined as inactive. The file `loader.hex` is only 68 bytes, with writes beginning at address 0x0006 in external data memory (Program Tag LMA_LOADER_OPTIONS).

Note: Of the 68 bytes contained in the `loader.hex` file, only 21 bytes are data. The remaining 47 bytes consist of the Intel format and checksum bytes.

3.3.2 Config Data Structure

The **config** data structure, defined in `config.h`, stores the enclosure configuration and data to be returned in response to any of the SAF-TE read commands. Additionally, this data structure polls intervals for refreshing enclosure status information, register storage, data memory, and program memory. The program memory stores the state machine that is generated by user responses to mappings of the MPLED and MPIO pins. To emphasize, `config.hex` was created for a specific system. Changes to that system will require change to this configuration file and the firmware. The initial write address for `config.hex` is address 0x0040 in external data memory (Program Tag CONFIG_BASE_ADDRESS). Execution of the utility is described in Chapter 5, “Configuration Data and the Configuration Utility.”

The **config** data structure can be thought of as having ten different elements or sections. The developer must understand how these elements are populated and their function is essential if the `safte.c` source code is modified. The main elements of the **config** data structure are:

1. General information - Contains miscellaneous configuration information.
2. Enclosure configuration - Lists the total number of fans, power supplies, etc.
3. Inquiry Response Information - Contains the data to be returned in response to an inquiry command.
4. Timer Setup Information - Specifies the polling interval and the 80C32 Timer 1 and Timer 2 initialization values.
5. TWS Temperature Sensor Information (on TWS bus) - Contains user specified sensor type and ID value.
6. LM78 Configuration - Contains the user specified configuration for each LM78 (a maximum of one per bus).
7. TWS Bus Configuration - Contains the user specified bus speed and ID information.

8. Register and Device State Maps - Not configurable regions. The register maps save the LSI53C040 system registers and the device state map is used to decode the Device-To-State member of the state machine.
9. Data memory - Contains the LED responses to different states, SCSI ID values, and initialization values. Data memory is modified during firmware execution.
10. Program memory - Contains the state machine instructions. Program memory is not modified during firmware execution.

For illustration purposes, a simple `config.hex` file was dissected and its contents are listed below. In the listing, **bold** indicates a user input, *italics* indicate a computed value based on user inputs, and regular text indicates values that cannot be altered by the user.

1. General Information:

   config structure length = 0xB105
   Scsi ID (in bit form) = 0x80

   Note: IDs between (0–7) represented as (1<<ID) << 8 while the three highest IDs are 0x0080, 0x0040 and 0x0020 respectively.

   Scsi lun = 0x00
   Parity (1 = enabled/0 = disabled) = 0
   Code load (1 = enabled/ 0 = disabled) = 0
   Temperature Units (1 = Celsius/0 = Fahrenheit) = 1
   Read Enclosure Status command length = 0x0F
   Read device slot status command length = 0x08
   Write device slot status command length = 0x06
   Data Memory Index: Read Enclosure Status = 0x42
   Data Memory Index: Temperature = 0x4C
   Data Memory Index: Temperature Bits = 0x4E
   Data Memory Index: SCSI ID Data = 0x48
   Data Memory Index: Device Slot Status = 0x51
   Data Memory Index: Slot Operation = 0x59
   Data Memory Index: Global Flags = 0x5F

2. Enclosure Configuration:

   Number of Fans (1-single, 1-dual, 1-TWS) = 3
   Number of Power Supplies (1-single, 1-dual, 1-TWS) = 3
   Number of Device Slots = 2
   Number of Door Locks = 1
   Number of Temperature Values (TWS sensors) = 2
   Number of Speakers = 1
   Number of Temperature Bits (single wire sensors) = 82

   Note: If Celsius, bit #7 is 1; else bit #7 is 0.

3. Inquiry Response Information:

   Peripheral Qualifier = 0x00
   ANSI Version = 0x02
   Response Format = 0x02
   Additional Length = 0x31
   Vendor ID = tester#1
   Product ID = this_is_product!
   Firmware Revision Level = B004
   Enclosure ID = enclos1
   Channel ID = 0
   Interface ID = SAF-TE
   Specification Level = 1.00

4. Timer Setup Information:

   Timer High = 0x0E
   Timer Low = 0xE0
   Timer 1 Rollover = 0x12
   Timer 2 Rollover = 0x06
   TWS Poll Interval = 20 (sec)

5. TWS Temperature Sensor Information:

Sensor Number	Chip ID	Chip Type	Chip Label	State 0 = unitialized 1 = initialized
0	**0x9A**	**0x01**	**LM75**	0
1	**0x97**	**0x02**	**DS1621**	0
2	*0x42*	*0x00*	*Unknown*	0
3	*0x42*	*0x00*	*Unknown*	0

Chip ID = 0x90 | (address << 1) | bus # (for LM75 and DS1621)

Chip ID = 0x50 | (address << 1) | bus # (for LM78)

6. LM78 Configuration:

LM78 on Bus # 0 (User specified NO LM78 on Bus 0)
State (0=uninitialized/1=initialized/0xFF=nonexistent) = 0xFF
No. of Errors during initialization = 0x00
*Fan Divisor = 0x00(Fan Divisor = (first*64 + second*16))*

Power Supply	Minimum Voltage	Maximum Voltage	Data Memory Index
0	*0x00*	*0x00*	*0x00*
1	*0x00*	*0x00*	*0x00*
2	*0x00*	*0x00*	*0x00*
3	*0x00*	*0x00*	*0x00*
4	*0x00*	*0x00*	*0x00*
5	*0x00*	*0x00*	*0x00*
6	*0x00*	*0x00*	*0x00*

Fan Number	Maximum Speed	Data Memory Index
0	*0x00*	*0x00*
1	*0x00*	*0x00*
2	*0x00*	*0x00*

LM78 on Bus # 1

State (0 = uninitialized/1 = initialized/0xFF = nonexistent) = 0x00

No. of Errors during initialization = 0x00

Fan Divisor = 0xF0 *Fan Divisor = (first*64 + second*16))*

Power Supply	Minimum Voltage	Maximum Voltage	Data Memory Index
0	**0x00**	**0x00**	*0x00*
1	**0x01**	**0xFE**	*0x47*
2	**0x00**	**0x00**	*0x00*
3	**0x00**	**0x00**	*0x00*
4	**0x00**	**0x00**	*0x00*
5	**0x00**	**0x00**	*0x00*
6	**0x00**	**0x00**	*0x00*

Fan Number	Maximum Speed	Data Memory Index
0	**0xFE**	*0x44*
1	**0x00**	*0x00*
2	**0x00**	*0x00*

7. TWS Bus Configuration:

Bus Number	Speed	Our ID
0	**0x16**	0x2A
1	**0x1C**	0x2A

Bus Speed: Answer 0 -> 0x02; 1 -> 0x01; 2 -> 0x16; 3 -> 0x1C

8. Register and Device State Maps: The **config** data structure has one member, `device_state`, which is of type `_dtl` and is populated automatically during the execution of the `config.exe` program. This member defines a series of masks that are used to determine the status of a particular device slot. The values in the structure specify particular pins. Each time the state machine executes, for each slot, the state OPC_DEVICE_TO_STATE is executed. This state results in successive application of the masks defined in the *`device_state`* member. The matched mask that has the highest priority is the value reported on a Read Slot Status command. Hence, the configuration program automatically prioritizes the possible reports.
9. Data Memory: This element is initialized with all zeros, and then elements at different data memory indices are modified to reflect the user selections. Since the SCSI Data memory index is 0x48 and there are 2 device slots, the first device slot is assigned SCSI ID 7 and the second is assigned ID 6. Likewise, at offset 0x04, the meaning assigned to each of the four light patterns is established. For example purposes only, some of the offsets are:

Offset	Description
0x02	fan (single)
0x04	fan (dual)
0x08	power supply (single)
0X0A	power supply (dual)
0x0E	door lock
0x10	speaker

Offset	Description
0x12	LED 0
0x22	LED 1
0x32	slot status

0x0000: *00 00 00 01 00 01 02 80 | 00 01 00 01 10 11 00 01*
0x0010: *00 01 00 00 03 01 01 01 | 01 01 01 00 00 01 02 00*
0x0020: *00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00*
0x0030: *00 00 02 02 02 02 03 05 | 03 03 02 02 02 02 05 05*
0x0040: *03 05 80 80 00 80 80 00 | 07 06 80 00 00 00 00 00*
0x0050: *00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00*
0x0060: *00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00*
0x0070: *00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00*
0x0080: *00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00*
0x0090: *00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00*
0x00A0: *00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00*
0x00B0: *00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00*
0x00C0: *00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00*
0x00D0: *00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00*
0x00E0: *00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00*
0x00F0: *00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00*

10. Program Memory: The program memory is populated using instructions for an 11-state state machine. The states that access MPIO or MPLED memory mapped registers are:

    Read Register Bit (OpCode 0x00)
    Write Register Bit (OpCode 0x01)
    Write LED Register Bit (OpCode 0x07)
    Device to State (OpCode 0x09)
    Write LED Bits (OpCode 0x0A).

The states that access the data memory are:

Read Memory Bit (OpCode 0x02)
Write Memory Bit (OpCode 0x03)
Read Memory Byte (OpCode 0x04)
Write Memory Byte (OpCode 0x05)
Read Memory Indexed (Opcode 0x06).

Termination of the state machine is accomplished with the last state:

Done (OpCode 0x08).

Using the answers to the questions posed by `config.exe`, the function *`fill_program`* clears and repopulates the data memory (256 bytes) and program memory (a maximum of 400 integers) members of the **config** data structure. It will, for each fan, power supply, door lock, etc., call the subroutine *`fill_instruction`* at least once.

Note: In those cases where the MPIO register is to be modified, *`fill_program`* will call the function *`fill_output_pin_instruction`* that in turn calls *`fill_instruction`*.

The function *`fill_instruction`* takes its arguments, program memory counter (beginning from 0), operation code (specifies the next state), Shift (bit shift, 0–3) and Address, and creates a 16-bit "instruction". The instruction

```
(operation code<< 12)|(Shift << 8)|(Address)
```

is then written to program memory at the offset specified by the counter. The SAF-TE source code restricts the maximum number of instructions to 400 (Program Tag PROGRAM_MEMORY_SIZE).

As shown in Table 3.2 below, the number of instructions written per device, is device specific.

Table 3.2 Instructions Per Device

Description	Number of Instructions
for each single state fan	3
for each dual state fan	9
for each single power supplie	3
for each dual power supplie	9
use of controlled door lock	4
use of monitored door lock	3
use of controlled speaker	4
use of monitored speaker	3
use of global ID LED	6
use global enclosure LED	8
use global drive LED	8
use global array LED	8
for each temperature sensor	2
for each device slot	
general	12
if at least 1 LED	+ 7
if no LED	+ 3
if use device present	+ 2
if use ready device	+ 2
if use remove device	+ 2
Termination	1

Remark: The actual bit sequence written to program memory will be unique to the configuration specified during the execution of `config.exe` because only those devices selected as being present are incorporated into the instruction set in program memory. Therefore, it is not possible to go to offset, for example: 0xA0, in program memory and say with certainty that is an instruction for a power supply.

For the example, the mapping of devices to MPIO and MPLED banks, is accomplished with the sequence of OpCodes as shown in Table 3.3.

Table 3.3 Mapping of devices to MPIO and MPLED Banks

Program Counter	OpCode	Description
0x0000:	0x00	Single Wire Fan # 1 MPIO Bank 2 Position 0
0x0001:	0x06	
0x0002:	0x05	
0x0003:	0x04	
0x0004:	0x05	
0x0005:	0x00	Dual Wire Fan # 1 MSB MPIO Bank 2 Position 2
0x0006:	0x03	
0x0007:	0x00	Dual Wire Fan # 1 LSB MPIO Bank 2 Position 1
0x0008:	0x03	
0x0009:	0x04	
0x000A:	0x06	
0x000B:	0x05	
0x000C:	0x00	Single Wire Power Supply # 1 MPIO Bank Position 0
0x000D:	0x06	
0x000E:	0x05	
0x000F:	0x04	
0x0010:	0x05	
0x0011:	0x00	Dual Power Supply # 1 MSB MPIO Bank 3 Position 2
0x0012:	0x03	
0x0013:	0x00	Dual Power Supply # 1 LSB MPIO Bank 3 Position 1
0x0014:	0x03	

Table 3.3 Mapping of devices to MPIO and MPLED Banks (Cont.)

Program Counter	OpCode	Description
0x0015:	0x04	
0x0016:	0x06	
0x0017:	0x05	
0x0018:	0x00	Monitored Door Locks MPIO Bank 0 Position 0
0x0019:	0x06	
0x001A:	0x05	
0x001B:	0x02	
0x001C:	0x01	Controlled Speakers MPIO Bank 0 Pos 1
0x001D:	0x06	
0x001E:	0x05	
0x001F:	0x04	
0x0020:	0x05	Global LED No. 0 ID
0x0021:	0x02	
0x0022:	0x03	
0x0023:	0x04	
0x0024:	0x0A	MPIO Bank 1 Position 0
0x0025:	0x04	
0x0026:	0x05	Global LED No. 1 Drive
0x0027:	0x02	
0x0028:	0x03	
0x0029:	0x02	
0x002A:	0x03	
0x002B:	0x04	
0x002C:	0x0A	MPIO Bank 1 Position 1
0x002D:	0x04	

Table 3.3 Mapping of devices to MPIO and MPLED Banks (Cont.)

Program Counter	OpCode	Description
0x002E:	0x05	Global LED No. 2 Array
0x002F:	0x02	
0x0030:	0x03	
0x0031:	0x02	
0x0032:	0x03	
0x0033:	0x04	
0x0034:	0x0A	MPIO Bank 1 Position 2
0x0035:	0x00	Single Wire Temp Sensor #0 MPIO Bank 0 Position 2
0x0036:	0x03	
0x0037:	0x00	Single Wire Temp Sensor #1 MPIO Bank 0 Position 3
0x0038:	0x03	
0x0039:	0x09	Configure Device Slot 0 (>0 LED) SCSI ID = 7
0x003A:	0x06	
0x003B:	0x0A	MPLED Bank 0 Position 0
0x003C:	0x09	Configure Device Slot 1 (>0 LED) SCSI ID = 6
0x003D:	0x06	
0x003E:	0x0A	MPLED Bank 0 Position 1
0x003F:	0x04	Initialize Device Slot 0 (>0 LED) SCSI ID = 7
0x0040:	0x05	
0x0041:	0x02	
0x0042:	0x03	
0x0043:	0x02	
0x0044:	0x03	
0x0045:	0x00	Using Device Present. MPLED Bank 1 Position 0

Table 3.3 Mapping of devices to MPIO and MPLED Banks (Cont.)

Program Counter	OpCode	Description
0x0046:	0x03	
0x0047:	0x02	
0x0048:	0x03	
0x0049:	0x04	
0x004A:	0x06	
0x004B:	0x05	
0x004C:	0x02	Using Ready Device
0x004D:	0x0A	MPLED Bank 1 Position 1
0x004E:	0x02	Using Remove Device
0x004F:	0x0A	MPLED Bank 1 Position 2
0x0050:	0x04	Initialize Device Slot 1 (>0 LED) SCSI ID = 6
0x0051:	0x05	
0x0052:	0x02	
0x0053:	0x03	
0x0054:	0x02	
0x0055:	0x03	
0x0056:	0x00	Using Device Present. MPLED Bank 2 Position 0
0x0057:	0x03	
0x0058:	0x02	
0x0059:	0x03	
0x005A:	0x04	
0x005B:	0x06	

Table 3.3 Mapping of devices to MPIO and MPLED Banks (Cont.)

Program Counter	OpCode	Description
0x005C:	0x05	
0x005D:	0x02	Using Ready Device
0x005E:	0x0A	MPLED Bank 2 Position 1
0x005F:	0x02	Using Remove Device
0x0060:	0x0A	MPLED Bank 2 Position 2
0x0061:	0x08	Done

3.4 SAF-TE Module

This section provides compilation instructions for the **safte.c** program and provides an high-level overview of the SAF-TE module, which contains the main program.

3.4.1 Compilation Instructions for safte.c

The component safte.hex is created by compiling the C-source file **safte.c**, linking the results, and creating the hex output file. As with the boot code, generation of the safte.hex involves multiple steps. The directory 8051 contains a makefile that will execute using the Borland 4.5 *make* function or the Watcom 11.0 Microsoft make clone, *nmake*. To recompile the safte.hex file, type on the command line:

```
c:\8051> nmake clean
c:\8051> nmake all
```

The first component will delete the object, hex, and other intermediate files while the second will generate new files. Note that individual targets may be recompiled or linked by following the make command with the target specification.

Like the boot code, the SAF-TE source code contains built-in switches to enable and disable different code features. Currently, these switches are set to TRUE or FALSE within the source code using #Define statements and cannot be set by the compiler. The default settings for all switches is FALSE. Table 3.4 lists the switch name and the action if the switch is TRUE.

Table 3.4 Switch Name and Action

Options	Action if TRUE
CODE_LOAD_ENABLED	Enable support for microcode downloads over the SCSI bus.
DMA_DATA_IN_ENABLED	Define the method of sending data over the SCSI bus (choices are using software or using DMA).
DMA_COMMAND_ENABLED	Defines the method of receiving CDB data over the SCSI bus (choices are using software or using DMA).
STATS_ENABLED	Controls the inclusion of statistics gathering and reporting by this code.
DEBUG_ENABLED	Controls the inclusion of debugging code that is used to print information for developers looking for problems in this code. This conditional should only be used in code that does **not** run at interrupt level.
DEBUG_SCSI_ENABLED	Controls the inclusion of debugging code that is used to print information for developers looking for problems in the SCSI code. This conditional should only be used in code that is part of the SCSI interrupt service routine. **Note:** DEBUG_ENABLED and DEBUG_SCSI_ENABLED cannot be used at the same time, because this would cause the *`printf`* library routine to be called from both the background process and the SCSI interrupt routine. This means that the *`printf`* routine is not re-entrant.
ZERO_CHECK_ENABLED	Controls the inclusion of code that checks reserved (or must be zero) fields of the SCSI and SEP commands. This code should be included in the final code but could be removed to save space and/or execution time. Thus, its only real purpose is to reject otherwise valid commands; removing this code would violate the SCSI specification, but the code would still “get the job done.”
PRINT_STATS_ENABLED	Controls the printing of statistics collected when the STATS_ENABLED option (above) is true.

Table 3.4 Switch Name and Action (Cont.)

Options	Action if TRUE
TRACER_ENABLED	Causes the SCSI interrupt routine to leave a record of its progress in a variable named TRACKER that can be displayed by the background process to help locate problems in the SCSI code.
OPTIONAL_ENABLED	Enables the handling of the optional SAF-TE commands. The routines that handle these commands are only placeholders and must be supplied by the OEM developers that purchase this code.
SET_SCSI_ID	Enables the handling of the optional SAF-TE command. The code that supports this command is supplied, but not enabled (by this conditional).
VERBOSE	Enables printing of various data at system start-up. This option is only useful to developers, and only if a serial line is attached to the LSI53C040.
MICRO_DEBUG	Enables the printing of debugging messages specific to the background process handling of the state machine used to monitor inputs and drive outputs (MPIO and MPLED pins).
TWS_DEBUG	Enables the printing of debugging messages specific to the TWS interface routines.
REV_A_CHIP	Enables code that is only necessary in working around DMA problems in the first (LSI Logic internal) version of the LSI53C040 chip.
DBG_BLOCK_SCSI	Set to block SCSI interrupt inside state machine (`execute_program_once`). Use this option only in debug versions.
SEP_CMD_DEBUG_CTRL	Set to enable specific SCSI I/Os to control output of printf's. VERBOSE and MICRO_DEBUG must be enabled.

3.4.2 Main Program

The main program consists of subroutines and functions pertaining to the LSI53C040 Enclosure Services Processor.

3.4.2.1 Background Module Overview

The top-level function in the SAF-TE source code is the module main. This function is responsible for:

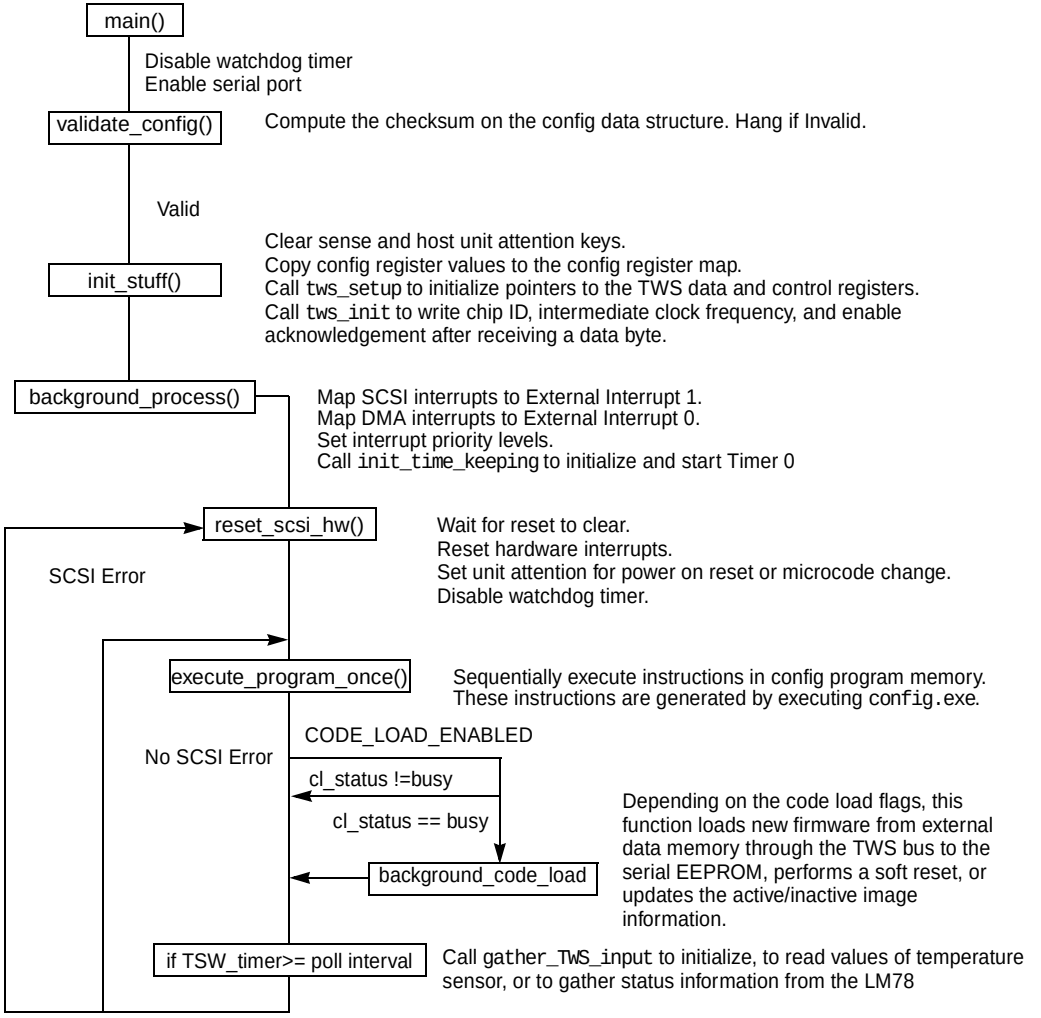
- transferring all TWS data
- executing the contents of program memory
- gathering enclosure status information periodically
- updating data memory;
- processing reboot and firmware update commands
- uploading new firmware from external data memory into the serial EEPROM attached to one of the TWS buses.

As stated earlier in this Programming Guide, upon power-up, the contents of the serial EEPROM are automatically downloaded to the external data memory (although this can be disabled by changing the jumper settings on the LSI53C040 board). Once the firmware has been loaded into external memory and a reset has occurred, the firmware begins executing.

A high-level flow diagram of the main module is shown in Figure 3.1. For convenience, a brief description of the subroutine actions is shown in *italics* next to each function name. The flow diagram illustrates that the function `background_process` can be viewed as having 4 functional components:

- reset_scsi_hw
- execute_program_once
- background_code_load, and
- gather_TWS_input.

Figure 3.1 Main Program



reset_scsi_hw()

If a reset is occurring, this routine delays until the reset is complete. Subsequently, the hardware interrupts are reset, the hardware select register is enabled and the sense keys are set for a Unit Attention.

background_code_load

The code load function loads new firmware, performs a soft reset, or updates the active/inactive image information.

gather_TWS_input

This function gathers status information of TWS peripherals. The status of each peripheral is read and the **config** data structure is updated when the global variable **TWS_timer** exceeds the threshold poll interval. The **TWS_timer** increments every two seconds and the poll interval specifies the number of 2-second intervals between queries. The user-configurable poll interval is set in response to the question "How many 2-second intervals would you like between 2-wire serial input passes?" which appears during execution of the `config.exe` program.

Note: If a value of zero (0) is entered, the firmware will poll continuously. The poll interval is the number of seconds between device polling. However, a system contains many devices, so the time (in seconds) between polls of Device A = (poll interval) x (total number of devices).

execute_program_once()

This function is designed to read and execute the instructions stored in the **config.program_memory** data structure member. These instructions are user-configurable and are generated as a result of the responses supplied to the `config.exe` program. Once a hardware reset has occurred and been detected, the *execute_program_once* begins to run. This function executes the set of instructions generated by the `config.exe` program. A hardware reset results in resetting the hardware to its state at power-up (the user specifies this state while the `config.exe` program is running).

Note: The program memory will not change while the firmware is running. However, the contents of the data memory will change during execution.

Each state in the state machine performs one or more operations and either assigns a value to the accumulator (as with reads) or uses the value in the accumulator to modify the contents of the **config** data structure. The first instruction in the program memory is always a read function (necessary to initialize the accumulator).

Function	Description
Read Register Bit	
Program Tag	OPC_READ_REG_BIT
Shift	Any value between 0 and 7 is legal.
Address	Lower 8-bits Only (0–7 valid)
Accumulator	Output
Action	Set accumulator to 1 if the data memory at the given address and shift is 1, otherwise 0.

Function	Description
Read Memory Bit	
Program Tag	OPC_READ_MEM_BIT
Shift	Any value between 0 and 7 is legal.
Address	Lower 8-bits Only (0–255 valid)
Accumulator	Output
Action	Set accumulator to 1 if the data memory at the given address and shift is 1, otherwise 0.

Function	Description
Read Memory Byte	
Program Tag	OPC_READ_MEM_BYTE
Shift	Not used.
Address	Lower 8-bits Only (0–255 valid)
Accumulator	Output
Action	Set accumulator to the value in **config.data_memory**[address].

Function	Description
Read Memory Indexed	
Program Tag	OPC_READ_MEM_INDEXED
Shift	Not used.
Address	Lower 8-bits Only (0–255 valid)
Accumulator	Output
Action	Set accumulator to the value in **config.data_memory**[address + accumulator].

Function	Description
Write Register Bit	
Program Tag	OPC_WRITE_REG_BIT
Shift	Any value between 0 and 7 is legal.
Address	Lower 8-bits Only (0–7 valid)
Accumulator	Input
Action	The register bit specified by the address and shift is cleared. If the accumulator LSB is 1, the bit is set to 1. If the accumulator LSB is 0, the bit is set to 0.

Function	Description
Write Memory Bit	
Program Tag	OPC_WRITE_MEM_BIT
Shift	Any value between 0 and 7 is legal.
Address	Lower 8-bits Only (0–255 valid)
Accumulator	Input
Action	The memory bit specified by the address and shift is cleared. If the accumulator LSB is 1, the bit is set to 1. If the accumulator LSB is 0, the bit is set to 0.

Function	Description
Write Memory Byte	
Program Tag	OPC_WRITE_MEM_BYTE
Shift	Not used.
Address	Lower 8-bits Only (0–255 valid)
Accumulator	Input
Action	**config.data_memory**[address] is set to the value in the accumulator.

Function	Description
Write LED Register Bit	
Program Tag	OPC_WRITE_LED_REG_BIT
Shift	Any value between 0 and 6 is legal.
Address	Lower 8-bits Only (0–7 valid)
Accumulator	Input
Action	Update the register specified by address. Two bits (based on the shift specified) are replaced by three times the accumulator LSB. The bits replaced are in the position specified by: 3 << shift. That is, if the shift is 0 bits, 0 and 1 are replaced; if the shift is 3 bits, 3 and 4 are replaced. **Note:** The maximum shift value is 6 (replace bits 6 and 7).

Function		Description
Write LED Bits		
	Program Tag	OPC_WRITE_LED_BITS
	Shift	Any value between 0 and 6 is legal.
	Address	Lower 8-bits Only (0-7 valid)
	Accumulator	Input
	Action	Update the register specified by address. Two bits (based on the shift specified) are replaced by the two LSBs in the accumulator. The bits replaced are in the position specified by: 3 << shift. That is, if the shift is 0 bits, 0 and 1 are replaced; if the shift is 3 bits, 3 and 4 are replaced. **Note:** The maximum shift value is 6 (replace bits 6 and 7).

Function		Description
Device to State		
	Program Tag	OPC_DEVICE_TO_STATE
	Shift	Not used.
	Address	Lower 8-bits Only (Maximum value is configuration dependent, but it is always less than 64)
	Accumulator	Output
	Action	The accumulator will return the status of the slot. The command Read Device Slot Status (0x04) requires a 4 Byte representation of the slot status. Bytes 0,1 (bits 0 and 1) and 3 (bits 0–2) contain information on the slot status. Beginning with byte 3, then 0 and finally byte 1, the individual bits in the data memory for the specified slot are checked. If a bit is set, then the accumulator is set according to the Table 3.5.

Table 3.5 Accumulator Settings

Byte	Bit	Accumulator	Comment
		0	initialized or no matches
3	1	2	ready for insertion/removal
3	2	3	prepared for operation
0	1	4	device faulty
0	2	5	device rebuilding
0	3	6	failed array
0	4	7	critical array
0	5	8	parity check
0	6	9	predicted fault
0	7	10	not configured
1	0	11	hot spare
1	1	12	rebuild stop

Function		Description
Done		
	Program Tag	OPC_DONE
	Shift	Not used.
	Address	Not used.
	Accumulator	Not used.
	Action	Return to calling program.

3.4.2.2 TWS Data Transfers

The three high-level subroutines that control all TWS data transfers are described briefly in Table 3.6. These functions all provide control, as opposed to the low-level functions that perform the actual data transfer.

Table 3.6 TWS High-Level Subroutines

Three High Level Subroutines	Functional Description
gather_TWS_input()	Initialize (function calls to *tws_write*) peripheral temperature sensors, power supplies, and fans (this may take multiple passes to initialize all devices). Once initialized, subsequent calls read peripheral temperature sensors (function calls to tws_read), convert to Fahrenheit (if required), and update the **config** data structure. Any failure results in the update value being set to 0xFF and requires initialization of sensor on next pass. Return: None. **Note:** No error messages are propagated to the calling process.
gather_LM78_input (LM78 index)	Invoked by *gather_TWS_input* to initialize the LM78 fans and power supplies. Subsequent calls read the status of the fans and power supplies and update the **config** data structure. Errors result in fan (power supply) status set to UNKNOWN and an error counter incremented. If too many errors occur, the device is initialized on next pass through. Return: None. **Note:** No error messages are propagated to the calling process.
background_code_load()	This function will force a soft reset, update the active/inactive image designation or update the firmware from the serial EEPROM. The actions are dependent on the settings of the code load flags. A detailed discussion of these flags is contained in Section 3.4.2.3, “Firmware Update.” Return: None.

Actual data transfers over the TWS bus are accomplished by manipulating the TWS registers. Detailed information on these registers is contained in the *LSI53C040 Enclosure Services Processor Technical Manual* chapter entitled “TWS Registers.” Two TWS buses are available, and the two separate sets of registers are located at addresses 0xFD00 through 0xFDFF in the external memory map. The SAF-TE source code uses the mappings shown in Table 3.7:

Table 3.7 SAF-TE Mappings

Variable Name	Descriptions
tws_data_ptr	Pointer to the TWS Own/Clock/Data Register
tws_car_ptr	Pointer to the TWS Control Register

On a write, the TWS bus transfers data from the external memory to a peripheral. On a read, the TWS bus transfers the measurements or data from the peripheral to external memory. These data are then used to update the **config** data structure. The subroutines *tws_read* and *tws_write* control all of the serial data transfers through the TWS bus and use the algorithms in Figures 2.11 and 2.12 in the *LSI53C040 Enclosure Services Processor Technical Manual.* All reads and writes enable the acknowledgment (bit 0) and clear all interrupts (bit 7) prior to a data transfer. The use of the ACK results in 9 bits being transferred for every byte requested. The ninth bit received is checked to determine that it is a logical 0 (ACK). If this bit is not a logical 0, the read/write subroutines return FALSE (error) to the calling program. If the transfer completes successfully, these subroutines return TRUE (no error).

In general, the low-level TWS data transfer subroutines return TRUE if processing is successful (no error) and FALSE if the directive failed. Exceptions are noted in Table 3.8.

Table 3.8 TWS Low-level Subroutines

Low-Level Subroutines	Functional Description
tws_setup(bus)	Initialize pointers to TWS registers. Must be called prior to any TWS data transfers. If Bus 0: **tws_data_ptr** = TWS_0_DATA (0xFD00) **tws_csr_ptr** = TWS_0_CSR (0xFD01) If Bus 1: **tws_data_ptr** = TWS_1_DATA (0xFD02) **tws_csr_ptr** = TWS_1_CSR (0xFD03) Return None.
tws_init (bus speed, ID on bus)	Initialize the clock speed for specified bus and ID. Return None.
tws_poll_pin()	Return TRUE if Data Register completed operation, FALSE if operation failed.
tws_poll_bb()	Return TRUE if bus free, FALSE if bus busy.
tws_poll_lrb()	Return TRUE if Last Received Bit (the ACK) is logical 0 (successful) or FALSE otherwise (failure).
tws_memory_write (chip ID, address, number of bytes to transfer, pointer to a buffer)	Invoked only on a code load operation. Parses the data to be written into small groups (1–64 bytes) and repeatedly calls the TWS write routine.
tws_write (chip ID, address, number of bytes to transfer, pointer to a buffer)	With the pointers to the bus initialized, this function writes the number of bytes from the given buffer to the specified address and chip ID.
tws_read (chip ID, address, number of bytes to transfer, pointer to a buffer)	With the pointers to the bus initialized, this function reads from the address and chip ID, the specified number of bytes and stores the data into the given buffer.

3.4.2.3 Firmware Update

To upload new firmware to the serial EEPROM, which is attached to the LSI53C040 chip by using a TWS bus, requires that the *CODE_LOAD_ENABLED* option be set during compilation of the SAF-TE source code. A successful upload of new firmware requires:

Step 1. Transfer of the new firmware using the SCSI bus to the external data memory on the LSI53C040 board.

Step 2. Transfer of the firmware from the external data memory over the TWS bus to the serial EEPROM.

Step 3. Error verification.

Step 4. Updating of the serial EEPROM tables.

Step 5. LSI53C040 chip reset.

Target Operations during a Code Load

Uploads of code to the serial EEPROM consist of a SCSI transfer between the Host and the Target's external memory followed by multiple transfers from external memory through the TWS bus to the EEPROM. The global variables are **cl_status**, **cl_addr**, and **cl_byte_count**. These variables contain the control information used to pass data from the SCSI code load subroutine (do_code_load*)* to the background process that transfers data over the TWS bus (background_code_load). The variables are described below:

- **cl_status** specifies the (current) status of the TWS bus. See Table 3.9.
- **cl_addr** is the EEPROM address to write the data. Set to the address in bytes 4 & 5 of a Mode 4 write buffer CDB (*do_code_load)*. See Table 3.10.
- **cl_byte_count** is the number of bytes to write to the EEPROM or the action to implement. The values of this variable and the resulting action are listed in Table 3.11.

Table 3.9 Subroutine background_code_lode

cl_status is set:	**by subroutine *background_code_load***
CLS_SUCCESS	if the memory write over the TWS bus was successful.
CLS_FAILURE_NVM	if the memory write over the TWS bus fails.
CLS_FAILURE_CHECKSUM	if the checksum over the data in code_buffer fails.
CLS_FAILURE_ADDRESS	if the byte count plus the address exceed the maximum image length (measured for the inactive image).

Table 3.10 Subroutine do_code_load

cl_addr is set:	**by subroutine *do_code_load***
CLS_BUSY	if a valid Mode 4 Write Buffer command with CDB[3] = 0 (load new data into external memory), 0xFD (set the flag to update the EEPROM) or 0xFE (set the flag to force a soft Reboot (See notes on Register 0xFE00). Note: The function *background_code_load* will disable interrupts and trip the watchdog timer by using an infinite loop.
CLS_IDLE	if a valid Mode 4 Write Buffer command with CDB[3] = 0xFF.

1. Note: **cl_status** is initialized (by `init_stuff`) to the value CLS_IDLE
2. Note: Mode 4 Write Buffer SCSI transfers require the code load status variable (**cl_status**) be in the idle state (CLS_IDLE). Attempts to perform SCSI transfers when the system is not idle result in a check condition.

Table 3.11 background_code_load operation

cl_byte_count value	background_code_load operation
CLBC_UPDATE_STRUCTURE (0)	Set up TWS bus, set bytes 0 and 1 of **code_buffer**, call *tws_memory_write* to update the EEPROM. If write is successful, set **cl_status** to CLS_SUCCESS; else set **cl_status** to CLS_FAILURE_NVM.
CLBC_REBOOT (1)	Turn off interrupts, set timer, and delay until a soft reset occurs.
Any value > 1	Validate write address. If invalid, set **cl_status** to CLS_FAILURE_ADDRESS. Validate image checksum. If invalid, set **cl_status** to CLS_FAILURE_CHECKSUM. Decrement the byte count pointer because any data transferred as part of a firmware update must have a checksum as the last byte. This byte is not part of the firmware. Update **cl_addr** with the starting address of the inactive image number. Set up TWS bus and call *tws_memory_write.* If write is successful, set **cl_status** to CLS_SUCCESS; else set **cl_status** to CLS_FAILURE_NVM.

Initiator Operations during a Code Load

Firmware uploads require that the Initiator issue a sequence of Mode 4 Write Buffer commands. As specified in the SAF-TE and SCSI-2 specifications, Write Buffer command CDBs use bytes 3–5 for the write address and bytes 6–8 for the transfer length, where the Most Significant Byte (MSB) is presented first.

The SAF-TE firmware requires that the MSB of the transfer length, that is, Byte 6, will always be 0. If this byte is not zero, the CDB will result in a check condition indicating an invalid field in the CDB.

The upper byte of the write address is utilized as a flag for the SAF-TE firmware. The choices for the upper byte and the corresponding actions are shown in Table 3.12.

Table 3.12 Upper Byte Choices

CDB[3] Value	Action
0x00	If **cl_status** not idle, set check condition for code load busy. If CDB invalid, set check condition for invalid CDB. Else, set **cl_status** to busy, set **cl_byte_count,** and populate **code_buffer** with data transmitted from Initiator.
0xFD	If **cl_status** not idle, set check condition for code load busy. Else, set **cl_status** to busy and set **cl_byte_count** to CLBC_REBOOT.
0xFE	If **cl_status** not idle, set check condition for code load busy. Else, set **cl_status** to busy and set **cl_byte_count** to CLBC_UPDATE_STRUCTURE.
0xFF	Set check condition based on value of **cl_status**. If **cl_status** other than in Idle or Busy states, reset to Idle state.

To update the SAF-TE firmware, the Initiator is required to send the following series of Mode 4 Write Buffer SCSI commands to the Target:

- CDB[3] = 0 to update the firmware
- CDB[3] = 0xFF to determine if an error has occurred
- CDB[3] = 0xFE to update the Flash ROM tables
- CDB[3] = 0xFD to reset the LSI53C040 chip.

Additionally, a Request Sense should be issued after the Write Buffer with CDB[3] = 0xFF to verify that the upload was successful.

Remark: The SAF-TE firmware allows transfers of, at most, 1 Kbyte (Tag CODE_BUFFER_SIZE) per SCSI call. However, the firmware itself will always be between 1 Kbyte and 12 Kbytes in size. Therefore, the Initiator issues a sequence of Mode 4 Write Buffer commands each time the firmware is updated.

Pseudo-code for a firmware upload is shown below where **firmware_size** is the size of the firmware to be loaded (in bytes) and

firmware_buffer is a large enough buffer to hold the entire firmware. Once a SCSI transfer has begun, the next SCSI transfer must wait until the firmware from the previous transfer is successfully moved to the serial EEPROM by using the TWS bus. Actual code should check for the BUSY or SUCCESS check conditions. If the status is BUSY, the code should wait until a check condition of SUCCESS or FAILURE is returned and the sense data should be read to verify that the command to update the structure was successful. Finally, the command to issue a REBOOT requires no additional validation, as it is purely a SCSI command.

The pseudo-code is presented below:

```
char send_buffer[1025]; //Generic buffer to hold data to
send.
dataptr = buffer;
status = GOOD;
bytes_left = firmware_size; //Initialize to size of firmware
image
while (firmware_size > 0)
{
  //While there is data left and previous write succeeded,
  // issue the next write buffer command.
      bytesToSend = bytes_left;
      if (bytesToSend > 1024)
        bytesToSend = 1024; // Reset maximum transfer to 1KB

  //Copy bytes from dataptr to send_buffer,
  //compute checksum and append
    Copy(bytesToSend, dataptr, send_buffer);
    checksum_byte = Compute_Checksum(bytesToSend, dataptr);
    send_buffer[bytesToSend +1] = checksum_byte;

     if ((status = SCSI_Write_Buffer(4, bytesToSend+1,
send_pointer)) != GOOD)
       {
         Request_Sense();      //Error on the SCSI Transfer
         return;
       }

    if ((status = SCSI_Write_Buffer(4, 0xFF0000, send_pointer)) !=  GOOD)
       {
         Request_Sense();      //Error on the SCSI Transfer
         return;
       }

       Request_Sense();  //Issue a request sense -if not
```

```
                        //SUCCESSFUL
                        //firmware upload failed.

      if (SENSE != 0x09 || ASC != 0x80 || ASCQ != 0x02)
         return;        // TWS data transfer error

      //Correct Sense Data - continue.
      bytes_left -= bytes_to_send; //Number of bytes left to send.
      dataptr += bytes_to_send;   //Increment the pointer
}

//Firmware upload to serial EEPROM is successful.
//Set cl_byte_count to CLBC_UPDATE_STRUCTURE
if ((status=SCSI_Write_Buffer (Mode_4, 0xFE0000, buffer)) != GOOD)
{
      Request_Sense();    //Error on the SCSI Transfer
      return;
}
//Make sure TWS has completed data transfer - should be in
//SUCCESS state.

//Set cl_byte_count to CLBC_REBOOT - ready to download to
//program memory and begin execution.
if ((status=SCSI_Write_Buffer (Mode_4, 0xFD0000, buffer)) != GOOD)
{
      Request_Sense();     //Error on the SCSI Transfer
      return;

}
```

3.4.3 Interrupts

The LSI53C040 block diagram, Figure 2.1 in the *LSI53C040 Enclosure Services Processor Technical Manual*, illustrates the primary elements of the chip: 80C32 microcontroller, DMA core, SCSI core, 8067 core, SRAM, dual TWS interfaces. In addition to these components, the LSI53C040 contains several timer registers and access to multipurpose I/O (MPIO) lines. The 80C32 microcontroller core has six interrupt sources: Timer 0, Timer 1, Timer 2, Serial Port, External 0, and External 1. These interrupt sources are controlled by writes to the Special Function Registers (SFR). The latter two interrupt sources, External 0 and External 1, provide the means by which interrupts from the other elements in the LSI53C040 chip can be processed.

The LSI53C040 provides three registers to route interrupts to one of the two external interrupts. A bit in the Interrupt Status Register (Program Tag INT_STATUS, 0xFE04) will go high when an interrupt of the appropriate type is pending. The Interrupt Mask Register (Program Tag INT_MASK, 0xFE0D) can be used to prevent an interrupt from being seen (and therefore processed) by the microcontroller core. The Interrupt Destination Register (Program Tag INT_DESTINATION, 0xFE0E) can be used to route any of the interrupts to External Interrupt 0 or External Interrupt 1. The SAF-TE source code disables all interrupts except SCSI and DMA, and routes these to External Interrupt 1 and 0 respectively. As implemented, the 80C32 microcontroller processes the following interrupts as shown in Table 3.13.

Table 3.13 Interrupts Processed by 80C32 Microcontroller

Interrupt Pin	Vector Address	safte.c
0	0x0003	External 0 (mapped to DMA Interrupts, if enabled)
1	0x000B	Timer 0 (SFRs 0x88–0x8A and 0x8C)
2	0x0013	External 1 (mapped to SCSI Interrupts)
3	0x001B	Timer 1 (SFR 0x88, 0x89, 0x8B, and 0x8D)
4	0x0023	Serial Port (SFR 0x98)
5	0x002B	Timer 2 (SFRs 0xC8, 0xCA–0xCD)

In addition to these interrupt sources, the SAF-TE firmware uses the internal watchdog timer (Program Tag WDT_CON, Register 0xFE00) to generate a soft reset (if the code load flag directs a reboot).

3.4.3.1 Interrupt Service Routines (ISR)

The interrupt service routines are executed when the timer 0 counter rolls over from 0xFFFF to 0.

The user data stored in **config.timer_init_hi** and **config.timer_init_low** initialize the 16-bit timer and the timer is started.

Timer 0 ISR – Used to implement a delay on an operation to load code to the serial EEPROM through the TWS bus. The subroutine *`tws_memory_write`* checks the value of the global variable **tim0_cnt** after each burst of data is transferred through the bus to the EEPROM. A delay routine requires three interrupts to Timer 0; that is, this timer must roll over three times before the next burst of data is sent over the TWS bus. For further information, see Section 3.4.2.2, “TWS Data Transfers,” page 3-30.

Function Name: `ir_timer0()`
Return Value: None.

Timer 1 ISR – Void Function. Timer 1 is initialized to be a 16-bit timer but is never utilized.

Function Name: `ir_timer1()`
Return Value: None.

Timer 2 ISR – Void Function. Timer 2 is initialized to be a 16-bit timer and started at the beginning of the main program. Timer 2 is used to clock (transmit and receive) the serial bus. It is initialized for a 40 MHz clock rate with autoreload capability. The output of the serial bus is used for debugging operations.

Function Name: `ir_timer2()`
Return Value: None.

Serial Port ISR – Void Function.

Function Name: `ir_serial()`
Return Value: None.

External Interrupt 0 ISR – If compile time options enable DMA data or commands, this ISR clears the DMA interrupt register, otherwise null routine.

To enable DMA data (commands), the #Define parameter DMA_DATA_IN_ENABLED must be set to TRUE prior to compilation.

> Note: If DMA data or commands are enabled, DMA interrupts are mapped to External Interrupt 0 and SCSI data transfers are handled by the DMA processor instead of through direct I/O. For more information, see the description of the External Interrupt 1 ISR and SCSI data transfer sections.

Function Name: `ir_external0()`
Return Value: None.

External Interrupt 1 ISR – Initiation of SCSI data transfers between the host and target result in the execution of this ISR. Depending on the #define settings, the data transfers between the Initiator and the Target (data memory) are performed by using DMA (see Figure 2.6 in the *LSI53C040 Enclosure Services Processor Technical Manual*) or by software using a Programmed I/O Method (see Figure 2.7 in the *LSI53C040 Enclosure Services Processor Technical Manual*).

The ISR flow diagrams (see Figure 3.2 and Figure 3.3 below) validate that the Target has been selected, the bus is not busy, and the ID bit has been set. Failure of one or more of these conditions results in two actions: an error message is sent to TRACER, and the ISR terminates. Next the ISR determines the Initiator SCSI ID and checks the SCSI control lines for a parity error. If any errors occur, the ISR sends an error message to TRACER and terminates. The SCSI attention (ATN) control line is then tested and these results are possible:

- If high, the target switches to message phase out and checks to see if the received byte (`rcv_byte`) is an Identify (0x80).
- If `rcv_byte` is an Identify (0x80), the message is saved and the code repeats until the ATN is low.
- If `rcv_byte` is not an Identify (0x80), the received byte is checked for the following cases:

No Operation (0x08), loop again until ATN is low; Abort (0x06) and Reset (0x0C); send an error message and terminate ISR; Other, send a reject message to the host.

Once the ATN line is low, the ISR switches to SCSI command phase and signals the Initiator to send the CDB. The subroutine `command_and_data_phases` is initiated. If this routine executes successfully (returns FALSE), the SCSI status followed by a command complete message is sent to the Initiator.

Function Name: `ir_external1()`
Return Value: None.

Figure 3.2 Flow Diagram of ir_external1()

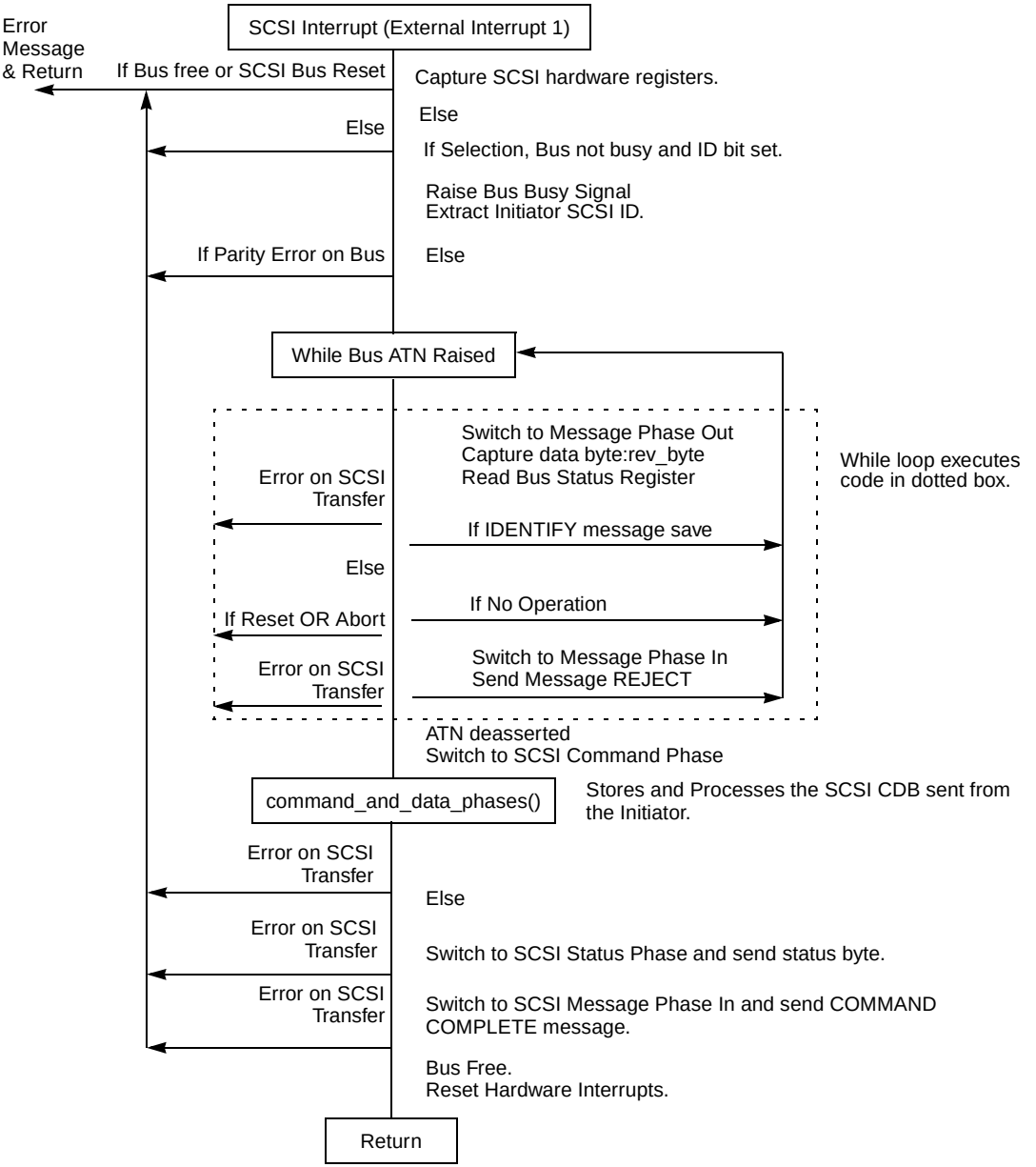
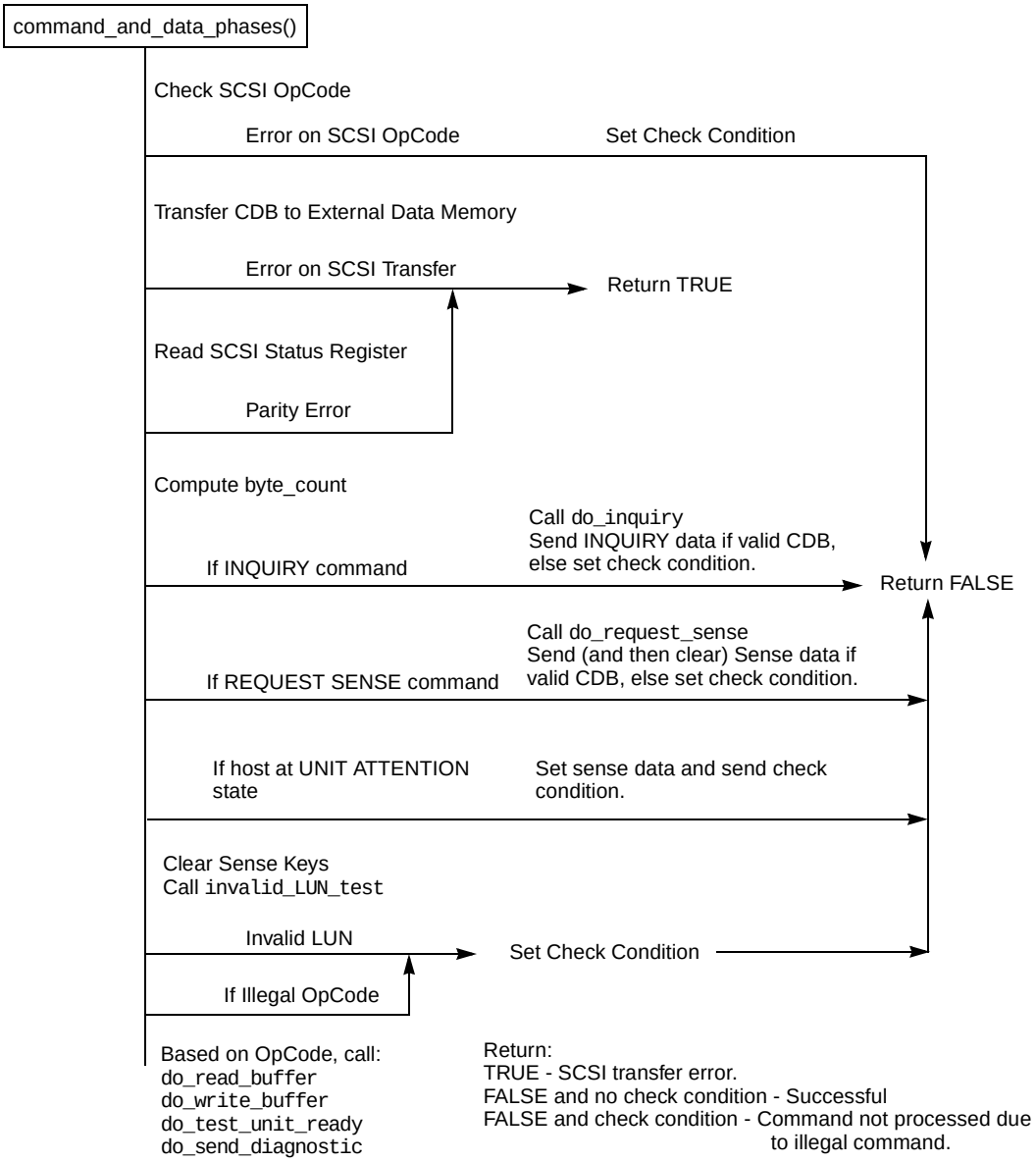


Figure 3.3 Flow Diagram of Command_and_Data_Phases( )



Unlike the other ISRs, `ir_external1` invokes a large number of subroutines. In general, these subroutines return FALSE if the request has been processed successfully and TRUE if the request has failed. For SCSI transfer routines, the validity of the CDB is first verified. If the CDB contains invalid entries, a check condition is generated and FALSE is returned to the calling function.

A brief description of each function is given in the right hand column of Table 3.14 through Table 3.17.

Table 3.14 Interrupt Service Routines - General

General Functions	Description
req_ack	On data transfers, this routine grabs each byte and stores the result in the global variable `rcv_byte`. This variable contains a copy of what was transferred over the SCSI bus, whether the phase is data in or data out. Returns TRUE if a Reset or the SCSI bus is not busy.
command_and_data_phases	Retrieves the CDB contents and processes the SCSI request. Based on the command line compile options, the CDB is written by the DMA engine or through direct I/O. CDBs with invalid OpCodes result in a check condition. Otherwise, processing is continued and the appropriate SCSI subroutines are invoked. Returns TRUE if a transfer is incomplete or if the data transfer of a byte of data failed. Returns FALSE with or without a check condition otherwise.
send_data_bytes	Transmits data from the supplied buffer to the Initiator. Data transfers occur through DMA (if enabled) or through direct I/O.
get_data_bytes	Receives data from the Initiator and stores them in the buffer supplied on the function call.
do_code_load	Specialized function for processing the Vendor Specific Mode 4 Write Buffer commands to upload firmware to the serial EEPROM. Based on the status of the TWS global flag, **cl_status**, and CDB[3], this function will populate the **code_buffer** or generate a check condition. The condition will occur if the TWS bus is busy of if there was an error on a previous Mode 4 Write Buffer command.

Table 3.15 Interrupt Service Routines - SCSI Commands

SCSI Command Subroutines	Functional Description
do_inquiry	Determine if queried LUN is valid and send the inquiry data to the Initiator.
do_request_sense	Determine if in a unit attention state. Send sense data to Initiator and then clear sense data.
invalid_LUN_test	If following an Identify message, returns TRUE if the identify message is not for the LUN specified in the **config** data structure. If not following an Identify, returns TRUE (error) if the queried LUN number is larger than 7.
do_send_diagnostic	Validates CDB. Null functionality.
do_test_unit_ready	Validates CDB. Null functionality.
do_read_buffer	Read data from the selected area of the **config** data structure and return the data to the host using the function send_data_bytes. Invalid Buffer ID values (CDB[2]) result in a check condition and an error condition being passed to the software.
do_write_buffer	Transfers data from the Initiator to the Target's external data memory (accessed through the **config** data structure at offset 0x3B80, variable **code_buffer**) or results in a check condition. Data transfer is accomplished using the **get_data_bytes** function. Mode 01 WRITE BUFFER commands write at most 64 Bytes of data. Invalid requests result in a check condition. Mode 04 WRITE BUFFER commands (update the SEP device) use the address to control whether the command is to write data, force a reboot, update the firmware or query for error messages. The upper address byte of the CDB, (that is, CDB[3]), provides the switch. A maximum of 1024 bytes of data may be transferred to the target from the host with one WRITE BUFFER command. See Section 3.4.2.3, "Firmware Update," page 3-33 for more detailed information.

Table 3.16 Interrupt Service Routines - SCSI Read

SCSI Read Buffer Subroutines	Functional Description
do_read_enclosure_configuration	Sends to the Initiator the current enclosure configuration (stored in **config.enclosure_config_data**)
do_read_enclosure_status	Updates temperature values and sends to the Initiator the current enclosure status (stored in **config.data_memory**)
do_read_global_flags	Sends to the Initiator the current global flag values (stored in **config.data_memory**)
do_read_usage_statistics	Null function.
do_read_device_insertions	Null function.
do_read_device_slot_status	Sends to the Initiator the current device slot status (stored in **config.data_memory**)
do_debug_r_mem	For debugging only - sends a maximum of 64 bytes of data memory at the address specified in the CDB to the Initiator.

Table 3.17 Interrupt Service Routines - SCSI Write

SCSI Write Buffer Subroutines	Functional Description
do_send_global_flags	Updates the contents of the global flag values (stored in **config.data_memory)**.
do_write_device_status	Updates the slot status for each device (except for those devices whose status bytes are all zero).
do_perform_slot_operation	If valid command, update the data memory (**config.data_memory**). Otherwise generate a check condition.
do_set_scsi_id	Updates the data memory (**config.data_memory**) if slot number is smaller than the total number of device slots, otherwise generates a check condition.

Table 3.17 Interrupt Service Routines - SCSI Write (Cont.)

SCSI Write Buffer Subroutines	Functional Description
do_set_fan_speed	Null function.
do_activate_power_supply	Null function.
do_debug_w_sbc	For debugging only - allows control of the subroutine *execute_program_once.*

3.4.4 Error Reporting

The SAF-TE source code's individual functions about error reporting mechanisms have been reviewed in earlier sections. Errors that occur during a SCSI data transfer result in a check condition or, in some cases a bus free. Errors that occur during a TWS data transfer manifest themselves by invalidating the data to be updated in the config data structure or through the code load status flag **cl_status** and a corresponding check condition. No other error reporting is provided.

3.5 Frequently Asked Questions (FAQ)

Table 3.18 lists possible questions that might be raised when using this source code along with appropriate answers.

Table 3.18 Source Code Issues

Question	Comment
How is the file `safte.hex` generated?	Use the makefile with the target specified as SAFTE. See Section 3.4.1, "Compilation Instructions for safte.c," and the compiler documentation for details.
How is the boot code placed on the TWS memory device?	For a new board, the chip can be placed in a TWS programmer or (for some boards) downloaded over the TWS bus. For boards with firmware in place, follow the instructions in the Section 3.4.2.3, "Firmware Update," page 3-33.
Where is the boot code stored once it is downloaded?	It is stored in a serial EEPROM.
What is the structure of the entire boot image?	`loader.hex + bootload.hex`

Table 3.18 Source Code Issues (Cont.)

Question	Comment
How do we change the flags used during compilation?	Currently, the source code #Define statements must be modified.
How are SCSI errors reported?	Errors during a SCSI transfer or with a CDB are reported by setting the appropriate check condition. Refer to Section 2.1.3 in the *LSI53C040 Enclosure Services Processor SAF-TE Firmware User's Guide* for more information.
How are TWS errors reported?	Invalid entries in **config** data structure, code load status variable cl_status are set to CLS_FAILURE_XXX. The XXX describes the failure and thus check conditions are set. The initiator may retrieve information by issuing a Mode 4 Write Buffer command with CDB[3] = 0xFF followed by a Request Sense command.
Is Timer 2 utilized?	Yes. Timer 2 is used to clock the transmission and reception of data on the serial bus.
What is the difference between a soft reset and a hard reset?	A hard reset is a power on reset. The chip, all registers, and external data memory are initialized using the contents of the serial EEPROM. A soft reset occurs when the watchdog timer expires. This forces only a chip reset. Note the code restarts but not all of the registers are initialized. See Chapter 2 in the *LSI53C040 Enclosure Services Processor Technical Manual* for more information.
What is a shadow register and how is it used?	It holds a duplicate of a register value in xdata data memory. Use of shadow registers allows for faster access in some cases.
Prior to transferring the data sent by a Mode 4 Write Buffer command over the TWS bus, the byte count variable is decremented by 1. Why?	All Mode 4 Write Buffer commands require that the last data byte transferred be the checksum of all previous data bytes. However, this checksum is **not** data to be saved in the serial EEPROM.
If the firmware is, say 10 Kbytes, but our SCSI data buffer is only 1025 Bytes, how is the firmware updated?	The Initiator, or an application program, must break the firmware into segments and write one segment at a time. See the discussion on Section 3.4.2.3, “Firmware Update,” page 3-33 for more information.

Chapter 4
SAF-TE Command Implementation

This chapter documents the SAF-TE commands supported by the LSI53C040, and the default firmware settings or user requirements for each command. These topics are discussed:

- Section 4.1, “SCSI Commands,” page 4-1. The LSI53C040 SAF-TE firmware supports these SCSI commands: Inquiry, Read Buffer, Request Sense, Send Diagnostic, Test Unit Ready, and Write Buffer.
- Section 4.2, “SAF-TE Read Buffer Commands,” page 4-7
- Section 4.3, “SAF-TE Write Buffer Commands,” page 4-18
- Section 4.4, “Unsupported SAF-TE Commands,” page 4-25

Please note that the LSI53C040 only supports LUN 0 at this time.

4.1 SCSI Commands

This section provides more detailed information about all the SCSI commands that are supported by the LSI53C040 enclosure services processor.

4.1.1 Inquiry

The host uses the Inquiry command to request parameter information from the enclosure. Table 4.1 shows the response data format from the Inquiry command.

Table 4.1 Inquiry Command Response Data

Byte	Bit	Description	Returned Value/ Notes
Byte 0	[7:5]	Peripheral Qualifier	0b000 if LUN 0 (the only valid LUN) is selected 0b011 if LUN 0 is not selected
	[4:0]	Peripheral Device Type	0x03 (SCSI Processor Device) if LUN 0 (the only valid LUN) is selected 0x1F (No device type) if LUN 0 is not selected
Byte 1	[7:0]	00h	Returns 0x00
Byte 2	[7:3]	0	0
	[2:0]	ANSI Approved Version	0x02 – Compliance with ANSI SCSI-2 specification
Byte 3	[7:4]	0	0
	[3:0]	Response Data Format	0x02 – Format defined in ANSI SCSI-2 specification
Byte 4	[7:0]	Additional Length	0x36 = 54 bytes
Byte 5	[7:0]	Reserved	Returns 0x00
Byte 6	[7:0]	Reserved	Returns 0x00
Byte 7	[7:0]	00h	Returns 0x00
Bytes 8–15	[7:0]	Vendor Identification	8-byte ASCII string defined in the configuration program
Bytes 16–31	[7:0]	Product Identification	16-byte ASCII string defined in the configuration program
Bytes 32–35	[7:0]	Firmware Revision Level	Returns a four-byte ASCII string representing the current SAF-TE firmware revision level
Bytes 36–42	[7:0]	Enclosure Unique Identifier	Returns a seven-byte ASCII ID number as defined in the configuration program
Byte 43	[7:0]	Channel Identifier	Returns a single ASCII character as defined in the configuration program
Bytes 44–49	[7:0]	SAF-TE Interface Identification String	ASCII string of “SAF-TE”
Bytes 50–53	[7:0]	SAF-TE Specification Revision Level	ASCII string of “1.00”

4.1.2 Read Buffer

The Read Buffer command is used to receive data from the LSI53C040 SAF-TE Processor. The data returned is dependent upon the content of the SAF-TE operation code field. These commands are included:

- Read Enclosure Configuration (SAF-TE operation code 0x00)
- Read Enclosure Status (0x01)
- Read Device Slot Status (0x04)
- Read Global Flags (0x05)

The format of these commands is described in Table 4.2 below. The mode field is 0x01 to indicate that a SAF-TE command is being sent. The transfer length is dependent upon which SAF-TE data is being returned.

Table 4.2 Read Buffer Data Format

Bit # => / Byte #	7	6	5	4	3	2	1	0
0	SCSI Operation Code (0x3C)							
1	Logical Unit Number			Reserved		Mode (0x01)		
2	SAF-TE Operation Code							
3	0x00							
4	0x00							
5	0x00							
6	0x00							
7	Transfer Length MSB							
8	Transfer Length LSB							
9	0x00							

4.1.3 Request Sense

Table 4.3 provides the sense key information supported by the LSI53C040 SAF-TE firmware.

Table 4.3 Sense Key Information

Sense Key	ASC	ASCQ	Error Condition
0x00			No Sense, No Error Condition
0x05			Illegal Request
	0x20	0x00	Invalid Command Operation Code
	0x24	0x00	Invalid Field in CDB
	0x25	0x00	Logical Unit not Supported
	0x26	0x02	Invalid SEP Command in Write Buffer Data Packet
0x06			Unit Attention
	0x29	0x00	Power-On, Reset, or Bus Device Reset Occurred
	0x3F	0x01	Microcode Changed
0x09			Vendor-Specific
	0x80	0xFF	Code Load Busy
	0x80	0x00	Code Load Idle
	0x80	0x01	Code Load Busy Writing
	0x80	0x02	Code Load Success
	0x80	0x03	Code Load Failure Bad Address
	0x80	0x04	Code Load Failure Bad Checksum
	0x80	0x05	Code Load NVM Write Failure

4.1.4 Send Diagnostic

This command is treated as a no operation and returns the status of GOOD.

4.1.5 Test Unit Ready

This command is implemented according to the SAF-TE specification.

4.1.6 Write Buffer

The Write Buffer command is used to send SAF-TE commands to the LSI53C040 SAF-TE Processor. These commands are included:

- Write Device Slot Status (0x10)
- Perform Slot Operation (0x12)
- Send Global Flags (0x15)

The format of these commands is described in Table 4.4 below. The mode field is 0x01 to indicate that a SAF-TE command is being sent. The transfer length is dependent upon which SAF-TE command is being sent.

Table 4.4 Write Buffer Data Format

Bit # => / Byte #	7	6	5	4	3	2	1	0
0	SCSI Operation Code (0x3B)							
1	Logical Unit Number			Reserved		Mode (0x01)		
2	0x00							
3	0x00							
4	0x00							
5	0x00							
6	0x00							
7	Transfer Length MSB							
8	Transfer Length LSB							
9	0x00							

The Write Buffer command is also used to update the SAF-TE firmware (stored in TWS Flash ROM). When used to upload firmware, the Write Buffer command format appears as follows:

Table 4.5 Write Buffer Data Format (Updating SAF-TE Firmware)

Bit # => / Byte #	7	6	5	4	3	2	1	0
0	Operation Code (0x3B)							
1	Logical Unit Number			Reserved		Mode (0x04)		
2	0x00							
3	Flag Byte							
4	Buffer Offset MSB							
5	Buffer Offset LSB							
6	0x00							
7	Transfer Length MSB							
8	Transfer Length LSB							
9	0x00							

The Mode field is 0x04 to indicate that SAF-TE firmware is being sent.

The Flag Byte can be one of the following values:

- 0x00 - upload firmware
- 0xFD - reset the LSI53C040 to run the new firmware
- 0xFE - upload firmware complete; update Flash ROM tables to use the new firmware
- 0xFF - request status of firmware upload; returned using next SCSI Request Sense command

The transfer length and buffer offset fields are used only when the Flag Byte is zero. When the Flag Byte is nonzero, the transfer length and buffer offset fields must contain zeros. The transfer length is a number between 0x02 and 0x4001, indicating how much data (including a 1-byte checksum) is being transferred. The buffer offset is a number between 0x00 and 0x3FFF, indicating which locations in the Flash ROM are to be updated.

4.2 SAF-TE Read Buffer Commands

This section provides information about all the various read buffer commands that can be sent to the LSI53C040 core.

4.2.1 Read Enclosure Configuration (0x00)

The application agent sends this command to the LSI53C040 to inquire about the number and type of system components in the enclosure. The LSI53C040 determines and returns this information based on the enclosure settings the user defines in the configuration program. At present, no vendor specific bytes are returned. Table 4.6 shows the return values:

Table 4.6 Read Enclosure Configuration Return Values

Byte	Bits	Field Description	Notes
0	[7:0]	Number of Fans (f)	Defined in Configuration Utility
1	[7:0]	Number of Power Supplies (p)	Defined in Configuration Utility
2	[7:0]	Number of Device Slots (d)	Defined in Configuration Utility
3	[7:0]	Door Lock Installed	Defined in Configuration Utility
4	[7:0]	Number of Temperature Sensors (t)	Defined in Configuration Utility
5	[7:0]	Audible Alarm	Defined in Configuration Utility

Table 4.6 Read Enclosure Configuration Return Values (Cont.)

Byte	Bits	Field Description	Notes
6	7	Celsius/Fahrenheit	Defined in Configuration Utility
	[6:4]	Reserved	
	[3:0]	Number of Thermostats	Defined in Configuration Utility
7 through 62	[7:0]	Reserved	Returns 00h
63	[7:0]	Number of Vendor Specific Bytes (v)	Returns 00h
64 through xx	[7:0]	Vendor Specific	Not supported

4.2.1.1 Fans (f)

This field contains the binary representation of the number of fans in the enclosure. This information reserves the appropriate number of bytes in the Read Enclosure Status field. The user defines this number in the configuration program.

4.2.1.2 Power Supplies (p)

This field contains the binary representation of the number of power supplies in the enclosure. This information reserves the appropriate number of bytes in the Read Enclosure Status field. The user defines this number in the configuration program.

4.2.1.3 Device Slots (d)

This field contains the binary representation of the number of available device slots in the enclosure. This information reserves the appropriate number of bytes in the Read Enclosure Status field. The user defines this number in the configuration program.

4.2.1.4 Door Lock

This field indicates whether the enclosure has a door lock. If there is no door lock, this field is 0. If a door lock is present, this field is 1. The user defines this field in the configuration program.

4.2.1.5 Number of Temperature Sensors (t)

This field contains the binary representation of the number of integer temperature sensors. This information reserves the appropriate number of bytes in the Read Enclosure Status field. This type of sensor will be connected to one of the TWS buses to transfer this integer value to the LSI53C040. The DS1621 is one example. If the user does not select the DS1621, the LM75, or the LM78 in the TWS device port mapping section of the firmware configuration program, then it is assumed that no integer temperature sensors are attached.

4.2.1.6 Audible Alarm

This field indicates whether the enclosure has a speaker. If there is no speaker, this field is 0. If a speaker is present, this field is 1. The user defines this field in the configuration program.

4.2.1.7 Celsius/Fahrenheit

This field indicates whether the integer temperatures (if there are any) will be reported in degrees Fahrenheit or Celsius. It is selectable by the user in the configuration program. A value of 1 indicates Celsius, and a value of 0 indicates Fahrenheit.

4.2.1.8 Number of Thermostats

This field indicates the number of binary temperature monitors. The user defines this field in the configuration program.

4.2.2 Read Enclosure Status (0x01)

The LSI53C040 processor returns the operational status of the these components in the enclosure:

- Fans
- Power supplies
- Slot SCSI IDs
- Door locks
- Speakers
- Integer temperatures
- Binary temperatures

Table 4.7 shows the Read Enclosure Status Return Values.

Table 4.7 Read Enclosure Status Return Values

Byte	Field Description	Notes
0	Fan 0 Status	Returns either: 0x00 Fan is operational 0x01 Fan is malfunctioning 0x02 Fan is not installed 0x80 unknown status, or status not reportable
f – 1	Fan f – 1 Status	Same as above
f	Power Supply 0 Status	Returns either: 0x00 Power Supply is operational and on 0x01 Power Supply is operational and off 0x10 Power Supply is malfunctioning and commanded on 0x11 Power Supply is malfunctioning and commanded off 0x20 Power Supply is not present 0x21 Power Supply is present 0x80 unknown status, or status not reportable
f + p – 1	Power Supply p – 1 Status	Same as above
f + p	Device Slot 0 SCSI ID	Returns binary encoded value of the SCSI ID
f + p + d – 1	Device Slot d – 1 SCSI ID	Same as above
f + p + d	Door Lock Status	Returns either: 0x00 Door is currently locked 0x01 Door is currently unlocked, or door lock not installed 0x80 Unknown status, or status not reportable
f + p + d + 1	Speaker Status	Returns either: 0x00 Speaker is currently off or no speaker installed 0x01 Speaker is currently on
f + p + d + 2	Temperature 0	Returns the integer value (0–255) of the DS1621 or LM75 temp sensor. Additionally, if no sensor is installed, no bytes are dedicated.
f + p + d + t + 1	Temperature t – 1	Same as above
f + p + d + t + 2	Temperature Out of Range Flags 1	Sets the ETA (bit 7) if temperature alert or 0 if no alert. See Section 4.2.2.7, “Temperature Out Of Range,” page 4-14.

Table 4.7 Read Enclosure Status Return Values (Cont.)

Byte	Field Description	Notes
f + p + d + t + 3	Temperature Out of Range Flags 2	See Section 4.2.2.7, “Temperature Out Of Range,” page 4-14.
f + p + d + t + 4	Number of Vendor Specific Bytes	0x00
f + p + d + t + 5	Vendor Specific	Not supported

4.2.2.1 Fan Status

For each fan in the enclosure, the configuration program defines whether the fan status is determined by either one or two input MPIO pins, or by an LM78. The Read Enclosure Configuration (0x00) command indicates whether fans are attached. If no fans are attached, this field is truncated from the Read Enclosure Status return values shown in Table 4.7. Based upon the input status, the LSI53C040 will return values. Table 4.8 shows these values:

Table 4.8 Fan Status Return Values

Value	Status
0x00	Fan is operational
0x01	Fan is malfunctioning
0x02	Fan is not installed
0x80	Unknown status, or status not reportable

The configuration program maps what response above should be returned for a single-bit input pattern of 0 or 1 and for a dual-input bit pattern of 0x00, 0x01, 0x10, or 0x11, or by the value read from the LM78.

4.2.2.2 Power Supply Status

For each power supply in the enclosure, the configuration program has defined whether the power supply status is determined by either one or two input MPIO pins, or by an LM78. The Read Enclosure Configuration (0x00) command indicates whether power supplies are present. If no power supplies are present, this field is truncated from the Read

Enclosure Status return values shown in Table 4.7. Based upon the status, the LSI53C040 will return values. These values are shown in Table 4.9:

Table 4.9 Power Supply Status Return Values

Value	Status
0x00	Power Supply is operational and on
0x01	Power Supply is operational and off
0x10	Power Supply is malfunctioning and commanded on
0x11	Power Supply is malfunctioning and commanded off
0x20	Power Supply is not present
0x21	Power Supply is present
0x80	Unknown status, or status not reportable

The configuration program maps what response above will be returned for a single-input bit pattern of 0 or 1 and for a dual-input bit pattern of 0x00, 0x01, 0x10, or 0x11, or by the value read from the LM78.

4.2.2.3 Device Slot SCSI ID

A SCSI ID (integer ID) is reported for each device slot in the enclosure. The configuration program maps what SCSI ID is assigned to each device slot. The SCSI ID is reported even if the drive is not present in a slot.

4.2.2.4 Door Lock Status

The state of one MPIO pin determines the door lock status. If the user has defined a host-controllable or a monitorable door lock in the configuration program, the LSI53C040 will return values. Table 4.10 lists the values and their status:

Table 4.10 Door Lock Status Return Values

Value	Status
0x00	Door is currently locked
0x01	Door is currently unlocked or no door lock is installed
0x80	Unknown status, or status not reportable

The configuration program maps what response above should be returned for single-input bit pattern of 0 or 1. If no door lock is defined, the LSI53C040 will return 0x01.

4.2.2.5 Speaker Status

The state of one MPIO pin determines the speaker status. If the user has defined the speaker status in the configuration program, the LSI53C040 will return values. Table 4.11 lists these values and their status:

Table 4.11 Speaker Status Return Values

Value	Status
0x00	Speaker is currently off (or no speaker installed)
0x01	Speaker is currently on

The configuration program maps what response above will be returned for a single-bit input pattern of 0 or 1. If speaker is not defined, the LSI53C040 will return 0x00.

4.2.2.6 Temperature

The integer (0 to 255) value of a temperature sensor(s) in degrees Fahrenheit or Celsius determines the temperature status. It is assumed that this type of sensor will be connected to one of the TWS buses as

the means for transferring this integer value to the LSI53C040. The DS1621 is one example. The Read Enclosure Configuration (0x00) command indicates whether a temperature sensor is attached. If no temperature sensors are attached, this field is truncated from the Read Enclosure Status return values shown in Table 4.7. The default state for this is field is 0. This field returns 255 if an error has occurred.

4.2.2.7 Temperature Out Of Range

This status returns whether an abnormal temperature has been detected on any thermostat hardware that only returns a binary value. Since up to 15 thermostat temperature sensors can be attached to the enclosure, up to 15 MPIO pins would be required.

A value of 1 on any of the dedicated MPIO pins indicates an abnormal temperature, and the corresponding flag will be set. When a value of 1 occurs on any of the dedicated MPIO pins, the ETA bit will be set (bit 7) in the Temperature Out of Range Flags (1 byte).

4.2.3 Read Device Slot Status (0x04)

This command returns information on the current state of each device/slot. The field that follows the device status bytes is a one byte field and indicates the number of Vendor Specific bytes to follow. This field will always be zero. Four bytes are associated with each device slot. Table 4.12 summarizes each of those bytes.

Table 4.12 Read Device Slot Status Command Return Values

Byte	Bit	Description	Notes
Byte 0	0	No Error Flag	Returns value as set by Write Device Slot Status Command
	1	Device Faulty Flag	Returns value as set by Write Device Slot Status Command
	2	Device Rebuilding Flag	Returns value as set by Write Device Slot Status Command
	3	In Failed Array Flag	Returns value as set by Write Device Slot Status Command
	4	In Critical Array Flag	Returns value as set by Write Device Slot Status Command
	5	Parity Check Flag	Returns value as set by Write Device Slot Status Command
	6	Predicted Fault Flag	Returns value as set by Write Device Slot Status Command
	7	No Drive Flag	Returns value as set by Write Device Slot Status Command
Byte 1	0	Hot Spare Flag	Returns value as set by Write Device Slot Status Command
	1	Rebuild Stopped Flag	Returns value as set by Write Device Slot Status Command
	[2:7]	Reserved	Returns 0x00
Byte 2	[0:7]	Reserved	Returns 0x00
Byte 3	0	(Slot) Device Inserted Flag	Returns either: 0 – no device inserted in slot 1 – device inserted in slot
	1	(Slot) Prepared for Insertion/Removal Flag	Returns either: 0 – device power is on (slot not ready for insertion/removal) 1 – device power is off (slot ready for insertion/removal)
	2	(Slot) Prepared for Operation Flag	Returns either: 0 – device power is off (slot not prepared for operation) 1 – device power is on (slot prepared for operation)
	[3:7]	Reserved	Returns 0x00

Table 4.13 lists the default slot status values that are set at power-on or reset.

Table 4.13 Power-On/Reset Default Slot Status

Value	Status
Byte 0	0x01
Byte 1	0x00
Byte 2	0x00
Byte 3	0x02

4.2.3.1 Device Inserted Bit

If the Device Present option was selected in the configuration program, this field returns the current state of a drive (whether or not it is installed in the device slot).

4.2.3.2 Prepared for Insertion/Removal

Setting this bit indicates the slot is ready for drive insertion or removal. It is the complement of the Prepared for Operation bit. See Perform Slot Operation for when this bit is set.

4.2.3.3 Prepared for Operation

This bit indicates that a drive has been inserted in a slot and is ready for operation. It is the complement of the Ready for Insertion/Removal bit. See Section 4.3.2, “Perform Slot Operation (0x12),” for when this bit is set.

4.2.4 Read Global Flags (0x05)

The Read Global Flags command is used to read from the LSI53C040 the most recent state of the global flags received in the Send Global Flags command (refer to Table 4.18). Sending this command will not modify the state of any global flag. Table 4.14 lists the return values for the Read Global Flags command.

Table 4.14 Read Global Flag Bytes

Byte	Bit	Global Bit Descriptions	LSI53C040 Action
Byte 0 (Global Flag 1)	0	Audible Alarm Control	Drives/Monitors an MPIO pin connected to an alarm signal (1 for on, 0 for off)
	1	Global Failure Indication	Drives LED
	2	Global Warning Indication	Drives LED
	3	Enclosure Power	Not Implemented
	4	Cooling Failure	Not Implemented
	5	Power Failure	Not Implemented
	6	Drive Failure	Drives LED
	7	Drive Warning	Drives LED
Byte 1 (Global Flag 2)	0	Array Failure	Drives LED
	1	Array Warning	Drives LED
	2	Enclosure Lock	Drives/Monitors an MPIO pin connected to a door lock
	3	Identify Enclosure	Drives LED
	[4:7]	Reserved	
Byte 2 (Global Flag 3)	[0:7]	Reserved	
Bytes 3–15		Reserved	

4.3 SAF-TE Write Buffer Commands

This section provides information about all the various write buffer commands that can be sent to the LSI53C040 core.

4.3.1 Write Device Slot Status (0x10)

This command informs the LSI53C040 of the state of each slot and activates device status LEDs. In general, the Write Device Slot Status is set by the RAID controller or host since it knows the status of the devices in each slot. Three bytes are associated with each device slot. Bytes 1, 2, and 3 contain the desired state for the device in slot #0; Bytes 4, 5, and 6 contain the desired state for the device in slot #1, etc. Table 4.15 summarizes these bytes and the associated actions of the LSI53C040.

Table 4.15 Write Device Slot Status Flag Bytes

Byte	Bit	State	Bit Descriptions	LSI53C040 Action
Byte 0			Operation Code (0x10)	
Byte n + 0	0	00	No Error Flag	Drives fault light LED(s) according to blink pattern
	1	03	Device Faulty Flag	Drives fault light LED(s) according to blink pattern
	2	04	Device Rebuilding Flag	Drives fault light LED(s) according to blink pattern
	3	05	In Failed Array Flag	Drives fault light LED(s) according to blink pattern
	4	06	In Critical Array Flag	Drives fault light LED(s) according to blink pattern
	5	07	Parity Check Flag	Drives fault light LED(s) according to blink pattern
	6	08	Predicted Fault Flag	Drives fault light LED(s) according to blink pattern
	7	09	No Drive Flag	Drives fault light LED(s) according to blink pattern

Table 4.15 Write Device Slot Status Flag Bytes (Cont.)

Byte	Bit	State	Bit Descriptions	LSI53C040 Action
Byte n + 1	0	10	Hot Spare Flag	Drives fault light LED(s) according to blink pattern
	1	11	Rebuild Stopped Flag	Drives fault light LED(s) according to blink pattern
	[2:7]		Reserved	
Byte n + 2	[0:7]		Reserved	

Note: Byte numbers for the Write Device Slot Status Flags are determined by using "n" = device slot number. Therefore, the above information repeats for each device slot specified.

The LSI53C040 will drive zero, one, or two LEDs for each device slot, depending on the option chosen in the configuration program. Table 4.16 shows the current default settings for each bit description in the Write Device Slot Status command for both the one-LED and two-LED options:

Table 4.16 Default LED Settings for Write Device Slot Status Flags

State	Bit Description	One-LED Option	Two-LED Option	
			LED 1	LED 2
0	Default/Nothing to Report	Off	Off	Off
1	Prepare for insertion/removal	Off	On	On
2	Prepare for operation	On	Off	Slow
3	Device Faulty	Slow	On	Off
4	Device Rebuilding	Slow	Off	Fast
5	In Failed Array	Slow	Fast	On
6	In Critical Array	Slow	Slow	Off
7	Parity Check operation	Slow	On	Off
8	Predicted Fault Failure	Slow	Fast	Off
9	No drive inserted	Off	On	Fast

Table 4.16 Default LED Settings for Write Device Slot Status Flags (Cont.)

State	Bit Description	One-LED Option	Two-LED Option	
			LED 1	LED 2
10	Hot Spare	Off	Fast	Fast
11	Rebuild Stopped	Slow	Off	On
12	Identify Slot	Fast	Slow	Fast

The bit descriptions are shown in increasing priority order. If more than one bit is set by the Write Device Slot Status command, the bit with the highest priority dictates the LED blink pattern.

4.3.2 Perform Slot Operation (0x12)

This command performs a specific operation on an intended device slot. In compliance with the SAF-TE specification, only one of these bits should be set at a time.

Table 4.17 Perform Slot Operation Flags

Byte	Bit	Bit Descriptions	Action
0		Operation Code (0x12)	
1		Slot Number	
2	0	Prepare for Operation Flag	Controls power to a device slot 0 – Turn slot power off 1 – Turn slot power on
	1	Prepare for Insertion/Removal Flag	Controls power to a device slot 0 – Turn slot power on 1 – Turn slot power off
	2	Identify Flag	Drives LED(s) according to blink pattern
	[3:7]	Reserved	
3–63		Reserved	

4.3.2.1 Prepare for Operation

This bit is set by the host to indicate that a drive has been inserted in a slot and is to be made ready for operation (that is, powered on). If the Ready Device for Use option is selected in the configuration program, the assigned MPIO pin will be asserted to turn slot power on.

These conditions apply if the Device Present option is chosen in the configuration program:

- If a device is present in the slot, and this bit is set, power will be turned on (by the MPIO pin). The Prepared for Operation bit will be set and the Ready for Insertion/Removal bit will be cleared in the Read Device Slot Status command.
- If a device is not present in the slot, and this bit is set, power will not be turned on. The Ready for Insertion/Removal bit will be set and the Prepared for Operation bit will be cleared in the Read Device Slot Status command.

These conditions apply if the Device Present option is not chosen in the configuration program:

- Power is applied to the slot, the Prepared for Operation bit will be set, and the Ready for Insertion/Removal bit will be cleared in the Read Device Slot Status command.

4.3.2.2 Prepare for Insertion/Removal

This bit is set by the host to indicate that the slot should be made ready for drive insertion or removal (that is, powered off). If the Prepare Device for Insertion/Removal option is selected in the configuration program, the assigned MPIO pin will be asserted to turn slot power off.

These conditions apply if the Device Present option is chosen in the configuration program:

- If a device is present in the slot, and this bit is set, power will be turned off (by the MPIO pin). The Prepared for Operation bit will be cleared and the Ready for Insertion/Removal bit will be set in the Read Device Slot Status command.

- If a device is not present in the slot, and this bit is set, power will remain off. The Ready for Insertion/Removal bit will be set and the Prepared for Operation bit will be cleared in the Read Device Slot Status command.

These conditions apply if the Device Present option is **not** chosen in the configuration program:

- Power will be turned off, the Prepared for Operation bit will be cleared, and the Ready for Insertion/Removal bit will be set in the Read Device Slot Status command.

4.3.2.3 Identify

This will drive an external LED(s) according to the blink pattern for the specific device slot. If one LED is chosen, the Identify Slot bit is set to the fast blink rate. If two LEDs are chosen, the LED1 bit is set to the slow blink rate and the LED2 bit is set to the fast blink rate.

4.3.3 Send Global Flags Command (0x15)

This command is used to send commands that apply to the enclosure. The Read Global Flags command (Table 4.14) is used to read the most current state of the global flags sent with this command:

Table 4.18 Send Global Flag Bytes

Byte	Bit	Global Bit Descriptions	LSI53C040 Action
Byte 0		Operation Code (0x15)	
Byte 1 (Global Flag 1)	0	Audible Alarm Control	Drives/Monitors an MPIO pin connected to an alarm signal (1 for on, 0 for off)
	1	Global Failure Indication	Drives LED
	2	Global Warning Indication	Drives LED
	3	Enclosure Power	Not Implemented
	4	Cooling Failure	Not Implemented
	5	Power Failure	Not Implemented
	6	Drive Failure	Drives LED

Table 4.18 Send Global Flag Bytes (Cont.)

Byte	Bit	Global Bit Descriptions	LSI53C040 Action
Byte 1 (Global Flag 1)	7	Drive Warning	Drives LED
Byte 2 (Global Flag 2)	0	Array Failure	Drives LED
	1	Array Warning	Drives LED
	2	Enclosure Lock	Drives/Monitors an MPIO pin connected to a door lock
	3	Identify Enclosure	Drives LED
	[4:7]	Reserved	
Byte 3 (Global Flag 3)	[0:7]	Reserved	

4.3.3.1 Audible Alarm Control

This bit is set to sound an alarm. If a controllable alarm is selected in the configuration program, setting this bit will sound the alarm. The alarm is turned off by clearing this bit.

4.3.3.2 Global Failure and Warning Indication

These bits are set to indicate a global failure or warning condition. If the Global Enclosure Status LED option is selected in the configuration program, setting of either the Global Failure or Warning Indication bits will drive the assigned LED as shown in Table 4.19:

Table 4.19 Global Failure/Global Warning LED Options

Global Failure	Global Warning	LED
0	0	Off
0	1	Slow
1	0	Fast
1	1	On

4.3.3.3 Drive Failure and Warning

These bits are set to indicate a drive failure or warning condition. If the Global Drive Status LED option is selected in the configuration program, setting of either the Drive Failure or Warning Indication bits will drive the assigned LED as shown in Table 4.20:

Table 4.20 Drive Failure/Drive Warning LED Options

Drive Failure	Drive Warning	LED
0	0	Off
0	1	Slow
1	0	Fast
1	1	On

4.3.3.4 Array Failure and Warning

These bits are set to indicate an array failure or warning condition. If the Global Array Status LED option is selected in the configuration program, setting of either the Drive Failure or Warning Indication bits will drive the assigned LED as shown in Table 4.21:

Table 4.21 Array Failure/Array Warning LED Options

Array Failure	Array Warning	LED
0	0	Off
0	1	Slow
1	0	Fast
1	1	On

4.3.3.5 Enclosure Lock

This bit is set to lock the enclosure. If a lock is selected in the configuration program, setting this bit will lock the enclosure. The enclosure is unlocked by clearing this bit.

4.3.3.6 Identify Enclosure

This bit is set to drive any global enclosure identify signal.

4.4 Unsupported SAF-TE Commands

The unsupported SAF-TE commands are:

- Read Usage Statistics (0x02)
- Read Device Insertions (0x03)
- Set SCSI ID (0x11)
- Set Fan Speed (0x13)
- Activate Power Supply (0x14)

Chapter 5
Configuration Data and the Configuration Utility

This chapter describes the Configuration Utility and includes these topics:

5.1 Using the Configuration Utility

The LSI53C040 SAF-TE firmware includes a configuration utility (`config.exe`) that maps specific enclosure monitoring functions to the MPIO and MPLED pins and sets up operating parameters for the specific enclosure environment. Also included with the firmware are the `bootload.hex` file and the `safte.hex` file. The configuration utility is a DOS-based program that asks the designer a series of questions about the enclosure design. The program uses this information to create an Intel compatible hex file called `config.hex`. The user concatenates the `config.hex` file with the Intel compatible SAF-TE firmware file (`safte.hex`) into a file called *`safcon.hex`*. This creates the image to be placed in the TWS flash memory device for downloading to the LSI53C040. Table 5.1 shows the files associated with the configuration utility.

Table 5.1 Configuration Utility Files

File Name	Description
`config.exe`	The main configuration program (the first element of the firmware) provided by LSI Logic. It displays the questions one at a time, beginning with general questions and progressing to more detailed questions about desired MPIO/MPLED pin assignments for the enclosure environment.
`bootload.hex`	The second element of the firmware provided by LSI Logic. This file contains the bootloader, which is used only if the designer selects the download option addressed in the first question of the configuration utility.
`loader.hex`	The hex output of `config.exe`. The designer concatenates this file with the `bootload.hex` firmware binary file into a file called *`boot.hex`* (or whatever the designer wishes to call this file), which is the image that is placed on the TWS memory device and downloaded to the LSI53C040. Refer to Section 5.3, "After Running the Configuration Utility," page 5-21 about using DOS commands to concatenate files.
`safte.hex`	The third element of the firmware provided by LSI Logic. This file is an Intel hex file which contains the SAF-TE firmware.
`config.hex`	The hex output of `config.exe`. The designer concatenates this file with the `safte.hex` firmware binary file into a file called *`safcon.hex`*, (or whatever the designer wishes to call this file), which is the image that is placed on the TWS memory device and downloaded to the LSI53C040. Refer to Section 5.3, "After Running the Configuration Utility," page 5-21 about using DOS commands to concatenate files.
`myinput``.txt`	The designer can create this file with a text editor using the answers from the questions asked in the configuration program. After this file is first created, the designer can change some of the values submitted to the configuration program, by editing this file and running it, instead of manually stepping through all of the questions in `config.exe`. Just run `config.exe` again, redirecting the input to the configuration utility to a file called *`myinput`*`.txt`, and redirecting the output of the configuration utility to a file called *`myinput`*`.log` using the DOS command: `config<`*`myinput`*`.txt>`*`myinput`*`.log`.
`myinput``.log`	When the file `myinput.txt` is used in the DOS command: `config<`*`myinput`*`.txt>`*`myinput`*`.log`, the resulting file (*`myinput`*`.log`) contains a summary of all the questions answered, as well as a summary of any errors in the input file. When the program runs to completion, the `myinput.log` file also contains a summary of the data structure sizes and the Enclosure ID field.

To start the configuration program, type `config` at the DOS prompt in the directory where the `config.exe` program resides. Before you start the program the first time, be aware of the following items:

- The configuration program cannot be stopped and restarted once you begin. If you exit the program you will have to start over from the beginning.
- The configuration program does not allow you to scroll back and view previous answers once they scroll off the screen.
- The configuration program requires at least 10 minutes for a novice user to enter data, depending on how many devices and options your enclosure supports.
- Some of the questions require `Yes/No` responses, others have field size limits, or other limits or expectations for the type of response you will give. The program will not allow you to assign an MPIO or an MPLED pin to more than one function. If you make one of these errors, the program will reprompt for a different response. In some cases additional information will display, such as the required format for pin assignments, either after an incorrect answer or after you press ENTER. The program will abort after about 20 incorrect responses to a specific question.

To run the configuration program as quickly as possible, you should have the following information ready when you start:

- SCSI ID for the LSI53C040 and each device slot in your system.
- Desired SCSI Bus High bits assignments for SCSI High ID 2, SCSI High ID 1, and SCSI High ID 0.
- Vendor ID, Product ID, Enclosure ID, and Channel Identifier.
- A list of MPIO and MPLED pins mapped to desired features and devices in the enclosure.
- A list of TWS devices and their respective bus numbers and addresses.
- Power On Configuration Options in the *LSI53C040 Enclosure Services Processor Technical Manual*.

The *LSI53C040 Enclosure Services Processor Technical Manual* describes many of the MPIO and MPLED pins and other device features in detail, and may be a useful reference as you run the configuration program.

5.1.1 Myinput.txt File

The designer can create the *myinput*.txt file using a simple text editor while answering the questions in the config.exe program. This file then contains a plain text summary of the responses entered for the most recent running of the config.exe program. The *myinput*.txt file can then be edited to change individual answers without the user having to step manually through config.exe all over again.

Since the *myinput*.txt file is a plain text file, any information can be commented as long as the line length is not exceeded. This file does not support line wrapping or carriage returns. If you enter new data, make sure it is in a format identical to that requested by the configuration program. The program looks for the proper number of fields (1, 2, or 3, with fields separated by a space) to answer each question; thus, any further information is considered to be comments.

For your convenience, an example of this text file follows. It does not include all the answers to the questions contained within the Configuration Utility. The ellipsis indicate more entries would be added based on your system's configuration requirements.

```
---begin---
n       code load
n       parity checking
eight---
sixteen.........
seven__
+
10      SCSI id2
9       SCSI id1
8       SCSI id0
1       SCSI id
0       fast blink value
3       slow blink value
n       controllable speaker?
y       monitorable speaker?
y       controllable door lock?
...
1024    length of download
---end---
```

5.2 Questions in the Configuration Utility

Table 5.2 through Table 5.4 and Table 5.6 through Table 5.10 discuss the types of questions that may occur when running the configuration utility. You may see slightly different questions, or you may not see all of these, depending on the type of system environment you specify in the general questions section. Italicized text provides additional information regarding the various questions asked and does not appear in the configuration utility program.

Table 5.2 General Questions

Question	Explanation/Required Input
Welcome to the 53C040 SafTe configuration program!	
First, some general questions:	
The `config.exe` *program opens with the following two questions.*	
Do you want to support microcode updates over the SCSI bus?	Enter `y` or `n`.
Do you want to support parity changes on the SCSI bus?	Enter `y` or `n`.
The following information is used to uniquely identify a specific enclosure. This information is reflected in the response data from an INQUIRY command.	
Enter text for Vendor ID to be returned in the SCSI INQUIRY command (8 characters).	Enter an 8-character ASCII string to identify the product vendor.
Enter text for Product ID to be returned in the SCSI INQUIRY command (16 characters).	Enter a 16-character ASCII string to specify the product ID.
Enter text for Enclosure ID to be returned in the SCSI INQUIRY command (7 characters).	Enter a 7-character ASCII string to specify a specific enclosure.
Enter text for Channel ID to be returned in the SCSI INQUIRY command (1 character).	Enter a 1-character ASCII character to specify the Channel ID.
The LSI53C040 has three LVD SCSI High ID pins (SHID[2:0] ±). These pins may be connected to any of the SCSI data signals from data bit 8 through 15. This enables the LSI53C040 SCSI core to respond to selection as a device with an ID greater than 7. The following questions assign specific SCSI data signals to each of the SHID pins.	
Which bit of the high byte of the SCSI data bus will the LSI53C040 see as bit SCSI High ID 2?	Enter a number between `8` and `15`.

Table 5.2 General Questions (Cont.)

<table>
<tr><th>Question</th><th>Explanation/Required Input</th></tr>
<tr><td>Which bit of the high byte of the SCSI data bus will the LSI53C040 see as bit SCSI High ID 1?</td><td>Enter a number between 8 and 15.</td></tr>
<tr><td>Which bit of the high byte of the SCSI data bus will the LSI53C040 see as bit SCSI High ID 0?</td><td>Enter a number between 8 and 15.
If you accidentally enter the same ID value more than once, the program will return to the beginning of these questions, so you have the opportunity to start over rather than being forced to select only from the remaining choices.</td></tr>
<tr><td>What SCSI ID do you want for the SAF-TE processor?</td><td>Enter a number between 0 and 15.
Based upon your SCSI Data signal assignments above, you can assign a SCSI ID between 0 and 15 to the LSI53C040.
Note: If the ID selected here is not possible based on your answers to the previous three questions, the program starts over at the beginning of the previous three questions.</td></tr>
<tr><td colspan="2">Four possible <u>slow</u> blink rates and four possible <u>fast</u> blink rates are available in the LSI53C040. The following questions assign specific rates for slow and fast blink rates. LEDs are used to indicate various conditions such as identify slot or device faulty. See the LED blink pattern section for default blink settings.</td></tr>
<tr><td>Prepare to enter fast and slow blink rates:
Blink rates are proportional to input clock frequency.
Fast and slow blink rates can be set from 0 to 3.
A blink rate of 2 is twice as fast as a blink rate of 3.
A blink rate of 1 is twice as fast as a blink rate of 2.
The fast blink rates are 4 times as fast as the slow blink rates.
Fast blink rates of 2 and 3 are the same as slow blink rates of 0 and 1.
Which of the 4 possible blink rates would you like for the fast blink rate (0 is the fastest, 3 is the slowest)?</td><td rowspan="2">The blink rates are proportional to the input clock frequency. See the "System Registers," chapter in the LSI53C040 Enclosure Services Processor Technical Manual for example blink rates based on a 20 MHz or a 40 MHz clock.
<u>Comparative Blink Rates</u>
<table><tr><th><u>Fast Blink Designator</u></th><th><u>Slow Blink Designator</u></th><th><u>Blink Rate</u></th></tr><tr><td>0</td><td>–</td><td>32 x BR</td></tr><tr><td>1</td><td>–</td><td>16 x BR</td></tr><tr><td>2</td><td>0</td><td>8 x BR</td></tr><tr><td>3</td><td>1</td><td>4 x BR</td></tr><tr><td>–</td><td>2</td><td>2 x BR</td></tr><tr><td>–</td><td>3</td><td>BR</td></tr></table></td></tr>
<tr><td>Which of the 4 possible blink rates would you like for the slow blink rate (0 is the fastest, 3 is the slowest)?</td></tr>
</table>

Table 5.2 General Questions (Cont.)

Question	Explanation/Required Input
The following information determines how many MPIO/MPLED pins to allocate for the number of speakers, door locks, fans, power supplies, device slots, and temperature devices in a specific enclosure, as well as Global Flags.	
Is there a controllable speaker?	Enter y or n.
Is there a monitorable speaker?	This question appears only if you replied no to the question about a controllable speaker. Enter y or n.
Is there a controllable door lock?	Enter y or n.
Is there a monitorable door lock?	This question appears only if you replied no to the question about a controllable door lock. Enter y or n.
Is there a Global Identify Enclosure LED?	Enter y or n. If yes, this will drive an LED to the identify enclosure signal. (Send Global Flags command).
Is there a Global Enclosure Status LED?	Enter y or n. If yes, this will drive an LED to indicate an error condition of global failure or global warning (Send Global Flags command).
Is there a Global Drive Status LED?	Enter y or n. If yes, this will drive an LED to indicate a drive error condition of drive failure or drive warning (Send Global Flags command).
Is there a Global Array Status LED?	Enter y or n. If yes, this will drive an LED to indicate an array error condition or array failure or array warning (Send Global Flags command).

Table 5.3 Enclosure Components Questions

Question	Explanation
These questions relate to HOW MANY various elements are in the enclosure.	
How many fans supplying a single wire input do you want to support?	Enter the number of fans in the enclosure that have one MPIO pin assigned for status. This allows up to two states to be determined for fan status. Up to six fans can be specified.
How many fans supplying a dual wire input do you want to support?	Enter the number of fans in the enclosure that have two MPIO pins assigned for status. This allows up to four states to be determined for fan status. Up to six fans can be specified.

Table 5.3 Enclosure Components Questions (Cont.)

Question	Explanation
How many power supplies supplying a single wire input do you want to support?	Enter the number of power supplies in the enclosure that have one MPIO pin assigned for status. This allows up to two states to be determined for power supply status. Up to six power supplies can be specified.
How many power supplies supplying a dual wire input do you want to support?	Enter the number of power supplies in the enclosure that have two MPIO pins assigned for status. This allows up to four states to be determined for power supply status. Up to 6 power supplies can be specified.
How many device slots do you want to support?	Enter the number of drive slots that the enclosure can support. Up to three MPIO pins and as many as two LEDs will be assigned for each drive slot. Up to 14 slots can be specified.
How many temperature inputs supplying a single wire input do you want to support?	Enter the number of temperature sensors that return a binary signal (under/over preset threshold) in the enclosure. One MPIO pin will be assigned for each of these temperature sensors. Up to 15 temperature inputs can be specified.
How many temperature inputs with 2-wire serial (TWS) input do you want to support?	Enter the number of temperature sensors in the enclosure that will be read over the TWS interface. This information is used to determine the number of integer temperature fields returned by the Read Enclosure Status command. Up to four TWS temperature inputs can be specified.
How many LM78 parts [max 1 per 2-wire serial (TWS) bus] do you want to support?	Enter `0`, `1`, or `2`.

Table 5.4 Pin Assignment Questions

Question	Explanation
More detailed questions are asked based on your previous input.	
The following questions assign specific MPIO and MPLED pins according to the questions answered above. Below are the specific pin assignments for each MPIO and MPLED pin. The MPIO and MPLED banks are organized as follows: Pin Type Bank Pins Available MPIO 0 [7:0] MPIO 1 [7:0] MPIO 2 [7:0] MPIO 3 [3:0] MPLED 0 [7:0] MPLED 1 [7:0] MPLED 2 [7:0] *The format of your answers should be:* `IO[space]bank number[space]pin number` *(for MPIO pins)* ***Example:*** IO 3 1 `LED[space]bank number[space]pin number` *(for MPLED pins)* ***Example:*** LED 2 2 *The MPLED pins can be used for general IO as well as LED functions if your design does not use all available MPLED pins. The MPIO pins cannot be used for LED functions, however, because they do not support blinking.*	
Enter input (output) pin to be used for door lock:	Enter the MPIO pin for the door lock signal.
Enter input (output) pin to be used for speaker:	Enter the MPIO pin for the speaker/alarm.
Enter input (output) pin to be used for Global Identify Enclosure LED:	Enter the MPLED pin for the global enclosure identification LED.
Enter input (output) pin to be used for Global Enclosure Status LED:	Enter the MPLED pin for the global enclosure status LED.
Enter output pin to be used for Global Drive Status LED:	Enter the MPLED pin for the global drive status LED.
Enter output pin to be used for Global Array Status LED:	Enter the MPLED pin for the global array status LED.
Enter input pin to be used for single input fan # x:	Enter the MPIO pin for each single input fan specified above. This question repeats for the number of single input fans specified.
Enter input pin to be used for dual input fan # x MSB:	Enter the MPIO pin for the MSB for each dual input fan specified above. Enter the MPIO pin for the LSB for each dual input fan specified above. These two questions repeat for the number of dual input fans specified.
Enter input pin to be used for dual input fan # x LSB:	

Table 5.4 Pin Assignment Questions (Cont.)

Question	Explanation
Enter input pin to be used for single input power supply # x:	Enter the MPIO pin for each single input power supply specified above. This question repeats for each single input power supply specified.
Enter input pin to be used for dual input power supply # x MSB:	Enter the MPIO pin for the MSB for each dual input power supply specified above. Enter the MPIO pin for the LSB for each dual input power supply specified above. These questions repeat for each dual input power supply specified.
Enter input pin to be used for dual input power supply # x LSB:	
How many LED outputs do you want to support per device slot?	Enter `0`, `1`, or `2` to specify the number of LEDs to drive for each device slot specified above.
Will there be a Device Present input signal for each device?	Enter `y` or `n` to indicate whether you want to detect when a device has been inserted or removed from its slot. This will assign one MPIO pin to each device slot supported.
Will there be a Ready Device for Use output signal for each device?	Enter `y` or `n` to indicate whether you want to control some specific operation for each device. One example would be to control power to a device slot. If yes, one MPIO pin will be assigned to each device slot supported. The Prepare for Operation bit (Byte 2 bit 0) in the Perform Slot Operation Command, will be used to activate this signal. This signal is the complement of the Prepare Device for Insertion/Removal output signal.

Table 5.4 Pin Assignment Questions (Cont.)

Question	Explanation
Will there be a Prepare Device for Insertion/Removal output signal for each device?	Enter y or n to indicate whether you want to control some specific operation for each device. Typically, only a Ready Device for Use output signal or Prepare Device for Insertion/Removal output signal will be chosen, but not both. If yes, one MPIO pin will be assigned to each device slot supported. The Prepare for Insertion bit (Byte 2 bit 1) in the Perform Slot Operation Command will be used to activate this signal. This signal is the complement of the Ready Device for Use output signal.
Do you want to override the LED patterns that display drive status?	Enter y or n. If you enter n, you accept the default LED settings shown in Table 5.5 on the next page. If you enter y, the program will prompt you with the questions shown in Table 5.6. The format of your responses to those questions depends on how you responded to the previous question, “How many LED outputs do you want to support per device slot?”.

Table 5.5 shows the current default LED settings for each bit description in the Write Device Slot Status Command. The bit descriptions are shown in increasing priority order. If more than one bit is set by the Write Device Slot Status Command, the bit with the highest priority will dictate the LED blink patterns.

Table 5.5 Default LED Settings for Write Device Slot Status Flags

State	Bit Description	One-LED Option	Two-LED Option	
			LED 1	LED 2
0	Default/Nothing to Report	Off	Off	Off
1	Prepare for insertion/removal	Off	On	On
2	Prepare for operation	On	Off	Slow
3	Device Faulty	Slow	On	Off
4	Device Rebuilding	Slow	Off	Fast
5	In Failed Array	Slow	Fast	On
6	In Critical Array	Slow	Slow	Off
7	Parity Check operation	Slow	On	Off
8	Predicted Fault Failure	Slow	Fast	Off
9	No drive inserted	Off	On	Fast
10	Hot Spare	Off	Fast	Fast
11	Rebuild Stopped	Slow	Off	On
12	Identify Slot	Fast	Slow	Fast

Table 5.6 Selections for Custom LED Settings for Write Device Slot Status Flags

Question	Explanation
Answers to the following questions permit the designer to change the LED settings for the Write Device Slot Status Flags. These questions appear in the configuration utility only if the designer has answered the previous question "Do you want to override the LED patterns that display drive status?" with a "yes" response. Table 5.5 shows the default settings for both the one and two-LED options. In this section, you will need to issue responses for each of the states 0 through 12. *Please specify one of:* *0 - for off 1 - for slow blink 2 - for fast blink 3 - for on*	
State 0 - Default/Nothing to Report: Off/0 Off/0 -	Enter new settings per the choices above after the "dash". The format of your answers should be: **Two-LED option:** LED1[space]LED2 **Example:** 3 3 **One-LED option:** LED1 **Example:** 0
State 1 - Ready for Insertion/Removal: On/3 On/3 -	Enter new settings after the dash (-), per the choices above.
State 2- Prepare for Operation: Off/0 Slow/1 -	Enter new settings after the dash (-), per the choices above.
State 3 - Device Faulty: On/3 Off/0 -	Enter new settings after the dash (-), per the choices above.
State 4 - Device Rebuilding: Off/0 Fast/2 -	Enter new settings after the dash (-), per the choices above.
State 5 - In Failed Array: Fast/2 On/3 -	Enter new settings after the dash (-), per the choices above.
State 6 - In Critical Array: Slow/1 Off/0 -	Enter new settings after the dash (-), per the choices above.
State 7 - Parity Check Operation: On/3 Off/0 -	Enter new settings after the dash (-), per the choices above.
State 8 - Predicted Fault Failure: Fast/2 Off/0 -	Enter new settings after the dash (-), per the choices above.
State 9 - No Drive Inserted (Unconfigured Drive): On/3 Fast/2 -	Enter new settings after the dash (-), per the choices above.
State 10 - Hot Spare: Fast/2 Fast/2 -	Enter new settings after the dash (-), per the choices above.
State 11 - Rebuild Stopped: Off/0 On/3 -	Enter new settings after the dash (-), per the choices above.
State 12 - Identify Slot: Slow/1 Fast/2 -	Enter new settings after the dash (-), per the choices above.

Table 5.7 Device Slot Operation Questions

Question	Explanation
What SCSI ID to you want associated with device slot x?	Enter the SCSI ID you want to associate with each device slot specified above. This question repeats for each device slot specified.
Enter output pin to be used for device slot x (SCSI ID y) LED MSB:	Enter the MPLED pin assignments for each device slot LED MSB and LSB specified above, using the format described at the beginning of Table 5.4. This pair of questions repeats for each device slot specified. **Note:** If only one LED was selected for each device slot in the previous questions, then only one MPLED pin will be assigned here for each device slot (that is, not an MSB *and* an LSB).
Enter output pin to be used for device slot x (SCSI ID y) LED LSB:	
Enter input pin to be used for device slot x (SCSI ID y) Device Present:	Enter the MPIO pin assignment for device present status for each device slot specified above, using the format described at the beginning of Table 5.4. This question repeats for each device slot specified, provided that the user has answered “y” to the previous question, “Will there be a Device Present input signal for each device?”
Enter output pin to be used for device slot x (SCSI ID y) Ready Device:	Enter the MPIO pin assignment for device ready for each device slot specified above, using the format described at the beginning of Table 5.4. This question repeats for each device slot specified, provided that the user has answered “y” to the previous question, “Will there be a Ready Device for Use output signal for each device?”
Enter output pin to be used for device slot x (SCSI ID y) Remove/Insert Device:	Enter the MPIO pin assignment for device remove/insert for each device slot specified above, using the format described at the beginning of Table 5.4. This question repeats for each device slot specified, provided that the user has answered “y” to the previous question, “Will there be a Prepare Device for Insertion/Removal output signal for each device?”
Enter input pin to be used for single input temp alarm #x:	Enter the MPIO pin assignment for each binary temperature sensor specified above, using the format described at the beginning of Table 5.4. This question repeats for each single input temperature sensor specified.

Table 5.8 Status Signal Questions

<table>
<tr><th>Question</th><th>Explanation</th></tr>
<tr><td colspan="2">The questions covered in this Table assign specific fan, power supply, door lock, and speaker status to be returned in response to the Read Enclosure Status command for single and dual inputs.</td></tr>
<tr><td colspan="2">For each fan, enter one of the following responses for each input pattern:
Choose:
0x00 Fan is operational
0x01 Fan is malfunctioning
0x02 Fan is not installed
0x80 Unknown status, or status not reportable</td></tr>
<tr><td>Enter the Read Enclosure Status command's response for a single input fan with an input bit pattern of 0:</td><td rowspan="2">The input bit pattern is read from the LSI53C040 MPIO pin(s) assigned to this fan.

Note: These two questions are asked only if the user has specified in the previous questions that there are single-wire input fans in this enclosure.</td></tr>
<tr><td>Enter the Read Enclosure Status command's response for a single input fan with an input bit pattern of 1:</td></tr>
<tr><td>Enter the Read Enclosure Status command's response for a dual input fan with an input bit pattern of 00:</td><td rowspan="4">The input bit pattern is read from the LSI53C040 MPIO pin(s) assigned to this fan.

Note: These four questions are asked only if the user has specified in the previous questions that there are dual-wire input fans in this enclosure.</td></tr>
<tr><td>Enter the Read Enclosure status command's response for a dual input fan with an input bit pattern of 01:</td></tr>
<tr><td>Enter the Read Enclosure status command's response for a dual input fan with an input bit pattern of 10:</td></tr>
<tr><td>Enter the Read Enclosure status command's response for a dual input fan with an input bit pattern of 11:</td></tr>
<tr><td colspan="2">For each power supply, enter one of the following responses for each input pattern:
Choose:
0x00 Power supply is operational and on
0x01 Power supply is operational and off
0x10 Power supply is malfunctioning and commanded on
0x11 power supply is malfunctioning and commanded off
0x20 Power supply is not present
0x21 Power supply is present
0x80 Unknown status, or status not reportable</td></tr>
</table>

Table 5.8 Status Signal Questions (Cont.)

Question	Explanation
Enter the Read Enclosure status command's response for a single input power supply with an input bit pattern of 0:	The input bit pattern is read from the LSI53C040 MPIO pin(s) assigned to this power supply. **Note:** These two questions are asked only if the user has specified in the previous questions that there are single-wire input power supplies in this enclosure.
Enter the Read Enclosure status command's response for a single input power supply with an input bit pattern of 1:	
Enter the Read Enclosure status command's response for a dual input power supply with an input bit pattern of 0x00:	The input bit pattern is read from the LSI53C040 MPIO pin(s) assigned to this power supply. **Note:** These two questions are asked only if the user has specified in the previous questions that there are dual-wire input power supplies in this enclosure.
Enter the Read Enclosure status command's response for a dual input power supply with an input bit pattern of 0x01:	
Enter the Read Enclosure status command's response for a dual input power supply with an input bit pattern of 0x10:	
Enter the Read Enclosure status command's response for a dual input power supply with an input bit pattern of 0x11:	
For door lock, enter one of the following responses for each input pattern: Choose: 0x00 Door is currently locked 0x01 Door is currently unlocked, or door lock not installed 0x80 Unknown status, or status not reportable	
Enter the Read Enclosure status command's response for a single input door lock with an input bit pattern of 0:	The input bit pattern is read from the LSI53C040 MPIO pin(s) assigned to this door lock. **Note:** These two questions are asked only if the user has specified in the previous questions that there are single-wire input door locks in this enclosure.
Enter the Read Enclosure status command's response for a single input door lock with an input bit pattern of 1:	

Table 5.8 Status Signal Questions (Cont.)

Question	Explanation
For speaker, enter one of the following responses for each input pattern: Choose: 0x00 Speaker is currently off, or no speaker installed 0x01 Speaker is currently on	
Enter the Read Enclosure status command's response for a single input speaker with an input bit pattern of 0:	The input bit pattern is read from the LSI53C040 MPIO pin(s) assigned to this speaker. **Note:** These two questions are asked only if the user has specified in the previous questions that there are single-wire input speakers in this enclosure.
Enter the Read Enclosure status command's response for a single input speaker with an input bit pattern of 1:	

Table 5.9 TWS Bus Operation Questions

Question	Explanation
The following questions are for TWS bus operation.	
For each 2-wire serial bus, specify the bus speeds desired for operation: 0 - 2-wire serial 78 KHz, system 20 MHz 1 - 2-wire serial 312 KHz, system 20 MHz 2 - 2-wire serial 78 KHz, system 40 MHz 3 - 2-wire serial 312 KHz, system 40 MHz	
Please select the speed for the 2-wire serial bus number 0 relative to the system clock you are using:	Enter `0`, `1`, `2`, or `3`.
Please select the speed for the 2-wire serial bus number 1 relative to the system clock you are using:	Enter `0`, `1`, `2`, or `3`.
Do you want the temperature reported in the Read Enclosure Status command's response reported in: 0 - Fahrenheit, or 1 - Celsius?	Enter `0` or `1`.
How many 2-second intervals would you like between 2-wire serial input passes?	Specify the desired sampling period.

Table 5.9 TWS Bus Operation Questions (Cont.)

Question	Explanation
Please select the chip type for 2-wire serial (TWS) temperature sensor number x: 0 - National LM75 1 - Dallas 1621 2 - National LM78	These questions repeat for each 2-wire serial temperature sensor specified.
Which 2-wire serial (TWS) bus (0 or 1) will this chip be on?	
At what address (0 to 7) will this chip be?	
The following questions refer to the LM78 on TWS bus number x. *These questions repeat for each serial bus specified.*	
What value is to be used for the first fan divisor?	Enter 0, 1, 2, or 3. For more details regarding the operation of the LM78, please refer to the LM78 specification.
What value is to be used for the second fan divisor?	Enter 0, 1, 2, or 3. For more details regarding the operation of the LM78, please refer to the LM78 specification.
Will there be a fan connected to fan monitor number m?	Enter y or n. This question and the next one repeat for each fan connected to a monitor (m = 0, 1, or 2).
What is the highest value that indicates that the fan is functioning correctly?	Enter a number between 1 and 254.
Will there be a power supply connected to voltage monitor number n?	Enter y or n. This question and the next two repeat for each power supply connected to a voltage monitor (n = 0, 1, ..., 5, or 6).
What is the lowest value that indicates that the power supply is functioning correctly?	Enter a number between 1 and 254.
What is the highest value that indicates that the power supply is functioning correctly?	Enter a number between 1 and 254.

Table 5.10 Questions for Firmware Bootloader

Questions	Explanation
The following questions are related to the hardware-based power-on serial ROM download, which will load and run the firmware bootloader.	
All the remaining questions are asked only if the user replied "yes" to support microcode updates over the SCSI bus.	
On which 2-wire serial (TWS) bus (0 or 1) will this download happen?	A pull-up resistor on LSI53C040, pin A11 changes the serial ROM download from TWS port 0 to port 1. Note that the answer to this question must be consistent with the use of a pull-up resistor on pin A11.
What chip address (0 to 7) will the download be from?	LSI53C040 pins A10, A9, and A8 define the address from which the LSI53C040 will attempt the initial configuration download. Note that the answer to this question must be consistent with the use of pull-up resistors on pin A10, A9, and A8. See Chapter 2 of the *LSI53C040 Enclosure Services Processor Technical Manual* for further information.
Do you want to use an LED to indicate bootload failures?	Enter y or n.
Select the LED bank (0–2).	The two questions are asked only if the user has chosen to use an LED to indicate bootload failures, by answering “y” to the previous question.
Select the LED (0–7).	

Table 5.10 Questions for Firmware Bootloader (Cont.)

Questions	Explanation
The following questions are related to the first SAF-TE firmware image that will be loaded, and run by the bootloader.	
On which 2-wire serial (TWS) bus (0 or 1) will this download happen?	A pull-up resistor on LSI53C040, pin A11 changes the serial ROM download from TWS port 0 to port 1. Note that the answer to this question must be consistent with the use of a pull-up resistor on pin A11.
What chip address (0 to 7) will the download be from?	LSI53C040 pins A10, A9, and A8 define the address from which the LSI53C040 will attempt the initial configuration download. Note that the answer to this question must be consistent with the use of a pull-up resistor on pin A10, A9, and A8. See pages 2-20 to 2-22 of the *LSI53C040 Enclosure Services Processor Technical Manual* for further information.
From which address will the download start?	Please answer the question with a number from `0` to `32767` (The recommended default value is `0`).
What length will the download be?	Please answer the question with a number from `1024` to 12192 (This value is equal to or greater than the number of bytes of the firmware program that will be downloaded).
The following questions are related to the second SAF-TE firmware image that will be loaded, and run by the bootloader.	
On which 2-wire serial (TWS) bus (0 or 1) will this download happen?	A pull-up resistor on LSI53C040, pin A11 changes the serial ROM download from TWS port 0 to port 1. Note that the answer to this question must be consistent with the use of a pull-up resistor on pin A11.
From which chip address (0 to 7) will the download be?	LSI53C040 pins A10, A9, and A8 define the address from which the LSI53C040 will attempt the initial configuration download. Note that the answer to this question must be consistent with the use of a pull-up resistor on pin A10, A9, and A8. See pages 2-20 to 2-22 of the *LSI53C040 Enclosure Services Processor Technical Manual* for further information.
From which address will the download start?	Please answer the question with a number from `0` to 32767 (The recommended default value is `0`).
What length will the download be?	Please answer the question with a number from `1024` to 12192 (This value is equal to or greater than the number of bytes of the firmware program that will be downloaded).

5.3 After Running the Configuration Utility

After all questions are answered, the utility creates the config.hex file, the loader.hex file, and provides the data structure sizes, as shown in the example below.

```
 94.=  0x05E bytes of 256.= 0x100 state machine data
       memory used
 50.=  0x032 words of 300.= 0x12C state machine program
       memory used
1194.= 0x4AA bytes used by the config_data structure
```

The Enclosure ID field of the SCSI Inquiry command is stored at offset

```
 61.=  0x03D from the start of the config_data structure
       which is at address
 64.=  0x04D which places the Enclosure ID field at
       address
125.=  0x07D
```

The program then returns to the DOS prompt. At this point, the config.hex file is ready to concatenate with the safte.hex file, and the loader.hex file is ready to concatenate with the bootload.hex file, using the following DOS commands:

```
copy  config.hex + safte.hex  safcon.hex
```

and

```
copy  loader.hex + bootload.hex  boot.hex
```

or you can refer to the myinput.txt file to see a summary of your answers.

Appendix A
LSI53C040 Board Utilities

A.1 Data Transfers

The LSI53C040 board supports serial, ISA, SCSI and 8067 data transfers. This section discusses these various data transfers.

A.1.1 Serial Port

The LSI53C040 board contains a serial port that is accessible if the MPIO Bank 3 Bits 0 and 1 are **not** used for other purposes. The SAF-TE firmware uses the 80C32 Timer 2 as the transmit and receive clock. It is configured for autoreload mode. The BAUD rate for the serial port is

BAUD = (Oscillator Frequency)/(# Clicks between Timer 2 rollovers)/32

The on-board oscillator operates at 40 MHz. The settings for the capture registers RCAP2H and RCAP2L, as well as the resulting error are shown in Table A.1.

Table A.1 Capture Register Settings

BAUD Rate	RCAPH2/RCAP2L	# Clicks	Error (%)
19200 (default)	0xFFBF	65	0.16
9600	0xFF7E	130	0.16
4800	0xFEFC	260	0.16

Any serial port management system can be used on the monitoring station. However, note that the data transfer is setup to transmit 8-bits plus a stop bit and the hardware setting is for XON/XOFF.

Information to be transmitted over the serial port is sent using the standard *printf* statement. To enable existing print statements, compile the SAF-TE source code with VERBOSE and/or DEBUG true. Additional statements can be added as necessary.

A.1.2 ISA

Any of the serial EEPROMs on the LSI53C040 board can be programmed with the ISA bus using the utility **cl_isa** (formerly named **cl2**). Typing **cl_isa** on the command line will result in the syntax for the utility. As designed, command line inputs for this utility are shown in Table A.2.

Table A.2 Command Line Inputs

Option	Description
-H*Address*	*Address* is Hardware address of the boards ISA bus. Valid values are 0xC000, 0xD000, 0xE000 or 0xF000. Default is 0xD000.
-b*Bus*	*Bus* is the Two Wire Serial bus number of the serial EEPROM. *Bus* should be 0 or 1. Default value is 0.
-c*Chip*	*Chip* is the TWS chip address. The utility supports addresses 0, 6, and E(e). Default value is 0.
-h	Suppresses adding the header and checksum used by the ISA hardware during download.
-a	Write address (in HEX) in serial EEPROM. Default is 0x0.
-s	Slow operation (not implemented).
-v	Suppresses writing to the TWS EEPROM. Performs a read from specified device and compares to loaded image.
filename (required)	Source filename. Must be in the Intel Hex format.

Note: If the ISA bus or a programmer is used to download the firmware, the firmware is not limited to 12192 (0x2FA0) bytes.

All of the serial EEPROMS are programmable through the ISA interface, however the firmware that will execute is set by the jumpers JP8 (A8), JP5 (A9) and JP6 (A10). Refer to the Section A.1.5, “LSI53C040 Board Layout/Jumper Settings,” page A-6 for more detailed information.

The ISA bus can be reset by running the program `reset.exe`.

A.1.3 SCSI

The LSI53C040 board can terminate or be inserted into the middle of a single-ended SCSI bus. As the board is self-terminating, in the former case, a terminator is not required on the unused SCSI port. To use the code load feature of the SAF-TE firmware:

Step 1. The firmware must be compiled with CODE_LOAD_ENABLED set to true.

Step 2. Execute `config.exe`.

Step 3. Create `boot.hex` (`loader.hex` + `bootload.hex`) and `firmware.hex` (`config.hex` + `safte.hex`).

Note. During the execution of `config.exe`, if a code download is allowed, you are required to enter the bus and chip numbers for the boot code, primary firmware image and secondary firmware image. The bus and chip numbers for the boot code MUST be entered to match the jumper settings on the board. The boot code contains information on the location of the primary (active) and secondary (inactive) images.

Step 4. Download the primary firmware image to the board using **cl_isa.** Select the command line arguments to suppress the creation of the four-byte header. As an example, to download firmware to the serial EEPROM on Bus 1 with Chip ID 0xE enter:

```
A:\>cl_isa  –b1 –cE –h firmware.hex
```

Step 5. Download the boot image using **cl_isa** (include the header).

Step 6. At this point, on a reboot, the boot code will be executed automatically by the hardware. It will copy the active firmware image to program memory and begin execution of the firmware itself.

Step 7. To update the firmware using the SCSI bus, execute **cl_scsi.** This program will automatically write the new firmware to the location specified for the inactive firmware image and modify the boot image to update the active/inactive image flags. (**cl_scsi** requires that an ASPI manager be installed).

Step 8. On a reboot, the image downloaded in Step 7 will be loaded into program memory and will be executed.

Note: Once Steps 1–6 have been performed, all future updates of the firmware through the SCSI bus can be accomplished with Step 7.

A.1.3.1 Example 1

Table A.3 Configuration Using Three Serial EEPROMS

The tail of the file *myInput.txt* contains this data:	
1	tws serial rom download bus
7	tws serial rom download chip
y	download error led
0	led bank
0	led bit
1	bus 0
3	chip 0
0	start 0
12192	length 0
0	bus 1
7	chip 1
0	start 1
12192	length 1

Upon execution of `config.exe`, these lines specify that the default EEPROM is on bus 1 at address E (7 << 1). The active (first) firmware is to be stored on bus 1, chip address 6 (3 << 1), with beginning address

of 0 and maximum length of 12192 bytes. The inactive (second) image is to be stored on bus 0, chip address E, with beginning address of 0 and maximum length of 12192 bytes.

Create `boot.hex` and `firmware.hex` (Step 3 above).

Download the active image (suppress the header):

```
c:\> cl_isa –b1 –c6 –h firmware.hex
```

Download the boot image:

```
c\:>cl_isa –b1 –cE boot.hex
```

Upon a reset (will occur automatically), `firmware.hex` will be loaded into program memory and executed. If **cl_scsi** is executed, the new firmware will be stored on bus 0 at chip address E, and upon a reset it will be downloaded into program memory and executed.

A.1.3.2 Example 2

Table A.4 Configuration Using Two Serial EEPROMS

The tail of the file *myInput.txt* contains this data:	
1	tws serial rom download bus
7	tws serial rom download chip
y	download error led
0	led bank
0	led bit
1	bus 0
7	chip 0
4186	start 0
12192	length 0
0	bus 1
7	chip 1

Table A.4 Configuration Using Two Serial EEPROMS (Cont.)

The tail of the file *myInput.txt* contains this data:	
0	start 1
12192	length 1

In `config.exe`, enable the code load option and specify that the default EEPROM is at address E on bus 1. The active (first) firmware is to be stored on bus 1, chip address E, with beginning address of 4186 (0x105A) and maximum length of 12192 bytes. The inactive (second) image is to be stored on bus 0, chip address E, with beginning address of 0 and maximum length of 12192 bytes. (The boot code limits the firmware to a maximum size of 0x2FA0 or 12912 bytes.)

Create `boot.hex` and `firmware.hex` (Step 3 above).

Download the active image (suppress the header and specify the hex address):

```
c:\> cl_isa –b1 –cE –h –a105A firmware.hex
```

Download the boot image:

```
c:\>cl_isa –b1 –cE boot.hex
```

With this configuration, the boot code is located in addresses 0x0000 through 0x1059 and the firmware is at addresses 0x105A through 0x4000. (The boot code limits the firmware to have a maximum size of 0x2FA0 or 12912 bytes). Upon a reset, the firmware is downloaded into program memory and executed. If **cl_scsi** is executed, the new firmware will be placed in the EEPROM on bus 1 at address 6.

A.1.4 8067 Utilities

Currently, there are no 8067 utilities available.

A.1.5 LSI53C040 Board Layout/Jumper Settings

The LSI53C040 board contains two banks of switches or jumpers that specify the behavior of the board hardware.

The board-reset push button is located in the upper left-hand corner of the board. Immediately to the right of the reset button, is the switch SW2. This switch controls the hardware address of the ISA bus. Table A.5 specifies the state of the switches relative to the address:

Table A.5 Switch Controls and Address

Switch 1	Switch 2	Address
On	On	0xF000
Off	On	0xE000
On	Off	0xD000
Off	Off	0xC000

The LSI53C040 board contains 4 serial EEPROMS, two on each bus, that can be used for firmware. The jumpers located in the bottom center of the board specify the bus number and the chip address of the serial EEPROM whose code will be executed upon a reset. For all jumpers, the state is 1 if the top two pins are connected and 0 if the bottom two pins are connected.

Table A.6 Jumpers and Chip Address

JP8	JP5	JP6	Chip Address
1	1	1	E
0	1	1	6

Note 1: The Chip address is (JP8 JP9 JP6) <<1, so a setting of 111 becomes 0b1110 = 0x0E.

Note 2: The default settings for the board have serial EEPROMS on bus 1 and bus 0 at addresses 6 and E.

Table A.7 Jumper and Bus

JP7	Bus
1	1
0	0

The branch address for the default serial EEPROM is selected by the settings on jumpers nine and three. Table A.8 specifies the relationship between the jumpers and the branch address.

Table A.8 Jumpers and Branch Address

JP9	JP3	Branch Address
1	1	0x8000
1	0	0x4000
0	1	0x0033
0	0	0x0

Table A.9 Jumper and Code Load

JP4	Code Load
1	Enabled
0	Disabled

Index

E

F

G

I

J

L

M

O

P

Q

R

S

T

U

V

W

X

Customer Feedback

We would appreciate your feedback on this document. Please copy the following page, add your comments, and fax it to us at the number shown.

If appropriate, please also fax copies of any marked-up pages from this document.

Important: Please include your name, phone number, fax number, and company address so that we may contact you directly for clarification or additional information.

Thank you for your help in improving the quality of our documents.

Reader's Comments

Fax your comments to: LSI Logic Corporation
Technical Publications
M/S E-198
Fax: 408.433.4333

Please tell us how you rate this document: *LSI53C040 Enclosure Services Processor Programming Guide.* Place a check mark in the appropriate blank for each category.

	Excellent	Good	Average	Fair	Poor
Completeness of information	____	____	____	____	____
Clarity of information	____	____	____	____	____
Ease of finding information	____	____	____	____	____
Technical content	____	____	____	____	____
Usefulness of examples and illustrations	____	____	____	____	____
Overall manual	____	____	____	____	____

What could we do to improve this document?

_____

_____

_____

_____

If you found errors in this document, please specify the error and page number. If appropriate, please fax a marked-up copy of the page(s).

_____

_____

_____

Please complete the information below so that we may contact you directly for clarification or additional information.

Name _____ Date _____

Telephone _____ Fax _____

Title _____

Department _____ Mail Stop _____

Company Name _____

Street _____

City, State, Zip _____

U.S. Distributors by State

A. E. Avnet Electronics
http://www.hh.avnet.com
B. M. Bell Microproducts, Inc. (for HAB's)
http://www.bellmicro.com
I. E. Insight Electronics
http://www.insight-electronics.com
W. E. Wyle Electronics
http://www.wyle.com

Alabama
Daphne
I. E. Tel: 334.626.6190
Huntsville
A. E. Tel: 256.837.8700
B. M. Tel: 256.705.3559
I. E. Tel: 256.830.1222
W. E. Tel: 800.964.9953

Alaska
A. E. Tel: 800.332.8638

Arizona
Phoenix
A. E. Tel: 480.736.7000
B. M. Tel: 602.267.9551
W. E. Tel: 800.528.4040
Tempe
I. E. Tel: 480.829.1800
Tucson
A. E. Tel: 520.742.0515

Arkansas
W. E. Tel: 972.235.9953

California
Agoura Hills
B. M. Tel: 818.865.0266
Granite Bay
B. M. Tel: 916.523.7047
Irvine
A. E. Tel: 949.789.4100
B. M. Tel: 949.470.2900
I. E. Tel: 949.727.3291
W. E. Tel: 800.626.9953
Los Angeles
A. E. Tel: 818.594.0404
W. E. Tel: 800.288.9953
Sacramento
A. E. Tel: 916.632.4500
W. E. Tel: 800.627.9953
San Diego
A. E. Tel: 858.385.7500
B. M. Tel: 858.597.3010
I. E. Tel: 800.677.6011
W. E. Tel: 800.829.9953
San Jose
A. E. Tel: 408.435.3500
B. M. Tel: 408.436.0881
I. E. Tel: 408.952.7000
Santa Clara
W. E. Tel: 800.866.9953
Woodland Hills
A. E. Tel: 818.594.0404
Westlake Village
I. E. Tel: 818.707.2101

Colorado
Denver
A. E. Tel: 303.790.1662
B. M. Tel: 303.846.3065
W. E. Tel: 800.933.9953
Englewood
I. E. Tel: 303.649.1800
Idaho Springs
B. M. Tel: 303.567.0703

Connecticut
Cheshire
A. E. Tel: 203.271.5700
I. E. Tel: 203.272.5843
Wallingford
W. E. Tel: 800.605.9953

Delaware
North/South
A. E. Tel: 800.526.4812
Tel: 800.638.5988
B. M. Tel: 302.328.8968
W. E. Tel: 856.439.9110

Florida
Altamonte Springs
B. M. Tel: 407.682.1199
I. E. Tel: 407.834.6310
Boca Raton
I. E. Tel: 561.997.2540
Bonita Springs
B. M. Tel: 941.498.6011
Clearwater
I. E. Tel: 727.524.8850
Fort Lauderdale
A. E. Tel: 954.484.5482
W. E. Tel: 800.568.9953
Miami
B. M. Tel: 305.477.6406
Orlando
A. E. Tel: 407.657.3300
W. E. Tel: 407.740.7450
Tampa
W. E. Tel: 800.395.9953
St. Petersburg
A. E. Tel: 727.507.5000

Georgia
Atlanta
A. E. Tel: 770.623.4400
B. M. Tel: 770.980.4922
W. E. Tel: 800.876.9953
Duluth
I. E. Tel: 678.584.0812

Hawaii
A. E. Tel: 800.851.2282

Idaho
A. E. Tel: 801.365.3800
W. E. Tel: 801.974.9953

Illinois
North/South
A. E. Tel: 847.797.7300
Tel: 314.291.5350
Chicago
B. M. Tel: 847.413.8530
W. E. Tel: 800.853.9953
Schaumburg
I. E. Tel: 847.885.9700

Indiana
Fort Wayne
I. E. Tel: 219.436.4250
W. E. Tel: 888.358.9953
Indianapolis
A. E. Tel: 317.575.3500

Iowa
W. E. Tel: 612.853.2280
Cedar Rapids
A. E. Tel: 319.393.0033

Kansas
W. E. Tel: 303.457.9953
Kansas City
A. E. Tel: 913.663.7900
Lenexa
I. E. Tel: 913.492.0408

Kentucky
W. E. Tel: 937.436.9953
Central/Northern/ Western
A. E. Tel: 800.984.9503
Tel: 800.767.0329
Tel: 800.829.0146

Louisiana
W. E. Tel: 713.854.9953
North/South
A. E. Tel: 800.231.0253
Tel: 800.231.5775

Maine
A. E. Tel: 800.272.9255
W. E. Tel: 781.271.9953

Maryland
Baltimore
A. E. Tel: 410.720.3400
W. E. Tel: 800.863.9953
Columbia
B. M. Tel: 800.673.7461
I. E. Tel: 410.381.3131

Massachusetts
Boston
A. E. Tel: 978.532.9808
W. E. Tel: 800.444.9953
Burlington
I. E. Tel: 781.270.9400
Marlborough
B. M. Tel: 800.673.7459
Woburn
B. M. Tel: 800.552.4305

Michigan
Brighton
I. E. Tel: 810.229.7710
Detroit
A. E. Tel: 734.416.5800
W. E. Tel: 888.318.9953
Clarkston
B. M. Tel: 877.922.9363

Minnesota
Champlin
B. M. Tel: 800.557.2566
Eden Prairie
B. M. Tel: 800.255.1469
Minneapolis
A. E. Tel: 612.346.3000
W. E. Tel: 800.860.9953
St. Louis Park
I. E. Tel: 612.525.9999

Mississippi
A. E. Tel: 800.633.2918
W. E. Tel: 256.830.1119

Missouri
W. E. Tel: 630.620.0969
St. Louis
A. E. Tel: 314.291.5350
I. E. Tel: 314.872.2182

Montana
A. E. Tel: 800.526.1741
W. E. Tel: 801.974.9953

Nebraska
A. E. Tel: 800.332.4375
W. E. Tel: 303.457.9953

Nevada
Las Vegas
A. E. Tel: 800.528.8471
W. E. Tel: 702.765.7117

New Hampshire
A. E. Tel: 800.272.9255
W. E. Tel: 781.271.9953

New Jersey
North/South
A. E. Tel: 201.515.1641
Tel: 609.222.6400
Mt. Laurel
I. E. Tel: 856.222.9566
Pine Brook
B. M. Tel: 973.244.9668
W. E. Tel: 800.862.9953
Parsippany
I. E. Tel: 973.299.4425
Wayne
W. E. Tel: 973.237.9010

New Mexico
W. E. Tel: 480.804.7000
Albuquerque
A. E. Tel: 505.293.5119

U.S. Distributors by State (Continued)

New York
Hauppauge
I. E. Tel: 516.761.0960
Long Island
A. E. Tel: 516.434.7400
W. E. Tel: 800.861.9953
Rochester
A. E. Tel: 716.475.9130
I. E. Tel: 716.242.7790
W. E. Tel: 800.319.9953
Smithtown
B. M. Tel: 800.543.2008
Syracuse
A. E. Tel: 315.449.4927

North Carolina
Raleigh
A. E. Tel: 919.859.9159
I. E. Tel: 919.873.9922
W. E. Tel: 800.560.9953

North Dakota
A. E. Tel: 800.829.0116
W. E. Tel: 612.853.2280

Ohio
Cleveland
A. E. Tel: 216.498.1100
W. E. Tel: 800.763.9953
Dayton
A. E. Tel: 614.888.3313
I. E. Tel: 937.253.7501
W. E. Tel: 800.575.9953
Strongsville
B. M. Tel: 440.238.0404
Valley View
I. E. Tel: 216.520.4333

Oklahoma
W. E. Tel: 972.235.9953
Tulsa
A. E. Tel: 918.459.6000
I. E. Tel: 918.665.4664

Oregon
Beaverton
B. M. Tel: 503.524.1075
I. E. Tel: 503.644.3300
Portland
A. E. Tel: 503.526.6200
W. E. Tel: 800.879.9953

Pennsylvania
Mercer
I. E. Tel: 412.662.2707
Philadelphia
A. E. Tel: 800.526.4812
B. M. Tel: 877.351.2355
W. E. Tel: 800.871.9953
Pittsburgh
A. E. Tel: 412.281.4150
W. E. Tel: 440.248.9996

Rhode Island
A. E. 800.272.9255
W. E. Tel: 781.271.9953

South Carolina
A. E. Tel: 919.872.0712
W. E. Tel: 919.469.1502

South Dakota
A. E. Tel: 800.829.0116
W. E. Tel: 612.853.2280

Tennessee
W. E. Tel: 256.830.1119
East/West
A. E. Tel: 800.241.8182
Tel: 800.633.2918

Texas
Arlington
B. M. Tel: 817.417.5993
Austin
A. E. Tel: 512.219.3700
B. M. Tel: 512.258.0725
I. E. Tel: 512.719.3090
W. E. Tel: 800.365.9953
Dallas
A. E. Tel: 214.553.4300
B. M. Tel: 972.783.4191
W. E. Tel: 800.955.9953
El Paso
A. E. Tel: 800.526.9238
Houston
A. E. Tel: 713.781.6100
B. M. Tel: 713.917.0663
W. E. Tel: 800.888.9953
Richardson
I. E. Tel: 972.783.0800
Rio Grande Valley
A. E. Tel: 210.412.2047
Stafford
I. E. Tel: 281.277.8200

Utah
Centerville
B. M. Tel: 801.295.3900
Murray
I. E. Tel: 801.288.9001
Salt Lake City
A. E. Tel: 801.365.3800
W. E. Tel: 800.477.9953

Vermont
A. E. Tel: 800.272.9255
W. E. Tel: 716.334.5970

Virginia
A. E. Tel: 800.638.5988
W. E. Tel: 301.604.8488
Haymarket
B. M. Tel: 703.754.3399
Springfield
B. M. Tel: 703.644.9045

Washington
Kirkland
I. E. Tel: 425.820.8100
Maple Valley
B. M. Tel: 206.223.0080
Seattle
A. E. Tel: 425.882.7000
W. E. Tel: 800.248.9953

West Virginia
A. E. Tel: 800.638.5988

Wisconsin
Milwaukee
A. E. Tel: 414.513.1500
W. E. Tel: 800.867.9953
Wauwatosa
I. E. Tel: 414.258.5338

Wyoming
A. E. Tel: 800.332.9326
W. E. Tel: 801.974.9953

Direct Sales Representatives by State (Components and Boards)

E. A. Earle Associates
E. L. Electrodyne - UT
GRP Group 2000
I. S. Infinity Sales, Inc.
ION ION Associates, Inc.
R. A. Rathsburg Associates, Inc.
SGY Synergy Associates, Inc.

Arizona
Tempe
E. A. Tel: 480.921.3305

California
Calabasas
I. S. Tel: 818.880.6480
Irvine
I. S. Tel: 714.833.0300
San Diego
E. A. Tel: 619.278.5441

Illinois
Elmhurst
R. A. Tel: 630.516.8400

Indiana
Cicero
R. A. Tel: 317.984.8608
Ligonier
R. A. Tel: 219.894.3184
Plainfield
R. A. Tel: 317.838.0360

Massachusetts
Burlington
SGY Tel: 781.238.0870

Michigan
Byron Center
R. A. Tel: 616.554.1460
Good Rich
R. A. Tel: 810.636.6060
Novi
R. A. Tel: 810.615.4000

North Carolina
Cary
GRP Tel: 919.481.1530

Ohio
Columbus
R. A. Tel: 614.457.2242
Dayton
R. A. Tel: 513.291.4001
Independence
R. A. Tel: 216.447.8825

Pennsylvania
Somerset
R. A. Tel: 814.445.6976

Texas
Austin
ION Tel: 512.794.9006
Arlington
ION Tel: 817.695.8000
Houston
ION Tel: 281.376.2000

Utah
Salt Lake City
E. L. Tel: 801.264.8050

Wisconsin
Muskego
R. A. Tel: 414.679.8250
Saukville
R. A. Tel: 414.268.1152

Sales Offices and Design Resource Centers

LSI Logic Corporation
Corporate Headquarters
1551 McCarthy Blvd
Milpitas CA 95035
Tel: 408.433.8000
Fax: 408.433.8989

NORTH AMERICA

California
Irvine
18301 Von Karman Ave
Suite 900
Irvine, CA 92612
♦ Tel: 949.809.4600
Fax: 949.809.4444

Pleasanton Design Center
5050 Hopyard Road, 3rd Floor
Suite 300
Pleasanton, CA 94588
Tel: 925.730.8800
Fax: 925.730.8700

San Diego
7585 Ronson Road
Suite 100
San Diego, CA 92111
Tel: 858.467.6981
Fax: 858.496.0548

Silicon Valley
1551 McCarthy Blvd
Sales Office
M/S C-500
Milpitas, CA 95035
♦ Tel: 408.433.8000
Fax: 408.954.3353
Design Center
M/S C-410
Tel: 408.433.8000
Fax: 408.433.7695

Wireless Design Center
11452 El Camino Real
Suite 210
San Diego, CA 92130
Tel: 858.350.5560
Fax: 858.350.0171

Colorado
Boulder
4940 Pearl East Circle
Suite 201
Boulder, CO 80301
♦ Tel: 303.447.3800
Fax: 303.541.0641

Colorado Springs
4420 Arrowswest Drive
Colorado Springs, CO 80907
Tel: 719.533.7000
Fax: 719.533.7020

Fort Collins
2001 Danfield Court
Fort Collins, CO 80525
Tel: 970.223.5100
Fax: 970.206.5549

Florida
Boca Raton
2255 Glades Road
Suite 324A
Boca Raton, FL 33431
Tel: 561.989.3236
Fax: 561.989.3237

Georgia
Alpharetta
2475 North Winds Parkway
Suite 200
Alpharetta, GA 30004
Tel: 770.753.6146
Fax: 770.753.6147

Illinois
Oakbrook Terrace
Two Mid American Plaza
Suite 800
Oakbrook Terrace, IL 60181
Tel: 630.954.2234
Fax: 630.954.2235

Kentucky
Bowling Green
1262 Chestnut Street
Bowling Green, KY 42101
Tel: 270.793.0010
Fax: 270.793.0040

Maryland
Bethesda
6903 Rockledge Drive
Suite 230
Bethesda, MD 20817
Tel: 301.897.5800
Fax: 301.897.8389

Massachusetts
Waltham
200 West Street
Waltham, MA 02451
♦ Tel: 781.890.0180
Fax: 781.890.6158

Burlington - Mint Technology
77 South Bedford Street
Burlington, MA 01803
Tel: 781.685.3800
Fax: 781.685.3801

Minnesota
Minneapolis
8300 Norman Center Drive
Suite 730
Minneapolis, MN 55437
♦ Tel: 612.921.8300
Fax: 612.921.8399

New Jersey
Red Bank
125 Half Mile Road
Suite 200
Red Bank, NJ 07701
Tel: 732.933.2656
Fax: 732.933.2643

Cherry Hill - Mint Technology
215 Longstone Drive
Cherry Hill, NJ 08003
Tel: 856.489.5530
Fax: 856.489.5531

New York
Fairport
550 Willowbrook Office Park
Fairport, NY 14450
Tel: 716.218.0020
Fax: 716.218.9010

North Carolina
Raleigh
Phase II
4601 Six Forks Road
Suite 528
Raleigh, NC 27609
Tel: 919.785.4520
Fax: 919.783.8909

Oregon
Beaverton
15455 NW Greenbrier Parkway
Suite 235
Beaverton, OR 97006
Tel: 503.645.0589
Fax: 503.645.6612

Texas
Austin
9020 Capital of TX Highway North
Building 1
Suite 150
Austin, TX 78759
Tel: 512.388.7294
Fax: 512.388.4171

Plano
500 North Central Expressway
Suite 440
Plano, TX 75074
♦ Tel: 972.244.5000
Fax: 972.244.5001

Houston
20405 State Highway 249
Suite 450
Houston, TX 77070
Tel: 281.379.7800
Fax: 281.379.7818

Canada
Ontario
Ottawa
260 Hearst Way
Suite 400
Kanata, ON K2L 3H1
♦ Tel: 613.592.1263
Fax: 613.592.3253

INTERNATIONAL

France
Paris
LSI Logic S.A.
Immeuble Europa
53 bis Avenue de l'Europe
B.P. 139
78148 Velizy-Villacoublay
Cedex, Paris
♦ Tel: 33.1.34.63.13.13
Fax: 33.1.34.63.13.19

Germany
Munich
LSI Logic GmbH
Orleansstrasse 4
81669 Munich
♦ Tel: 49.89.4.58.33.0
Fax: 49.89.4.58.33.108

Stuttgart
Mittlerer Pfad 4
D-70499 Stuttgart
♦ Tel: 49.711.13.96.90
Fax: 49.711.86.61.428

Italy
Milan
LSI Logic S.P.A.
Centro Direzionale Colleoni Palazzo
Orione Ingresso 1
20041 Agrate Brianza, Milano
♦ Tel: 39.039.687371
Fax: 39.039.6057867

Japan
Tokyo
LSI Logic K.K.
Rivage-Shinagawa Bldg. 14F
4-1-8 Kounan
Minato-ku, Tokyo 108-0075
♦ Tel: 81.3.5463.7821
Fax: 81.3.5463.7820

Osaka
Crystal Tower 14F
1-2-27 Shiromi
Chuo-ku, Osaka 540-6014
♦ Tel: 81.6.947.5281
Fax: 81.6.947.5287

Sales Offices and Design Resource Centers (Continued)

Korea
Seoul
LSI Logic Corporation of Korea Ltd
10th Fl., Haesung 1 Bldg.
942, Daechi-dong,
Kangnam-ku, Seoul, 135-283
Tel: 82.2.528.3400
Fax: 82.2.528.2250

The Netherlands
Eindhoven
LSI Logic Europe Ltd
World Trade Center Eindhoven
Building 'Rijder'
Bogert 26
5612 LZ Eindhoven
Tel: 31.40.265.3580
Fax: 31.40.296.2109

Singapore
Singapore
LSI Logic Pte Ltd
7 Temasek Boulevard
#28-02 Suntec Tower One
Singapore 038987
Tel: 65.334.9061
Fax: 65.334.4749

Sweden
Stockholm
LSI Logic AB
Finlandsgatan 14
164 74 Kista
♦ Tel: 46.8.444.15.00
Fax: 46.8.750.66.47

Taiwan
Taipei
LSI Logic Asia, Inc.
Taiwan Branch
10/F 156 Min Sheng E. Road
Section 3
Taipei, Taiwan R.O.C.
Tel: 886.2.2718.7828
Fax: 886.2.2718.8869

United Kingdom
Bracknell
LSI Logic Europe Ltd
Greenwood House
London Road
Bracknell, Berkshire RG12 2UB
♦ Tel: 44.1344.426544
Fax: 44.1344.481039

♦ Sales Offices with Design Resource Centers

International Distributors

Australia
New South Wales
Reptechnic Pty Ltd
3/36 Bydown Street
Neutral Bay, NSW 2089
♦ Tel: 612.9953.9844
Fax: 612.9953.9683

Belgium
Acal nv/sa
Lozenberg 4
1932 Zaventem
Tel: 32.2.7205983
Fax: 32.2.7251014

China
Beijing
LSI Logic International Services Inc.
Beijing Representative Office
Room 708
Canway Building
66 Nan Li Shi Lu
Xicheng District
Beijing 100045, China
Tel: 86.10.6804.2534 to 38
Fax: 86.10.6804.2521

France
Rungis Cedex
Azzurri Technology France
22 Rue Saarinen
Sillic 274
94578 Rungis Cedex
Tel: 33.1.41806310
Fax: 33.1.41730340

Germany
Haar
EBV Elektronik
Hans-Pinsel Str. 4
D-85540 Haar
Tel: 49.89.4600980
Fax: 49.89.46009840

Munich
Avnet Emg GmbH
Stahlgruberring 12
81829 Munich
Tel: 49.89.45110102
Fax: 49.89.42.27.75

Wuennenberg-Haaren
Peacock AG
Graf-Zepplin-Str 14
D-33181 Wuennenberg-Haaren
Tel: 49.2957.79.1692
Fax: 49.2957.79.9341

Hong Kong
Hong Kong
AVT Industrial Ltd
Unit 608 Tower 1
Cheung Sha Wan Plaza
833 Cheung Sha Wan Road
Kowloon, Hong Kong
Tel: 852.2428.0008
Fax: 852.2401.2105

Serial System (HK) Ltd
2301 Nanyang Plaza
57 Hung To Road, Kwun Tong
Kowloon, Hong Kong
Tel: 852.2995.7538
Fax: 852.2950.0386

India
Bangalore
Spike Technologies India Private Ltd
951, Vijayalakshmi Complex,
2nd Floor, 24th Main,
J P Nagar II Phase,
Bangalore, India 560078
♦ Tel: 91.80.664.5530
Fax: 91.80.664.9748

Israel
Tel Aviv
Eastronics Ltd
11 Rozanis Street
P.O. Box 39300
Tel Aviv 61392
Tel: 972.3.6458777
Fax: 972.3.6458666

Japan
Tokyo
Daito Electron
Sogo Kojimachi No.3 Bldg
1-6 Kojimachi
Chiyoda-ku, Tokyo 102-8730
Tel: 81.3.3264.0326
Fax: 81.3.3261.3984

Global Electronics Corporation
Nichibei Time24 Bldg. 35 Tansu-cho
Shinjuku-ku, Tokyo 162-0833
Tel: 81.3.3260.1411
Fax: 81.3.3260.7100
Technical Center
Tel: 81.471.43.8200

Marubeni Solutions
1-26-20 Higashi
Shibuya-ku, Tokyo 150-0001
Tel: 81.3.5778.8662
Fax: 81.3.5778.8669

Shinki Electronics
Myuru Daikanyama 3F
3-7-3 Ebisu Minami
Shibuya-ku, Tokyo 150-0022
Tel: 81.3.3760.3110
Fax: 81.3.3760.3101

Yokohama-City
Innotech
2-15-10 Shin Yokohama
Kohoku-ku
Yokohama-City, 222-8580
Tel: 81.45.474.9037
Fax: 81.45.474.9065

Macnica Corporation
Hakusan High-Tech Park
1-22-2 Hadusan, Midori-Ku,
Yokohama-City, 226-8505
Tel: 81.45.939.6140
Fax: 81.45.939.6141

The Netherlands
Eindhoven
Acal Nederland b.v.
Beatrix de Rijkweg 8
5657 EG Eindhoven
Tel: 31.40.2.502602
Fax: 31.40.2.510255

Switzerland
Brugg
LSI Logic Sulzer AG
Mattenstrasse 6a
CH 2555 Brugg
Tel: 41.32.3743232
Fax: 41.32.3743233

Taiwan
Taipei
Avnet-Mercuries Corporation, Ltd
14F, No. 145,
Sec. 2, Chien Kuo N. Road
Taipei, Taiwan, R.O.C.
Tel: 886.2.2516.7303
Fax: 886.2.2505.7391

Lumax International Corporation, Ltd
7th Fl., 52, Sec. 3
Nan-Kang Road
Taipei, Taiwan, R.O.C.
Tel: 886.2.2788.3656
Fax: 886.2.2788.3568

Prospect Technology Corporation, Ltd
4Fl., No. 34, Chu Luen Street
Taipei, Taiwan, R.O.C.
Tel: 886.2.2721.9533
Fax: 886.2.2773.3756

Wintech Microeletronics Co., Ltd
7F., No. 34, Sec. 3, Pateh Road
Taipei, Taiwan, R.O.C.
Tel: 886.2.2579.5858
Fax: 886.2.2570.3123

United Kingdom
Maidenhead
Azzurri Technology Ltd
16 Grove Park Business Estate
Waltham Road
White Waltham
Maidenhead, Berkshire SL6 3LW
Tel: 44.1628.826826
Fax: 44.1628.829730

Milton Keynes
Ingram Micro (UK) Ltd
Garamonde Drive
Wymbush
Milton Keynes
Buckinghamshire MK8 8DF
Tel: 44.1908.260422

Swindon
EBV Elektronik
12 Interface Business Park
Bincknoll Lane
Wootton Bassett,
Swindon, Wiltshire SN4 8SY
Tel: 44.1793.849933
Fax: 44.1793.859555

♦ Sales Offices with Design Resource Centers